2023-2 공학설계

3주차 주간보고

11조

20191446 박형욱

20192074 임지영

프로젝트 진행 상황

W2 (10/31 - 11/6)

- Median Filter 구현
- Gaussian Filter 구현

W3 (11/7 - 13)

- Median Filter 최적화
- Gaussian Filter 최적화
- Bilateral Filter 구현

W4 (11/14 - 20)

- 중간 발표

금주 진행 상황

Median Filter 최적화

Median Filter 커널 사이즈에 따른 PSNR 값 비교

```
 8    k size : 3,   sigma = 7
 9    PSNR : 26.916010196318577
10    stride 1, filter size : 3
11    k size : 5,   sigma = 1
12    PSNR : 27.092299276174035
13    stride 1, filter size : 3
14    k size : 5,   sigma = 4
15    PSNR : 27.161295722165132
16    stride 1, filter size : 3
17    k size : 5,   sigma = 7
18    PSNR : 27.148359033237472
19    stride 1, filter size : 3
20    k size : 7,   sigma = 1
21    PSNR : 27.12090264185008
```

```python
#median filter

def median_filter(img, filter_size=(3, 3), stride=1):
    img_shape = np.shape(img)
    result_shape = tuple(np.int64((np.array(img_shape[:2]) - np.array(filter_size)) / stride) + 1) + (img_shape[2],)
    result = np.zeros(result_shape)

    for h in range(0, result_shape[0], stride):
        for w in range(0, result_shape[1], stride):
            for c in range(img_shape[2]):
                tmp = img[h:h + filter_size[0], w:w + filter_size[1], c].ravel()
                tmp = np.sort(tmp)
                result[h, w, c] = tmp[int(len(tmp) / 2)]

    return result
```

- 커널 사이즈 4일 때 가장 높은 PSNR 도출

금주 진행 상황

Gaussian Filter 최적화

Gaussian Filter 커널 사이즈 및 표준 편차 값에 따른 PSNR 값 비교

```
 8    k size : 3,  sigma = 7
 9    PSNR : 26.916010196318577
10    stride 1, filter size : 3
11    k size : 5,  sigma = 1
12    PSNR : 27.092299276174035
13    stride 1, filter size : 3
14    k size : 5,  sigma = 4
15    PSNR : 27.161295722165132
16    stride 1, filter size : 3
17    k size : 5,  sigma = 7
18    PSNR : 27.148359033237472
19    stride 1, filter size : 3
20    k size : 7,  sigma = 1
21    PSNR : 27.12090264185008
```

```python
for k in range(0,3):
    mfs = 3 + (k * 2)
    filter_size = (mfs,mfs)
    stride = 1
    median_img = median_filter(noisy_image, filter_size, stride)
    median_img_resized = resize(median_img, img.shape, mode ='constant', anti_aliasing=True)
    median_img_resized = np.clip(median_img_resized, 0., 1.0)
    for i in range(0, 3):
        for j in range(0,3):
            k_size = 3 + 2 * i
            sigma = 3 * j + 1
            res = gaussian_filtering(median_img_resized, k_size = k_size, sigma = sigma)
            title = "stride 1, filter size : " + str(mfs) + "\nk size : " + str(k_size) + ",  sigma = " + str(sigma) + "\nPSNR : " + str(peak_signal_noise_ratio(img, res))
            print(title)
```

- 커널 사이즈 5, 표준 편차 4일 때 가장 높은 PSNR 도출

금주 진행 상황

Bilateral Filter 구현

이미지에 Gaussian Filter 적용 시,

엣지 부근에서 픽셀 값을 평탄하게 만드는 단점을 보완하기 위해 Bilateral Filter 구현

ALGORITHM 5.13 Perform bilateral filtering on an image

BILATERALFILTER$(I, \sigma_s, \sigma_r, n_{iter})$

Input: grayscale image I, standard deviations σ_s and σ_r of Gaussian spatial and range kernels, number n_{iter} of iterations

Output: bilateral-filtered image

```
1  for k ← 1 to n_iter do                                  ➤ For each iteration,
2      for (x, y) ∈ I do                                   and for each pixel in the image,
3          val ← 0                                         initialize the value to zero,
4          norm ← 0                                        and the normalization factor to zero.
5          for (δ_x, δ_y) ∈ W do                           ➤ For each pixel in a ±2.5σ_s window,
6              d_s^2 ← δ_x * δ_x + δ_y * δ_y               compute squared spatial distance,
7              d_r ← I(x, y) − I(x + δ_x, y + δ_y)          and range difference
8              w ← exp(−d_s^2/(2 * σ_s^2)) * exp(−(d_r * d_r)/(2 * σ_r^2))   to compute weight.
9              val ←+ w * I(x + δ_x, y + δ_y)              ➤ Accumulate weighted sum
10             norm ←+ w                                   and update normalization factor.
11         I'(x, y) ← val/norm                             ➤ Set output to normalized weighted sum.
12     I ← I'                                              ➤ Copy entire output image to input for next iteration.
13 return I'
```

```python
def bilateral_filter(noisy_img, k_size=5, sigma_space=4, sigma_intensity=0.2):
    h, w, ch = noisy_img.shape
    bilateral_noisy_img = np.zeros((h, w, ch))
    spatial_filter = gaussian_kernel(k_size, sigma_space)
    for c in range(ch):
        for i in range(h):
            for j in range(w):
                intensity_center = noisy_img[i, j, c]
                weighted_sum = 0.0
                normalization_factor = 0.0
                for m in range(-k_size//2, k_size//2 + 1):
                    for n in range(-k_size//2, k_size//2 + 1):
                        i_neighbor = i + m
                        j_neighbor = j + n

                        if 0 <= i_neighbor < h and 0 <= j_neighbor < w:
                            intensity_neighbor = noisy_img[i_neighbor, j_neighbor, c]
                            weight_intensity = np.exp(-(intensity_center - intensity_neighbor)**2 / (2 * sigma_intensity**2))
                            weight_spatial = spatial_filter[m + k_size//2, n + k_size//2]
                            weight = weight_intensity * weight_spatial
                            weighted_sum += intensity_neighbor * weight
                            normalization_factor += weight

                bilateral_noisy_img[i, j, c] = weighted_sum / normalization_factor

    return bilateral_noisy_img
```

