Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Уральский федеральный университет

имени первого Президента России Б.Н. Ельцина»

Физико-Технологический Институт

Кафедра технической физики

Отчет по лабораторной работе №3 по теме

«**АНАЛИЗ МОРФОЛОГИЧЕСКИХ ХАРАКТЕРИСТИК ИЗОБРАЖЕНИЙ»**

|  |  |
| --- | --- |
| Преподаватель: | Новоселов Иван Эдуардович |
| Студенты: | Черняков Матвей Сергеевич |
| Группа: | Фт-420008 |

Екатеринбург

2025

**ОГЛАВЛЕНИЕ**

[ЗАДАНИЕ 3](#_Toc213541980)

[ВВЕДЕНИЕ 4](#_Toc213541981)

[ОПИСАНИЕ РЕШЕНИЯ 5](#_Toc213541982)

[1. Теоретические основы 5](#_Toc213541983)

[2. Реализация 5](#_Toc213541984)

[РЕЗУЛЬТАТЫ И ВЫВОДЫ 7](#_Toc213541985)

[СПИСОК ЛИТЕРАТУРЫ 14](#_Toc213541986)

[ПРИЛОЖЕНИЕ А. ПРОГРАММНЫЙ КОД 15](#_Toc213541987)

# ЗАДАНИЕ

Изучить и применить морфологические операции для анализа изображений, оценить влияние морфологических фильтров на извлекаемые признаки.

Задачи:

1. Загрузить исходные изображения;
2. Применить морфологические операции: эрозия, дилатация, открытие, закрытие, top-hat, black-hat;
3. Построить бинарные маски, выделить границы и области интереса (ROI);
4. Извлечь морфологические признаки объектов;
5. Провести сравнение характеристик до и после морфологических фильтров;
6. Визуализировать результаты обработки.

# ВВЕДЕНИЕ

Морфологический анализ изображений является фундаментальным инструментом в области компьютерного зрения и обработки изображений [1]. Морфологические операции позволяют изменять форму объектов на изображении, выделять структурные элементы, удалять шум и извлекать значимые признаки.

Базовые морфологические операции — эрозия и дилатация — служат основой для более сложных преобразований, таких как открытие, закрытие, морфологический градиент, top-hat и black-hat трансформации [2]. Эти операции широко применяются в задачах сегментации изображений, распознавания образов, медицинской диагностике и промышленном контроле качества.

Цель работы — освоить методы морфологического анализа изображений, изучить влияние морфологических фильтров на извлекаемые признаки объектов и продемонстрировать практическое применение с использованием Python библиотек OpenCV и scikit-image [3].

# ОПИСАНИЕ РЕШЕНИЯ

## 1. Теоретические основы

Морфологические операции основаны на теории множеств и оперируют со структурными элементами (ядрами). Основные операции [4]:

* Эрозия — уменьшает размер объектов, удаляет мелкие детали;
* Дилатация — увеличивает размер объектов, заполняет дыры;
* Открытие = эрозия + дилатация — удаляет мелкий шум;
* Закрытие = дилатация + эрозия — заполняет небольшие дыры;
* Морфологический градиент = дилатация – эрозия — выделяет границы;
* Top-hat = исходное – открытие — выделяет светлые детали;
* Black-hat = закрытие – исходное — выделяет тёмные детали.

## 2. Реализация

Импорт библиотек:

import cv2

from skimage import morphology, measure

Создание структурного элемента:

kernel = cv2.getStructuringElement(

cv2.MORPH\_ELLIPSE, (5, 5))

Морфологические операции:

erosion = cv2.erode(img, kernel)

dilation = cv2.dilate(img, kernel)

opening = cv2.morphologyEx(

img, cv2.MORPH\_OPEN, kernel)

closing = cv2.morphologyEx(

img, cv2.MORPH\_CLOSE, kernel)

Извлечение признаков:

labeled = measure.label(binary)

props = measure.regionprops(labeled)

area = prop.area

perimeter = prop.perimeter

eccentricity = prop.eccentricity

# РЕЗУЛЬТАТЫ И ВЫВОДЫ

В ходе выполнения работы были применены все основные морфологические операции и проведён анализ их влияния на характеристики объектов.

