

Цель работы

Получение практических навыков использования алгоритмов улучшения качества изображений, применяемых в пространственной области, на примерах методов шумоподавления, выделения контуров и градационных преобразований.

1. Реализация модели аддитивного шума.

При выполнении этого и последующих пунктов будет проводиться работа с яркостной составляющей изображения Y-компонентой, значение в пикселе которой вычисляется по следующей формуле:

$$Y = 0.299R + 0.587G + 0.114B,$$

где R , G , и B – значения цветовых компонент формата RGB для соответствующего пикселя.

Модель формирования аддитивного шума для 8-битных значений интенсивностей пикселей изображения можно представить следующим образом:

$$I'_{x,y} = \text{Clipp}(I_{x,y} + N_{x,y} 0.255),$$

где $N_{x,y}$ – значение шума на позиции пикселя с координатами (x, y) , а Clipp – операция клиппирования, соответствующая следующей формуле:

$$\text{Clipp}(I_{x,y}, I^{min}, I^{max}) = \begin{cases} I^{min}, & \text{если } I_{x,y} < I^{min} \\ I^{max}, & \text{если } I_{x,y} > I^{max} \\ I_{x,y}, & \text{иначе} \end{cases}$$

Значение шума $N_{x,y}$ можно сгенерировать, воспользовавшись преобразованием Бокса-Мюллера:

Пусть r и φ – независимые случайные величины, равномерно распределенные на интервале $(0, 1]$ и z_0 и z_1 определены как

$$z_0 = \cos(2\pi\varphi) \sqrt{-2 \ln(r)}$$

$$z_1 = \sin(2\pi\varphi) \sqrt{-2 \ln(r)}$$

тогда z_0 и z_1 – независимы и имеют нормально распределение с математическим ожиданием 0 и дисперсией 1.

Чтобы перейти к общему нормальному распределению воспользуемся формулой:

$$\varepsilon = \mu + \sigma z$$

где μ – математическое ожидание, σ – стандартное отклонение, а ε – случайная величина с Гауссовским распределением $N(\mu, \sigma^2)$.

Результаты формирования аддитивного шума:



Рис. 1. Изображение с шумом при $\sigma = 50$.

2. Реализация модели импульсного шума.

Модель формирования импульсного шума для 8-битных значений интенсивностей описывается следующей формулой:

$$I_{x,y} = \begin{cases} 0, \text{ с вероятностью } p_a \\ 255, \text{ с вероятностью } p_b \\ I_{x,y}, \text{ с вероятностью } 1 - p_a - p_b \end{cases},$$

где p_a и p_b – параметры системы.

Результаты формирования шума:



Рис. 2. Изображение с шумом при $ra = 0.03, pb = 0.07$

Как видно на изображениях импульсный шум более выражен чем аддитивный даже при небольшом значение вероятности, т.к. зашумлённые пиксели принимают крайние значения 0 или 255.

3. Построение графиков PSNR для аддитивного и импульсного шумов.

Необходимо построить графики $PSNR(\sigma)$ и $PSNR(p_a, p_b)$.

PSNR (пиковое отношение сигнал-шум) – соотношение между максимально возможным значением сигнала и мощности шума, искажающего значение сигнала. Значение PSNR вычисляется по следующей формуле:

$$PSNR = 10 \lg \frac{WH(2^L - 1)^2}{\sum_{i=1}^H \sum_{j=1}^W \left(I_{i,j}^{(A)} - \hat{I}_{i,j}^{(A)} \right)^2},$$

где $I_{i,j}^{(A)}$ – исходное значение компоненты A по координатам (i, j) , $\hat{I}_{i,j}^{(A)}$ – значение компоненты A по этим же координатам в искаженном шумом изображении, W и H – ширина и высота изображения соответственно, а L – количество бит, выделенных на значение компоненты.

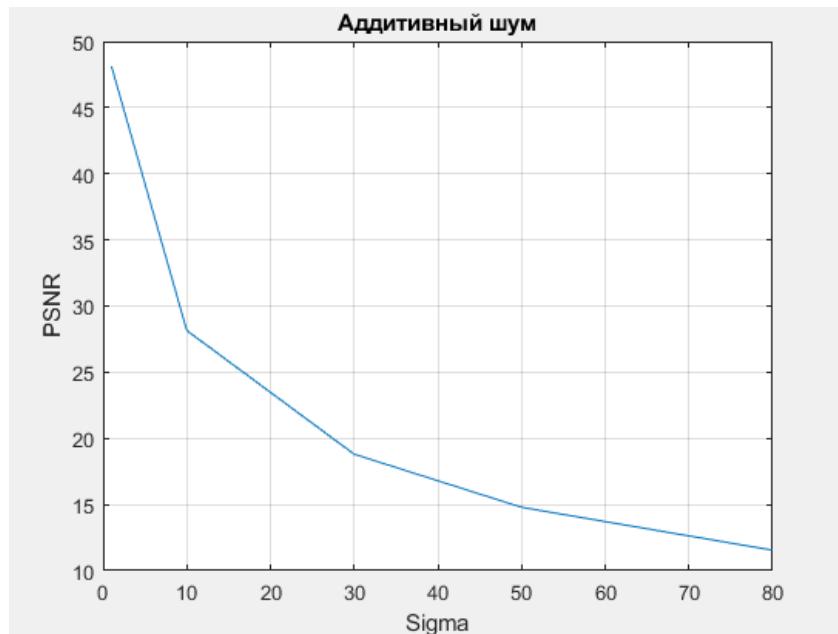


Рис. 3. График $PSNR(\sigma)$ при $\sigma = 1, 10, 30, 50$ и 80 .

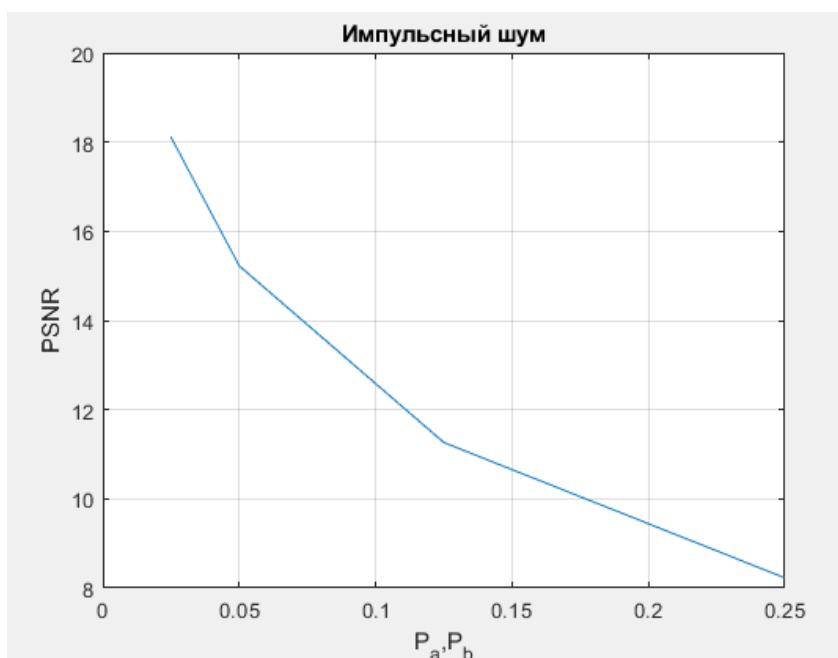


Рис. 4. График $PSNR(pa, pb)$ при $pa = pb = 0.025, 0.05, 0.125, 0.25$.

По данным графикам видно какие искажения вносят импульсный и аддитивные шумы. Импульсный шум даже при малых параметрах сильно ухудшает качество картинки, т.к. максимальное значение PSNR чуть больше 18, а, чтобы картинку можно было восстановилась, PSNR должен быть не меньше 35.

4. Обработка изображений с гауссовским шумом

В процессе выполнения фильтрации входного (зашумленного) изображения $I_{[H \times W]}$ формируется новое изображение $I'_{[H \times W]}$ с теми же размерами. Каждый пиксель $I'_{y,x}$ формируется в результате применения некоторого оператора к пикслю $I_{y,x}$ и подмножеству соседних с ним пикселей, образующих так называемую (апертуру) фильтра. Применяемый оператор может быть как линейным, так и нелинейным. При линейной фильтрации используется следующее правило для расчета значений от фильтрованных пикселей:

$$I'_{y,x} = \frac{1}{Z} \sum_{k=-R}^R \sum_{m=-R}^R w_{k,m} I_{y+k, x+m},$$

где R – радиус фильтра, определяющий апертуру, $w_{k,m}$ – весовые коэффициенты фильтра, Z – коэффициент нормировки, определяемый по формуле:

$$Z = \sum_{k=-R}^R \sum_{m=-R}^R w_{k,m},$$

Следует отметить, что использование термина «радиус» является общепринятым, хотя в реальности апертура имеет форму квадрата со стороной $2R+1$. На краях массива I , где апертура выходит за границы изображения, недостающее значение, как правило, заменяется интенсивностью ближайшего по Евклидову расстоянию пикселя.

4.1. Реализация метода скользящего среднего

В методе скользящего среднего значения весовых коэффициентов постоянны и не зависят от расстояния до центрального пикселя. Все весовые коэффициенты равны единице, в связи с этим выход фильтра рассчитывается следующей формуле:

$$I'_{y,x} = \frac{1}{(2R + 1)} \sum_{k=-R}^R \sum_{m=-R}^R I_{y+k,x+m}$$

Результаты применения фильтра:



Рис. 5. Изображение с шумом при $\sigma = 50$.



Рис. 6. Отфильтрованное изображение при размере окна $R = 1$.

Как видно из приведенных рисунков – отфильтрованное изображение получилось немного размытым, но шум удалось «сгладить». Значение PSNR увеличилось с 14.7886 до 23.486.

4.2. Подборка оптимального размера окна R

Определим значение R для $\sigma = 1, 10, 30, 50$ и 80 такое, что значение PSNR будет максимальным.

Результаты работы программы:

sigma = 1	
R	PSNR
1	32.6276
2	29.2675
3	27.8106

4	26.8641
5	26.1449

sigma = 10	
R	PSNR
1	31.4325
2	29.0381
3	27.7258
4	26.8245
5	26.1241

sigma = 30	
R	PSNR
1	26.8865
2	27.5895
3	27.0948
4	26.4797
5	25.9071

sigma = 50	
R	PSNR
1	23.486
2	25.5878
3	25.8957
4	25.6761
5	25.3228

sigma = 80	
R	PSNR
1	20.1039
2	22.5968
3	23.3479
4	23.4961
5	23.4202

4.3. Реализация метода Гауссовой фильтрации

Поскольку на фотoreалистичном изображении соседние пиксели сильно коррелируют, значения весовых коэффициентов, как правило,

рассчитываются на базе некоторой функции от расстояния до центральной позиции $I_{i,j}$, значения которой убывают по мере удаления позиции коэффициента от центральной. Однако такой подход имеет негативный эффект с точки зрения визуального восприятия. В окрестности контуров (резких перепадов) применение линейных фильтров приводит к «размытию» контура. Поэтому данный класс фильтров также называют фильтрами размытия. Функции расстояния для расчета значений весовых коэффициентов формируется на базе функции Гаусса от двух переменных (отсюда и название фильтра):

$$w_{k,m} = \exp\left(\frac{-(k^2 + m^2)}{2\sigma^2}\right),$$

Где σ – параметр фильтра, определяющий скорость убывания коэффициентов $w_{k,m}$ по мере удаления от позиции центрального пикселя.

Результат применения фильтра:



Рис. 7. Изображение с шумом при $\sigma = 50$.



Рис. 8. Отфильтрованное изображение при размере окна $R = 1$ и $\sigma = 1$.

После фильтрации так же присутствует эффект размытия. Значения PSNR удалось увеличить с 14.7886 до 23.1862 за счет «сглаживания» шума.

4.4. Построение графиков $PSNR(\sigma)$

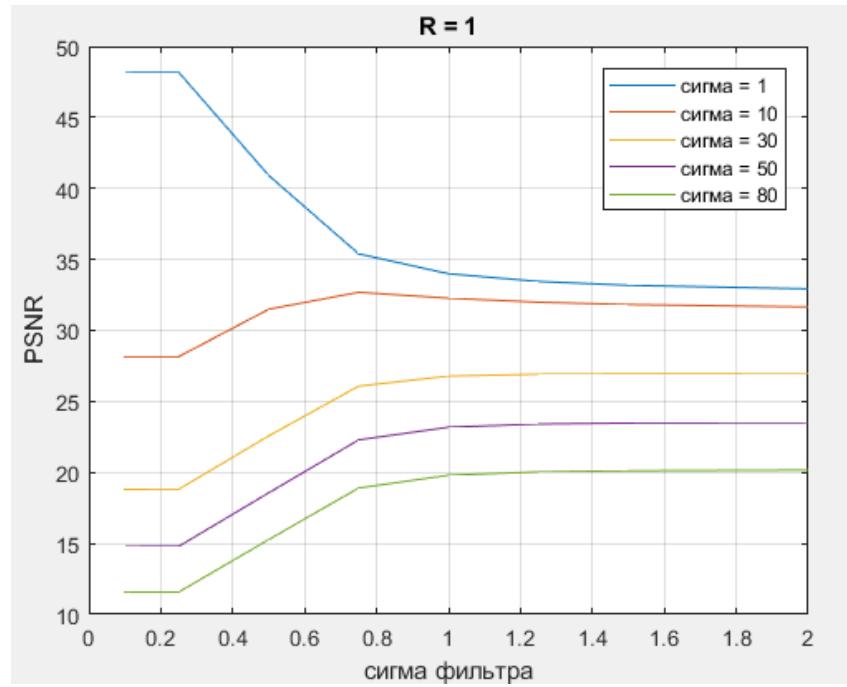


Рис. 9. График $PSNR(\sigma)$ при $R = 1$.

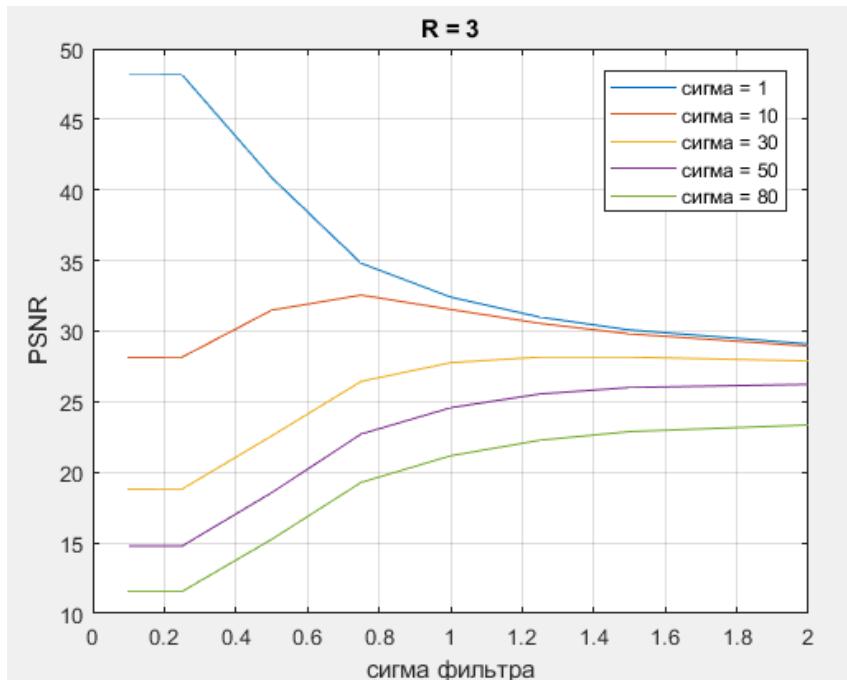


Рис. 10. График $PSNR(\sigma)$ при $R = 3$.

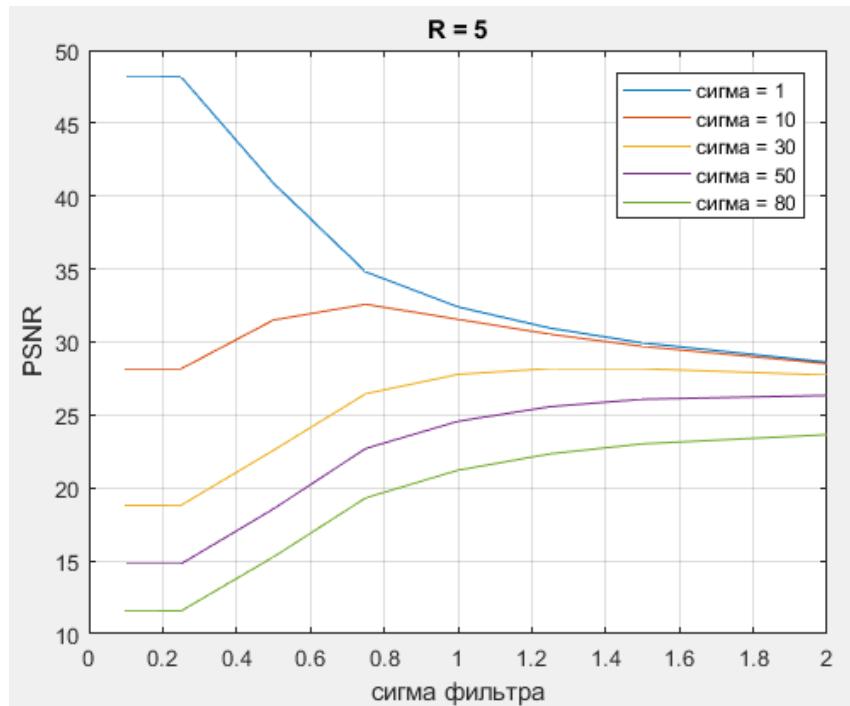


Рис. 11. График $PSNR(\sigma)$ при $R = 5$.

Из полученных значениях σ (параметр фильтра) > 1.4 значения PSNR при всех рассматриваемых шумах и значениях R практически перестают меняться.

