Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ, информационные технологии»

ДОМАШНЯЯ РАБОТА №2

«Исследование методов символического описания объектов»

ДИСЦИПЛИНА: «Программные системы распознавания и обработки информации»

Выполнил: студент гр. ИУК4-31М _____ (Сафронов Н.С,)
(подпись) (Ф.И.О.)

Проверил: _____ (Гагарин Ю.Е.)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2025

Цель:

Формирование практических навыков исследования одного из методов символического описания объектов, в котором прямолинейные и криволинейные отрезки образуют основную структуру объекта.

Задачи:

1. Определение параметров нормального уравнения прямых, проходящих через множество исходных точек;

2. Построение гистограмм для параметров нормального уравнения прямой.

3. Построение прямых, проходящих через наибольшее количество исходных точек, определяющих объект.

Задание

Вариант 6

1. Задать случайным образом n точек $(x_i y_i)$ из заданного диапазона;

2. Построить график расположения точек $(x_i y_i)$ на плоскости;

3. Определить возможный диапазон значений параметров нормального уравнения прямой $\rho = xcos\theta + ysin\ \theta$, проходящей через набор исходных точек $(x_i y_i)$;

4. Задавая значения параметров (ρ, θ) из определенного диапазона, построить одномерные гистограммы распределения параметров (ρ, θ) от количества точек k, удовлетворяющих уравнению прямой;

5. Построить двумерную гистограмму распределения параметров (ρ, θ) от количества точек k и определить два максимальных значения этой гистограммы;

6. На графике расположения точек $(x_i y_i)$ на плоскости построить две прямые, параметры которых соответствуют максимальным значениям k, полученным в п.5.

Диапазон значений исходных данных $(x_i y_i)$: $x \in [10, 90], y \in [-80, -10]$, $n = 70$.

Результаты выполнения работы

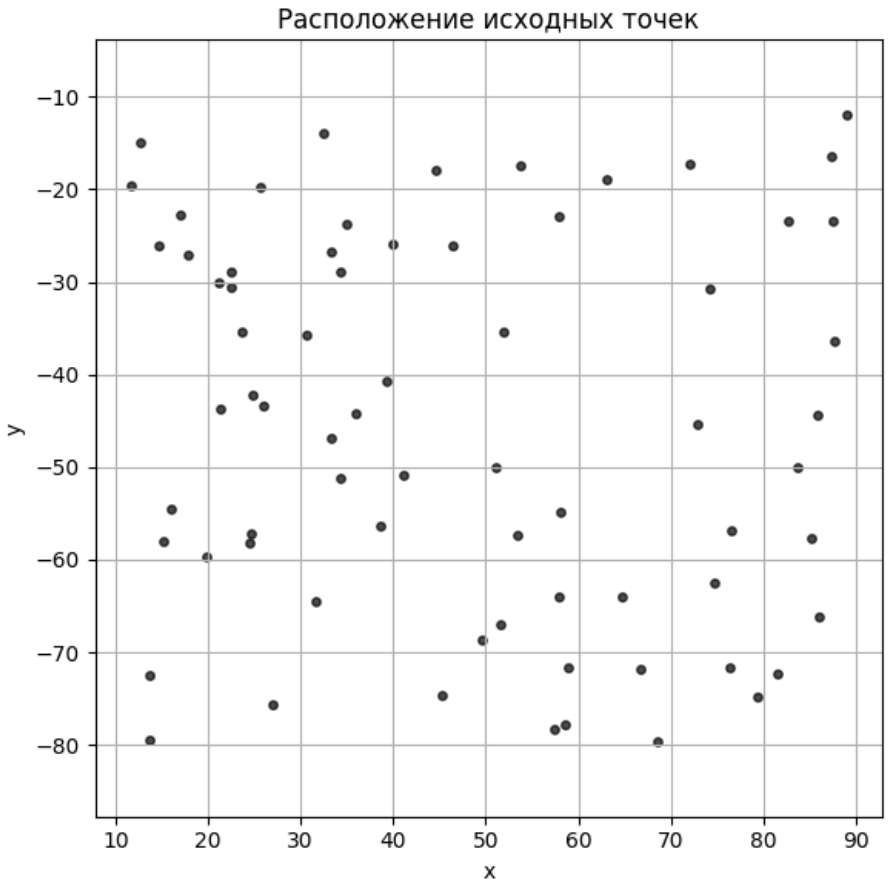

Рисунок 1 – Расположение исходных точек

```
Диапазон θ: (0, 3.141592653589793)
Диапазон ρ: (np.float64(-89.75343500943784), np.float64(89.75343500943784))
```