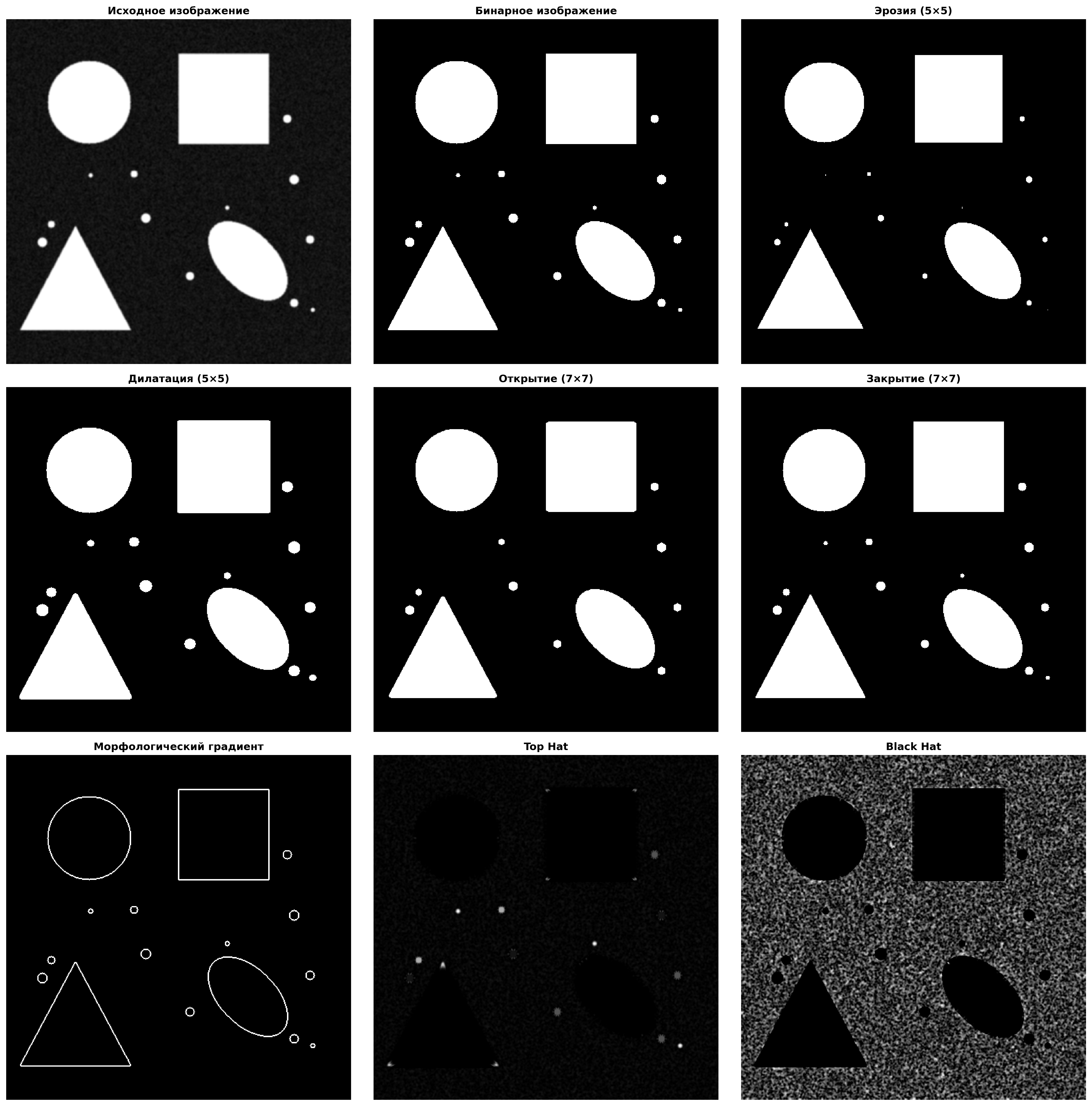


Рисунок 1 — Морфологические операции

На рисунке 1 представлены результаты применения базовых морфологических операций. Эрозия уменьшает объекты и удаляет мелкие детали (46538 белых пикселей против 50798 в оригинале). Дилатация увеличивает объекты (55461 пикселей). Открытие эффективно удаляет шум, сохраняя крупные объекты. Закрытие заполняет небольшие дыры в объектах. Морфологический градиент выделяет контуры. Top-hat выявляет светлые детали меньше структурного элемента, black-hat — тёмные.

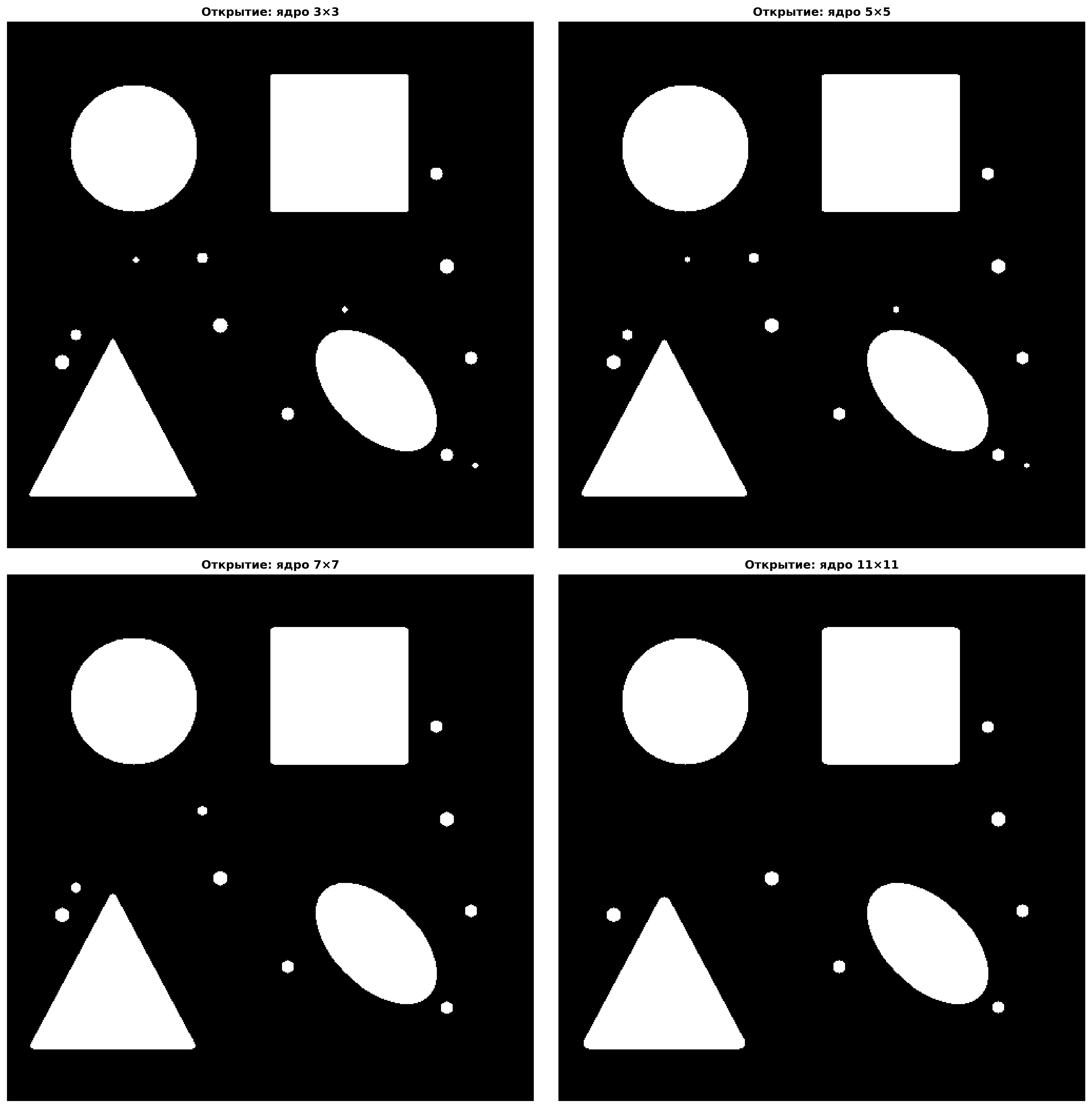


Рисунок 2 — Влияние размера структурного элемента

Размер структурного элемента критически влияет на результат (рисунок 2). При увеличении ядра с 3×3 до 11×11 количество белых пикселей после открытия уменьшается с 50778 до 50402, что свидетельствует об удалении всё более крупных деталей. Малые ядра сохраняют мелкие объекты, большие — удаляют их. Выбор размера ядра зависит от масштаба шума и значимых деталей на изображении.

