Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Российский химико-технологический университет имени Д.И. Менделеева»
Кафедра информационных компьютерных технологий

ОТЧЕТ ПО РАБОТЕ

Обучение модели машинного обучения для обнаружения частиц вещества на снимках, полученных с помощью электронного микроскопа.

Выполнили студенты группы КС-33: Чахалиди В.Э.
Вагенлейтнер Н.С.
Вельмакина Д.Е.
Динмухаметов Г.В.

Принял: к.т.н. доцент Зубов Д.В.

Москва 2025

Оглавление

Описание задачи.

Целью данной работы является автоматическое обнаружение и измерение частиц вещества на изображениях, полученных с помощью электронного микроскопа. В частности, задача заключается в извлечении информации о количестве и размере частиц (в нанометрах), что важно для анализа структуры материалов в нанотехнологиях, материаловедении и смежных дисциплинах.

Описание метода/модели.

Реализованный метод включает следующие основные этапы:

Загрузка и предобработка изображений

Изображения загружаются из заданной директории. Алгоритм работает с форматами .jpg, .png, .tif. На этом этапе также осуществляется преобразование изображения в оттенки серого для дальнейшей обработки.

Определение масштаба изображения (нм/пиксель)

В нижней части микроскопических снимков присутствует шкала, длина которой известна (например, 50 или 100 нм).

Из этой области:

1. вырезается прямоугольник, содержащий шкалу;
2. применяется оператор Canny для выделения границ;
3. через преобразование Хафа определяется самая длинная линия, принимаемая за масштаб;
4. рассчитывается отношение длины этой линии (в пикселях) к её реальной длине (в нанометрах), что даёт масштаб: **нм/пиксель**.

Предобработка изображения для сегментации

После сглаживания (Gaussian Blur) и бинаризации (Otsu thresholding), изображение проходит морфологическую очистку (открытие), что удаляет мелкий шум и соединяет разрозненные участки частиц.

Сегментация методом водораздела (Watershed)

Метод водораздела применяется для разделения частиц, сливающихся друг с другом:

1. выделяются области "уверенного" переднего и заднего плана;
2. промежуточные области считаются неизвестными;
3. применяется Watershed-алгоритм, который размечает каждую частицу отдельной меткой.

Измерение частиц

Для каждой размеченной частицы:

1. находится минимальная описывающая окружность;

2. рассчитывается её диаметр в пикселях и затем пересчитывается в нанометры с использованием ранее найденного масштаба;
3. размеры сохраняются, а частицы визуализируются на изображении.

Сохранение результатов

1. Результирующие изображения сохраняются в выходной директории.
2. Строится и сохраняется гистограмма распределения диаметров частиц.
3. Основные статистики (имя файла, количество частиц, средний диаметр) записываются в CSV-таблицу.

Выполнение задачи.