Рисунок 2 – Диапазоны ρ и θ

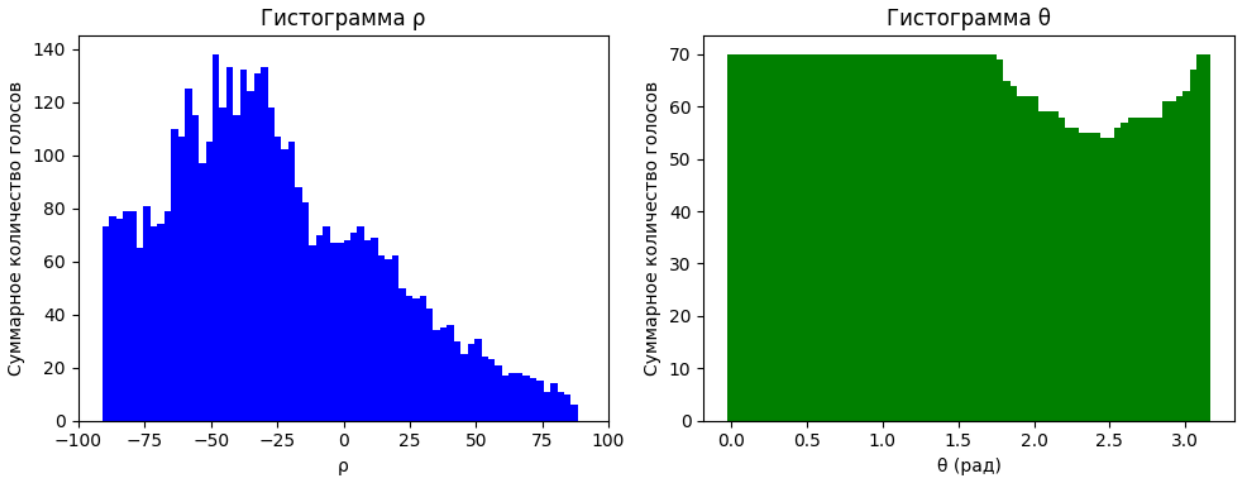

Рисунок 3 – Одномерные гистограммы распределения параметров ρ и θ от количества точек k

