Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(УНИВЕРСИТЕТ ИТМО)

Факультет «Систем управления и робототехники»

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ №5

По дисциплине «Техническое зрение»
на тему:
«Преобразование Хафа»

Студенты:
Гуров Михаил Алексеевич 408510 R3243
Зыкин Леонид Витальевич 470912 R3335
Куликов Илья Вячеславович 470122 R3243

Преподаватель:
Шаветов Сергей Васильевич

г. Санкт-Петербург
2025

Цель работы:
Освоение преобразования для поиска геометрических примитивов.

Теоретическое обоснование применяемых методов и функций геометрических преобразований:

В лабораторной работе использовалось **преобразование Хафа** — метод, основанный на идее голосования точек изображения за параметры геометрических объектов (прямых, окружностей и др.). Этот метод позволяет обнаруживать объекты даже при наличии шумов или частичной утраты формы.

Поиск прямых: преобразование Хафа для прямых основано на параметрическом уравнении $\rho = x\cos\theta + y\sin\theta$. Каждая точка изображения порождает синусоиду в пространстве параметров (ρ, θ), а точки пересечения этих синусоид указывают на существование прямой.

Поиск окружностей: для окружностей используется уравнение $(x-a)^2 + (y-b)^2 = R^2$. Каждая точка изображения "голосует" за возможные центры окружностей. Локальные максимумы аккумуляторного пространства указывают на положение окружностей.

Дифференциальные операторы: Фильтры (Собеля, Лапласа, Гаусса) применялись для усиления границ и подавления шума перед применением преобразования Хафа. Это повышает точность и уменьшает количество ложных срабатываний. Методы являются устойчивыми к поворотам, сдвигам и масштабированию, что делает их надёжным инструментом в задачах технического зрения.

Ход выполнения работы:

Исходные изображения:

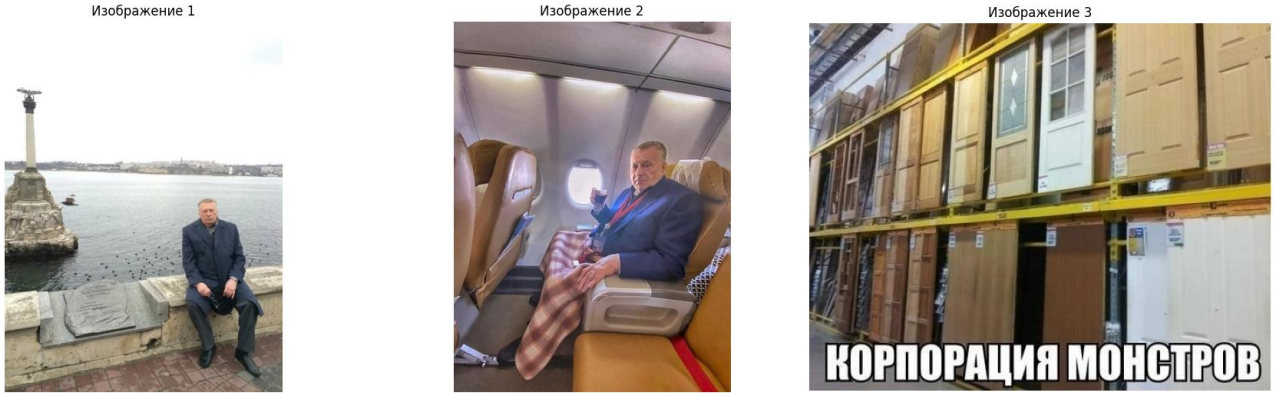

Рисунок 1 – исходные изображения для задания 1

Рисунок 2 – исходные изображения для задания 2

Листинги программных реализаций:
Задание 1. Поиск прямых

Импортируем необходимые библиотеки, загружаем первое изображение с прямыми, преобразуем его в оттенки серого для дальнейшей обработки.

```
import cv2 as cv
import numpy as np
import math
from pa_utils import ShowImages
img1 = cv.imread("1.jpg")
gray1 = cv.cvtColor(img1, cv.COLOR_BGR2GRAY)
```