4.5. Подбор оптимальных параметров

Параметр шума: $sigma = 1$		
$sigma$ фильтра	R	PSNR
0.25	1	48.1461
Параметр шума: $sigma = 10$		
$sigma$ фильтра	R	PSNR
0.75	1	32.5541
Параметр шума: $sigma = 30$		
$sigma$ фильтра	R	PSNR
1.25	3	28.1624

Параметр шума: sigma = 50		
sigma фильтра	R	PSNR
1.25	5	26.2313
Параметр шума: sigma = 80		
sigma фильтра	R	PSNR
2	5	23.3570

Как видно из полученных значений – чем больше значение шума, тем больше значение σ , при котором PSNR максимальна.

4.7. Метод медианной фильтрации

Медианный фильтр является фильтром, основанным на порядковых статистиках – частном случае нелинейной фильтрации.

Алгоритм медианной фильтрации для пикселя $I_{i,j}$ можно представить в виде следующих шагов:

- Формирование одномерного массива A из апертуры пикселя $I_{i,j}$;
- Взятие медианы одномерного массива A .

Взятие медианы по массиву A происходит следующим образом:

- Производится сортировка элементов одномерного массива A (неважно по возрастанию или по убыванию);
- Сохранить в $I'_{i,j}$ элемент, находящийся в середине отсортированного массива A . Для апертуры радиуса R необходимо взять из отсортированного массива элемент с индексом $\left(\left\lfloor \frac{(2R+1)^2}{2} \right\rfloor + 1\right)$.



Рис. 12. Изображение с шумом при $\sigma = 50$.



Рис. 13. Отфильтрованное изображение с $R = 5$.

На выходе фильтра получено сильно размытое изображение. Шум на изображении удалось сгладить, контуры размыты. Значение PSNR увеличилось с 14.7886 до 25.543. Визуально отфильтрованное изображение воспринимается сложнее, чем зашумленное.

4.8. Подборка оптимального R

Определим значение R для $\sigma = 1, 10, 30, 50$ и 80 такое, что значение PSNR будет максимальным:

sigma = 1	
R	PSNR
1	34.110
3	29.211
5	27.431

sigma = 10	
R	PSNR
1	31.532
3	28.812
5	27.227

sigma = 30	
R	PSNR
1	25.461
3	27.185
5	26.332

sigma = 50	
R	PSNR
1	21.592
3	25.519
5	25.540

sigma = 80	
R	PSNR
1	17.926
3	23.231
5	24.335

При зашумленности $\sigma < 30$ PSNR принимает максимальное значение при $R = 1$. В случае большой зашумленности, когда $\sigma = 80$, наибольшее значение достигается при $R = 5$. Отсюда можно сделать вывод, что чем сильнее шум на изображении, тем больший размер окна следует брать.

4.9. Анализ PSNR после применения различных фильтров

Сформируем изображения с различными аддитивными шумами при $\sigma = 1, 10, 30, 50, 80$. Проведем фильтрацию этого изображения различными фильтрами и проанализируем максимальные значение PSNR, полученные в результате работы каждого фильтра.

Для параметра шума $\sigma = 1$



Рис. 14. Изображение с шумом. PSNR = 48.1458



Рис. 15. Изображение после фильтрации скользящего среднего. PSNR = 32.627



Рис. 16. Изображение после фильтрации Гаусса. PSNR = 48.1409



Рис. 17. Изображение после медианной фильтрации. PSNR = 34.108

Для параметра шума $\sigma = 10$



Рис. 18. Изображение с шумом. PSNR = 28.1405



Рис. 19. Изображение после фильтрации скользящего среднего. PSNR = 31.426



Рис. 20. Изображение после фильтрации Гаусса. PSNR = 32.662



Рис. 21. Изображение после медианной фильтрации. PSNR = 31.5564

Для параметра шума $\sigma = 30$



Рис. 22. Изображение с шумом. PSNR = 18.7972



Рис. 23. Изображение после фильтрации скользящего среднего. PSNR = 27.5889



Рис. 24. Изображение после фильтрации Гаусса. PSNR = 28.0655



Рис. 25. Изображение после медианной фильтрации. PSNR = 27.214

Для параметра шума $\sigma = 50$



Рис. 26. Изображение с шумом. PSNR = 14.782



Рис. 27. Изображение после фильтрации скользящего среднего. PSNR = 25.852



Рис. 28. Изображение после фильтрации Гаусса. PSNR = 25.5196



Рис. 29. Изображение после медианной фильтрации. PSNR = 25.51434

Для параметра шума $\sigma = 80$



Рис. 30. Изображение с шумом. PSNR = 11.5525



Рис. 31. Изображение после фильтрации скользящего среднего. PSNR = 23.5434



Рис. 32. Изображение после фильтрации Гаусса. PSNR = 23.339



Рис. 33. Изображение после медианной фильтрации. PSNR = 23.1946

Визуально можно заметить, что метод Гаусса и скользящего среднего эффективнее улучшает изображение. Метод медианной фильтрации даже при слабом шуме заметно размывает изображение, соответственно, у метода медианной фильтрации практически отсутствует зернистость, как у фильтра Гаусса и скользящего среднего.

4.10. Анализ PSNR после применения различных фильтров

Построим графики зависимости $PSNR(\sigma)$ для $PSNR$ между исходным и зашумленным изображениями, а также максимальных $PSNR$ между исходным и отфильтрованными изображениями для каждого фильтра;

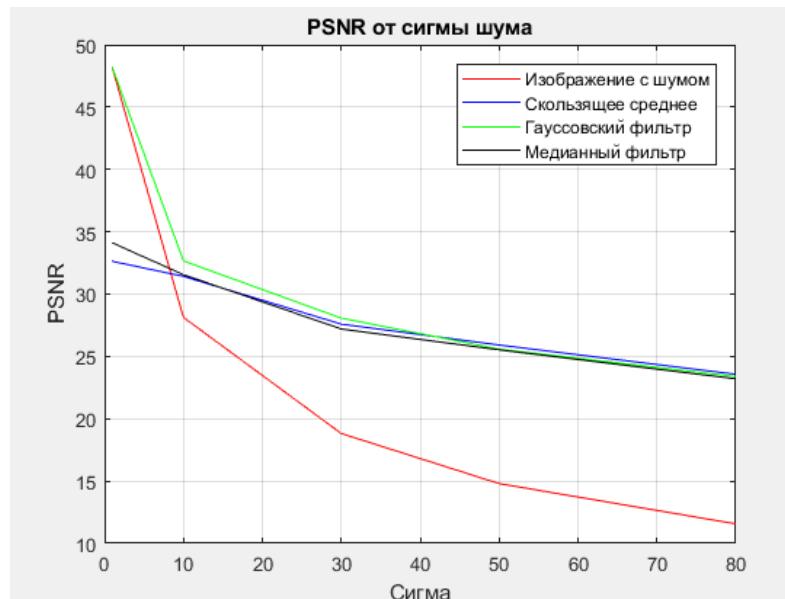


Рис. 34. Графики зависимости PSNR.

По графикам видно, что для слабозашумлённых изображений ($\sigma = 1, \sigma = 10, \sigma = 30$) лучшей фильтрацией является фильтр Гаусса. Для $\sigma > 30$ все фильтры дают примерно одинаковый результат, однако визуально для медианного фильтра изображение воспринимается гораздо хуже.

Обработка изображений с импульсным шумом.

5.1 Наложение шума.

Наложим на изображение импульсные шума так, чтобы доли искаженных пикселей составляли: 5%, 10%, 25% и 50%.



Рис. 35. Изображение с шумом при $ra = pb = 0.025$.



Рис. 36. Изображение с шумом при $ra = pb = 0.05$.



Рис. 37. Изображение с шумом при $ra = pb = 0.125$.



Рис. 38. Изображение с шумом при $ra = pb = 0.25$.

5.2. Вычисление PSNR по зашумленным изображениям

Определим значения PSNR для изображений, сформированных в предыдущем пункте.

Таблица. Результаты вычислений

	PSNR
5%	18.1291
10%	15.2408
25%	11.2541
50%	8.2455

5.3. Применение медианной фильтрации



Рис. 39. Отфильтрованное изображение с шумом при $ra = pb = 0.025$ и $R = 1$.



Рис. 40. Изображение с шумом при $ra = pb = 0.05$ и $R = 1$.



Рис. 41. Изображение с шумом при $ra = pb = 0.125$ и $R = 2$.



Рис. 42. Изображение с шумом при $ra = pb = 0.25$ и $R = 3$.

5.4. Анализ PSNR разных вариантов фильтрации

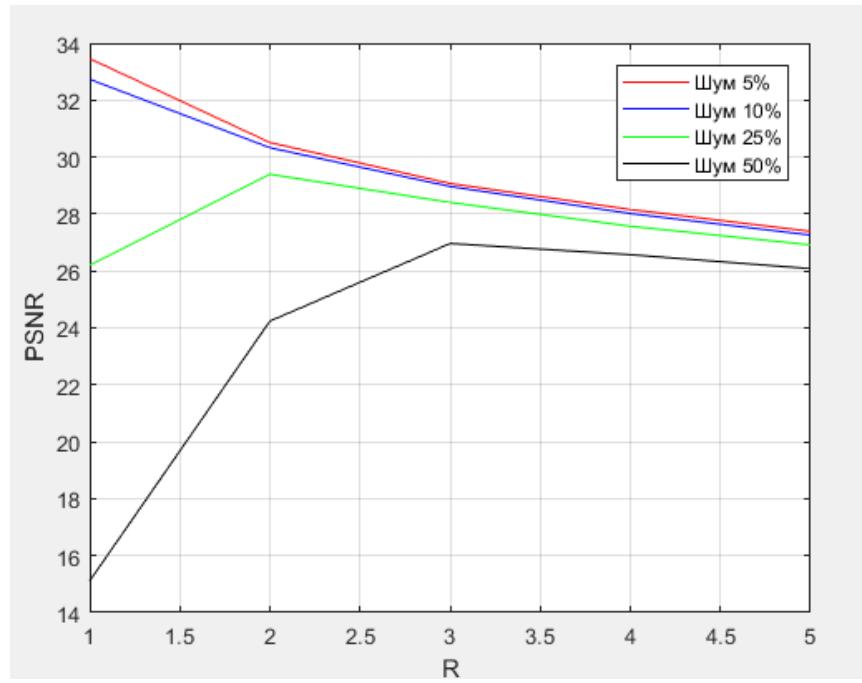


Рис. 43. График зависимости PSNR(R).

Из полученных графиков видно, что при низкой зашумленности (5% и 10% деформированных пикселей) наибольшие значения PSNR достигаются при размере окна $R = 1$. При сильной зашумленности наибольшие значения PSNR достигаются при размере окон R равных 2 и 3 соответственно.

6. Реализация методов выделения контура

6.1. – 6.2. Применение оператора Лапласа и формирование изображения по отклику

Необходимо применить оператор Лапласа $L(I)$ к изображению I и получить отклик I' , воспользовавшись формулой (4.1), где в качестве весовых коэффициентов $w_{i,j}$ будут использоваться элементы масок:

0	1	0
1	-4	1
0	1	0

0	-1	0
-1	4	-1
0	-1	0

Результаты работы:

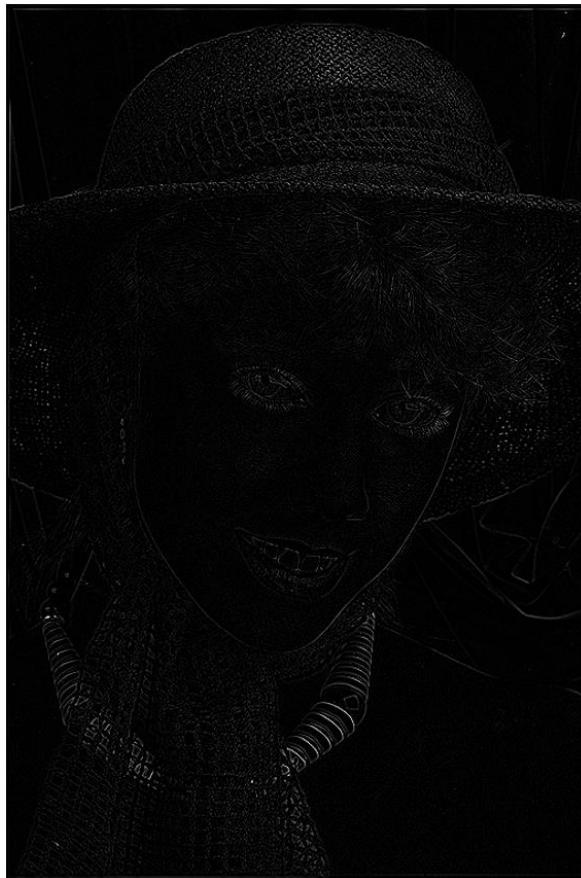


Рис. 44. Отклик после первой маски.

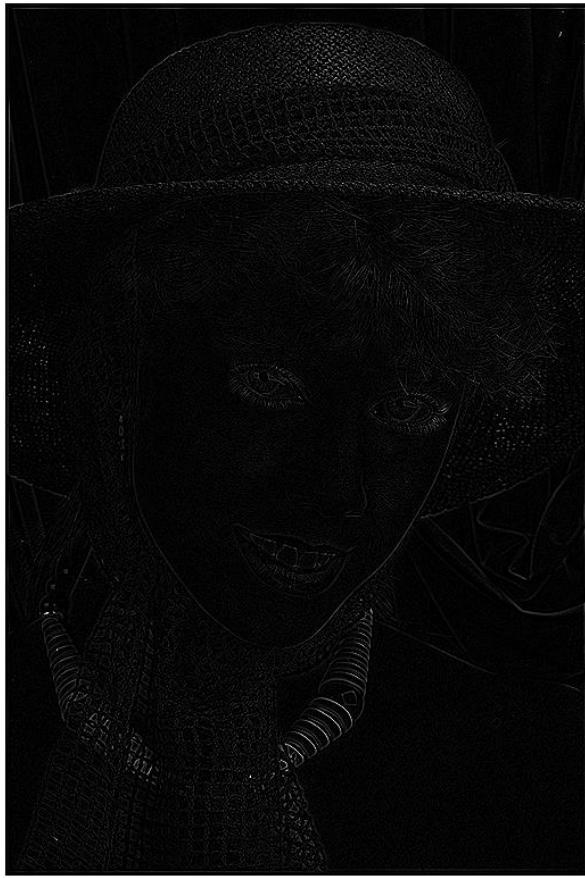


Рис. 45. Отклик после второй маски.

По полученному отклику оператора Лапласа I' необходимо сформировать изображение. Значение пикселя на позиции (x, y) вычисляется по следующей формуле:

$$Clip(I'_{x,y} + 128, 0, 255)$$

Полученный результат:

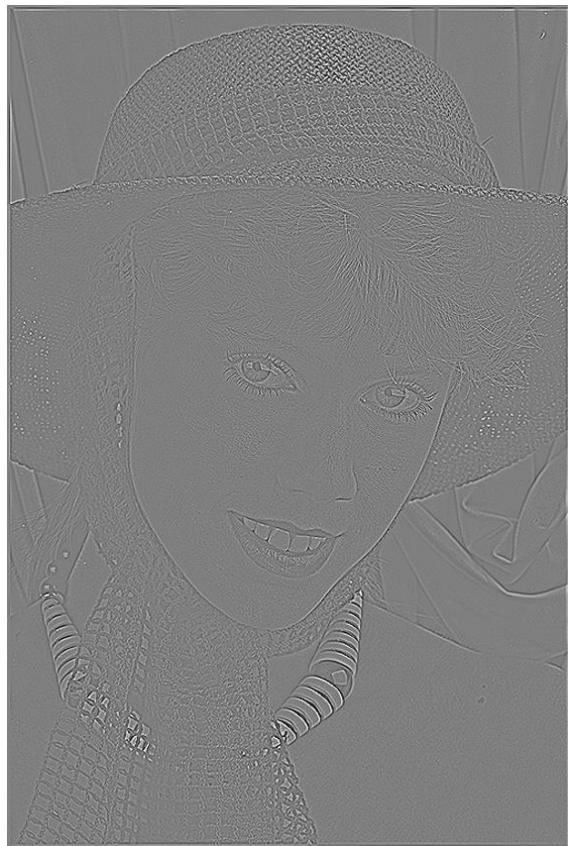


Рис. 46. Изображение после первой маски.

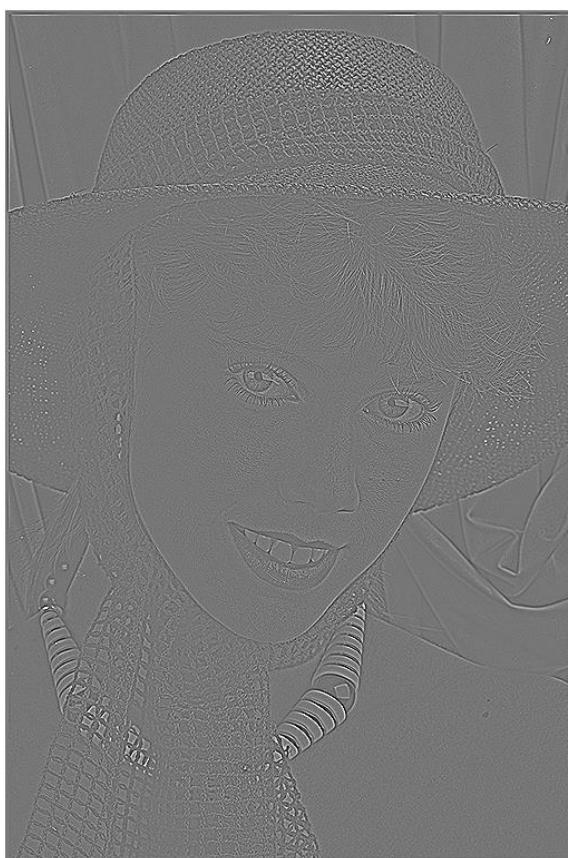


Рис. 47. Изображение после второй маски.

По полученным изображениям можно увидеть разницу в применении масок. Например, после использования второй маски у решеток на окнах пропадает дополнительный контур.