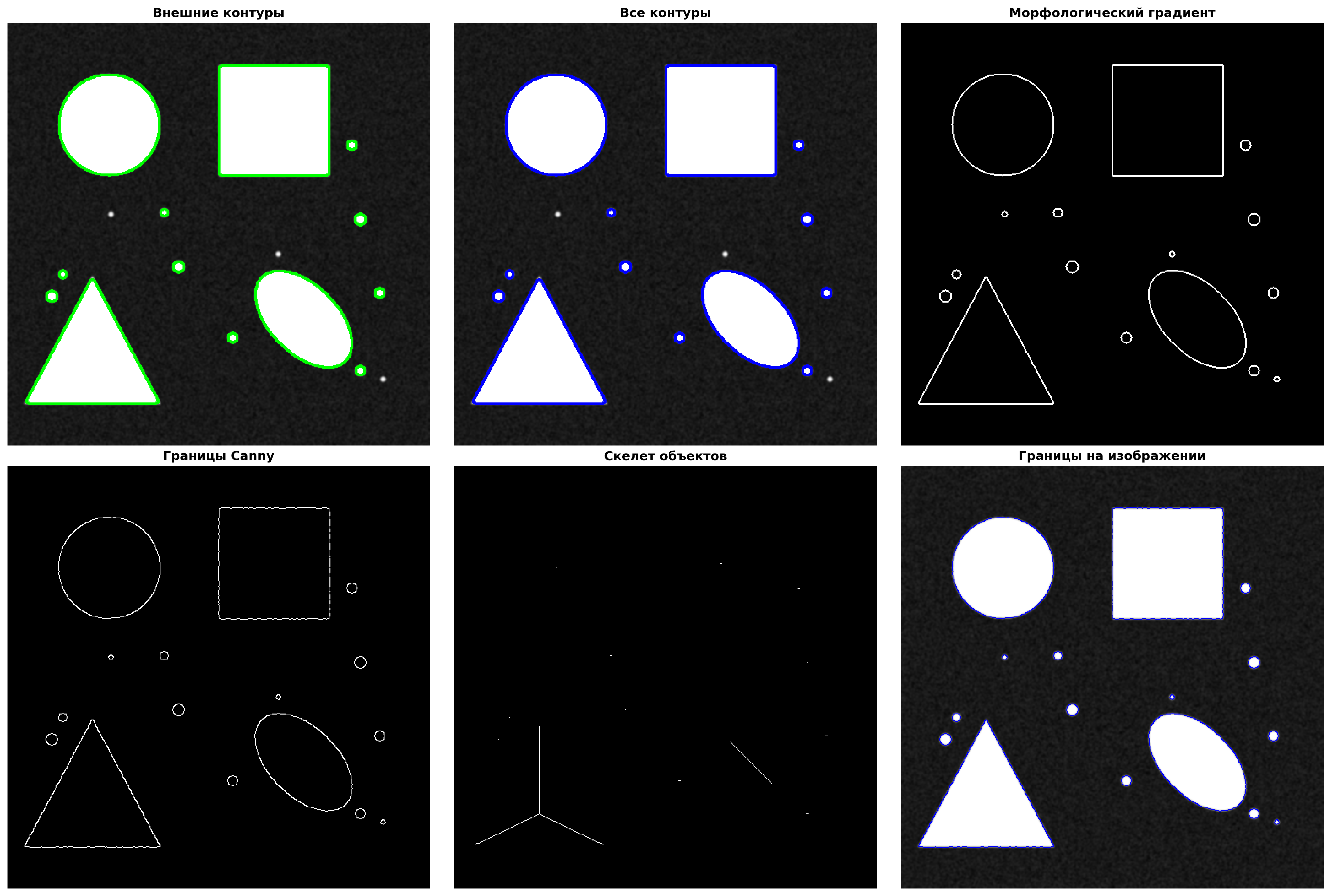


Рисунок 3 — Методы выделения границ

Реализованы различные методы выделения границ (рисунок 3). Обнаружено 13 внешних контуров. Морфологический градиент выделяет границы за счёт разности дилатации и эрозии. Детектор Canny обеспечивает тонкие, чёткие границы. Скелетонизация сводит объекты к одномерным линиям, сохраняя топологию. Наложение границ на исходное изображение демонстрирует точность детектирования контуров объектов.