На полученном изображении выделяются контуры методом Кэнни, после чего применяется cv.HoughLinesP() для детекции прямых. Каждая найденная прямая отображается зелёной линией, а её начальные и конечные точки выделяются красными кружками. Параллельно считается длина каждой линии.

```
edges1 = cv.Canny(gray1, 50, 150, apertureSize=3)
lines1 = cv.HoughLinesP(edges1, 1, np.pi/180, threshold=80, minLineLength=30,
maxLineGap=10)

img1_lines = img1.copy()
lengths1 = []

if lines1 is not None:
    for x1, y1, x2, y2 in lines1[:, 0]:
        cv.line(img1_lines, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv.circle(img1_lines, (x1, y1), 4, (0, 0, 255), -1)
        cv.circle(img1_lines, (x2, y2), 4, (0, 0, 255), -1)
        lengths1.append(math.hypot(x2 - x1, y2 - y1))

ShowImages([("Прямые на исходном изображении", img1_lines)])
```

Изображение фильтруется с помощью оператора Собеля, чтобы усилить вертикальные границы. Это уменьшает количество ложных контуров. После фильтрации применяется тот же метод Хафа для поиска прямых. Выводится информация о количестве найденных прямых, а также длина самой короткой и самой длинной прямой до и после применения фильтра.

Для наглядного сравнения отображаются два результата: до фильтрации и после. Подписи содержат числовую сводку по количеству и длинам прямых.

```
sobelx1 = cv.Sobel(gray1, cv.CV_64F, 1, 0, ksize=3)
sobelx1 = cv.convertScaleAbs(sobelx1)
edges_sobel1 = cv.Canny(sobelx1, 50, 150)

lines_sobel1 = cv.HoughLinesP(edges_sobel1, 1, np.pi/180, 80, 30, 10)
img1_sobel_lines = img1.copy()
lengths_sobel1 = []

if lines_sobel1 is not None:
    for x1, y1, x2, y2 in lines_sobel1[:, 0]:
        cv.line(img1_sobel_lines, (x1, y1), (x2, y2), (255, 0, 0), 2)
        cv.circle(img1_sobel_lines, (x1, y1), 3, (0, 255, 255), -1)
        cv.circle(img1_sobel_lines, (x2, y2), 3, (0, 255, 255), -1)
        lengths_sobel1.append(math.hypot(x2 - x1, y2 - y1))

ShowImages([("Прямые после фильтрации (Sobel)", img1_sobel_lines)])

print("[Оригинал]")
print("Количество прямых:", len(lengths1))
print("Мин длина: {:.2f}".format(min(lengths1)))
print("Макс длина: {:.2f}".format(max(lengths1)))

print("\n[После Собеля]")
print("Количество прямых:", len(lengths_sobel1))
print("Мин длина: {:.2f}".format(min(lengths_sobel1)))
print("Макс длина: {:.2f}".format(max(lengths_sobel1)))

fig, axs = plt.subplots(1, 2, figsize=(14, 7))

# Первое изображение: без фильтрации
axs[0].imshow(cv.cvtColor(img1_lines, cv.COLOR_BGR2RGB))
axs[0].set_title(f"До фильтрации\nПрямых: {len(lengths1)}\nМин:
{min(lengths1):.1f}, Макс: {max(lengths1):.1f}")
axs[0].axis('off')

# Второе изображение: после Лапласа
axs[1].imshow(cv.cvtColor(img1_sobel_lines, cv.COLOR_BGR2RGB))
axs[1].set_title(f"После Sobel\nПрямых: {len(lengths_sobel1)}\nМин:
{min(lengths_sobel1):.1f}, Макс: {max(lengths_sobel1):.1f}")
axs[1].axis('off')
```

```
plt.tight_layout()
plt.show()
```

Для изображения 2 можно повторить ту же процедуру (с фильтрацией Собеля).

Для изображения 3 — заменить фильтрацию на оператор Лапласа (Laplacian) и сравнить результат.

```
laplacian3 = cv.Laplacian(gray3, cv.CV_64F)
laplacian3 = cv.convertScaleAbs(laplacian3)
edges_laplace3 = cv.Canny(laplacian3, 50, 150)

lines_laplace3 = cv.HoughLinesP(edges_laplace3, 1, np.pi/180, 80, 30, 10)
img3_laplace_lines = img3.copy()
lengths_laplace3 = []

if lines_laplace3 is not None:
    for x1, y1, x2, y2 in lines_laplace3[:, 0]:
        cv.line(img3_laplace_lines, (x1, y1), (x2, y2), (0, 255, 255), 2)
        cv.circle(img3_laplace_lines, (x1, y1), 3, (255, 0, 255), -1)
        cv.circle(img3_laplace_lines, (x2, y2), 3, (255, 0, 255), -1)
        lengths_laplace3.append(math.hypot(x2 - x1, y2 - y1))
```

Задание 2. Поиск окружностей

Загружаем первое изображение с окружностями и переводим его в оттенки серого.

```
img1 = cv.imread("4.jpg")
gray1 = cv.cvtColor(img1, cv.COLOR_BGR2GRAY)
```