Для реализации использовался язык Python

```
# === Импорт библиотек ===

import cv2  # OpenCV — для обработки изображений
import numpy as np  # NumPy — для работы с массивами и математикой
import matplotlib.pyplot as plt  # Matplotlib — для построения гистограмм
import pandas as pd  # pandas — для сохранения данных в CSV
import glob  # glob — для поиска файлов по маске
import os  # os — для работы с файловой системой
from tqdm import tqdm  # tqdm — для отображения прогресса обработки

# === Папки ===
image_dir = "dataset"  # Папка с входными изображениями
output_dir = "output"  # Папка для выходных данных
os.makedirs(output_dir, exist_ok=True)  # Создаём папку, если её нет
results_file = os.path.join(output_dir, "results.csv")  # Файл для сохранения результатов

# === Индивидуальные масштабы изображений (в нанометрах) ===
custom_scales = {
    "Image7.jpg": 100,
    "Image8.jpg": 100,
    "Image9.jpg": 50,
    "Image10.jpg": 50,
    "Image811.jpg": 50,
}
```

```
    # === Функция для получения путей ко всем изображениям ===
    def get_image_paths(folder):
        image_paths = []
        for ext in ["*.jpg", "*.png", "*.tif"]:
            image_paths.extend(glob.glob(os.path.join(folder, ext)))
        return image_paths

    # === Получить масштаб для конкретного изображения ===
    def get_scale_for_image(filename):
        return custom_scales.get(filename)

    # === Определение масштаба (нм/пиксель) по шкале в правом нижнем углу ===
    def extract_scale(gray_img, nm_in_scale):
        scale_region = gray_img[-50:, -150:]  # Вырезаем область шкалы
        edges = cv2.Canny(scale_region, 50, 150)  # Поиск границ (Canny)
        lines = cv2.HoughLinesP(edges, 1, np.pi / 180, 30, minLineLength=20, maxLineGap=5)  # Поиск прямых (Хафа)

        if lines is not None:
            # Выбираем самую длинную прямую как шкалу
            longest = max(lines, key=lambda l: np.linalg.norm([l[0][0] - l[0][2], l[0][1] - l[0][3]]))
            x1, y1, x2, y2 = longest[0]
            pixel_len = np.linalg.norm([x1 - x2, y1 - y2])
            return nm_in_scale / pixel_len  # Возвращаем нм/пиксель
        else:
            return None  # Если шкала не найдена

    # === Предобработка изображения ===
    def preprocess_image(gray_img):
        blur = cv2.GaussianBlur(gray_img, (5, 5), 0)  # Сглаживание шума
        _, thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)  # Бинаризация (Otsu)
        kernel = np.ones((3, 3), np.uint8)
        opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=2)  # Удаление мелких шумов
```

```
    return opening

# === Сегментация методом водораздела (Watershed) ===
def segment_watershed(original_img, opening):
    kernel = np.ones((3, 3), np.uint8)
    sure_bg = cv2.dilate(opening, kernel, iterations=3)  # Явный фон
    dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)  # Расстояние до фона
    _, sure_fg = cv2.threshold(dist_transform, 0.4 * dist_transform.max(), 255, 0)  # Явный передний план
    sure_fg = np.uint8(sure_fg)
    unknown = cv2.subtract(sure_bg, sure_fg)  # Неизвестная область
    _, markers = cv2.connectedComponents(sure_fg)  # Маркировка
    markers = markers + 1
    markers[unknown == 255] = 0  # Устанавливаем маркер для неизвестной области

    return cv2.watershed(original_img, markers)  # Выполняем водораздел

# === Анализ частиц: подсчёт и измерение диаметров ===
def analyze_particles(markers, nm_per_pixel, original_img):
    diameters = []
    result_img = original_img.copy()

    for label in range(2, markers.max() + 1):  # Пропускаем фон и границы
        mask = np.uint8(markers == label)
        cnts, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

        for cnt in cnts:
            area = cv2.contourArea(cnt)
            if area < 10:
                continue  # Отбрасываем шум

            (x, y), radius = cv2.minEnclosingCircle(cnt)
            diameter = 2 * radius * nm_per_pixel
            diameters.append(diameter)
            cv2.circle(result_img, (int(x), int(y)), int(radius), (0, 255, 0), 1)  # Визуализация
```

```
    return diameters, result_img

# === Сохранение гистограммы распределения диаметров ===
def save_histogram(diameters, filename):
    plt.figure()
    plt.hist(diameters, bins=20)
    plt.xlabel("Диаметр (нм)")
    plt.ylabel("Количество частиц")
    plt.title(filename)
    plt.savefig(os.path.join(output_dir, f"{filename}_hist.png"))
    plt.close()

# === Сохранение результатов в таблицу CSV ===
def save_results_to_csv(filename, diameters):
    df = pd.DataFrame({
        "filename": [filename],
        "count": [len(diameters)],
        "mean_diameter_nm": [np.mean(diameters) if diameters else 0]
    })
    df.to_csv(results_file, mode='a', header=not os.path.exists(results_file), index=False)

# === Главный цикл обработки изображений ===
image_paths = get_image_paths(image_dir)
print(f"Найдено изображений: {len(image_paths)}")

for path in tqdm(image_paths, desc="Обработка изображений"):
    filename = os.path.basename(path)
    print(f"\n📷 Обработка файла: {filename}")

    img = cv2.imread(path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Получаем масштаб
    nm_in_scale = get_scale_for_image(filename)
    if nm_in_scale is None:
        print("⚠️ Нет масштаба для этого изображения, пропущено.")
```

```
        continue

    # Определяем масштаб в нм/пиксель
    nm_per_pixel = extract_scale(gray, nm_in_scale)
    if nm_per_pixel is None:
        print("⚠️ Не удалось определить шкалу.")
        continue

    # Обработка и анализ
    opening = preprocess_image(gray)
    markers = segment_watershed(img, opening)
    diameters, result_img = analyze_particles(markers, nm_per_pixel, img)

    # Вывод результатов
    print(f"🔍 Найдено частиц: {len(diameters)}")
    print(f"📏 Средний диаметр: {np.mean(diameters):.1f} нм")

    # Сохранение результатов
    cv2.imwrite(os.path.join(output_dir, filename), result_img)
    save_histogram(diameters, filename)
    save_results_to_csv(filename, diameters)
```