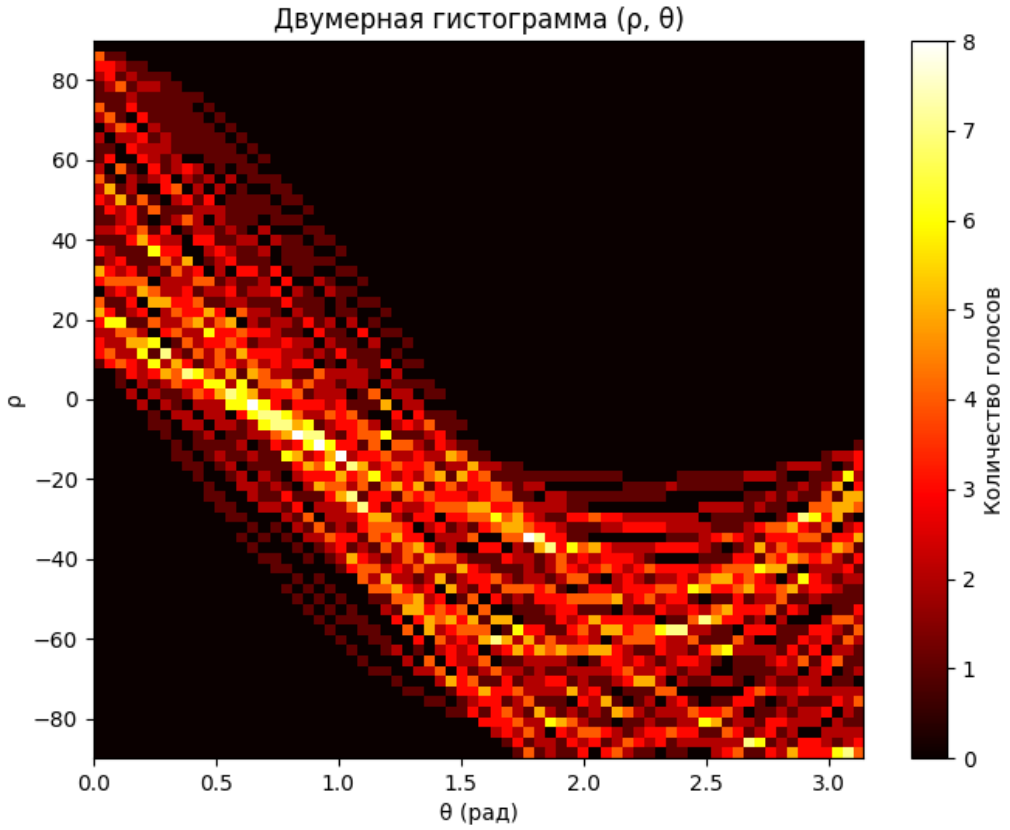

Рисунок 4– Двумерная гистограмма распределения параметров ρ и θ от количества точек k

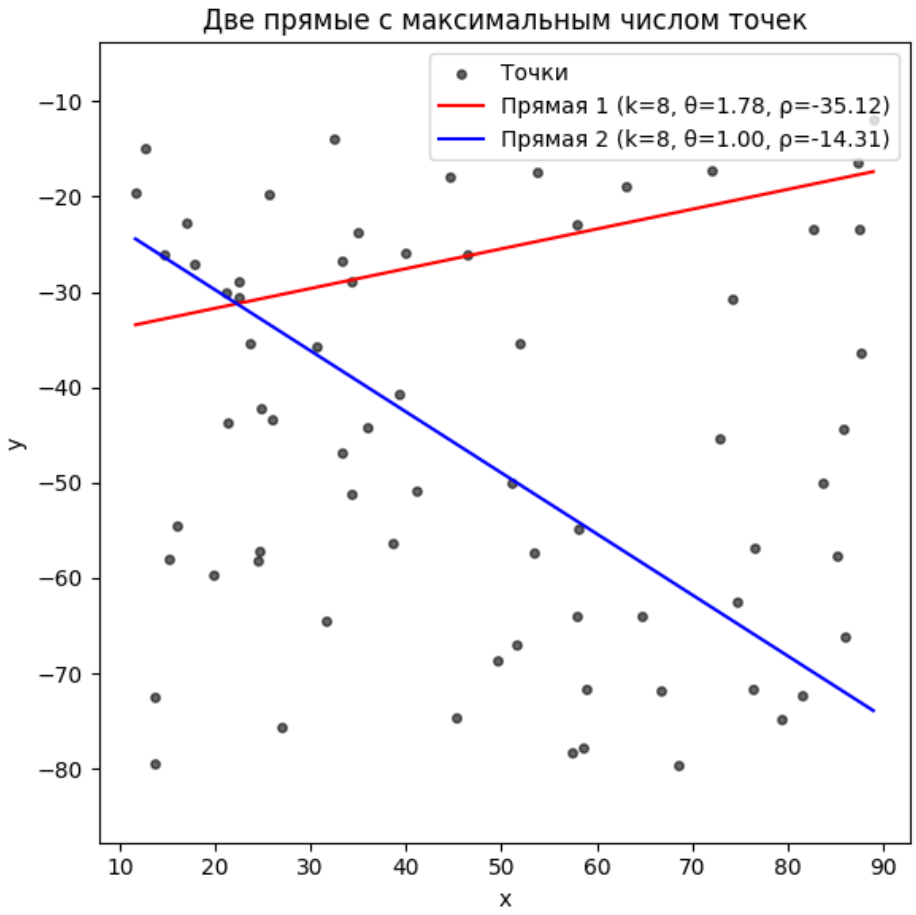

Рисунок 5 – Две прямые с максимальным числом точек

Вывод: в результате выполнения домашней работы были сформированы практические навыки исследования одного из методов символического описания объектов, в котором прямолинейные и криволинейные отрезки образуют основную структуру объекта..