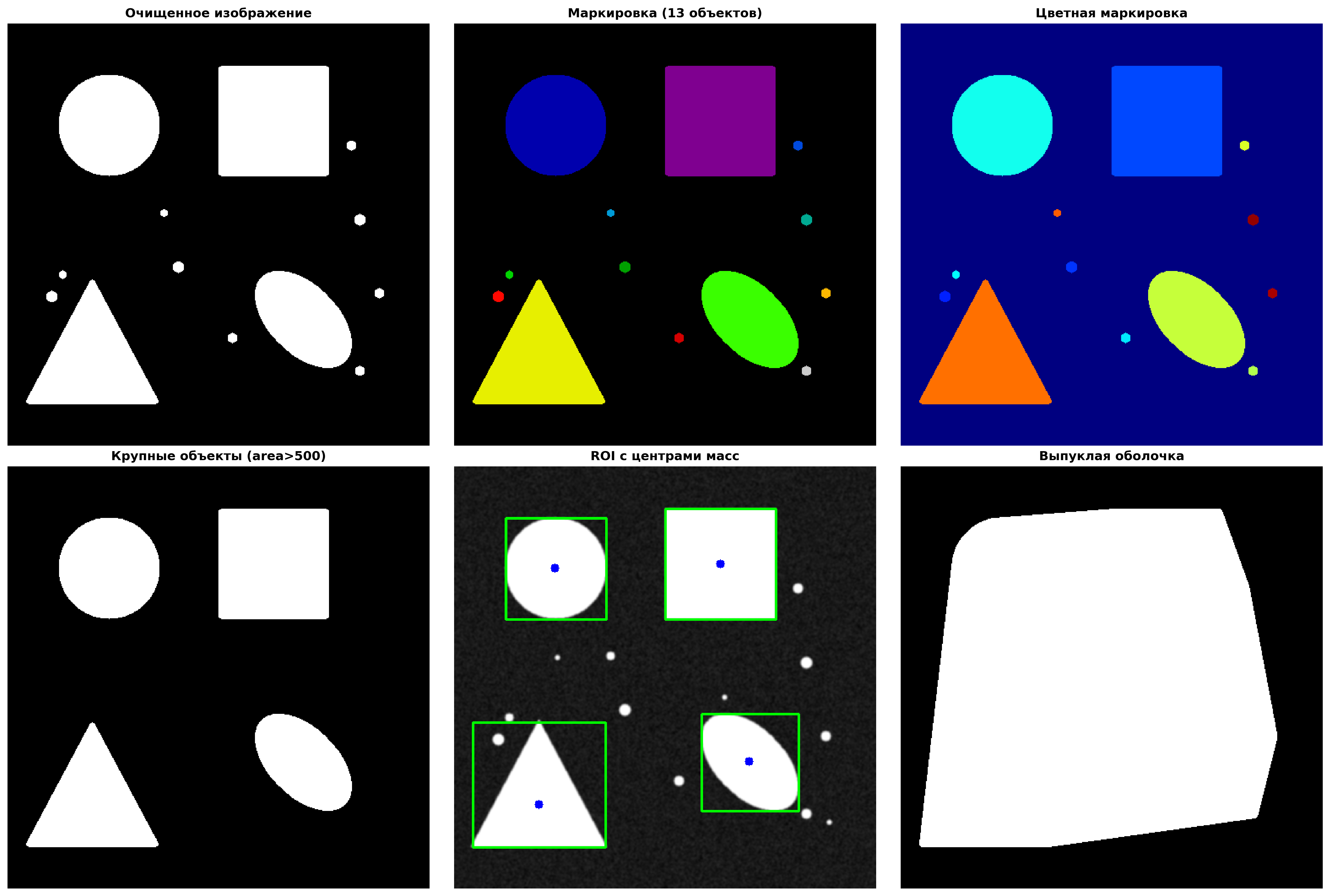


Рисунок 4 — Выделение областей интереса (ROI)

Проведена маркировка связных компонент (рисунок 4): выделено 13 объектов. Цветная маркировка позволяет визуально различать отдельные объекты. Выделены 4 крупных объекта (area > 500 пикселей): квадрат (площадь 17145), круг (11293), эллипс (8979) и треугольник (12207). Для каждого построены ограничивающие прямоугольники и отмечены центры масс. Выпуклая оболочка объединяет все объекты в единую область.