Вывод программы

```
Найдено изображений: 5
Обработка изображений:   0%|0/5 [00:00<?, ?it/s]
📷 Обработка файла: Image10.jpg
🔍 Найдено частиц: 7
📏 Средний диаметр: 195.5 нм
Обработка изображений:  20%|██| 1/5 [00:00<00:03, 1.05it/s]
📷 Обработка файла: Image7.jpg
🔍 Найдено частиц: 54
📏 Средний диаметр: 144.2 нм
Обработка изображений:
40%|████| 2/5 [00:02<00:03,  1.19s/it]
📷 Обработка файла: Image8.jpg
```

🔍 Найдено частиц: 50

📏 Средний диаметр: 91.9 нм

Обработка изображений: 60%|[redacted]| 3/5 [00:03<00:02, 1.14s/it]

📷 Обработка файла: Image811.jpg

🔍 Найдено частиц: 10

📏 Средний диаметр: 46.2 нм

Обработка изображений: 80%|[redacted]| 4/5 [00:03<00:00, 1.26it/s]

📷 Обработка файла: Image9.jpg

🔍 Найдено частиц: 26

📏 Средний диаметр: 68.3 нм

Обработка изображений: 100%|[redacted]5/5 [00:04<00:00, 1.12it/s]

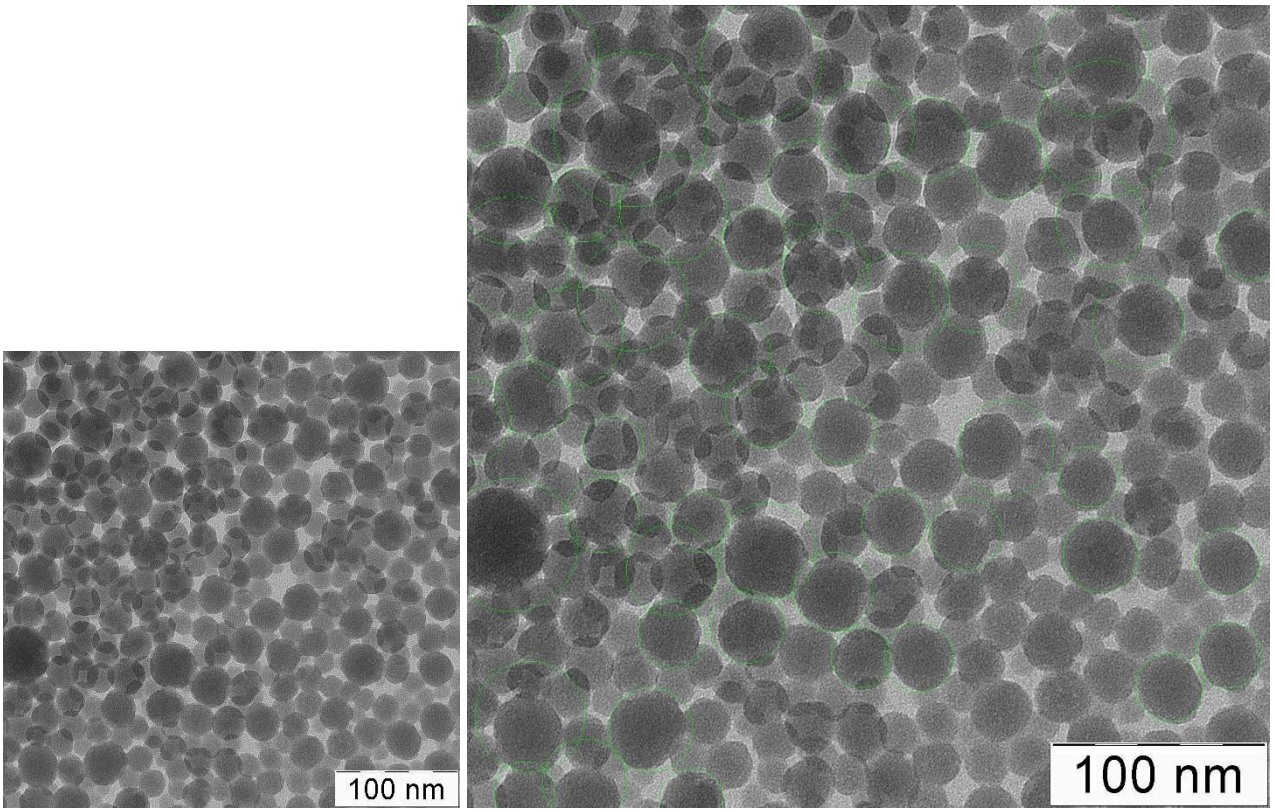

Рис. 1 – Результаты для изображения Image7.jpg

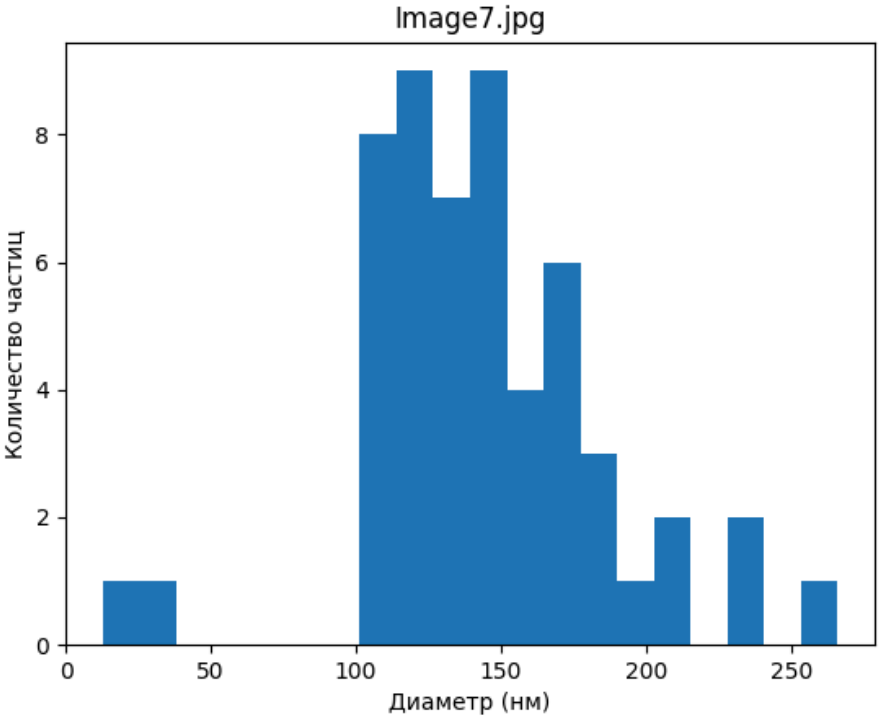
Image7.jpg
Количество частиц
0
2
4
6
8
0
50
100
150
200
250
Диаметр (нм)