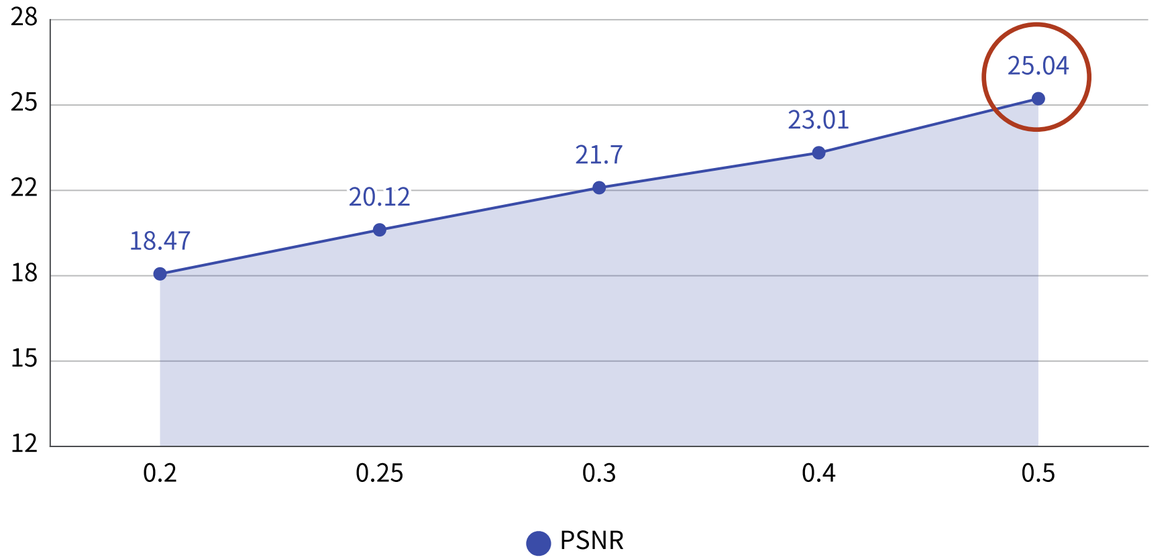
금주 진행 상황

Bilateral Filter

Filter 표준 편차 값에 따른 PSNR 값 비교

- 표준 편차 5일 때, 가장 높은 PSNR 도출

```
sigma_intemsity=0.2 - PSNR: 18.465199371341477
sigma_intemsity=0.25 - PSNR: 20.11810428632533
sigma_intemsity=0.3 - PSNR: 21.69718677355458
sigma_intemsity=0.35 - PSNR: 23.00701856295732
sigma_intemsity=0.4 - PSNR: 23.966817433556663
sigma_intemsity=0.5 - PSNR: 25.0380909386099
```



금주 진행 상황

Bilateral Filter 구현 결과

```python
def bilateral_filter(noisy_img, k_size=5, sigma_space=4, sigma_intensity=0.2):
    h, w, ch = noisy_img.shape
    bilateral_noisy_img = np.zeros((h, w, ch))
    spatial_filter = gaussian_kernel(k_size, sigma_space)
    for c in range(ch):
        for i in range(h):
            for j in range(w):
                intensity_center = noisy_img[i, j, c]
                weighted_sum = 0.0
                normalization_factor = 0.0
                for m in range(-k_size//2, k_size//2 + 1):
                    for n in range(-k_size//2, k_size//2 + 1):
                        i_neighbor = i + m
                        j_neighbor = j + n

                        if 0 <= i_neighbor < h and 0 <= j_neighbor < w:
                            intensity_neighbor = noisy_img[i_neighbor, j_neighbor, c]
                            weight_intensity = np.exp(-(intensity_center - intensity_neighbor)**2 / (2 * sigma_intensity**2))
                            weight_spatial = spatial_filter[m + k_size//2, n + k_size//2]
                            weight = weight_intensity * weight_spatial
                            weighted_sum += intensity_neighbor * weight
                            normalization_factor += weight

                bilateral_noisy_img[i, j, c] = weighted_sum / normalization_factor

    return bilateral_noisy_img
```

- Dog_noisy (4.9 Mb)
  약 8분 소요
- Baby_noisy (25.8 Mb)
  약 50분 소요

→ 기존 코드보다 더 빠르게 실행 결과를 얻을 수 있도록 코드 최적화

감사합니다