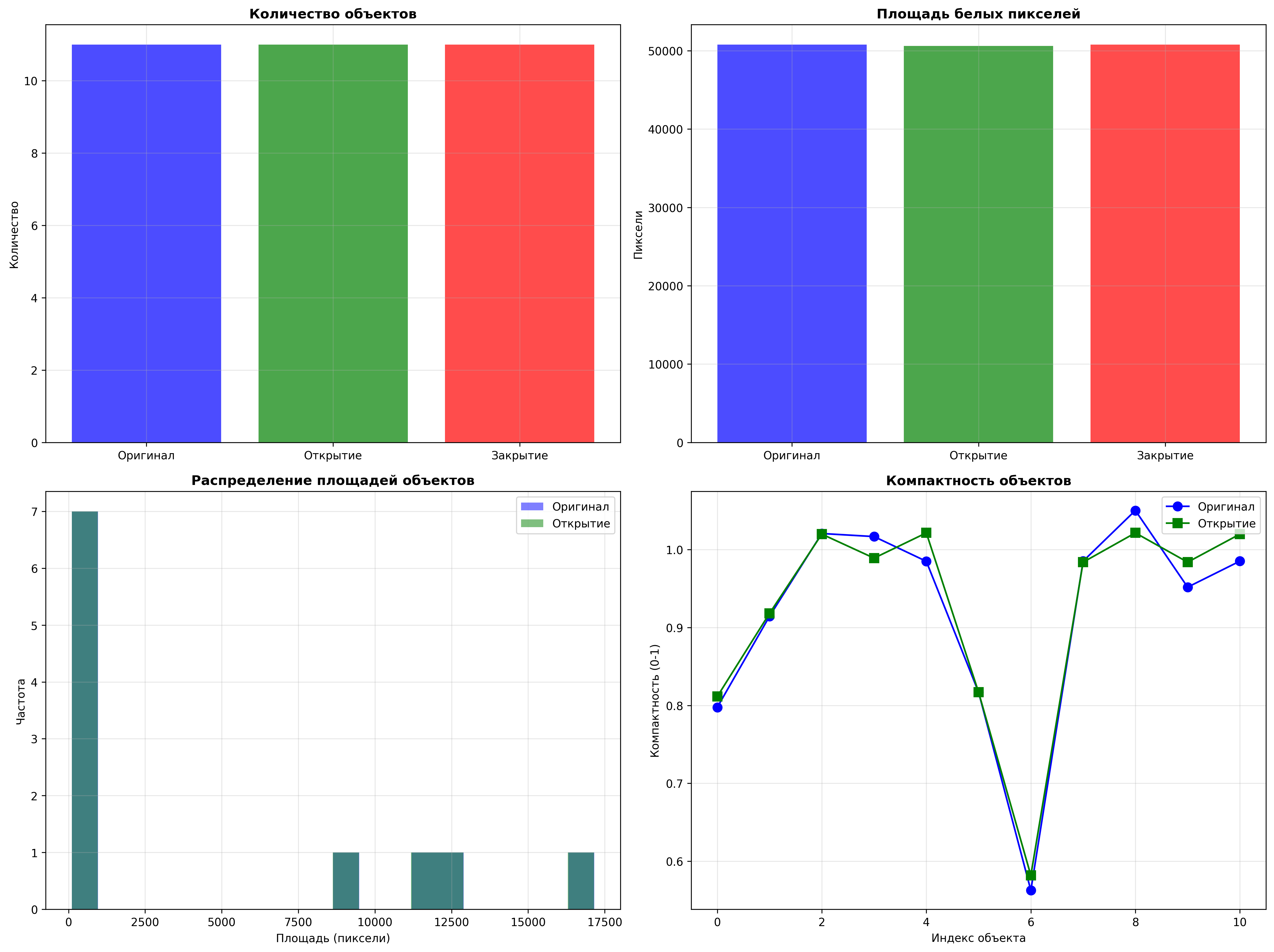


Рисунок 5 — Сравнительный анализ характеристик

Количественное сравнение (рисунок 5) показывает, что открытие незначительно уменьшает общую площадь объектов (50613 против 50798 пикселей) при сохранении количества объектов (11). Закрытие практически не изменяет характеристики. Распределение площадей смещается в сторону меньших значений после открытия. Средняя компактность объектов увеличивается с 0.917 до 0.924, что указывает на сглаживание контуров.

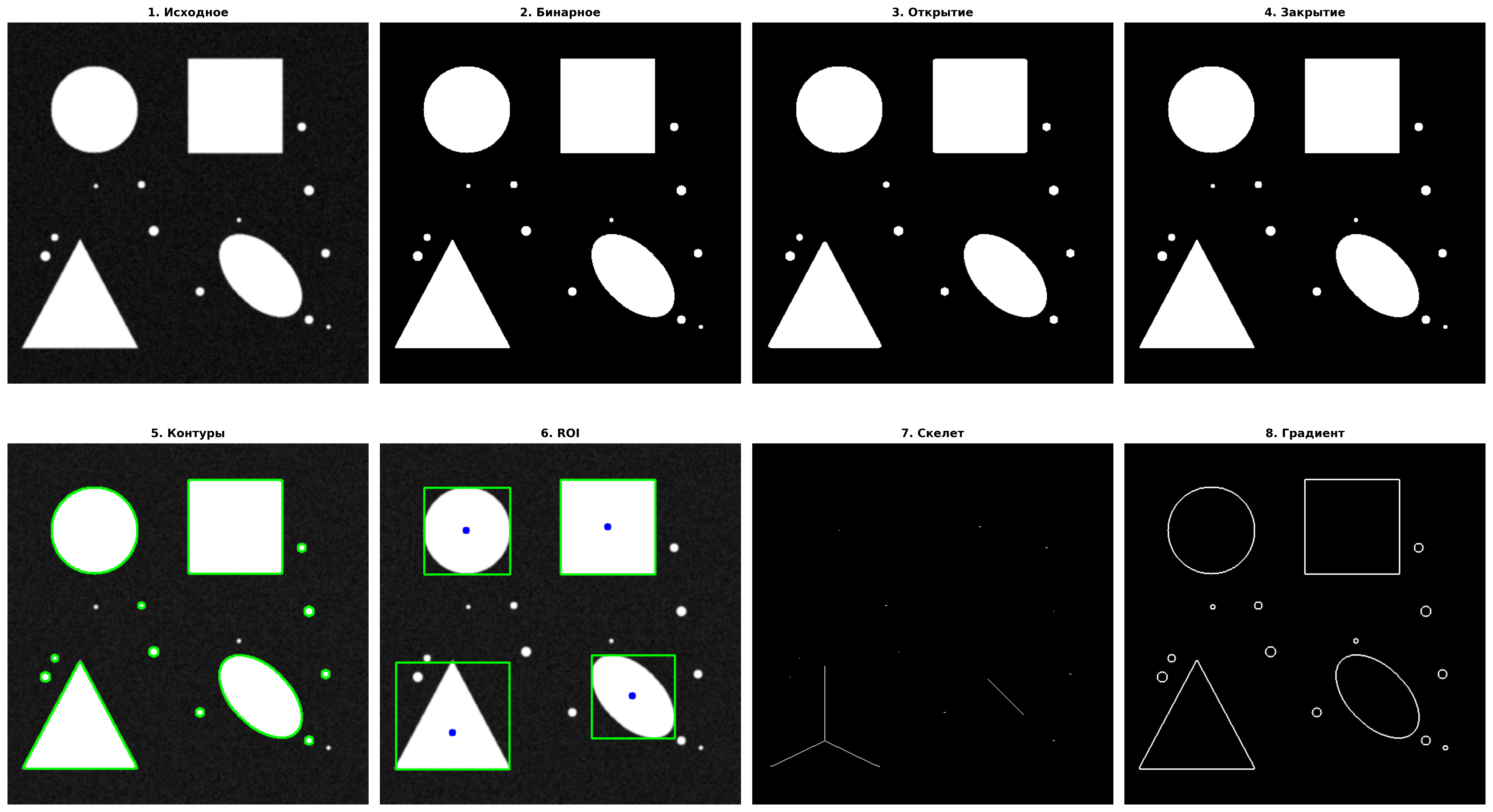


Рисунок 6 — Итоговая визуализация этапов обработки

Итоговая визуализация (рисунок 6) демонстрирует полный конвейер морфологического анализа: от исходного изображения через бинаризацию и морфологические операции до выделения ROI, построения скелета и морфологического градиента. Каждый этап решает специфическую задачу: очистка от шума, выделение объектов, определение границ, извлечение структурных признаков.