6.3. Синтез изображения с усилением высоких частот

Необходимо синтезировать изображение с усилением высоких частот, используя разность исходного изображения I и отклика оператора Лапласа I' , применив маску.

0	-1	0
-1	$\alpha+4$	-1
0	-1	0

Значение пикселя на позиции (x, y) синтезированного изображения вычисляется по следующей формуле:

$$\text{Clip}(I'_{x,y} + I_{x,y}, 0, 255)$$

Полученный результат:



Рис. 48. Синтезированное изображение с усилением высоких частот.



Рис. 49. Синтезированное изображение после клиппирования.

Изображение после клиппирования оказалось идентично изображению до, следовательно, значения пикселей не выходили за границы.

6.4.-6.5. Синтез изображений с различными значениями параметра α

Необходимо синтезировать изображения, применив маску для разных значений параметра α . Параметр α изменяется в интервале от 1 до 1.5 с шагом 0.1.



Рис. 50. Синтезированное изображение при $\alpha = 1$.



Рис. 51. Синтезированное изображение при $\alpha = 1.1$.



Рис. 52. Синтезированное изображение при $\alpha = 1.2$.



Рис. 53. Синтезированное изображение при $\alpha = 1.3$.



Рис. 54. Синтезированное изображение при $\alpha = 1.4$.



Рис. 55. Синтезированное изображение при $\alpha = 1.5$.

По полученным изображениям можно сделать вывод, что, при увеличении α общая яркость изображения увеличивается, а резкость – уменьшается.

Синтезированное изображение совпадает с I'' при $\alpha = 1$. Так как именно при этом значении достигается максимальный эффект увеличения резкости. При увеличении значения α растет яркость изображения.

6.6. Средние значения яркости

Необходимо рассчитать среднее значение яркости для каждого из изображений, полученных в предыдущем пункте.

Таблица. Результаты вычислений

α	средняя яркость
1	88.3045
1.1	105.4839
1.2	122.2671
1.3	138.2026
1.4	152.6773
1.5	165.3547

6.7. Построение и анализ гистограмм

Необходимо построить и проанализировать гистограммы для исходного изображения, а также изображений, полученных в результате выполнения пункта 6.4.

Результаты построения:

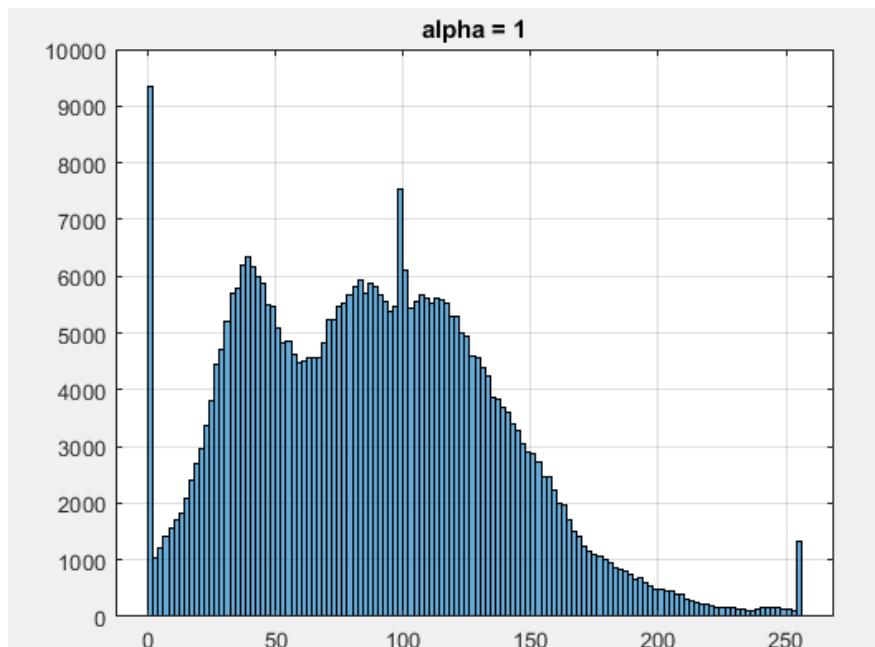


Рис. 56. Гистограмма исходного изображения.

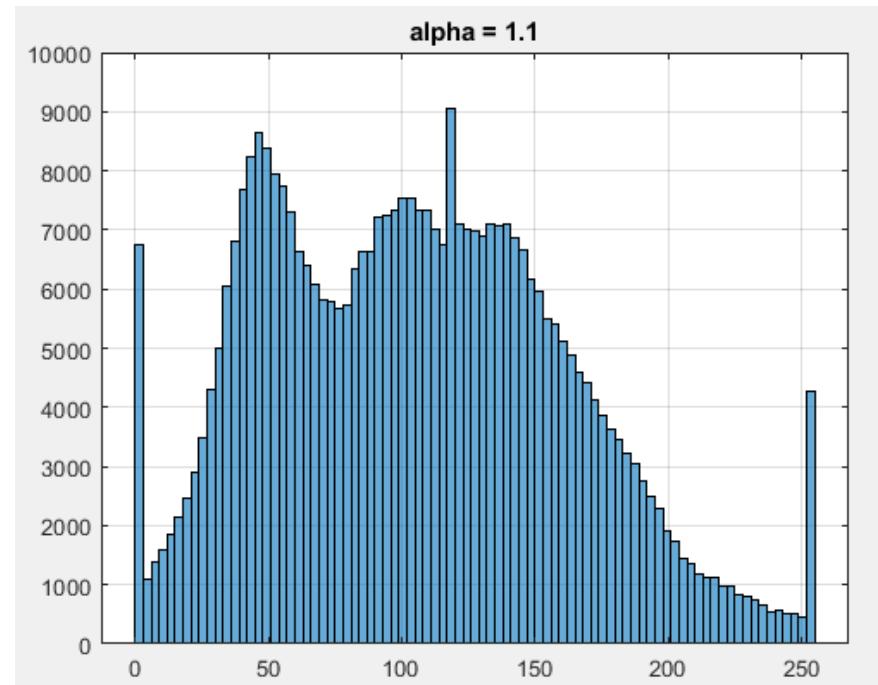


Рис. 57 Гистограмма изображения при $\alpha = 1.1$.

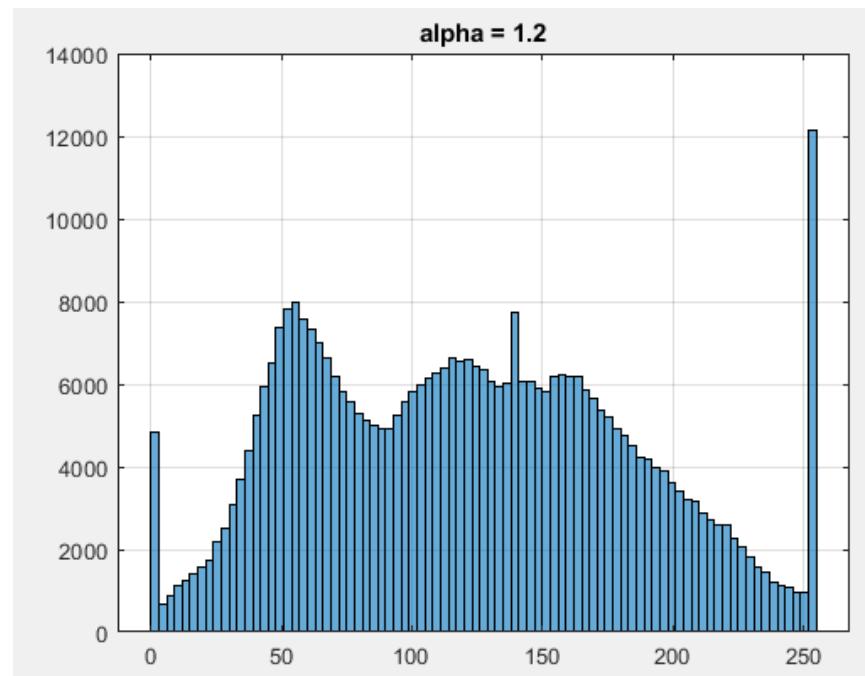


Рис. 58. Гистограмма изображения при $\alpha = 1.2$.

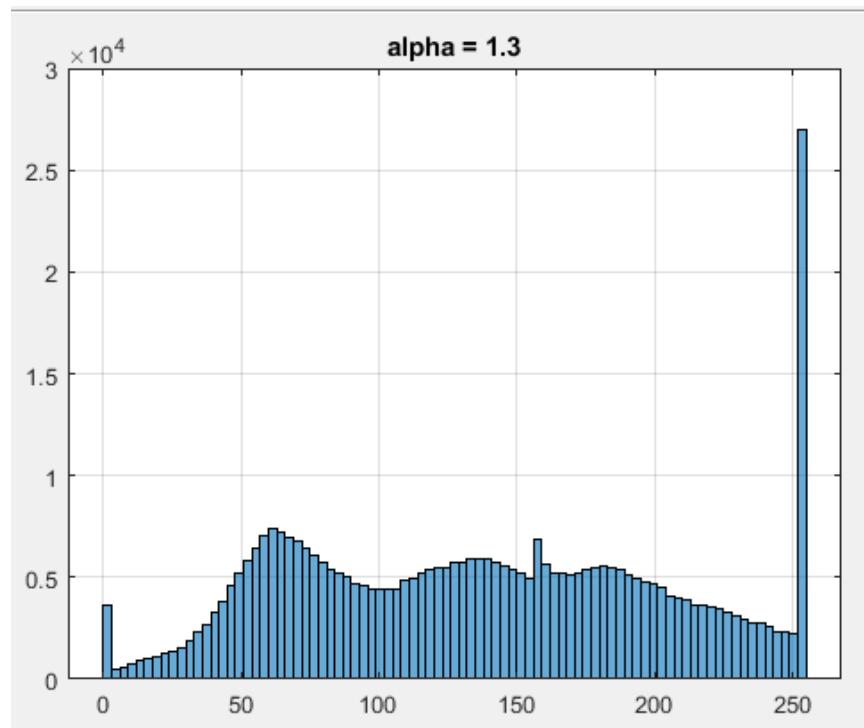


Рис. 59. Гистограмма изображения при $\alpha = 1.3$.

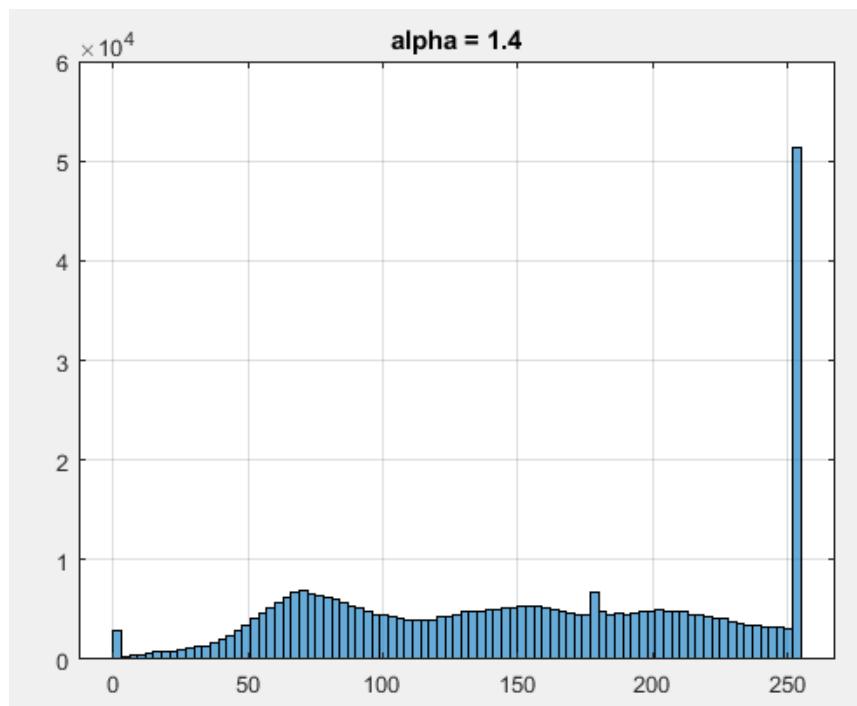


Рис. 60. Гистограмма изображения при $\alpha = 1.4$.

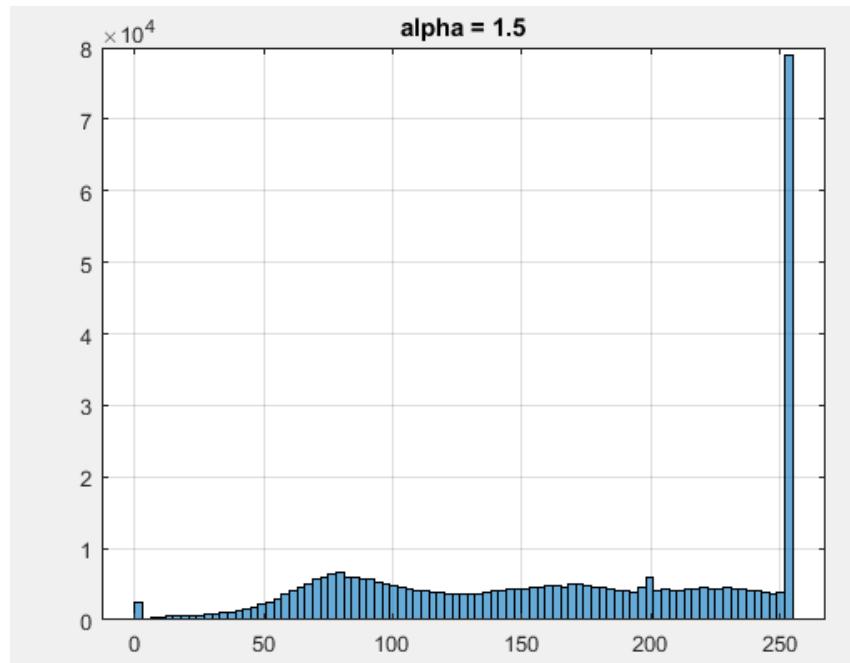


Рис. 61. Гистограмма изображения при $\alpha = 1.5$.

По полученным гистограммам отчетливо можно видеть, что, при увеличении α , уменьшается общее количество пикселей с интенсивностью компоненты < 255 и увеличивается крайний правый пик – количество пикселей с интенсивностью компоненты равной 255. Это связано с постепенным осветлением изображения – все больше компонент пикселей принимают свое максимальное значение.

6.8.-6.9. Реализация оператора Собеля

Необходимо применить оператор Собеля к исходному изображению I.

Оператор Собеля является ключевым во многих алгоритмах анализа контуров изображения. В основе оператора Собеля лежит расчет двух производных: по вертикали и по горизонтали. Соответствующие этим операциям маски фильтров приведены на рис. 62.

а)

-1	0	1
-2	0	2
-1	0	1

б)

1	2	1
0	0	0
-1	-2	-1

Рис. 62. Маски фильтров, используемые в операторе Собеля: а) – по горизонтали, б) – по вертикали.

Алгоритм применения оператора:

- Рассчитать значения откликов $G_{x,y}^h$ и $G_{x,y}^v$, используя фильтры с масками, показанными на рис. 56.
- Рассчитать величину силы (длины) контура $|\nabla I_{x,y}|$ по формуле

$$|\nabla I_{x,y}| = \sqrt{{G_{x,y}^h}^2 + {G_{x,y}^v}^2}$$

1. Рассчитать направление градиента $\theta_{x,y}$ по формуле

$$\theta_{x,y} = \arctg \frac{G_{x,y}^v}{G_{x,y}^h}$$

Результат:



Рис. 63. Результат применения оператора Собеля.

6.10-6.11 Формирование изображений с различными порогами thr.



Рис. 64. Результат применения оператора Собеля при $thr = 20$.



Рис. 65. Результат применения оператора Собеля при $thr = 40$.



Рис. 66. Результат применения оператора Собеля при $thr = 60$.



Рис. 67. Результат применения оператора Собеля при $thr = 90$.



Рис. 68. Результат применения оператора Собеля при $thr = 120$.

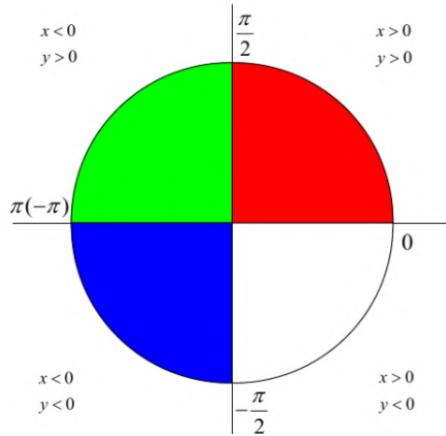


Рис. 69. Результат применения оператора Собеля при $thr = 160$.

По сформированным изображениям можно сделать вывод, что наиболее точной бинарной картой является карта с порогом $thr = 60$. В случае, когда $thr < 40$ изображения перегружены черными пикселями. При пороге > 90 некоторые контуры становятся чётче различимы, а некоторые исчезают вовсе.

6.12 Карта направлений градиентов

Необходимо построить 4-цветную карту направлений градиентов на полученных изображениях по такому правилу:



Результат



Рис. 70. Карта направлений градиентов.

7. Реализация градационных преобразований

Назовем множество пикселей, формирующих изображение, пространственной областью. Пространственные методы повышения качества изображений оперируют непосредственно значениями этих пикселей и могут описываться следующим выражением:

$$I' = T[I]$$

где I – входные данные, I' – обработанные данные и $T[*$] – оператор над I , определенный в некоторой окрестности точки с координатами (x, y) .

В простейшем случае функция отображения не зависит от местоположения обрабатываемого пикселя и неизменна для всего изображения. Градационное преобразование можно представить в виде “черного” ящика с отсутствующей памятью, который для конкретного входного значения формирует соответствующее выходное на основе заранее определенной функции.

7.1. Синтез засвеченного, затемненного и сбалансированного изображений



Рис. 71. Сбалансированное изображение.



