2023-2 공학설계

5주차 주간 보고

11조

20191446 박형욱

20192074 임지영

프로젝트 진행 상황

W4 (11/14 - 20)

- Bilateral Filter 구현
- Bilateral Filter 최적화

W5 (11/21 - 27)

- Non-local Means Filter 구현
- 파라미터 값에 따른 영상 품질 비교

W6 (11/28 - 12/4)

- Non-local Means Filter 최적화
- 파라미터 값에 따른 영상 품질 비교

White Gaussian Noise

Non-local Means Filter

ALGORITHM 5.12 Perform non-local means filtering on an image

NonLocalMeans(I, w)

Input: grayscale image I, set of pixels $\mathcal{W}$ specifying window
Output: smoothed image from applying non-local means filtering

```
1  for (x, y) ∈ I' do                          ➤ For each pixel in the output image (same size as input image),
2      val ← 0                                   initialize value to zero,
3      norm ← 0                                  and normalization factor to zero.
4      for (x', y') ∈ I do                       ➤ For each pixel in input image,
5          d ← 0                                 initialize distance to zero.
6          for (δx, δy) ∈ W do                   ➤ Compute the dissimilarity between the two windows.
7              d ←+ I(x + δx, y + δy) − I(x' + δx, y' + δy)
8          w ← exp(−d * d/(2 * σ * σ))           ➤ Set the weight to the similarity.
9          val ←+ w * I(x', y')                  ➤ Update the sum of the weighted pixels,
10         norm ←+ w                             and the normalization factor.
11     I'(x, y) ← val/norm                       ➤ Set the output pixel to the normalized value.
12 return I'
```

영상 내에서 특정 픽셀 주변 영역과 비슷한 패턴을 갖는 영역들을 찾아내 평균을 구하는 알고리즘

White Gaussian Noise

Non-local Means Filter 구현

```python
def NLMeans_Filter(noisy_img, h=0.1, patch_size=5, search_window=11):
    denoised_img = np.zeros_like(noisy_img)
    padded_img = np.pad(noisy_img, ((patch_size//2,),(patch_size//2,),(0,)), 'reflect')
    height, width, channels = noisy_img.shape

    patches = np.array([
        padded_img[i:i + patch_size, j:j + patch_size, :]
        for i in range(height)
        for j in range(width)
    ])

    for y in range(height):
        for x in range(width):
            patch = patches[y * width + x];  weight_sum = 0.0;  weighted_sum = np.zeros((channels,))

            for dy in range(-search_window//2, search_window//2 + 1):
                for dx in range(-search_window//2, search_window//2 + 1):
                    yy, xx = y + dy, x + dx
                    if 0 <= yy < height and 0 <= xx < width:
                        neighbor_patch = patches[yy * width + xx];  patch_diff = np.sum((neighbor_patch - patch) ** 2)
                        weight = np.exp(-patch_diff / (h ** 2));    weighted_sum += weight * neighbor_patch[patch_size//2, patch_size//2, :]
                        weight_sum += weight

            denoised_img[y, x, :] = weighted_sum / weight_sum

    return denoised_img
```

모든 픽셀에 대해 Patch 생성

검색 윈도우 내 모든 이웃한 픽셀 고려

Image Denoising

Denoising 결과 (Median, Bilateral Filter 적용)

- Median Filter: filter size = 3, stride = 1
- Bilateral Filter: kernel size = 7, sigma space = 7, intensity = 0.2

	Baby	Bagle	Beach	Book	Dog	Girl	Lego	Kitty	House	Street
PSNR	28.35	24.66	26.61	27.12	26.84	26.04	25.34	28.61	21.39	22.78
SSIM	0.86	0.80	0.86	0.87	0.79	0.89	0.81	0.88	0.65	0.69