Выводы:

1. Освоены базовые морфологические операции: эрозия, дилатация, открытие, закрытие, top-hat, black-hat, морфологический градиент.
2. Установлено, что размер структурного элемента критически влияет на результат обработки.
3. Реализованы методы выделения границ и контуров объектов, включая скелетонизацию.
4. Выполнена маркировка связных компонент и выделение областей интереса (ROI) с построением ограничивающих прямоугольников.
5. Извлечены морфологические признаки: площадь, периметр, эксцентриситет, плотность, компактность, ориентация.
6. Проведён количественный анализ влияния морфологических фильтров на характеристики объектов.
7. Показано, что открытие эффективно удаляет шум, увеличивая среднюю компактность объектов с 0.917 до 0.924.
8. Все операции реализованы с использованием Python библиотек OpenCV и scikit-image.

Практическая значимость: освоенные методы морфологического анализа применяются в задачах сегментации медицинских изображений, распознавания символов, промышленного контроля качества, анализа спутниковых снимков и других областях компьютерного зрения.

# СПИСОК ЛИТЕРАТУРЫ

1. Soille, P. Morphological Image Analysis: Principles and Applications / P. Soille. — 2nd ed. — Berlin : Springer, 2003. — 391 p.

2. Serra, J. Image Analysis and Mathematical Morphology / J. Serra. — London : Academic Press, 1982. — 610 p.

3. Van der Walt, S. scikit-image: image processing in Python / S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias [et al.] // PeerJ. — 2014. — Vol. 2. — e453. — DOI: 10.7717/peerj.453.

4. Gonzalez, R. C. Digital Image Processing / R. C. Gonzalez, R. E. Woods. — 4th ed. — New York : Pearson, 2018. — Chapter 9: Morphological Image Processing. — P. 629–710.

ПРИЛОЖЕНИЕ А. ПРОГРАММНЫЙ КОД

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from scipy.stats import pearsonr  
from scipy.cluster import hierarchy  
import warnings  
  
warnings.filterwarnings('ignore')  
  
plt.rcParams['figure.figsize'] = (14, 10)  
plt.rcParams['font.size'] = 10  
sns.set\_style("whitegrid")  
  
  
*# ==================== ГЕНЕРАЦИЯ ДАННЫХ ====================*def generate\_xps\_data():  
 *"""Генерация примерных данных РФЭС"""* np.random.seed(42)  
 samples = []  
  
 modifier\_types = ['Без модификатора', 'C-модификатор', 'N-модификатор', 'S-модификатор']  
 electron\_doses = [0, 1e15, 5e15, 1e16, 5e16]  
  
 for modifier in modifier\_types:  
 for dose in electron\_doses:  
 base\_noise = np.random.normal(0, 0.5)  
 dose\_effect = dose / 1e16  
  
 sample = {  
 'Модификатор': modifier,  
 'Доза\_e\_см2': dose,

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А.**

'Pb\_4f7': 138.2 + base\_noise + dose\_effect \* 0.3,  
 'I\_3d5': 619.0 + base\_noise - dose\_effect \* 0.4,  
 'Cs\_3d5': 724.8 + base\_noise + dose\_effect \* 0.2,  
 'N\_1s': 400.2 + base\_noise if 'N' in modifier else 399.8 + base\_noise,  
 'C\_1s': 284.8 + base\_noise + (1.0 if 'C' in modifier else 0),  
 'O\_1s': 531.5 + base\_noise + dose\_effect \* 0.5,  
 'Pb\_I\_ratio': 1.5 - dose\_effect \* 0.3 + np.random.normal(0, 0.1),  
 'Organic\_%': (15 if modifier != 'Без модификатора' else 5) - dose\_effect \* 3,  
 'Degrad\_index': dose\_effect \* 10 + np.random.normal(0, 1),  
 'Roughness\_nm': 2.5 + dose\_effect \* 5 + np.random.normal(0, 0.5),  
 'Band\_gap\_eV': 1.6 - dose\_effect \* 0.15 + np.random.normal(0, 0.02),  
 }  
 samples.append(sample)  
  
 return pd.DataFrame(samples)  
  
  
*# ==================== ОСНОВНОЙ АНАЛИЗ ====================*print("=" \* 80)  
print("АНАЛИЗ ДАННЫХ РФЭС ПЕРОВСКИТОВ")  
print("=" \* 80)

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А.**

df = generate\_xps\_data()  
print(f"\n✓ Загружено {df.shape[0]} образцов, {df.shape[1]} параметров")  
  
*# Сохранение таблицы*df.to\_csv('outputs/xps\_data.csv', index=False)  
print("✓ Таблица: xps\_data.csv")  
  
*# ==================== ВИЗУАЛИЗАЦИЯ 1: Распределения ====================*fig, axes = plt.subplots(2, 3, figsize=(18, 12))  
fig.suptitle('Распределение энергий связи элементов (РФЭС)', fontsize=16, fontweight='bold')  
  
elements = ['Pb\_4f7', 'I\_3d5', 'Cs\_3d5', 'N\_1s', 'C\_1s', 'O\_1s']  
for ax, element in zip(axes.flat, elements):  
 for modifier in df['Модификатор'].unique():  
 data = df[df['Модификатор'] == modifier][element]  
 ax.hist(data, alpha=0.5, label=modifier, bins=8)  
 ax.set\_xlabel('Энергия связи (эВ)', fontsize=10)  
 ax.set\_ylabel('Частота', fontsize=10)  
 ax.set\_title(element, fontsize=12, fontweight='bold')  
 ax.legend(fontsize=8)  
 ax.grid(True, alpha=0.3)  
  
plt.tight\_layout()  
plt.savefig('outputs/fig1\_distributions.png', dpi=300, bbox\_inches='tight')  
print("✓ График 1: fig1\_distributions.png")  
plt.close()