Рис. 72. Синтезированное засвеченное изображение.



Рис. 73. Синтезированное затемненное изображение.

7.2. Преобразование методом на основе опорных точек

К полученным в предыдущем пункте изображениям нужно применить градационное преобразование на базе двух опорных точек для повышения их качества.

Метод опорных точек – один из простейших методов улучшения качества изображения на основе анализа и видоизменения гистограммы основывается на использовании метода опорных точек.

На основе точек $(0,0)$ и $(255,255)$ и двух опорных точек, которые задаёт человек, входное значение преобразуется с помощью линейной интерполяции в новое значение.



Рис. 74, 75. Сбалансированное изображение,
Сбалансированное изображение на основе опорных точек $(80, 110)$ и $(170, 200)$.



Рис. 76. Затемненное изображение.

Рис. 77. Затемненное изображение на основе опорных точек (90, 200) и (210, 190).



Рис. 78. Осветленное изображение.

Рис. 79. Осветленное изображение на основе опорных точек (70, 10) и (250, 150).

7.3. Формирование гистограмм

Необходимо сформировать и проанализировать гистограммы для исходных и полученных изображений из предыдущего пункта.

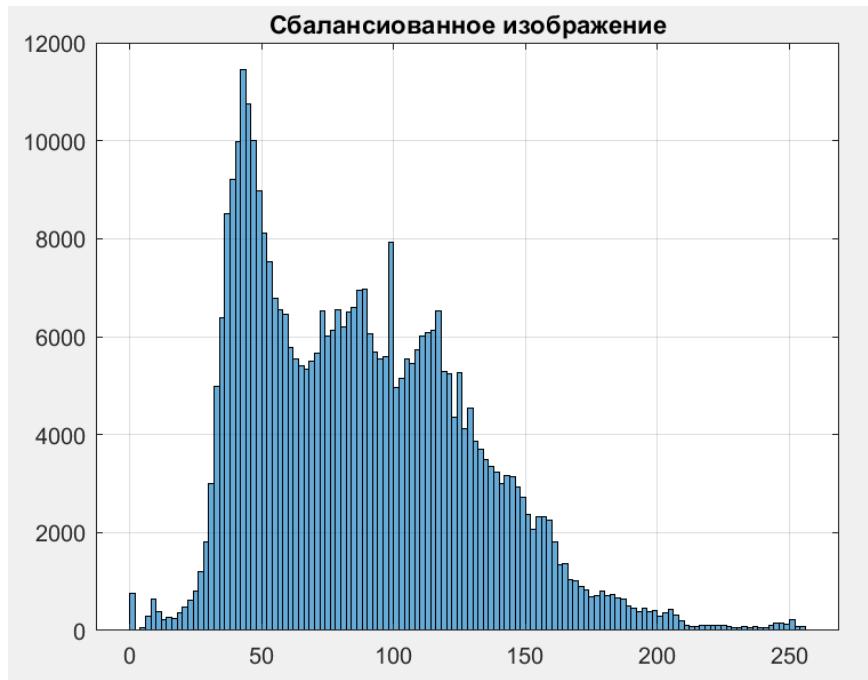


Рис. 80. Гистограмма сбалансированного изображения.



Рис. 81. Гистограмма сбалансированного изображения на опорных точках.

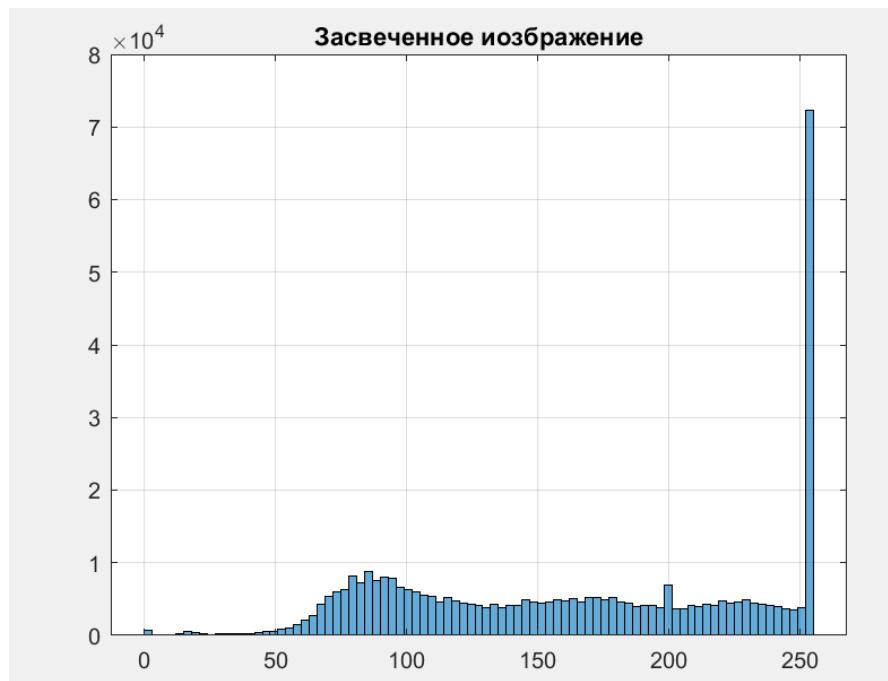


Рис. 82. Гистограмма засвеченного изображения.



Рис. 83. Гистограмма засвеченного изображения на опорных точках.

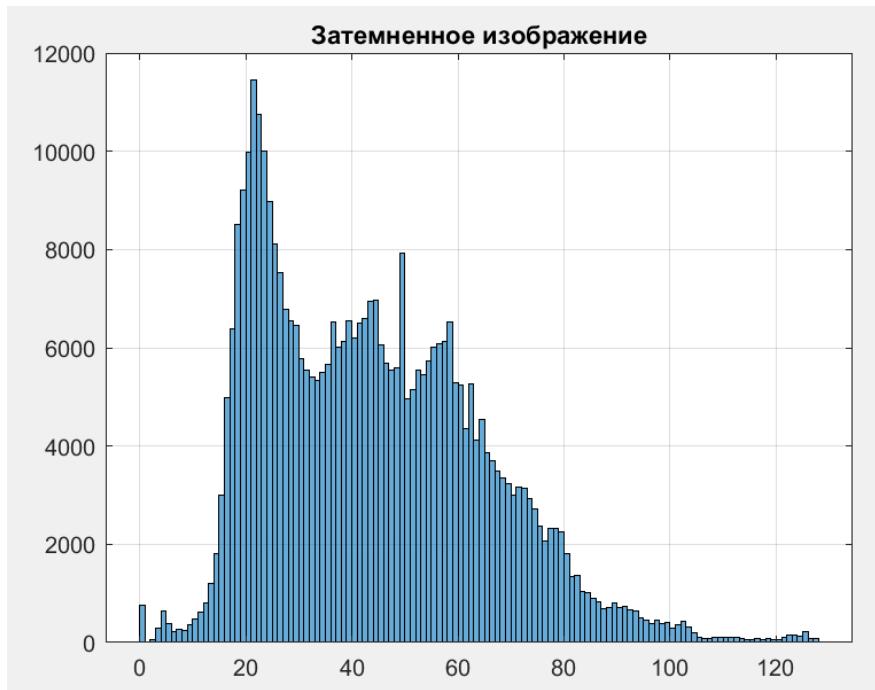


Рис. 84. Гистограмма затемненного изображения.



Рис. 85. Гистограмма затемненного изображения на опорных точках.

По полученным гистограммам можно увидеть, что преобразование практически не влияет на количество пикселей с крайними значениями компонент относительно других пикселей, но поднимает количество удаленных от крайних пикселей.

7.4.-7.5. Гамма преобразования

Необходимо получить графики уравнения $I'_{x,y} = c(I_{x,y})^\gamma$ для различных значений параметра γ для каждого из трех входных изображений.

Степенные преобразования выполняются по следующей формуле:

$$I'_{x,y} = c(I_{x,y})^\gamma$$

где c и γ являются положительными величинами.

При этом перед выполнением расчета по формуле входное значение $I_{y,x}$ следует привести в диапазон $[0,1]$. Выходное значение $I'_{y,x}$ возвращают в диапазон $[0, 255]$ путем умножения на 255.

Результат:



Рис. 86 – Сбалансированное изображение.



Рис. 87 – Сбалансированное изображение при $\gamma = 0,04$.



Рис. 88 - Сбалансированное изображение при $\gamma = 0,5$.



Рис. 89 - Сбалансированное изображение при $\gamma = 1$.



Рис. 90 - Сбалансированное изображение при $\gamma = 8$.



Рис. 91 - Сбалансированное изображение при $\gamma = 25$.



Рис. 92 – Затемненное изображение.



Рис. 93 - Затемненное изображение при $\gamma = 0,04$.



Рис. 94 - Затемненное изображение при $\gamma = 0,5$.



Рис. 95 - Затемненное изображение при $\gamma = 1$.

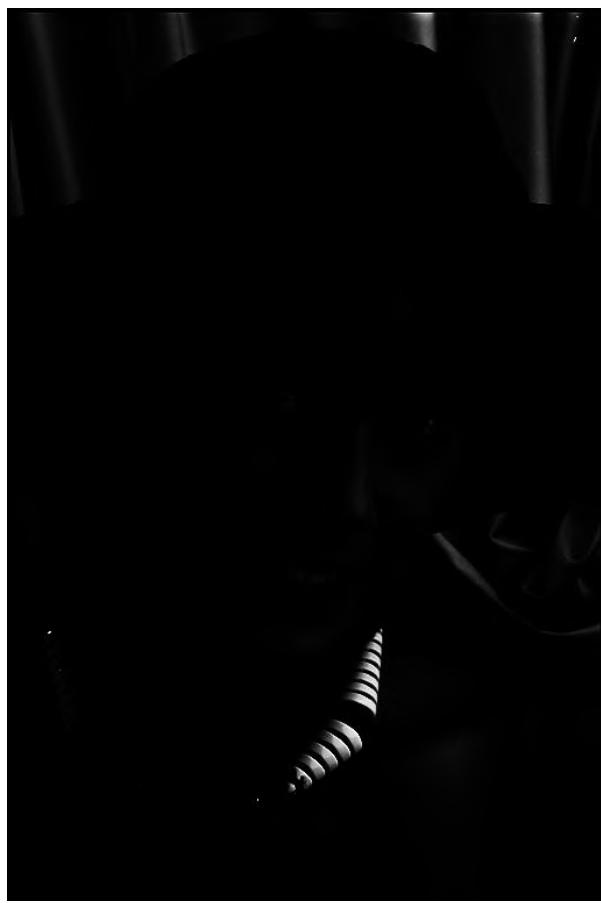


Рис. 96 - Затемненное изображение при $\gamma = 8$.

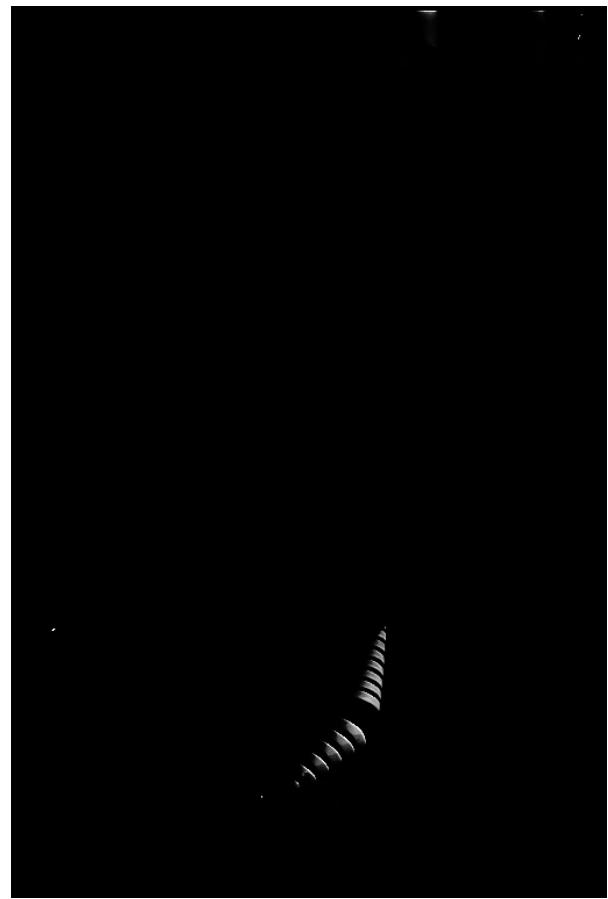


Рис. 97 - Затемненное изображение при $\gamma = 25$.



Рис. 98 - Осветленное изображение.



Рис. 99 – Осветленное изображение при $\gamma = 0,04$.



Рис. 100 – Осветленное изображение при $\gamma = 0,5$.



Рис. 101 – Осветленное изображение при $\gamma = 1$.



Рис. 102 – Осветленное изображение при $\gamma = 8$.

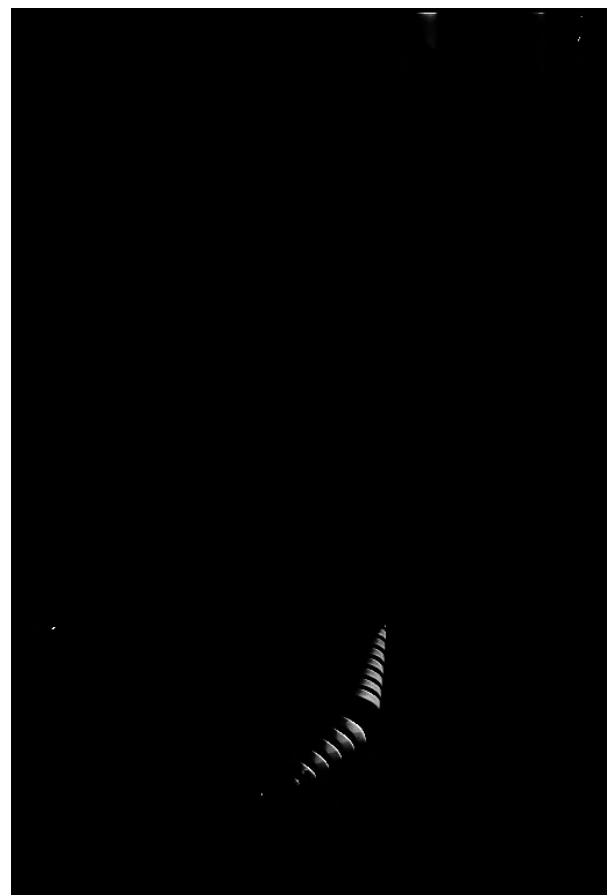


Рис. 103 – Осветленное изображение при $\gamma = 25$.

По полученным гистограммам можно сделать вывод, что при значениях $\gamma < 1$ яркость изображения увеличивается. А при $\gamma > 1$ наоборот уменьшается.

7.7. Формирование гистограмм

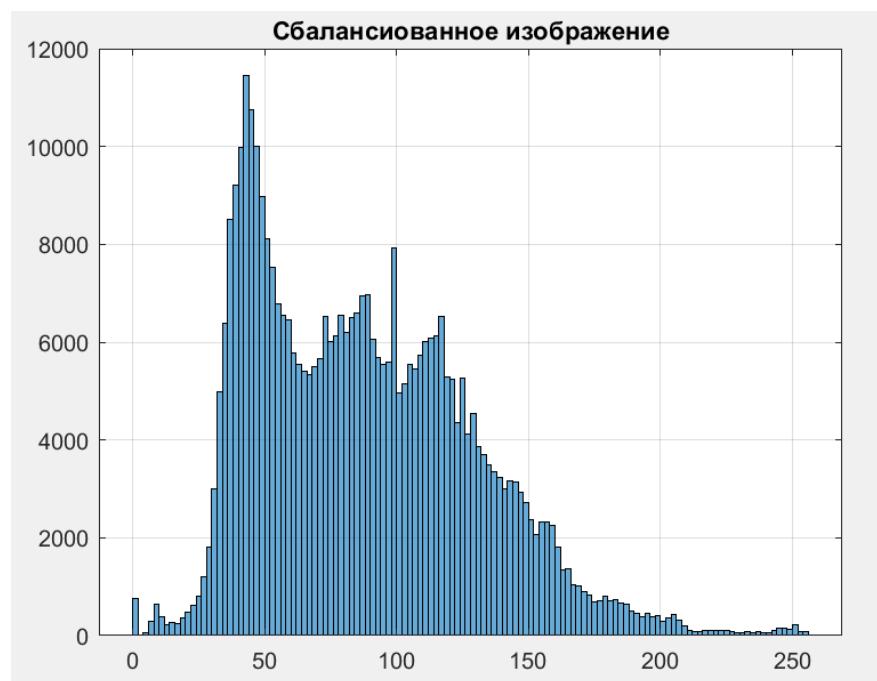


Рис. 104 – Гистограмма сбалансированного изображения.

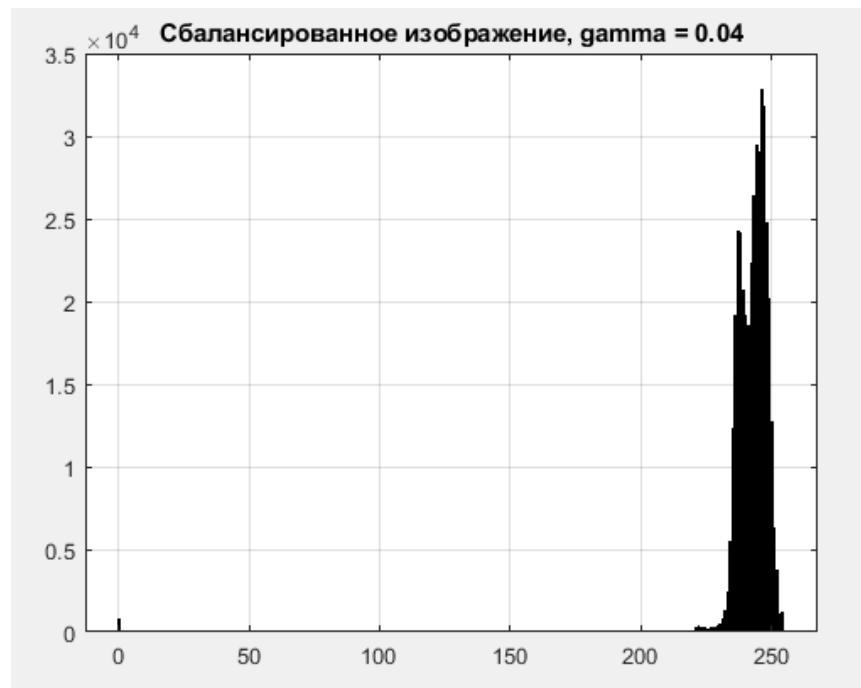


Рис. 105 – Гистограмма сбалансированного изображения при $\gamma = 0,04$.

