

Дата: 12.06.23

ФИО: Козлов Евгений Юрьевич

Группа: 224-322

ЛАБОРАТОРНАЯ РАБОТА №2

Сравнение различных фильтров для устранения шумов в изображении

1. Цель работы

Проанализировать возможности фильтров для устранения различных шумовых структур, подобрать параметры фильтрации под конкретное изображение.

2. Содержание работы

1. Проанализировать предложенные изображения определить тип шумовой структуры.
2. Обосновать выбор фильтров, которые могут быть использованы для устранения шумов в предложенных изображениях [4]
3. Провести фильтрацию изображений для устранения шумовой структуры [4, 6]
4. Оценить эффективность устранения шумов с помощью показателя PSNR [5]
5. Построить диаграмму, отражающую изменение показателя PSNR для изображений до и после фильтрации





3. Исходные данные и программное обеспечение

Используемая среда программирования: Visual Studio Code

Используемый язык программирования: Python 3.11.1 64-bit

Используемые библиотеки: numpy, scipy, skimage, matplotlib

4. Выполнение работы

Изображение с шумовой структурой	test2_1	test2_2	test2_3	test2_4	
					
	Флуктуационный шум	Шум квантования	Импульсный шум	Шум дискретизации	
	TV Chambolle	TV Bregman	Медианный фильтр	Гауссовский фильтр	
	Вариационные фильтры усредняют шум, сохраняя резкость контуров объектов. Шум распределён равномерно по всему изображению.	Фильтр здесь должен быть более чувствителен для определения шума. Был использован метод на регуляризации Брегмана.	Медианный фильтр оперирует на небольших областях изображения, заменяя резкие пики медианным значениями.	Гауссовский фильтр плавно размывает изображение, стирая резкие границы, но изображение при этом размывается.	
Параметры фильтрации	weight = 0.1, eps = 0.00000015	weight = 7.5, eps = 0.000005	Маска: прямоугольник 5×9 для выреза галочки	sigma_x = 5, _y = 4 (как шаг пикс. сетки)	
PSNR	Ориг/Шум	27.98	34.46	31.66	26.637853
	Шум/Фильтр	28.86	33.90	28.67	25.547059
	Ориг/Фильтр	31.90	33.86	31.40	24.512642

Результаты

Слева представлено зашумленное изображение, справа – его отфильтрованный результат.

1. Флуктуационный шум.



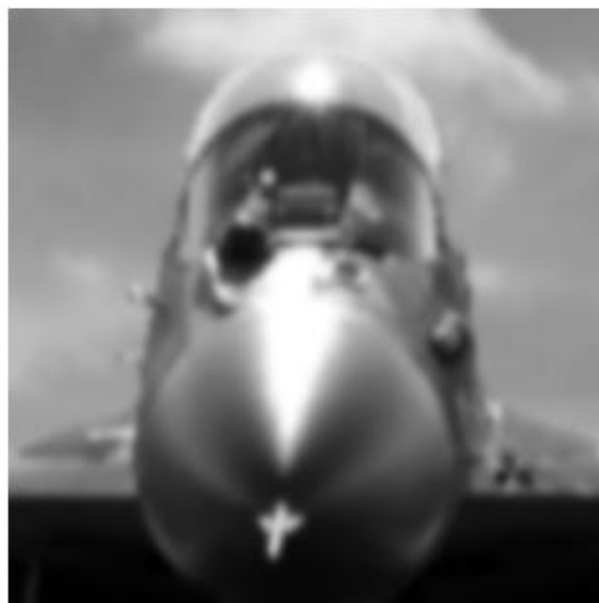
2. Шум квантования (постеризация).



3. Импульсный шум.



4. Шум дискретизации.



Вывод

Было проведено сравнение различных фильтров для устранения шумов в изображении. Необходимо понимать тип шумовой структуры для выбора подходящего фильтра, иначе обработка может нанести ещё большие потери

информации. В работе фильтров используются статистические методы, благодаря им происходит полное размытие изображения до значений среднего, медианного и т. п. уровня.

Почти все фильтры не различают зашумлённые и незашумлённые области. Чтобы минимизировать потери, стоит применять фильтры избирательно, и если шумовая структура это позволяет — на определённой области.

Все изображения размещены на гугл-диске по адресу:

<https://drive.google.com/drive/folders/1UWP1KEAZR4wI52x2OGs2OXu3Za2xE0BF?usp=sharing>