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А.**

*# ==================== ВИЗУАЛИЗАЦИЯ 2: Влияние дозы ====================*fig, axes = plt.subplots(2, 2, figsize=(16, 12))  
fig.suptitle('Влияние дозы электронов на свойства перовскитов', fontsize=16, fontweight='bold')  
  
properties = [  
 ('Pb\_I\_ratio', 'Отношение Pb/I'),  
 ('Organic\_%', 'Органика (%)'),  
 ('Degrad\_index', 'Индекс деградации'),  
 ('Band\_gap\_eV', 'Ширина з.з. (эВ)')  
]  
  
for ax, (prop, label) in zip(axes.flat, properties):  
 for modifier in df['Модификатор'].unique():  
 subset = df[df['Модификатор'] == modifier]  
 ax.plot(subset['Доза\_e\_см2'], subset[prop], marker='o', label=modifier, linewidth=2, markersize=6)  
 ax.set\_xlabel('Доза электронов (см⁻²)', fontsize=11)  
 ax.set\_ylabel(label, fontsize=11)  
 ax.set\_title(label, fontsize=12, fontweight='bold')  
 ax.legend(fontsize=9)  
 ax.grid(True, alpha=0.3)  
 ax.ticklabel\_format(style='scientific', axis='x', scilimits=(0, 0))  
  
plt.tight\_layout()  
plt.savefig('outputs/fig2\_dose\_effects.png', dpi=300, bbox\_inches='tight')  
print("✓ График 2: fig2\_dose\_effects.png")  
plt.close()

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А.**

*# ==================== КОРРЕЛЯЦИЯ ПИРСОНА ====================*print("\n" + "=" \* 80)  
print("КОРРЕЛЯЦИОННЫЙ АНАЛИЗ")  
print("=" \* 80)  
  
corr\_data = df.select\_dtypes(include=[np.number])  
pearson\_corr = corr\_data.corr(method='pearson')  
  
*# Сохранение матрицы*pearson\_corr.to\_csv('outputs/pearson\_correlation.csv')  
print("\n✓ Матрица Пирсона: pearson\_correlation.csv")  
  
*# Визуализация*fig, ax = plt.subplots(figsize=(14, 12))  
mask = np.triu(np.ones\_like(pearson\_corr, dtype=bool), k=1)  
sns.heatmap(pearson\_corr, mask=mask, annot=True, fmt='.2f', cmap='coolwarm', center=0,  
 square=True, linewidths=1, cbar\_kws={"shrink": 0.8}, ax=ax, vmin=-1, vmax=1)  
ax.set\_title('Матрица корреляции Пирсона', fontsize=16, fontweight='bold', pad=20)  
plt.tight\_layout()  
plt.savefig('outputs/fig3\_pearson.png', dpi=300, bbox\_inches='tight')  
print("✓ График 3: fig3\_pearson.png")  
plt.close()

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А.**

*# ==================== P-VALUES ====================*def calculate\_pvalues(df):  
 cols = df.columns  
 p\_matrix = np.zeros((len(cols), len(cols)))  
 for i, col1 in enumerate(cols):  
 for j, col2 in enumerate(cols):  
 if i != j:  
 \_, p\_val = pearsonr(df[col1].dropna(), df[col2].dropna())  
 p\_matrix[i, j] = p\_val  
 return pd.DataFrame(p\_matrix, columns=cols, index=cols)  
  
  
pvalues = calculate\_pvalues(corr\_data)  
pvalues.to\_csv('outputs/pvalues.csv')  
print("✓ P-values: pvalues.csv")  
  
*# Визуализация*fig, ax = plt.subplots(figsize=(14, 12))  
mask = np.triu(np.ones\_like(pvalues, dtype=bool), k=1)  
sns.heatmap(pvalues, mask=mask, annot=True, fmt='.3f', cmap='RdYlGn\_r',  
 square=True, linewidths=1, ax=ax, vmin=0, vmax=0.05)  
ax.set\_title('Матрица значимости (p-values)', fontsize=16, fontweight='bold', pad=20)  
plt.tight\_layout()  
plt.savefig('outputs/fig4\_pvalues.png', dpi=300, bbox\_inches='tight')  
print("✓ График 4: fig4\_pvalues.png")  
plt.close()

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А.**

*# ==================== КЛАСТЕРНЫЙ АНАЛИЗ ====================*fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 8))  
  
linkage\_vars = hierarchy.linkage(pearson\_corr, method='ward')  
dendro = hierarchy.dendrogram(linkage\_vars, labels=pearson\_corr.columns,  
 ax=ax1, orientation='right', leaf\_font\_size=10)  
ax1.set\_title('Дендрограмма параметров', fontsize=14, fontweight='bold')  
ax1.set\_xlabel('Расстояние', fontsize=12)  
  
idx = dendro['leaves']  
corr\_ordered = pearson\_corr.iloc[idx, idx]  
sns.heatmap(corr\_ordered, annot=True, fmt='.2f', cmap='coolwarm', center=0,  
 square=True, linewidths=0.5, ax=ax2, vmin=-1, vmax=1)  
ax2.set\_title('Упорядоченная матрица корреляций', fontsize=14, fontweight='bold')  
  
plt.tight\_layout()  
plt.savefig('outputs/fig5\_cluster.png', dpi=300, bbox\_inches='tight')  
print("✓ График 5: fig5\_cluster.png")  
plt.close()  
  
*# ==================== PhiK КОРРЕЛЯЦИЯ ====================*try:  
 from phik import phik\_matrix, significance\_matrix

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А.**

phik\_corr = phik\_matrix(df)  
 phik\_corr.to\_csv('outputs/phik\_correlation.csv')  
 print("✓ PhiK матрица: phik\_correlation.csv")  
  
 fig, ax = plt.subplots(figsize=(14, 12))  
 mask = np.triu(np.ones\_like(phik\_corr, dtype=bool), k=1)  
 sns.heatmap(phik\_corr, mask=mask, annot=True, fmt='.2f', cmap='plasma',  
 square=True, linewidths=1, ax=ax, vmin=0, vmax=1, annot\_kws={'size': 8})  
 ax.set\_title('Матрица PhiK корреляции', fontsize=16, fontweight='bold', pad=20)  
 plt.tight\_layout()  
 plt.savefig('outputs/fig6\_phik.png', dpi=300, bbox\_inches='tight')  
 print("✓ График 6: fig6\_phik.png")  
 plt.close()  
  