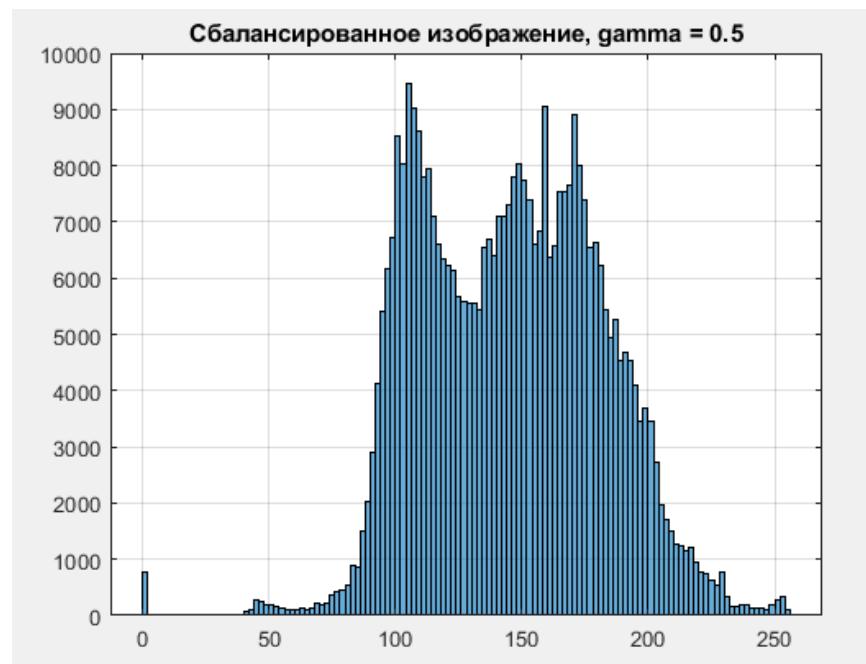


Рис. 106 - Гистограмма сбалансированного изображения при $\gamma = 0,5$.

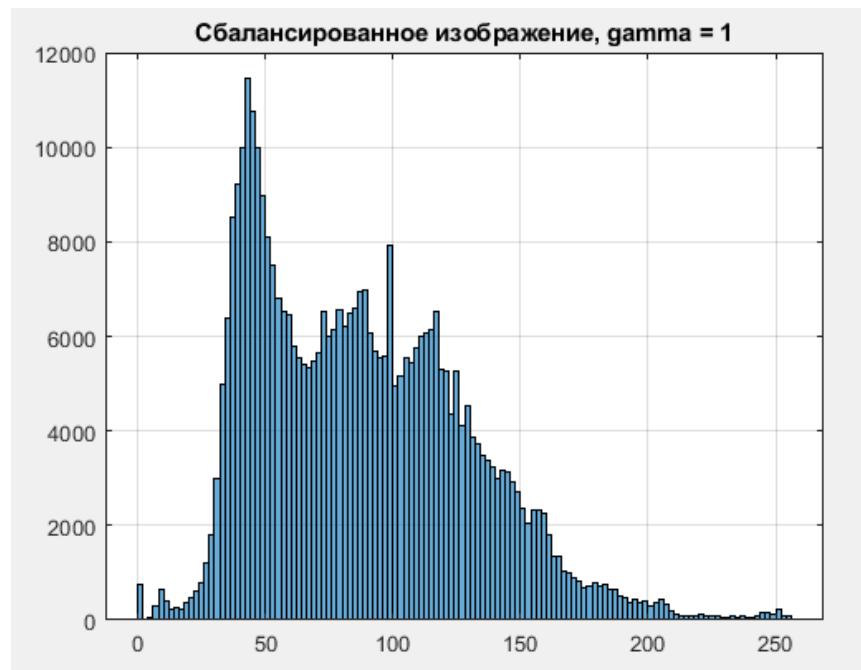


Рис. 107 - Гистограмма сбалансированного изображения при $\gamma = 1$.

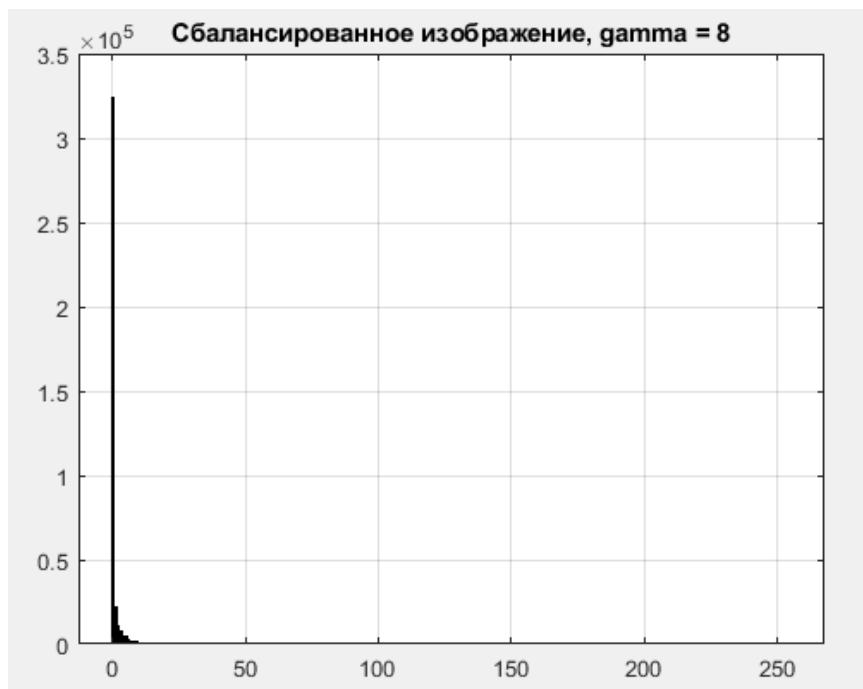


Рис. 108 - Гистограмма сбалансированного изображения при $\gamma = 8$.

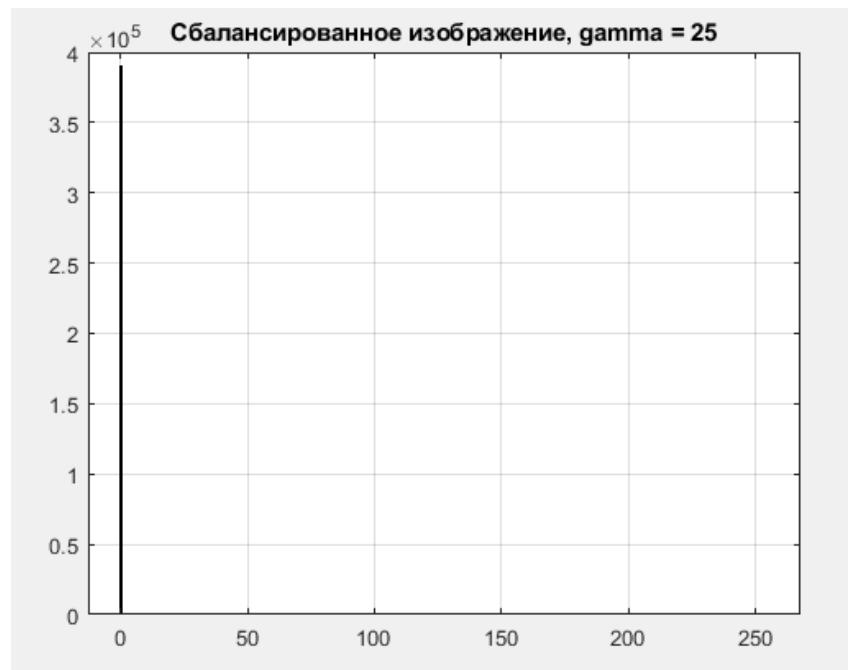


Рис. 109 - Гистограмма сбалансированного изображения при $\gamma = 25$.



Рис. 110 – Гистограмма затемненного изображения.

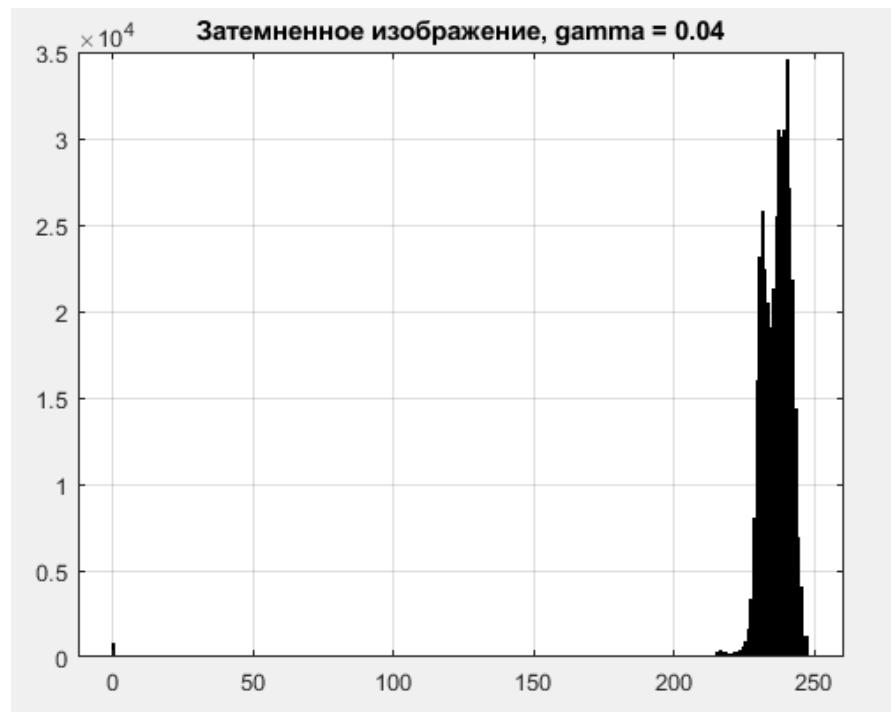


Рис. 111 - Гистограмма затемненного изображения при $\gamma = 0,04$.

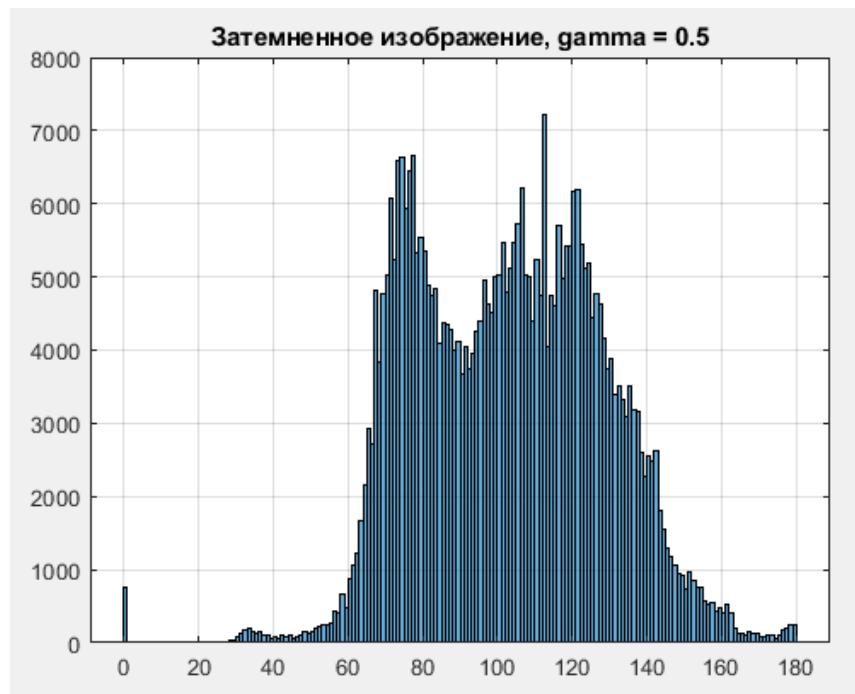


Рис. 112 - Гистограмма затемненного изображения при $\gamma = 0,5$.



Рис. 113 - Гистограмма затемненного изображения при $\gamma = 1$.

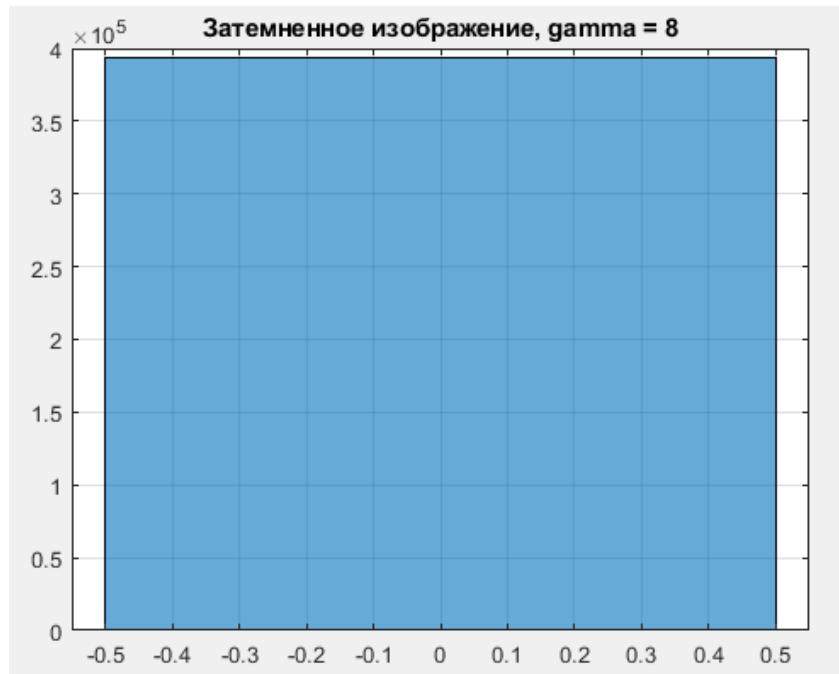


Рис. 114 - Гистограмма затемненного изображения при $\gamma = 8$.

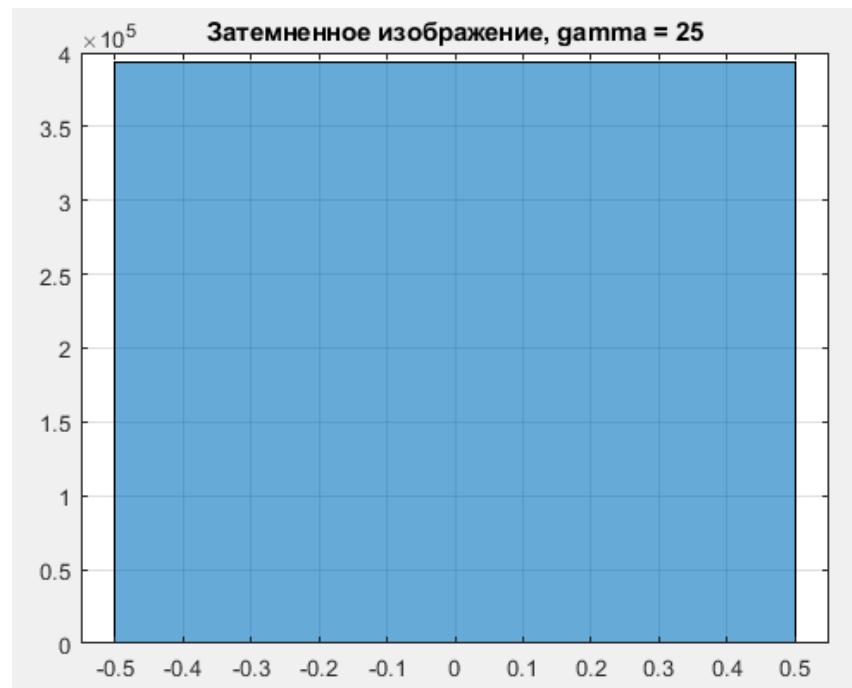


Рис. 115 - Гистограмма затемненного изображения при $\gamma = 25$.

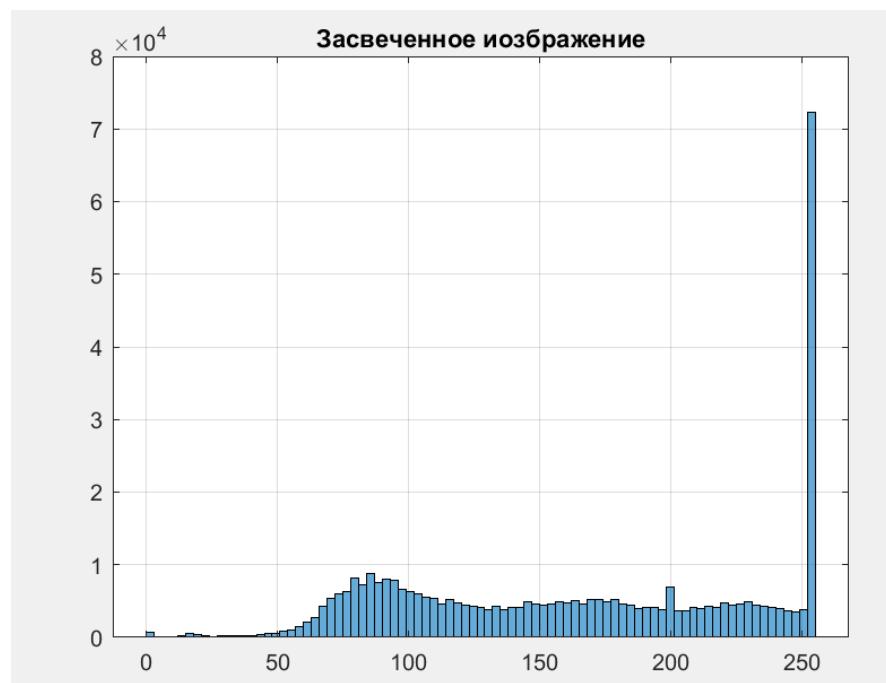


Рис. 116 - Гистограмма освещенного изображения.