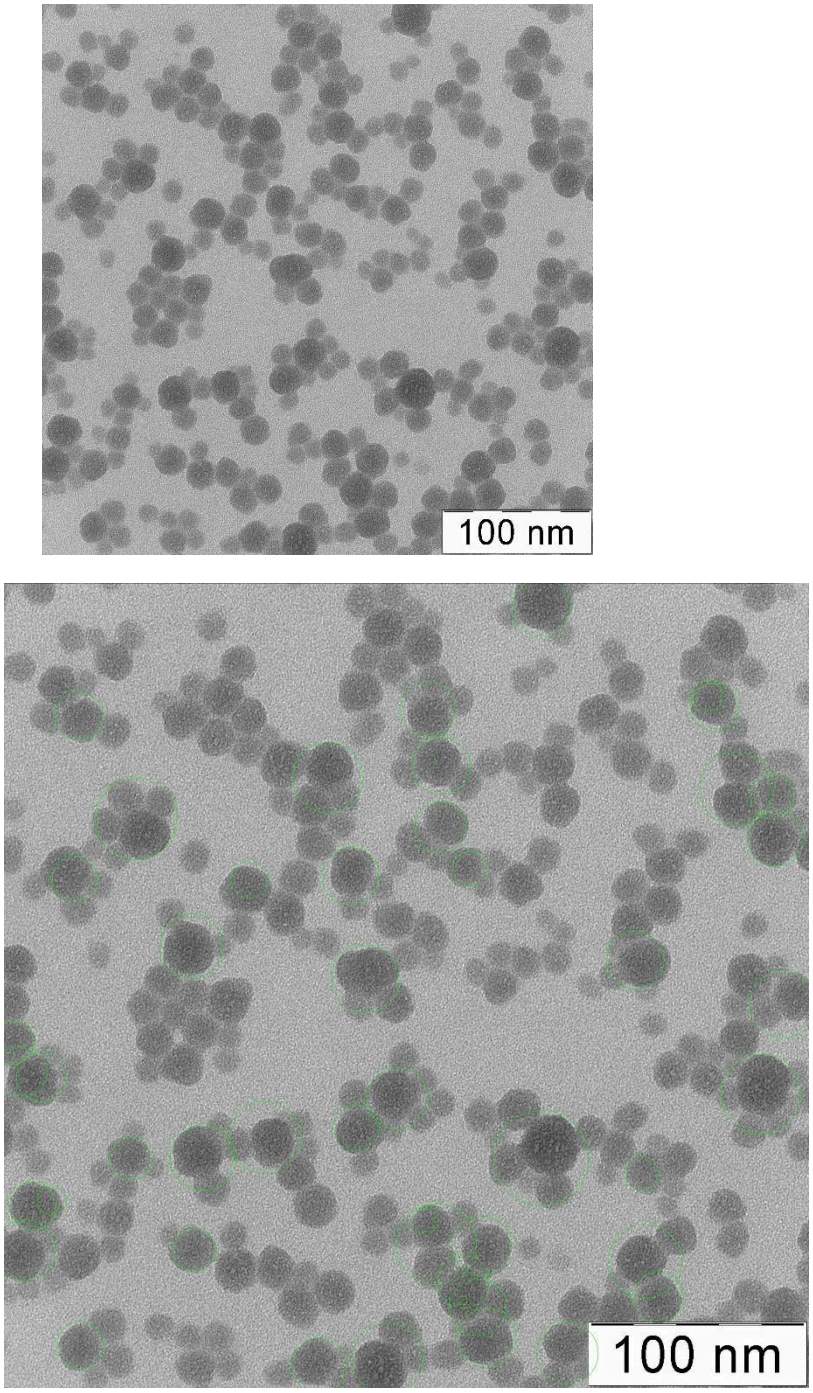

Рис. 2 – Результаты для изображения Image8.jpg

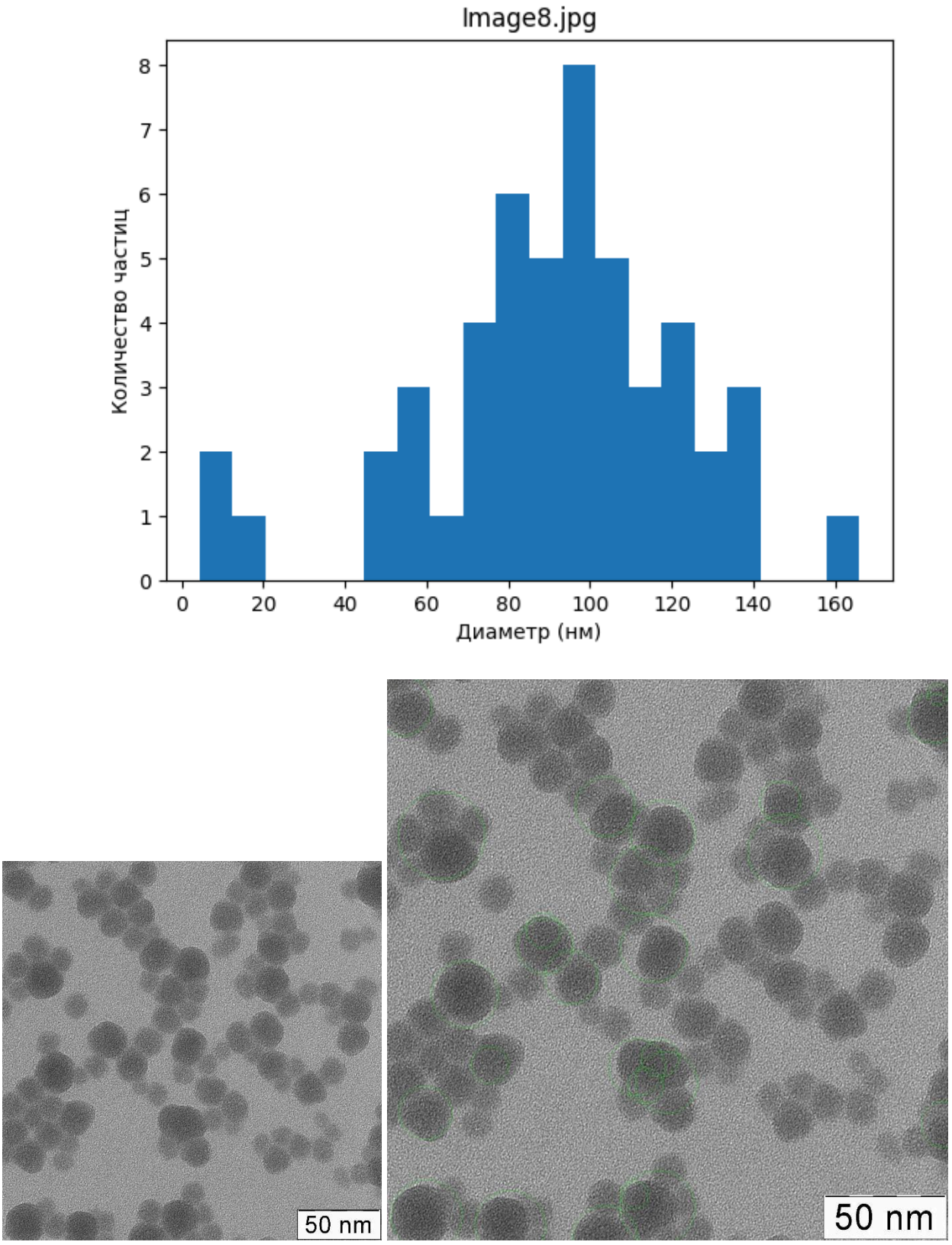

Рис. 3 – Результаты для изображения Image9.jpg

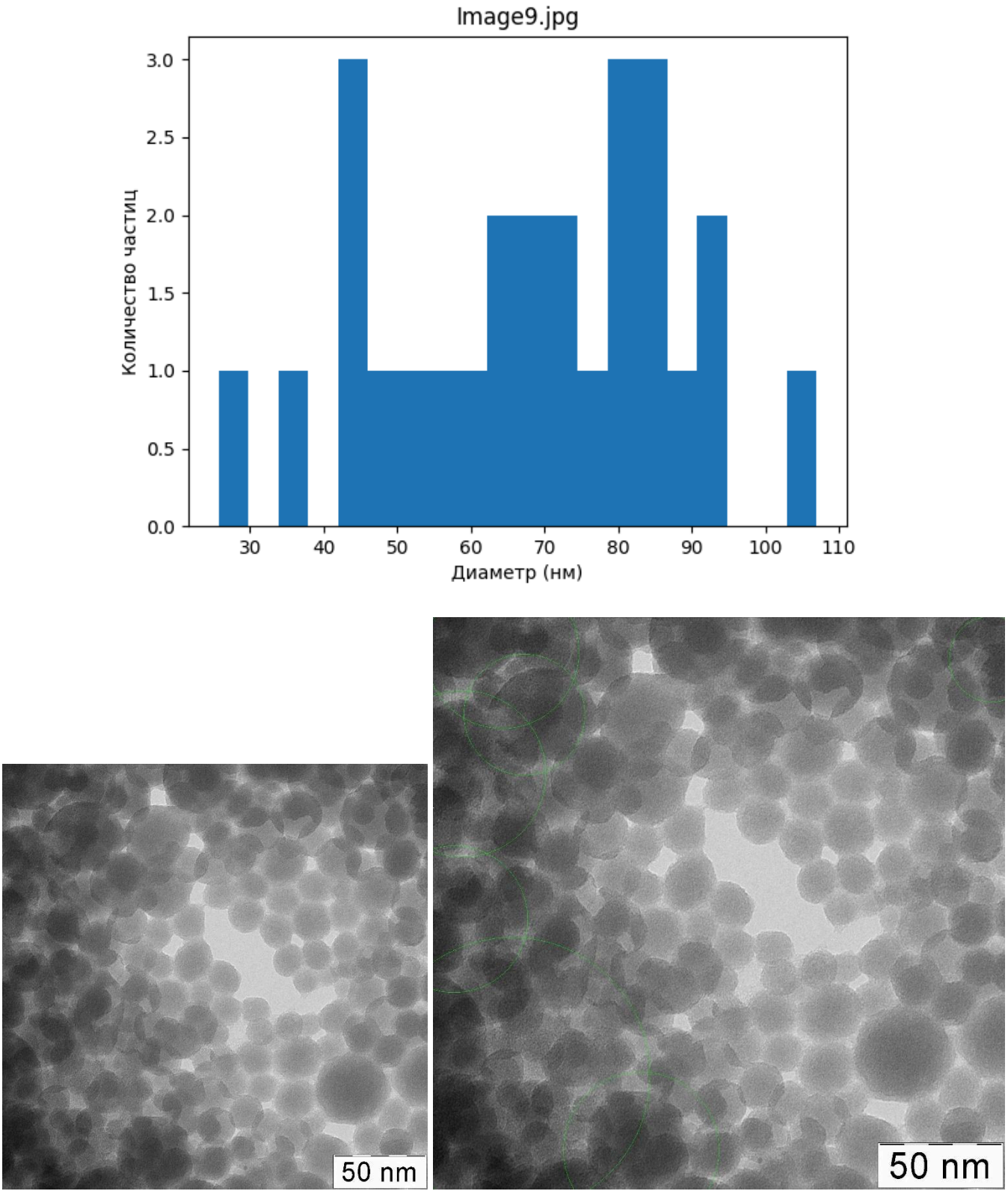

Рис. 4 – Результаты для изображения Image10.jpg

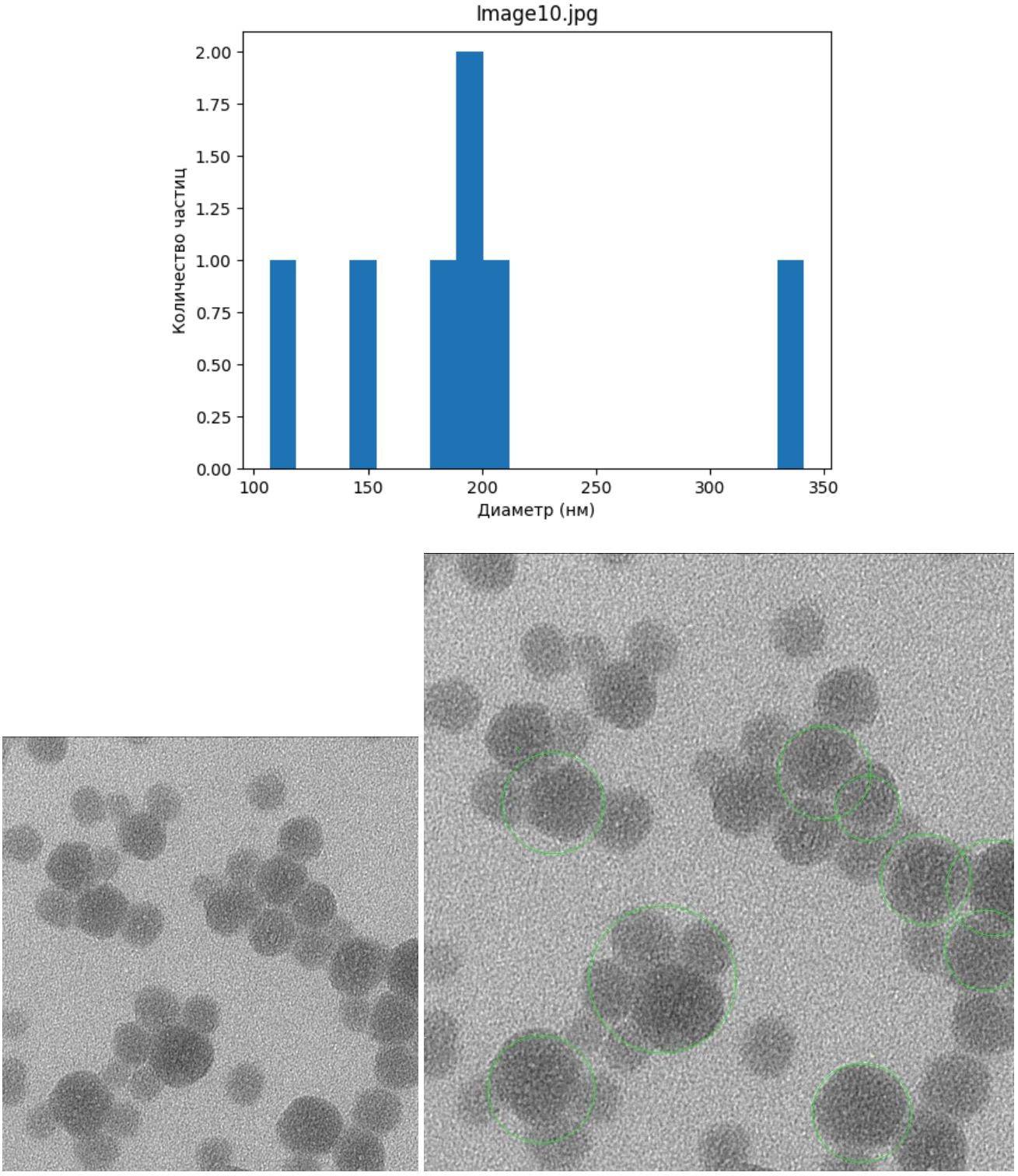

Рис. 5 – Результаты для изображения Image811.jpg

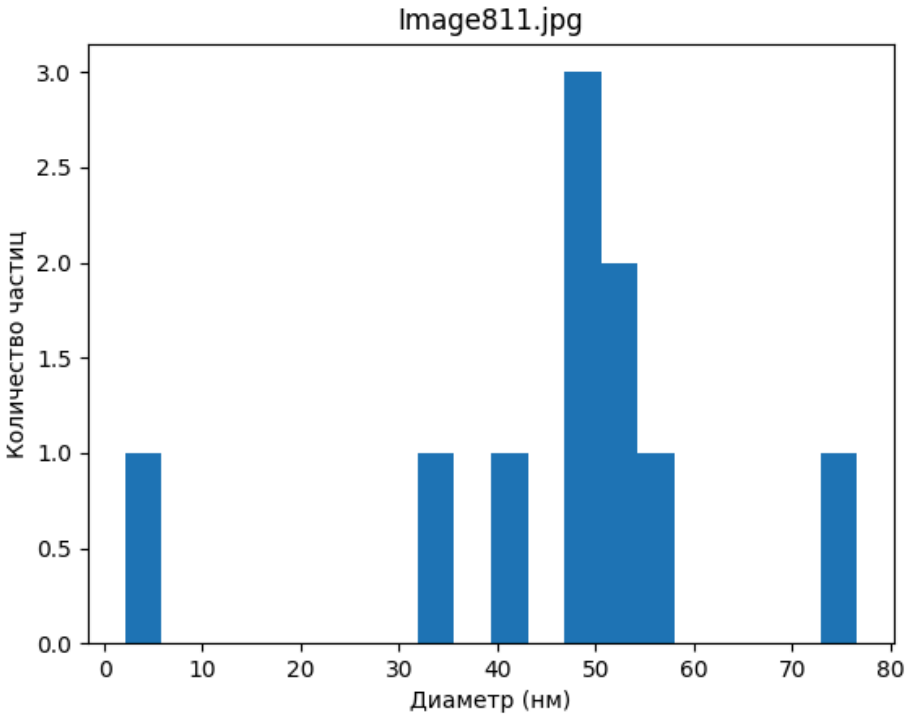
Image811.jpg
Количество частиц
Диаметр (нм)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