С помощью cv.HoughCircles() найдены окружности с радиусами из заданного диапазона (15–50 пикселей). Они отрисованы зелёными кругами, центры помечены красной точкой.

```
circles1 = cv.HoughCircles(
    gray1,
    cv.HOUGH_GRADIENT,
    dp=1,
    minDist=20,
    param1=100,
    param2=30,
    minRadius=15,
    maxRadius=50
)
```

```
img1_circles = img1.copy()
if circles1 is not None:
    circles1 = np.uint16(np.around(circles1))
    for x, y, r in circles1[0, :]:
        cv.circle(img1_circles, (x, y), r, (0, 255, 0), 2)
        cv.circle(img1_circles, (x, y), 2, (0, 0, 255), 3)
```

Для подавления шума и усиления краёв изображение фильтруется оператором Лапласа. Повторный запуск HoughCircles позволяет уточнить контуры окружностей.

```
Sobel1 = cv.Sobel(gray1, cv.CV_64F, 1, 1, ksize=3)
Sobel1 = cv.convertScaleAbs(sobel1)

Circles1_sobel = cv.HoughCircles(
    Sobel1,
    cv.HOUGH_GRADIENT,
    dp=1,
    minDist=20,
    param1=100,
    param2=30,
    minRadius=10,
    maxRadius=40
)

Img1_circles_sobel = img2.copy()
if circles1_sobel is not None:
    circles1_sobel = np.uint16(np.around(circles2_sobel))
    for x, y, r in circles1_sobel[0, :]:
        cv.circle(img1_circles_sobel, (x, y), r, (255, 0, 255), 2)
        cv.circle(img1_circles_sobel, (x, y), 2, (0, 255, 255), 3)
```