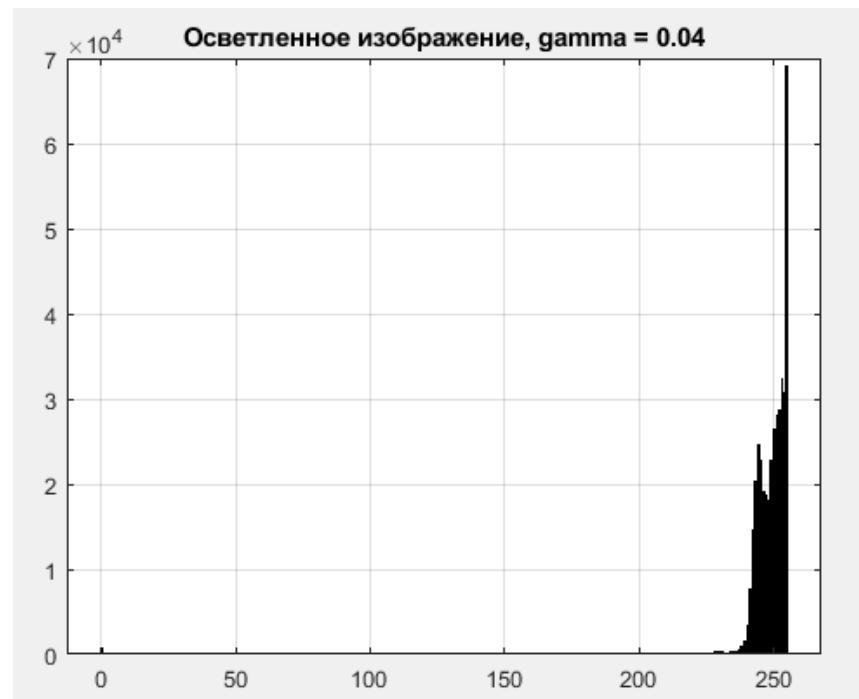


Рис. 117 – Гистограмма осветленного изображения при $\gamma = 0,04$.



Рис. 118 – Гистограмма осветленного изображения при $\gamma = 0,5$.

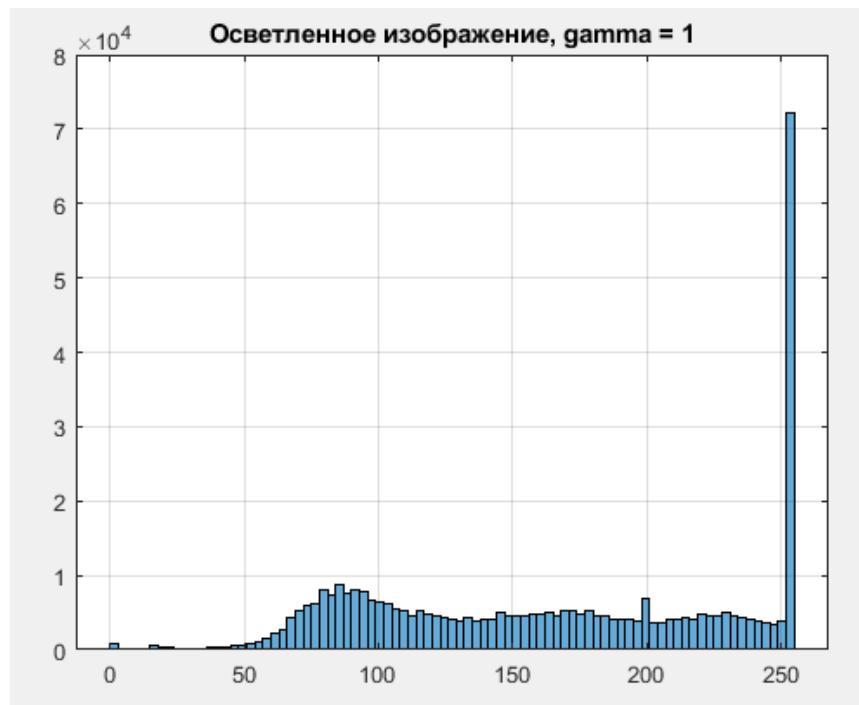


Рис. 119 – Гистограмма осветленного изображения при $\gamma = 1$.



Рис. 120 – Гистограмма осветленного изображения при $\gamma = 8$.



Рис. 121 – Гистограмма осветленного изображения при $\gamma = 25$.

По полученным данным можно убедиться, что вывод, сделанный в предыдущем пункте, верный.

7.8.-7.9. Методы выравнивания гистограмм

Выравнивание гистограммы может быть использовано в задаче улучшения качества изображений. Данный подход может увеличить контрастность обрабатываемого изображения.

Для выполнения операции выравнивания гистограмм следует воспользоваться формулой на базе эмпирической функции распределения:

$$s_k = \frac{2^L - 1}{N} \sum_{j=0}^k n_j$$

где $2^L - 1$ – максимально допустимое число градаций интенсивности на изображении, s_k – значение интенсивности в выходном изображении, в которое будет преобразовываться r_k , N – общее число пикселей и n_j – число пикселей, имеющих яркость r_k .

Необходимо синтезировать засвеченное, затемненное, а также сбалансированное изображения и применить к ним алгоритм выравнивания гистограмм в соответствии с формулой.



Рис. 122 – Исходное сбалансированное изображение. Рис. 123 – Обработанное сбалансированное изображение.



Рис. 124 – Исходное затемненное изображение. Рис. 125 – Обработанное затемненное изображение.



Рис. 126 – Исходное осветленное изображение. Рис. 127 – Обработанное осветленное изображение.

7.10. Формирование гистограмм



Рис. 128 – Гистограмма сбалансиированного изображения.

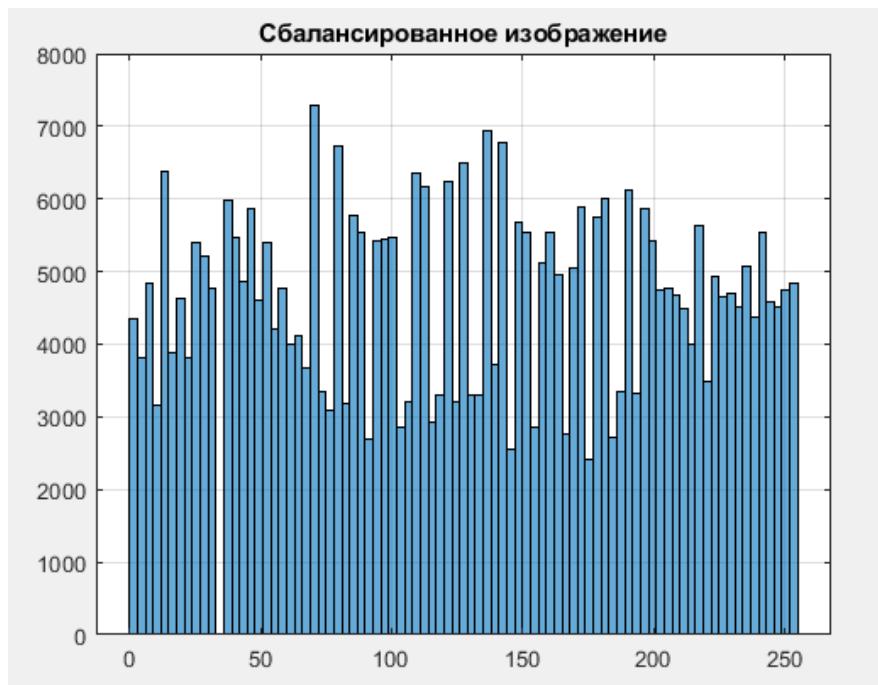


Рис. 129 – Гистограмма обработанного сбалансированного изображения.

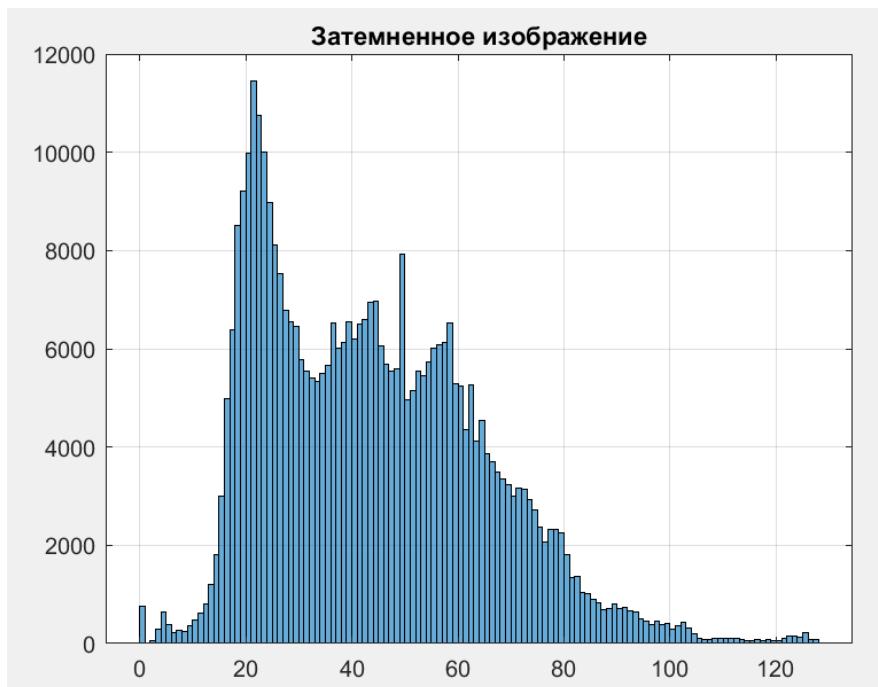


Рис. 130 – Гистограмма затемненного изображения.

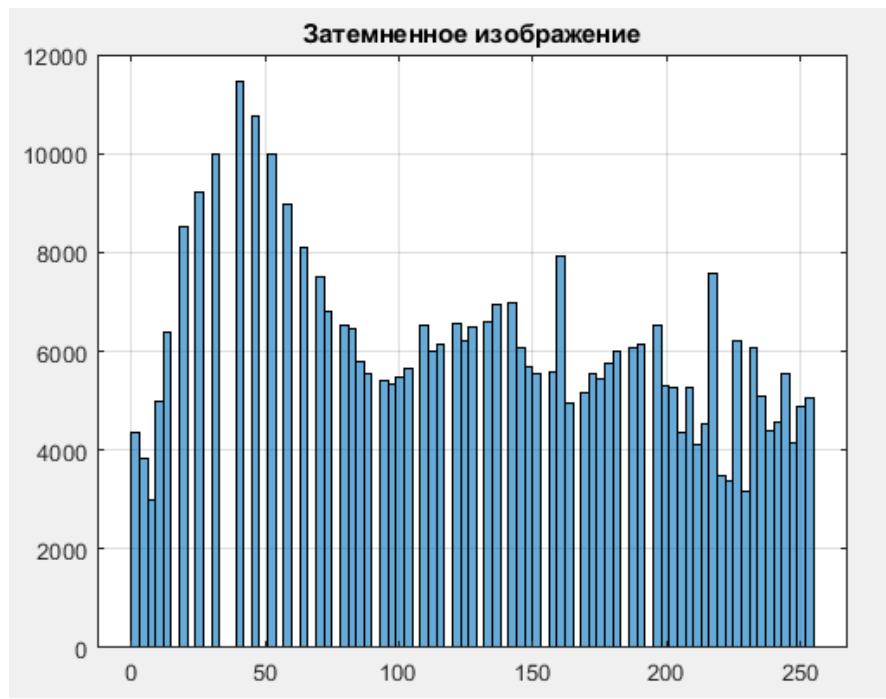


Рис. 131 – Гистограмма обработанного затемненного изображения.

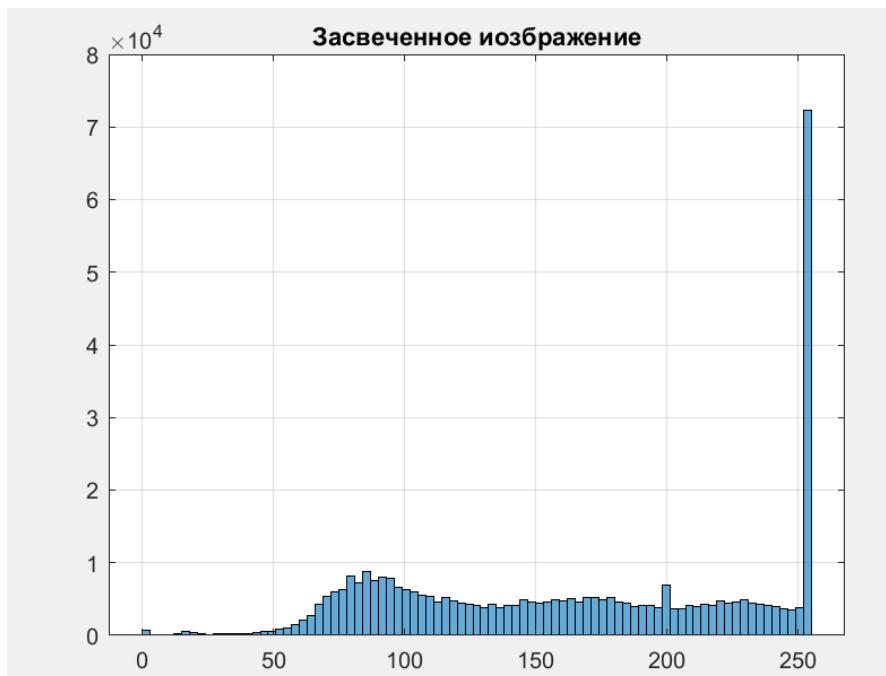


Рис. 132 – Гистограмма освещенного изображения.



Рис. 133 – Гистограмма обработанного осветленного изображения.

По полученным данным видно, что все пиксели выравниваются, образуя равномерное распределение. Визуально видно, что после обработки осветленного и затемнённого изображений, они стали ближе к сбалансированному.

7.11. Методы построения карты контуров на основе градационных преобразований.

Метод построение карты контуров заключается в следующем: в зависимости от выбора порога T выходной пиксель бинарного значения соответствует белому (интенсивность соответствующего пикселя исходного изображения превышает порог) или черному цвету (иначе).

Необходимо синтезировать градационную функцию преобразования для построения бинарного изображения. Порог T принадлежит интервалу $[16, 240]$ и изменяется с фиксированным шагом.



Рис. 134 – Исходное изображение.



Рис. 135 – Бинарное изображение при $t = 20$.



Рис. 136 – Бинарное изображение при $t = 40$.



Рис. 136 – Бинарное изображение при $t = 60$.



Рис. 137 – Бинарное изображение при $t = 80$.



Рис. 138 – Бинарное изображение при $t = 100$.



Рис. 139 – Бинарное изображение при $t = 120$.



Рис. 140 – Бинарное изображение при $t = 140$.



Рис. 141 – Бинарное изображение при $t = 160$.

Рис. 142 – Бинарное изображение при $t = 180$.



Рис. 143 – Бинарное изображение при $t = 200$.

Рис. 144 – Бинарное изображение при $t = 220$.

7.12.-7.13. Формирование наглядного бинарного изображения



Рис. 145 – Бинарное изображение при $t = 100$.

Изображение, полученное при $T = 100$, получилось наиболее наглядным, т.к. достаточно хорошо видны мелкие детали, при этом общий контур изображения так же хорошо различим.

Отсюда можно сделать вывод, что значение T находящееся приблизительно в середине диапазона для сбалансированного изображения является лучшим. Значения гораздо выше середины являются в данном случае худшими, т.к. на изображении просто белое. Значения гораздо больше среднего с увеличением порога T теряют точность, однако разобрать возможно.

8. Выводы

В ходе выполнение лабораторной работы было установлено:

1. Импульсный шум намного больше, чем аддитивный, ухудшает качество изображения, поскольку пиксели принимают крайние значения 0 или 255.

2. Для аддитивного шума, при малом количестве шума, лучшим фильтром оказался фильтр Гаусса. При большом количестве шума после применения любого фильтра у изображения нельзя различить основные детали.
3. С импульсным шумом практически идеально справляется медианный фильтр. Он полностью убирает шум при небольшом его количестве, а при большом количестве шума на изображении можно разглядеть основные детали.
4. Оператор Лапласа увеличивает яркость изображения при увеличении α . При $\alpha = 1$ яркость не меняется, но контуры становятся более четкими.
5. Оператор Собеля наиболее точную бинарную карту дает при пороге $thr = 60$. В случае, когда $thr < 40$ изображения перегружены черными пикселями. При пороге > 90 некоторые контуры становятся чётче различимы, а некоторые исчезают вовсе.
6. Методы на основе опорных точек и гамма-преобразования позволяют менять оттенки серого.
7. Метод выравнивания гистограмм для засвеченного и затемнённого изображений приближают картинки к сбалансированному и выравнивают значения пикселей по всему диапазону.
8. Наиболее точная карта контуров получается при значении порога $T = 100$, видны мелкие детали, а общий контур изображения ещё хорошо различим.