Код работы:

```
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib.pyplot as plt

from skimage.io import imread, imshow, imsave
from skimage import data, img_as_float, metrics, restoration, filters,
morphology as morph

# https://sci-hub.ru/https://www.sciencedirect.com/science/arti-
# cle/abs/pii/S016727899290242F
# https://eeweb.engineering.nyu.edu/iselesni/lecture_notes/TVDmm/
# https://www.youtube.com/watch?v=6cRwZ19iiHo
# https://scikit-image.org/skimage-tutorials/lectures/1_image_fil-
# ters.html

# skimage.io некорректно работает с сохранением изображения с
# low_contrast, даже если флаг check_contrast=False
# https://github.com/scikit-image/scikit-image/issues/3819
# тем не менее, библиотека позволяет их выводить на экран, поэтому для
# данной лабораторной работы был использован
# функционал Jupiter Notebook

emap = lambda f, xs: [f(x) for x in xs]
unzip = lambda xs: [[a for a, _ in xs], [b for _, b in xs]]

show1 = lambda pic: (lambda ax: ax.axis('off') and ax.imshow(pic,
cmap='gray'))(plt.subplots()[1])
float64 = lambda x: x.astype(np.float64) / 255.0
loadf64 = lambda x: float64(imread(x))

psnr = metrics.peak_signal_noise_ratio
compare = lambda fn, o, n, f: {'Ориг/Шум': fn(o, n), 'Шум/Фильтр': fn(n,
f), 'Ориг/Фильтр': fn(o, f)}
```

```

psnr_ssim1 = lambda o, n, f: {'': compare(psnr, o, n, f)}
psnr_ssim = lambda o, n, f: pd.DataFrame(psnr_ssim1(o, n, f))

def show(pics, title=None):
    cols, rows = 2, (len(pics) // 2) + (len(pics) % 2)
    fig, axs = plt.subplots(rows, cols, squeeze=True, constrained_layout=True)
    show1 = lambda pic, ax: ax.axis('off') and ax.imshow(pic,
cmap='gray')
    [show1(pic, ax) for pic, ax in zip(pics, axs.flat)]

IMG_ARR = emap(loadf64, ['img/init/test2_0.jpg',
'img/dist/1/test2_1.jpg', 'img/dist/2/test2_2.jpg',
'img/dist/3/test2_3.jpg', 'img/dist/4/test2_4.jpg'])

INIT_IMG, STAT, POST, SQRT, PIXL = IMG_ARR
emap(show1, IMG_ARR)

# Флуктуационный шум
# Вариационные фильтры усредняют шум, сохраняя резкость контуров объектов. Шум распределён равномерно по всему изображению.
FILTERED_1 = restoration.denoise_tv_chambolle(stat, 0.08,
eps=0.00000015)
show([STAT, FILTERED_1])
psnr_ssim(INIT_IMG, STAT, FILTERED_1).T.plot.barh(title='PSNR, Флуктуационный шум', color=['yellowgreen', 'teal', 'mediumseagreen'])

def some_highlights(styler, min_color="red", max_color="green"):
    styler.highlight_min(color=min_color, axis=None)
    styler.highlight_max(color=max_color, axis=None)
    return styler

psnr_ssim(INIT_IMG, STAT, FILTERED_1).style.pipe(some_highlights)

# Шум квантования
# Фильтр здесь должен быть более чувствителен для определения шума. Был использован метод на регуляризации Брегмана.
FILTERED_2 = restoration.denoise_tv_bregman(POST, 7.5, eps=0.000005,
max_num_iter=5000)
show([POST, FILTERED_2])
psnr_ssim(INIT_IMG, POST, FILTERED_2).T.plot.barh(title='PSNR, Шум квантования (постеризация)', color=['yellowgreen', 'teal', 'mediumseagreen'])
psnr_ssim(INIT_IMG, POST, FILTERED_2).T.T.style.pipe(some_highlights)

# Импульсный шум
# Медианный фильтр оперирует на небольших областях изображения, заменяя резкие пики медианным знаменами.
FILTERED_3 = filters.median(SQRT, morph.rectangle(5, 9))
show([SQRT, FILTERED_3])

```

```
psnr_ssim(INIT_IMG, SQRT, FILTERED_3).T.plot.barh(title='PSNR, Импульс-  
ный шум', color=['yellowgreen', 'teal', 'mediumseagreen'])  
psnr_ssim(INIT_IMG, SQRT, FILTERED_3).T.T.style.pipe(some_highlights)  
  
# Шум дискретизации  
# Гауссовский фильтр плавно размывает изображение, стирая резкие гра-  
ницы, но изображение при этом размывается.  
FILTERED_4 = filters.gaussian(PIXL, sigma=(5, 4))  
show([PIXL, FILTERED_4])  
psnr_ssim(INIT_IMG, PIXL, FILTERED_4).T.plot.barh(title='PSNR, Шум дис-  
кретизации', color=['yellowgreen', 'teal', 'mediumseagreen'])  
psnr_ssim(INIT_IMG, PIXL, FILTERED_4).T.T.style.pipe(some_highlights)  
  
# Вывод данных  
pd.DataFrame({  
    'Флуктуационный шум': psnr_ssim1(INIT_IMG, STAT, FILTERED_1),  
    'Шум квантования': psnr_ssim1(INIT_IMG, POST, FILTERED_2),  
    'Импульсный шум': psnr_ssim1(INIT_IMG, SQRT, FILTERED_3),  
    'Шум дискретизации': psnr_ssim1(INIT_IMG, PIXL, FILTERED_4),  
}).T
```