Листинг программы

```
import argparse

import numpy as np
import matplotlib.pyplot as plt



def parse_args() -> argparse.Namespace:
    """Выполняет парсинг аргументов."""
    parser = argparse.ArgumentParser(description="Symbolic Object Description")
    parser.add_argument(
        "--n",
        type=int,
        help="Number of points in distribution",
        required=False,
        default=70,
    )
    parser.add_argument(
        "--x_min",
        type=float,
        help="Minimal value of x-axis",
        required=False,
        default=10,
    )
    parser.add_argument(
        "--x_max",
        type=float,
        help="Maximal value of x-axis",
        required=False,
        default=90,
    )
    parser.add_argument(
        "--y_min",
        type=float,
        help="Minimal value of y-axis",
        required=False,
        default=-80,
    )
```

```
    parser.add_argument(
        "--y_max",
        type=float,
        help="Maximal value of y-axis",
        required=False,
        default=-10,
    )
    return parser.parse_args()


def generate_distribution(
        x_range: tuple[float, float],
        y_range: tuple[float, float],
        n: int,
) -> tuple[np.ndarray, np.ndarray]:
    """Сгенерировать распределение."""
    np.random.seed(42)
    return (
        np.random.uniform(x_range[0], x_range[1], n),
        np.random.uniform(y_range[0], y_range[1], n),
    )


def plot_distribution(x: np.ndarray, y: np.ndarray):
    """Построить график распределения точек."""
    plt.figure(figsize=(6, 6))
    plt.scatter(x, y, s=15, color='black', alpha=0.7)
    plt.title('Расположение исходных точек')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.grid(True)
    plt.axis('equal')
    plt.tight_layout()
    plt.show()


def calculate_parameter_range(
        x: np.ndarray,
        y: np.ndarray,
) -> tuple[tuple[float, float], tuple[float, float]]:
```

```
    """
    Вычислить значения параметров ρ и θ, где
        ρ = x*cosθ + y*sinθ.
    Таким образом, получаем
        |ρ| = |x*cosθ+y*sinθ| ≤ √(x**2+y**2)
    :returns:
    - θ - угол между перпендикуляром из начала координат к прямой и осью x;
    - ρ - расстояние от начала координат до прямой.
    """
    theta_min, theta_max = 0, np.pi
    max_dist = np.sqrt(max(x) ** 2 + max(y) ** 2)
    rho_min, rho_max = -max_dist, max_dist
    return (theta_min, theta_max), (rho_min, rho_max)


def plot_histogram(
        x: np.ndarray,
        y: np.ndarray,
        theta_range: tuple[float, float],
        rho_range: tuple[float, float],
        *,
        num_theta: int = 180,
        num_rho: int = 360,
) -> np.ndarray:
    """
    Построить гистограммы для параметров ρ и θ.
    :returns:
    Аккумулятор Хафа.
    """
    theta_vals = np.linspace(
        start=theta_range[0],
        stop=theta_range[1],
        num=num_theta,
    )
    rho_vals = np.linspace(
        start=rho_range[0],
        stop=rho_range[1],
        num=num_rho,
    )
```

```
accumulator = np.zeros((num_rho, num_theta), dtype=int)

for xi, yi in zip(x, y):
    rho_curve = xi * np.cos(theta_vals) + yi * np.sin(theta_vals)
    rho_indices = np.digitize(rho_curve, rho_vals) - 1
    valid = (rho_indices >= 0) & (rho_indices < num_rho)
    if np.any(valid):
        accumulator[rho_indices[valid], np.arange(num_theta)[valid]] += 1

rho_hist = np.sum(accumulator, axis=1)
theta_hist = np.sum(accumulator, axis=0)

plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.bar(
    x=rho_vals,
    height=rho_hist,