Листинг программы

Первая часть

```

close all
clear, clc

img = fopen('kodim04.bmp','r');

%BITMAPFILEHEADER

FH = struct('bfTYPE','bfSIZE','bfR1','bfR2','bfOFFBITS','');
FH.bfTYPE = fread(img, 2, 'uchar');
FH.bfSIZE = fread(img, 1, 'ulong');
FH.bfR1 = fread(img, 2, 'uint8');
FH.bfR2 = fread(img, 2, 'uint8');
FH.bfOFFBITS = fread(img, 1, 'ulong');

%BITMAPINFOHEADER

```

```

IH = struct('biSIZE','','biWIDTH','','biHEIGHT','','biPLANES','...
,'biBITCOUNT','','biCOMPRESSION','','biSIZEIMAGE','','biXPELSPERMETER',...
,'biYPELSPERMETER','','biCLRUSED','','biCLRIMPORTANT','tmp');

IH.biSIZE = fread(img, 1, 'ulong');
IH.biWIDTH = fread(img, 1, 'ulong');
IH.biHEIGHT = fread(img, 1, 'ulong');
IH.biPLANES = fread(img, 1, 'uint16');
IH.biBITCOUNT = fread(img, 1, 'uint16');
IH.biCOMPRESSION = fread(img, 1, 'ulong');
IH.biSIZEIMAGE = fread(img, 1, 'ulong');
IH.biXPELSPERMETER = fread(img, 1, 'ulong');
IH.biYPELSPERMETER = fread(img, 1, 'ulong');
IH.biCLRUSED = fread(img, 1, 'ulong');
IH.biCLRIMPORTANT = fread(img, 1, 'ulong');
IH.tmp = fread(img, FH.bfOFFBITS-54, 'uint8');
PIXELS = fread(img, 'uint8');
fclose(img);

```

```

j = 1;
for i = 1 : 3 : length(PIXELS)
    R(j) = PIXELS(i);
    G(j) = PIXELS(i + 1);
    B(j) = PIXELS(i + 2);
    j = j + 1;
end

```

Y = 0.299 * R + 0.587 * G + 0.114 * B;

% 1 Аддитивный шум

```

sigma_add = 50;
for i = 1 : length(Y)
    Y_add(i) = Y(i) + box_muller(sigma_add);
end
Y_add = cliping(Y_add, 0, 255);
Y_sigma_add = Y_add;
save_image(Y_add, Y_add, Y_add, 'add_noize_1.bmp', FH, IH);

```

% 2 Импульсный шум

```

pa = 0.03;
pb = 0.07;
for i = 1 : length(Y)
    p = rand();
    Y_imp(i) = Y(i);
    if (p <= pa)
        Y_imp(i) = 0;
    end
    if (p > pa && p <= pa + pb)
        Y_imp(i) = 255;
    end
end
save_image(Y_imp, Y_imp, Y_imp, 'imp_noize.bmp', FH, IH);

```

```

%3 Графики PSNR
disp('1.3 пункт');
sigma = [1, 10, 30, 50, 80];
for j = 1 : length(sigma)
    for i = 1 : length(Y)
        Y_add(j, i) = Y(i) + box_muller(sigma(j));
    end
    Y_add(j, :) = cliping(Y_add(j,:), 0, 255);
    save_image(Y_add(j,:), Y_add(j,:), Y_add(j,:),
    ['add_dnoize_',num2str(j),'.bmp'], FH, IH);
    psnrA(j) = PSNR(Y, Y_add(j,:), FH, IH);

    disp(['Для сигмы равной ',num2str(sigma(j)), ' PSNR =
    ',num2str(psnrA(j))]);
end
disp(' ');
%
Pab = [0.025, 0.05, 0.125, 0.25];
for j = 1 : length(Pab)
    for i = 1 : length(Y)
        p = rand;
        Y_imp(j,i) = Y(i);
        if (p <= Pab(j))
            Y_imp(j,i) = 0;
        end
        if (p > Pab(j) && p <= 2 * Pab(j))
            Y_imp(j,i) = 255;
        end
    end
end

```

```

    end
end
save_image(Y_imp(j,:), Y_imp(j,:), Y_imp(j,:),
['imp_noize_',num2str(j),'.bmp'], FH, IH);
psnrI(j) = PSNR(Y, Y_imp(j,:), FH, IH);
disp(['Для Pa = Pb = ',num2str(Pab(j)), ' PSNR = ',num2str(psnrI(j))]);
end
disp(' ');
figure();
plot(sigma, psnrA);
xlabel('Sigma');
ylabel('PSNR');
grid on
title('Аддитивный шум');
figure();
plot(Pab, psnrI);
xlabel('P_a,P_b');
ylabel('PSNR');
grid on
title('Импульсный шум');

```

%4.1 Скользящее среднее

```

disp('1.4.1 пункт');
Ymatrix = reshape(Y_sigma_add, IH.biWIDTH, IH.biHEIGHT);
R = 1;
for i = 1 : IH.biHEIGHT
    for j = 1 : IH.biWIDTH
        Yslide(j,i) = slide(Ymatrix, j, i, R, IH);
    end
end
Yslide = reshape(Yslide, 1, IH.biWIDTH * IH.biHEIGHT);
psnr = PSNR(Y_sigma_add, Yslide, FH, IH);
disp(['PSNR для скользящего среднего = ',num2str(psnr)])
disp(' ');
save_image(Yslide, Yslide, Yslide, 'Slide_ex.bmp', FH, IH);

```

%4.2 Подбор размера окна R

```

disp('1.4.2 пункт');
for s = 1 : length(sigma)
    Ynm = reshape(Y_add(s, :), IH.biWIDTH, IH.biHEIGHT);
    for R = 1 : 5

```

```

for i = 1 : IH.biHEIGHT
    for j = 1 : IH.biWIDTH
        Yslider(j,i,R) = slide(Ynm, j, i, R, IH);
    end
end
Yline = reshape(Yslider(:,:,R), 1, IH.biWIDTH * IH.biHEIGHT);
psnr(R) = PSNR(Y, Yline, FH, IH);
[res, ind] = max (psnr);
end
disp(['PSNR для сигмы = ',num2str(sigma(s)),': ',num2str(psnr)])
disp(['Максимальное PSNR = ',num2str(res)]);
disp(['Максимальный номер окна R = ',num2str(ind)]);
disp (' ');
figure();
plot(1:5, psnr);
xlabel('R');
ylabel('PSNR');
grid on
title(['Сигма = ',num2str(sigma(s))]);
end

```

```

% 4.3 Гауссовская фильтрация
disp('1.4.3 пункт');
R = 1;
sigmaFilter = 1;
Ymatrix = reshape(Y_sigma_add, IH.biWIDTH, IH.biHEIGHT);
for i = 1 : IH.biHEIGHT
    for j = 1 : IH.biWIDTH
        YGauss_ex(j,i) = gauss(Ymatrix, j, i, R, sigmaFilter, IH);
    end
end
Ygline = reshape(YGauss_ex, 1, IH.biWIDTH * IH.biHEIGHT);
save_image(Ygline, Ygline, Ygline, 'Gauss_ex.bmp', FH, IH);

```

```

%4.5 Графики PSNR для Гаусса
sigmaFilter = [0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 2];
R = 5;
figure();
for s = 1 : length(sigma)
    Yng = reshape(Y_add(s, :), IH.biWIDTH, IH.biHEIGHT);
    for sf = 1 : length(sigmaFilter)
        for i = 1 : IH.biHEIGHT

```

```

for j = 1 : IH.biWIDTH
    YGauss(sf,j,i) = gauss(Yng, j, i, R, sigmaFilter(sf), IH);
end
end
Ygline = reshape(YGauss(sf,:,:), 1, IH.biWIDTH * IH.biHEIGHT);
psnr(s, sf) = PSNR(Y, Ygline, FH, IH);
end

plot(sigmaFilter, psnr(s, :));
xlabel('сигма фильтра');
ylabel('PSNR');
hold on;
grid on
title(['R = ',num2str(R)]);
legend('сигма = 1', 'сигма = 10', 'сигма = 30', 'сигма = 50', 'сигма = 80');
Ys = reshape(YGauss(5,:,:), 1, IH.biWIDTH * IH.biHEIGHT);
end

%4.7 Медианный фильтр
Ymatrix = reshape(Y_sigma_add, IH.biWIDTH, IH.biHEIGHT);
R_ = 5;
for i = 1 : IH.biHEIGHT
    for j = 1 : IH.biWIDTH
        YMedian_ex(j,i) = mediana(Ymatrix, j, i, R_, IH);
    end
end
Ygline = reshape(YMedian_ex, 1, IH.biWIDTH * IH.biHEIGHT);
save_image(Ygline, Ygline, Ygline, 'Mediana_ex.bmp', FH, IH);

%4.8 Подбор размера окна R
disp ('1.4.8 пункт');
sigma = [1, 10, 30, 50, 80];
R = [1, 3, 5];
PSNR_2 = zeros(1, length(R));
figure();
for s = 1 : length(sigma)
    Yng = reshape(Y_add(s,:), IH.biWIDTH, IH.biHEIGHT);
    for r = 1 : length(R)
        for i = 1 : IH.biHEIGHT
            for j = 1 : IH.biWIDTH
                YMedian(r,j,i) = mediana(Yng, j, i, R(r), IH);
            end
        end
    end
end

```

```

end
Ymline = reshape(YMedian(r,:,:), 1, IH.biWIDTH * IH.biHEIGHT);
PSNR_2(s, r) = PSNR(Y, Ymline, FH, IH);
[res, ind] = max(PSNR_2(s, :));
end
disp(['PSNR для сигмы = ',num2str(sigma(s)),': ',num2str(PSNR_2(s, :))])
disp(['Максимальное PSNR = ',num2str(res)]);
disp(['Максимальный номер окна R = ',num2str(ind)]);
disp(' ');
plot(R, PSNR_2(s,:));
xlabel('R');
ylabel('PSNR');
hold on;
grid on
end
title('Медианный фильтр');
legend('сигма = 1', 'сигма = 10', 'сигма = 30', 'сигма = 50', 'сигма = 80');

```

%4.6 4.9

```

sigNoize = 5; %1,2,3,4,5
Ymatrix = reshape(Y_add(sigNoize, :), IH.biWIDTH, IH.biHEIGHT);

R_s = 4; %1, 1, 2, 3, 4
R_g = 3; %1, 1, 2, 3, 3
R_m = 3; %1, 1, 2, 3, 3
sigm = 2; %0.25, 0.75, 1.25, 1.25, 2

for i = 1 : IH.biHEIGHT
    for j = 1 : IH.biWIDTH
        Yslide1(j,i) = slide(Ymatrix, j, i, R_s, IH);
    end
end
Yslide1 = reshape(Yslide1, 1, IH.biWIDTH * IH.biHEIGHT);
save_image(Yslide1, Yslide1, Yslide1, ['slide_',num2str(sigNoize),'.bmp'],
FH, IH);

for i = 1 : IH.biHEIGHT
    for j = 1 : IH.biWIDTH
        Yslide2(j,i) = gauss(Ymatrix, j, i, R_g, sigm, IH);
    end
end
Yslide2 = reshape(Yslide2, 1, IH.biWIDTH * IH.biHEIGHT);

```

```

save_image(Yslide2, Yslide2, Yslide2, ['gauss_',num2str(sigNoize),'.bmp'],
FH, IH);

for i = 1 : IH.biHEIGHT
    for j = 1 : IH.biWIDTH
        Yslide3(j,i) = mediana(Ymatrix, j, i, R_m, IH);
    end
end
Yslide3 = reshape(Yslide3, 1, IH.biWIDTH * IH.biHEIGHT);
save_image(Yslide3, Yslide3, Yslide3,
['mediana_',num2str(sigNoize),'.bmp'], FH, IH);

%4.10
Rslide = [1, 1, 2, 3, 4];
Rgauss = [1, 1, 2, 3, 3];
Rmediana = [1, 1, 2, 3, 3];
sigmaFilter = [0.25, 0.75, 1.25, 1.25, 2];

Yslide = zeros(IH.biWIDTH, IH.biHEIGHT);
Ygauss = zeros(IH.biWIDTH, IH.biHEIGHT);
Ymediana = zeros(IH.biWIDTH, IH.biHEIGHT);
for s = 1 : length(sigma)
    Ymatrix = reshape(Y_add(s, :), IH.biWIDTH, IH.biHEIGHT);
    for i = 1 : IH.biHEIGHT
        for j = 1 : IH.biWIDTH
            Yslide(j,i) = slide(Ymatrix, j, i, Rslide(s), IH);
            Ygauss(j,i) = gauss(Ymatrix, j, i, Rgauss(s), sigmaFilter(s), IH);
            Ymediana(j,i) = mediana(Ymatrix, j, i, Rmediana(s), IH);
        end
    end
    YslideR = reshape(Yslide, 1, IH.biWIDTH * IH.biHEIGHT);
    YgaussR = reshape(Ygauss, 1, IH.biWIDTH * IH.biHEIGHT);
    YmedianaR = reshape(Ymediana, 1, IH.biWIDTH * IH.biHEIGHT);
    psnrS(s) = PSNR(Y, YslideR, FH, IH);
    psnrG(s) = PSNR(Y, YgaussR, FH, IH);
    psnrM(s) = PSNR(Y, YmedianaR, FH, IH);
end

figure();
plot(sigma, psnrA, 'r');
hold on; grid on

```

```

plot(sigma, psnrS, 'b');
hold on; grid on
plot(sigma, psnrG, 'g');
hold on; grid on
plot(sigma, psnrM, 'k');
hold on; grid on
xlabel('Сигма');
ylabel('PSNR');
title('PSNR от сигмы шума');
legend('Изображение с шумом', 'Скользящее среднее', 'Гауссовский
фильтр', 'Медианный фильтр');

%5.3
Ymatrix = reshape(Y_imp(4, :), IH.biWIDTH, IH.biHEIGHT);%1 0.05 2 0.1
3 0.25 4 0.5
R = 4; %1 1 2 4
for i = 1 : IH.biHEIGHT
    for j = 1 : IH.biWIDTH
        YmedianaIMP(j,i) = mediana(Ymatrix, j, i, R, IH);
    end
end
YmedianaIR = reshape(YmedianaIMP, 1, IH.biWIDTH * IH.biHEIGHT);
save_image(YmedianaIR, YmedianaIR, YmedianaIR, 'imp_noize_R4.bmp',
FH, IH);%1-2-3-4

%5.4
for t = 1 : 4
    Ymatrix = reshape(Y_imp(t, :), IH.biWIDTH, IH.biHEIGHT);
    for r = 1 : 5
        for i = 1 : IH.biHEIGHT
            for j = 1 : IH.biWIDTH
                YmedianaNEW(r,j,i) = mediana(Ymatrix, j, i, r, IH);
            end
        end
        YmedianaR = reshape(YmedianaNEW(r,:,:), 1, IH.biWIDTH *
IH.biHEIGHT);
        psnrIMP(t, r) = PSNR(Y, YmedianaR, FH, IH);
    end
end
r = 1 : 5;
figure();

```



```

res = 10*log10((IH.biWIDTH * IH.biHEIGHT * (2^8 - 1)^2) / sum((A -
B).^2));
end

function save_image(r, g, b, filename, FH, IH)
    FID = fopen(filename, 'w');
    fwrite(FID, FH.bfTYPE, 'uchar');
    fwrite(FID, FH.bfSIZE, 'ulong');
    fwrite(FID, FH.bfR1, 'uint8');
    fwrite(FID, FH.bfR2, 'uint8');
    fwrite(FID, FH.bfOFFBITS, 'ulong');

    fwrite(FID, IH.biSIZE, 'ulong');
    fwrite(FID, IH.biWIDTH, 'ulong');
    fwrite(FID, IH.biHEIGHT, 'ulong');
    fwrite(FID, IH.biPLANES, 'uint16');
    fwrite(FID, IH.biBITCOUNT, 'uint16');
    fwrite(FID, IH.biCOMPRESSION, 'ulong');
    fwrite(FID, IH.biSIZEIMAGE, 'ulong');
    fwrite(FID, IH.biXPELSPERMETER, 'ulong');
    fwrite(FID, IH.biYPELSPERMETER, 'ulong');
    fwrite(FID, IH.biCLRUSED, 'ulong');
    fwrite(FID, IH.biCLRIMPORTANT, 'ulong');
    fwrite(FID, IH.tmp, 'uint8');
    for i = 1 : length(r)
        fwrite(FID, r(i), 'uint8');
        fwrite(FID, g(i), 'uint8');
        fwrite(FID, b(i), 'uint8');
    end
    fclose(FID);
end

function res = slide(Y, indexX, indexY, R, IH)
    summ = 0;
    for i = -R : R
        tmpI = i + indexY;
        if (i + indexY < 1)
            tmpI = 1;
        end
        if (i + indexY > IH.biHEIGHT)
            tmpI = IH.biHEIGHT;
        end
    end
    res = summ;
end

```

```

for j = -R : R
    tmpJ = j + indexX;
    if (j + indexX < 1)
        tmpJ = 1;
    end
    if (j + indexX > IH.biWIDTH)
        tmpJ = IH.biWIDTH;
    end
    summ = summ + Y(tmpJ, tmpI);
end
res = (1 / (2*R + 1)^2) * summ;
end

```

```

function res = point(Y, x1, x2, y1, y2)
z = zeros(1, 256);
for i = 1 : x1
    z(i) = (y1 * i)/x1;
end
for i = x1 + 1 : x2
    z(i) = y1 + ((y2-y1)/(x2-x1))*(i - x1);
end
for i = x2 + 1 : 256
    z(i) = y2 + ((255 - y2)/(255 - x2))*(i - x2);
end
figure();
plot(0 : 255, z);
grid on
res = zeros(1, length(Y));
for i = 1 : length(Y)
    res(i) = z(floor(Y(i))+1);
end
end

```

```

function res = gauss(Y, indexX, indexY, R, sigma, IH)
summ = 0;
Z = 0;
for i = -R : R
    tmpI = i + indexY;
    if (i + indexY < 1)
        tmpI = 1;
    end

```

```

if (i + indexY > IH.biHEIGHT)
    tmpI = IH.biHEIGHT;
end
for j = -R : R
    tmpJ = j + indexX;
    if (j + indexX < 1)
        tmpJ = 1;
    end
    if (j + indexX > IH.biWIDTH)
        tmpJ = IH.biWIDTH;
    end
    w = exp((-j^2 + i^2)/(2*sigma^2));
    Z = Z + w;
    summ = summ + Y(tmpJ, tmpI) * w;
end
res = (1 / Z) * summ;
end