10
20
30
40
50
60
70
80

Заключение.

Разработанный метод автоматического анализа изображений электронного микроскопа позволяет выполнять базовую сегментацию частиц, определять их размеры и собирать статистику по их распределению. Он использует ручные эвристики, такие как определение масштаба по шкале на изображении, морфологическую обработку и метод водораздела для разделения частиц.

Однако метод **имеет ряд существенных недостатков**:

1. Он **ошибочно определяет границы частиц**, особенно в случае, если объекты перекрываются, имеют низкую контрастность или нестандартную форму.
2. Алгоритм **неустойчив к шуму** и артефактам на изображении, что приводит к **ложным срабатываниям** или **пропущенным частицам**.
3. Расчёт диаметров часто **некорректен**, так как основан на минимальных описывающих окружностях, что не всегда соответствует реальной форме и размеру частиц.
4. Подход **не учитывает контекст или структуру частиц**, из-за чего сложные случаи анализируются неправильно.

В результате **данная методика не обеспечивает необходимой точности и надёжности для подготовки данных или обучения моделей машинного обучения**. Для построения устойчивого ML-решения необходимо использовать размеченные датасеты и современные методы компьютерного зрения, основанные на глубоких нейросетях (например, U-Net, Mask R-CNN), способных адаптироваться к особенностям изображений и минимизировать ошибки сегментации.