    width=(rho_vals[1] - rho_vals[0]),
    color='blue',
)
plt.title('Гистограмма ρ')
plt.xlabel('ρ')
plt.ylabel('Суммарное количество голосов')

plt.subplot(1, 2, 2)
plt.bar(
    x=theta_vals,
    height=theta_hist,
    width=(theta_vals[1] - theta_vals[0]),
    color='green',
)
plt.title('Гистограмма θ')
plt.xlabel('θ (рад)')
plt.ylabel('Суммарное количество голосов')
plt.tight_layout()
plt.show()

plt.figure(figsize=(8, 6))
plt.imshow(
    accumulator,
```

```
        aspect='auto',
        origin='lower',
        extent=(theta_range[0], theta_range[1], rho_range[0], rho_range[1]),
        cmap='hot',
    )
    plt.colorbar(label='Количество голосов')
    plt.title('Двумерная гистограмма (ρ, θ)')
    plt.xlabel('θ (рад)')
    plt.ylabel('ρ')
    plt.show()

    return accumulator


def plot_found_lines(
        accumulator: np.ndarray,
        x: np.ndarray,
        y: np.ndarray,
        theta_range: tuple[float, float],
        rho_range: tuple[float, float],
        *,
        num_theta: int = 180,
        num_rho: int = 360,
):
    theta_vals = np.linspace(
        start=theta_range[0],
        stop=theta_range[1],
        num=num_theta,
    )
    rho_vals = np.linspace(
        start=rho_range[0],
        stop=rho_range[1],
        num=num_rho,
    )

    acc_copy = accumulator.copy()
    max_points = []
    for _ in range(2):
        idx = np.unravel_index(np.argmax(acc_copy), acc_copy.shape)
        max_points.append((idx, acc_copy[idx]))
```

```
            r, t = idx
            r_min, r_max = max(0, r - 5), min(num_rho, r + 6)
            t_min, t_max = max(0, t - 5), min(num_theta, t + 6)
            acc_copy[r_min:r_max, t_min:t_max] = 0

        plt.figure(figsize=(6, 6))
        plt.scatter(x, y, s=15, color='black', alpha=0.6, label='Точки')

        colors = ['red', 'blue']
        for idx, ((r_idx, t_idx), cnt) in enumerate(max_points):
            rho = rho_vals[r_idx]
            theta = theta_vals[t_idx]
            if np.abs(np.sin(theta)) > 1e-3:
                x_line = np.linspace(min(x), max(x), 100)
                y_line = (rho - x_line * np.cos(theta)) / np.sin(theta)
                plt.plot(
                    x_line,
                    y_line,
                    color=colors[idx],
                    label=f'Прямая {idx + 1} (k={cnt}, θ={theta:.2f}, ρ={rho:.2f})',
                )
            else:
                x_vert = rho / np.cos(theta)
                plt.axvline(
                    x_vert,
                    color=colors[idx],
                    label=f'Прямая {idx + 1} (k={cnt}, θ={theta:.2f}, ρ={rho:.2f})',
                )

        plt.title('Две прямые с максимальным числом точек')
        plt.xlabel('x')
        plt.ylabel('y')
        plt.legend()
        # plt.grid(True)
        plt.axis('equal')
        plt.tight_layout()
        plt.show()


if __name__ == '__main__':
```

```
args = parse_args()
x, y = generate_distribution(
    x_range=(args.x_min, args.x_max),
    y_range=(args.y_min, args.y_max),
    n=args.n,
)
plot_distribution(x, y)
theta_range, rho_range = calculate_parameter_range(x, y)
print(f'Диапазон θ: {theta_range}')
print(f'Диапазон ρ: {rho_range}')

accumulator = plot_histogram(
    x=x,
    y=y,
    theta_range=theta_range,
    rho_range=rho_range,
    num_rho=args.n,
    num_theta=args.n,
)
plot_found_lines(
    x=x,
    y=y,
    theta_range=theta_range,
    rho_range=rho_range,
    num_rho=args.n,
    num_theta=args.n,
    accumulator=accumulator,
)
```