```

```

function res = mediana(Y, indexX, indexY, R, IH)
ind = 1;
for i = -R : R
    tmpI = i + indexY;
    if (i + indexY < 1)
        tmpI = 1;
    end
    if (i + indexY > IH.biHEIGHT)
        tmpI = IH.biHEIGHT;
    end
    for j = -R : R
        tmpJ = j + indexX;
        if (j + indexX < 1)
            tmpJ = 1;
        end
        if (j + indexX > IH.biWIDTH)
            tmpJ = IH.biWIDTH;
        end
        A(ind) = Y(tmpJ, tmpI);
        ind = ind + 1;
    end
end
A = sort(A);

```

```
    res = A(floor(((2*R+1)^2)/2) + 1);
end
```

Вторая часть

```
close all
clear, clc
```

```
FID = fopen('kodim04.bmp','r');
```

```
%BITMAPFILEHEADER
```

```
FH = struct('bfTYPE','bfSIZE','bfR1','bfR2','bfOFFBITS','');
FH.bfTYPE = fread(FID, 2, 'uchar');
FH.bfSIZE = fread(FID, 1, 'ulong');
FH.bfR1 = fread(FID, 2, 'uint8');
FH.bfR2 = fread(FID, 2, 'uint8');
FH.bfOFFBITS = fread(FID, 1, 'ulong');
```

```
%BITMAPINFOHEADER
```

```
IH =
struct('biSIZE','biWIDTH','biHEIGHT','biPLANES','biBITCOUNT','',
biCOMPRESSION','biSIZEIMAGE','biXPELSPERMETER','biYPELSP
ERMETER','biCLRUSED','biCLRIMPORTANT','tmp');
IH.biSIZE = fread(FID, 1, 'ulong');
IH.biWIDTH = fread(FID, 1, 'ulong');
IH.biHEIGHT = fread(FID, 1, 'ulong');
IH.biPLANES = fread(FID, 1, 'uint16');
IH.biBITCOUNT = fread(FID, 1, 'uint16');
IH.biCOMPRESSION = fread(FID, 1, 'ulong');
IH.biSIZEIMAGE = fread(FID, 1, 'ulong');
IH.biXPELSPERMETER = fread(FID, 1, 'ulong');
IH.biYPELSPERMETER = fread(FID, 1, 'ulong');
IH.biCLRUSED = fread(FID, 1, 'ulong');
IH.biCLRIMPORTANT = fread(FID, 1, 'ulong');
IH.tmp = fread(FID, FH.bfOFFBITS-54, 'uint8');
PIXELS = fread(FID, 'uint8');
fclose(FID);
```

```
j = 1;
for i = 1 : 3 : length(PIXELS)
```

```

R(j) = PIXELS(i);
G(j) = PIXELS(i + 1);
B(j) = PIXELS(i + 2);
j = j + 1;
end
%2.2.1.1-2
Y = 0.299 * R + 0.587 * G + 0.114 * B;
figure();
histogram(Y);
grid on
title('Y');
disp(['Средняя яркость: ',num2str(mean(Y))]);
Mask = [0, -1, 0; -1, 4, -1; 0, -1, 0];
L = laplasArr(Y, Mask, IH, 0);
saveimg(L, L, L, 'laplas_11.bmp', FH, IH);
Y1 = clip(L + 128, 0, 255);
saveimg(Y1, Y1, Y1, 'laplas_11_clipp.bmp', FH, IH);

%2.2.1.3
saveimg(L+Y, L+Y, L+Y, 'laplas_2.bmp', FH, IH);
Y2 = clip(L + Y, 0, 255);
saveimg(Y2, Y2, Y2, 'laplas_22_clipp.bmp', FH, IH);

%2.2.1.4-5
Mask = [0, -1, 0; -1, 5, -1; 0, -1, 0];
L = laplasArr(Y, Mask, IH, 1);
Y2 = clip(L, 0, 255);
figure();
histogram(Y2);
grid on
title('alpha = 1');
disp(['Средняя яркость (alpha = 1): ',num2str(mean(Y2))]);
saveimg(Y2, Y2, Y2, 'laplas_alpha_1.bmp', FH, IH);

Mask = [0, -1, 0; -1, 5.1, -1; 0, -1, 0];
L = laplasArr(Y, Mask, IH, 1.1);
Y2 = clip(L, 0, 255);
figure();
histogram(Y2);
grid on
title('alpha = 1.1');
disp(['Средняя яркость (alpha = 1.1): ',num2str(mean(Y2))]);

```

```

saveimg(Y2, Y2, Y2, 'laplas_alpha_11.bmp', FH, IH);

Mask = [0, -1, 0; -1, 5.2, -1; 0, -1, 0];
L = laplasArr(Y, Mask, IH, 1.2);
Y2 = clip(L, 0, 255);
figure();
histogram(Y2);
grid on
title('alpha = 1.2');
disp(['Средняя яркость (alpha = 1.2): ',num2str(mean(Y2))]);
saveimg(Y2, Y2, Y2, 'laplas_alpha_12.bmp', FH, IH);

Mask = [0, -1, 0; -1, 5.3, -1; 0, -1, 0];
L = laplasArr(Y, Mask, IH, 1.3);
Y2 = clip(L, 0, 255);
figure();
histogram(Y2);
grid on
title('alpha = 1.3');
disp(['Средняя яркость (alpha = 1.3): ',num2str(mean(Y2))]);
saveimg(Y2, Y2, Y2, 'laplas_alpha_13.bmp', FH, IH);

Mask = [0, -1, 0; -1, 5.4, -1; 0, -1, 0];
L = laplasArr(Y, Mask, IH, 1.4);
Y2 = clip(L, 0, 255);
figure();
histogram(Y2);
grid on
title('alpha = 1.4');
disp(['Средняя яркость (alpha = 1.4): ',num2str(mean(Y2))]);
saveimg(Y2, Y2, Y2, 'laplas_alpha_14.bmp', FH, IH);

Mask = [0, -1, 0; -1, 5.5, -1; 0, -1, 0];
L = laplasArr(Y, Mask, IH, 1.5);
Y2 = clip(L, 0, 255);
figure();
histogram(Y2);
title('alpha = 1.5');
grid on
disp(['Средняя яркость (alpha = 1.5): ',num2str(mean(Y2))]);
saveimg(Y2, Y2, Y2, 'laplas_alpha_15.bmp', FH, IH);

%2.2

```

```

Y = 0.299 * R + 0.587 * G + 0.114 * B;
Mask1 = [-1, 0, 1; -2, 0, 2; -1, 0, 1]; %горизонталь
Mask2 = [1, 2, 1; 0, 0, 0; -1, -2, -1]; %вертикаль

L1 = laplasArr(Y, Mask1, IH, 0);
L2 = laplasArr(Y, Mask2, IH, 0);
l = [10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160];
for j = 1 : length(l)
    Mod = sqrt(L1.^2 + L2.^2);
    Thet = atan(L2./L1);
    thr = l(j);
    for i = 1 : length(Mod)
        if (Mod(i) > thr)
            Mod(i) = 0;
        else
            Mod(i) = 255;
        end
    end
    saveimg(Mod, Mod, Mod, ['thr_',num2str(l(j)),'.bmp'], FH, IH);
end

Rs = zeros(1, length(Y));
Gs = zeros(1, length(Y));
Bs = zeros(1, length(Y));
for i = 1 : length(Y)
    if (L2(i) > 0)
        if (L1(i) > 0)
            Rs(i) = 255;
        else
            Rs(i) = 255;
            Gs(i) = 255;
            Bs(i) = 255;
        end
    else
        if (L1(i) > 0)
            Gs(i) = 255;
        else
            Bs(i) = 255;
        end
    end
end
saveimg(Rs, Gs, Bs, 'color.bmp', FH, IH);

```



```

    end
    if (i(j) < min)
        res(j) = min;
    end
end

function res = laplasArr(Y, Mask, IH, alp)
    Y = reshape(Y, IH.biWIDTH, IH.biHEIGHT);
    for i = 1 : IH.biHEIGHT
        for j = 1 : IH.biWIDTH
            res(j,i) = laplas(Y, j, i, Mask, IH, alp);
        end
    end
    res = reshape(res, 1, IH.biHEIGHT * IH.biWIDTH);
end

function res = laplas(Y, indexX, indexY, Mask, IH, alp)
    summ = 0;
    if (alp > 0)
        for i = 1 : size(Mask,1)
            tmpI = indexY - (floor(size(Mask,1)/2) + 1) + i;
            if (tmpI < 1)
                tmpI = 1;
            end
            if (tmpI > IH.biHEIGHT)
                tmpI = IH.biHEIGHT;
            end
            for j = 1 : size(Mask,2)
                tmpJ = indexX - (floor(size(Mask,1)/2) + 1) + j;
                if (tmpJ < 1)
                    tmpJ = 1;
                end
                if (tmpJ > IH.biWIDTH)
                    tmpJ = IH.biWIDTH;
                end
                summ = summ + (Y(tmpJ, tmpI) * Mask(i,j));
            end
        end
        res = summ;
        if (Mask(floor(size(Mask,1)/2) + 1, floor(size(Mask,2)/2) + 1) > 0)
            res = (alp-1)*Y(indexX,indexY) + summ;
        end
    end
end

```

```

else
    res = (alp-1)*Y(indexX,indexY) - summ;
end
end
if (alp == 0)
    for i = 1 : size(Mask,1)
        tmpI = indexY - (floor(size(Mask,1)/2) + 1) + i;
        if (tmpI < 1)
            tmpI = 1;
        end
        if (tmpI > IH.biHEIGHT)
            tmpI = IH.biHEIGHT;
        end
        for j = 1 : size(Mask,2)
            tmpJ = indexX - (floor(size(Mask,1)/2) + 1) + j;
            if (tmpJ < 1)
                tmpJ = 1;
            end
            if (tmpJ > IH.biWIDTH)
                tmpJ = IH.biWIDTH;
            end
            summ = summ + (Y(tmpJ, tmpI) * Mask(i,j));
        end
    end
    res = summ;
    if (Mask(floor(size(Mask,1)/2) + 1, floor(size(Mask,2)/2) + 1) > 0)
        res = (alp)*Y(indexX,indexY) + summ;
    else
        res = (alp)*Y(indexX,indexY) - summ;
    end
end
end

```

Третья часть

close all
clear, clc

FID = fopen('kodim04.bmp','r');

```

%BITMAPFILEHEADER
FH = struct('bfTYPE','bfSIZE','bfR1','bfR2','bfOFFBITS','');
FH.bfTYPE = fread(FID, 2, 'uchar');
FH.bfSIZE = fread(FID, 1, 'ulong');
FH.bfR1 = fread(FID, 2, 'uint8');
FH.bfR2 = fread(FID, 2, 'uint8');
FH.bfOFFBITS = fread(FID, 1, 'ulong');

%BITMAPINFOHEADER
IH =
struct('biSIZE','biWIDTH','biHEIGHT','biPLANES','biBITCOUNT','
biCOMPRESSION','biSIZEIMAGE','biXPELSPERMETER','biYPELSP
ERMETER','biCLRUSED','biCLRIMPORTANT','tmp');
IH.biSIZE = fread(FID, 1, 'ulong');
IH.biWIDTH = fread(FID, 1, 'ulong');
IH.biHEIGHT = fread(FID, 1, 'ulong');
IH.biPLANES = fread(FID, 1, 'uint16');
IH.biBITCOUNT = fread(FID, 1, 'uint16');
IH.biCOMPRESSION = fread(FID, 1, 'ulong');
IH.biSIZEIMAGE = fread(FID, 1, 'ulong');
IH.biXPELSPERMETER = fread(FID, 1, 'ulong');
IH.biYPELSPERMETER = fread(FID, 1, 'ulong');
IH.biCLRUSED = fread(FID, 1, 'ulong');
IH.biCLRIMPORTANT = fread(FID, 1, 'ulong');
IH.tmp = fread(FID, FH.bfOFFBITS-54, 'uint8');
PIXELS = fread(FID, 'uint8');
fclose(FID);

j = 1;
for i = 1 : 3 : length(PIXELS)
    R(j) = PIXELS(i);
    G(j) = PIXELS(i + 1);
    B(j) = PIXELS(i + 2);
    j = j + 1;
end

Y = 0.299 * R + 0.587 * G + 0.114 * B;
figure();
histogram(Y);
title('Сбалансиованное изображение');
grid on

```

```
figure();
Black = 0.5 .* Y;
histogram(Black);
title('Затемненное изображение');
grid on

figure();
Light = clip(2 .* Y, 0, 255);
histogram(Light);
title('Засвеченное изображение');
grid on

X1 = 70;
Y1 = 10;
X2 = 250;
Y2 = 150;
Yl = point(Light, X1, X2, Y1, Y2);
figure();
histogram(Yl);
title('Засвеченное изображение после преобразования');
grid on

X1 = 90;
Y1 = 200;
X2 = 210;
Y2 = 190;
Yb = point(Black, X1, X2, Y1, Y2);
figure();
histogram(Yb);
title('Затемненное изображение после преобразования');
grid on

X1 = 80;
Y1 = 110;
X2 = 170;
Y2 = 200;
Yy = point(Y, X1, X2, Y1, Y2);
figure();
histogram(Yy);
title('Сбалансированное изображение после преобразования');
grid on

saveimg(Y, Y, Y, 'normal.bmp', FH, IH);
```

```

saveimg(Black, Black, Black, 'black.bmp', FH, IH);
saveimg(Light, Light, Light, 'light.bmp', FH, IH);

saveimg(Yy, Yy, Yy, 'normal_2.bmp', FH, IH);
saveimg(Yb, Yb, Yb, 'black_2.bmp', FH, IH);
saveimg(Yl, Yl, Yl, 'light_2.bmp', FH, IH);

%ГАММА
Y = 0.299 * R + 0.587 * G + 0.114 * B;
Black = 0.5 .* Y;
Light = clip(2 .* Y, 0, 255);

p = 0 : 255;
g = [0.04, 0.5, 1, 8, 25];
for i = 1 : length(g)
    Yy(i,:) = gammaP(Y, g(i));
    saveimg(Yy(i,:), Yy(i,:), Yy(i,:), ['GLight_',num2str(i),'.bmp'], FH, IH);
    figure(i);
    histogram(Yy(i,:));
    title(['Засвеченное изображение, gamma = ',num2str(g(i))]);
    grid on
end

z1 = (p/255).^g(1);
z2 = (p/255).^g(2);
z3 = (p/255).^g(3);
z4 = (p/255).^g(4);
z5 = (p/255).^g(5);
figure();
plot(p,z1, p,z2, p,z3, p,z4, p,z5);
grid on;
legend('gamma = 0.04', 'gamma = 0.5', 'gamma = 1', 'gamma = 8', 'gamma = 25');

% выравнивание гистограмм
Y = 0.299 * R + 0.587 * G + 0.114 * B;
Black = 0.5 .* Y;
Light = clip(2 .* Y, 0, 255);

Yy = lineHist(Y);
figure();
histogram(Yy);

```



```

fwrite(FID, FH.bfTYPE, 'uchar');
fwrite(FID, FH.bfSIZE, 'ulong');
fwrite(FID, FH.bfR1, 'uint8');
fwrite(FID, FH.bfR2, 'uint8');
fwrite(FID, FH.bfOFFBITS, 'ulong');

fwrite(FID, IH.biSIZE, 'ulong');
fwrite(FID, IH.biWIDTH, 'ulong');
fwrite(FID, IH.biHEIGHT, 'ulong');
fwrite(FID, IH.biPLANES, 'uint16');
fwrite(FID, IH.biBITCOUNT, 'uint16');
fwrite(FID, IH.biCOMPRESSION, 'ulong');
fwrite(FID, IH.biSIZEIMAGE, 'ulong');
fwrite(FID, IH.biXPELSPERMETER, 'ulong');
fwrite(FID, IH.biYPELSPERMETER, 'ulong');
fwrite(FID, IH.biCLRUSED, 'ulong');
fwrite(FID, IH.biCLRIMPORTANT, 'ulong');
fwrite(FID, IH.tmp, 'uint8');
for i = 1 : length(r)
    fwrite(FID, r(i), 'uint8');
    fwrite(FID, g(i), 'uint8');
    fwrite(FID, b(i), 'uint8');
end
fclose(FID);
end

```

```

function res = point(Y, x1, x2, y1, y2)
z = zeros(1, 256);
for i = 1 : x1
    z(i) = (y1 * i)/x1;
end
for i = x1 + 1 : x2
    z(i) = y1 + ((y2-y1)/(x2-x1))*(i - x1);
end
for i = x2 + 1 : 256
    z(i) = y2 + ((255 - y2)/(255 - x2))*(i - x2);
end
figure();
plot(0 : 255, z);
grid on
res = zeros(1, length(Y));

```

```

for i = 1 : length(Y)
    res(i) = z(floor(Y(i))+1);
end
end

function res = clip(i, min, max)
    res = i;
    for j = 1 : length(i)
        if (i(j) > max)
            res(j) = max;
        end
        if (i(j) < min)
            res(j) = min;
        end
    end
end

function res = gammaP(Y, gam)
    res = Y ./ 255;
    res = res.^gam;
    res = clip(floor(res * 255), 0, 255);
end

function res = lineHist(Y)
    t = zeros(1, 256);
    for i = 1 : length(Y)
        t(floor(Y(i))+1) = t(floor(Y(i))+1) + 1;
    end
    res = zeros(1, length(Y));
    for i = 1 : length(Y)
        summ = 0;
        for j = 1 : Y(i)+1
            summ = summ + t(j);
        end
        res(i) = floor(255 * summ / length(Y));
    end
end

function res = bin(Y, t)
    res = zeros(1, length(Y));
    for i = 1 : length(Y)
        if (Y(i) <= t)
            res(i) = 0;
        end
    end

```

```
else
    res(i) = 255;
end
end
```