 *# Сравнение Pearson vs PhiK* fig, axes = plt.subplots(1, 2, figsize=(24, 10))  
  
 mask = np.triu(np.ones\_like(pearson\_corr, dtype=bool), k=1)  
 sns.heatmap(pearson\_corr, mask=mask, annot=True, fmt='.2f', cmap='coolwarm', center=0,  
 square=True, linewidths=1, ax=axes[0], vmin=-1, vmax=1, annot\_kws={'size': 7})  
 axes[0].set\_title('Pearson (линейные)', fontsize=14, fontweight='bold')

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А.**

common\_cols = list(set(pearson\_corr.columns) & set(phik\_corr.columns))  
 phik\_aligned = phik\_corr.loc[common\_cols, common\_cols]  
 mask\_phik = np.triu(np.ones\_like(phik\_aligned, dtype=bool), k=1)  
 sns.heatmap(phik\_aligned, mask=mask\_phik, annot=True, fmt='.2f', cmap='viridis',  
 square=True, linewidths=1, ax=axes[1], vmin=0, vmax=1, annot\_kws={'size': 7})  
 axes[1].set\_title('PhiK (линейные + нелинейные)', fontsize=14, fontweight='bold')  
  
 plt.tight\_layout()  
 plt.savefig('outputs/fig7\_comparison.png', dpi=300, bbox\_inches='tight')  
 print("✓ График 7: fig7\_comparison.png")  
 plt.close()  
  
except ImportError:  
 print("\n⚠ PhiK не установлен. Установите: pip install phik")  
  
*# ==================== SCATTER PLOTS ====================*fig, axes = plt.subplots(2, 2, figsize=(16, 14))  
fig.suptitle('Диаграммы рассеяния ключевых зависимостей', fontsize=16, fontweight='bold')

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А.**

scatter\_pairs = [  
 ('Доза\_e\_см2', 'Degrad\_index'),  
 ('Pb\_I\_ratio', 'Degrad\_index'),  
 ('Organic\_%', 'Band\_gap\_eV'),  
 ('Доза\_e\_см2', 'Pb\_4f7')  
]  
  
for ax, (x\_var, y\_var) in zip(axes.flat, scatter\_pairs):  
 for modifier in df['Модификатор'].unique():  
 subset = df[df['Модификатор'] == modifier]  
 ax.scatter(subset[x\_var], subset[y\_var], label=modifier, alpha=0.7, s=80,  
 edgecolors='black', linewidth=0.5)  
  
 z = np.polyfit(df[x\_var], df[y\_var], 1)  
 p = np.poly1d(z)  
 x\_trend = np.linspace(df[x\_var].min(), df[x\_var].max(), 100)  
 ax.plot(x\_trend, p(x\_trend), "r--", alpha=0.5, linewidth=2, label='Тренд')  
  
 r, p\_val = pearsonr(df[x\_var], df[y\_var])  
 ax.set\_xlabel(x\_var, fontsize=11)  
 ax.set\_ylabel(y\_var, fontsize=11)  
 ax.set\_title(f'{x\_var} vs {y\_var}\nr={r:.3f}, p={p\_val:.4f}', fontsize=11, fontweight='bold')  
 ax.legend(fontsize=8)  
 ax.grid(True, alpha=0.3)

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А.**

plt.tight\_layout()  
plt.savefig('outputs/fig8\_scatter.png', dpi=300, bbox\_inches='tight')  
print("✓ График 8: fig8\_scatter.png")  
plt.close()  
  
*# ==================== СИЛЬНЫЕ КОРРЕЛЯЦИИ ====================*print("\n" + "=" \* 80)  
print("СИЛЬНЫЕ КОРРЕЛЯЦИИ (|r| > 0.7)")  
print("=" \* 80)  
  
strong\_corr = []  
for i in range(len(pearson\_corr.columns)):  
 for j in range(i + 1, len(pearson\_corr.columns)):  
 corr\_val = pearson\_corr.iloc[i, j]  
 if abs(corr\_val) > 0.7:  
 strong\_corr.append({  
 'Переменная 1': pearson\_corr.columns[i],  
 'Переменная 2': pearson\_corr.columns[j],  
 'r': corr\_val,  
 'p-value': pvalues.iloc[i, j]  
 })  
  
if strong\_corr:  
 strong\_df = pd.DataFrame(strong\_corr).sort\_values('r', key=abs, ascending=False)  
 print(strong\_df.head(15).to\_string(index=False))  
 strong\_df.to\_csv('outputs/strong\_correlations.csv', index=False)

**ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А.**

print("\n✓ Таблица: strong\_correlations.csv")  
  
*# ==================== ИТОГ ====================*print("\n" + "=" \* 80)  
print("АНАЛИЗ ЗАВЕРШЕН")  
print("=" \* 80)  
print("\n📁 Созданные файлы:")  
print(" • xps\_data.csv - исходные данные")  
print(" • pearson\_correlation.csv - матрица Пирсона")  
print(" • pvalues.csv - статистическая значимость")  
print(" • phik\_correlation.csv - матрица PhiK")  
print(" • strong\_correlations.csv - сильные корреляции")  
print("\n📊 Графики:")  
print(" • fig1\_distributions.png - распределения энергий")  
print(" • fig2\_dose\_effects.png - влияние дозы")  
print(" • fig3\_pearson.png - корреляция Пирсона")  
print(" • fig4\_pvalues.png - значимость")  
print(" • fig5\_cluster.png - кластерный анализ")  
print(" • fig6\_phik.png - PhiK корреляция")  
print(" • fig7\_comparison.png - сравнение методов")  
print(" • fig8\_scatter.png - диаграммы рассеяния")  
print("\n" + "=" \* 80)