Визуально сравниваются результаты поиска окружностей до и после фильтрации. На изображениях отмечены как окружности, так и их центры.

```
fig, axs = plt.subplots(1, 2, figsize=(14, 7))

axs[0].imshow(cv.cvtColor(img1_circles, cv.COLOR_BGR2RGB))
axs[0].set_title("До фильтрации (исходное изображение)")
axs[0].axis('off')

axs[1].imshow(cv.cvtColor(img1_circles_sobel, cv.COLOR_BGR2RGB))
axs[1].set_title("После Собеля (фильтрованное изображение)")
axs[1].axis('off')

plt.suptitle("Поиск окружностей — изображение 2")
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

Аналогично, применим этот код к изображениям 2 и 3.

Результирующие изображения:

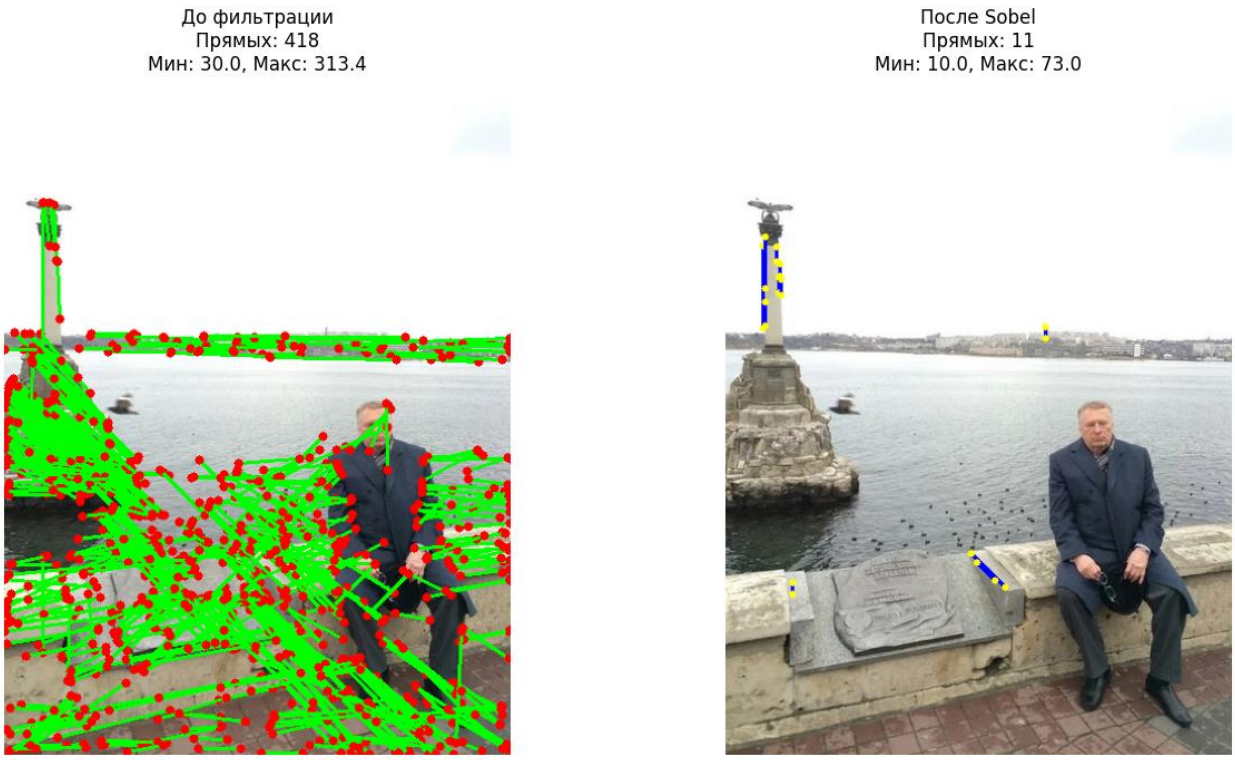

Рисунок 3 - Задание 1, изображение 1

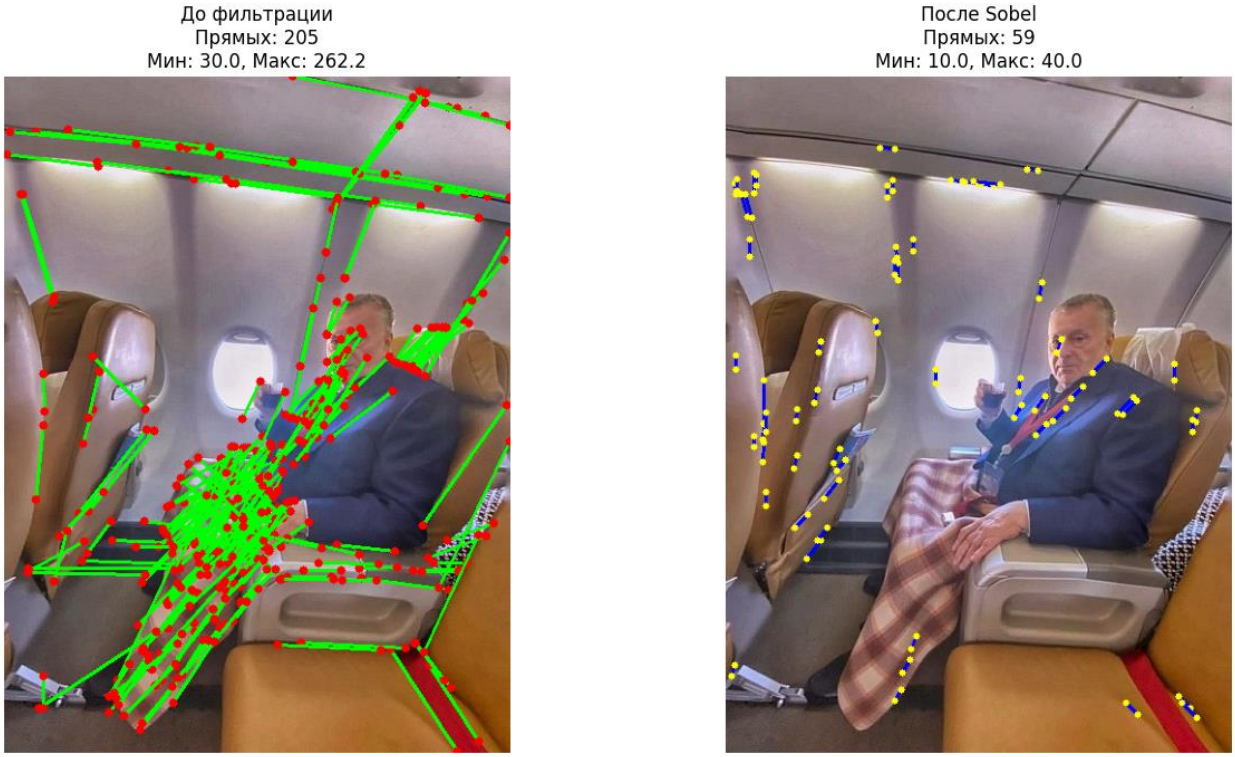

Рисунок 4 - Задание 1, изображение 2

Рисунок 5 - Задание 1, изображение 3

Рисунок 6 - Задание 2, изображение 1

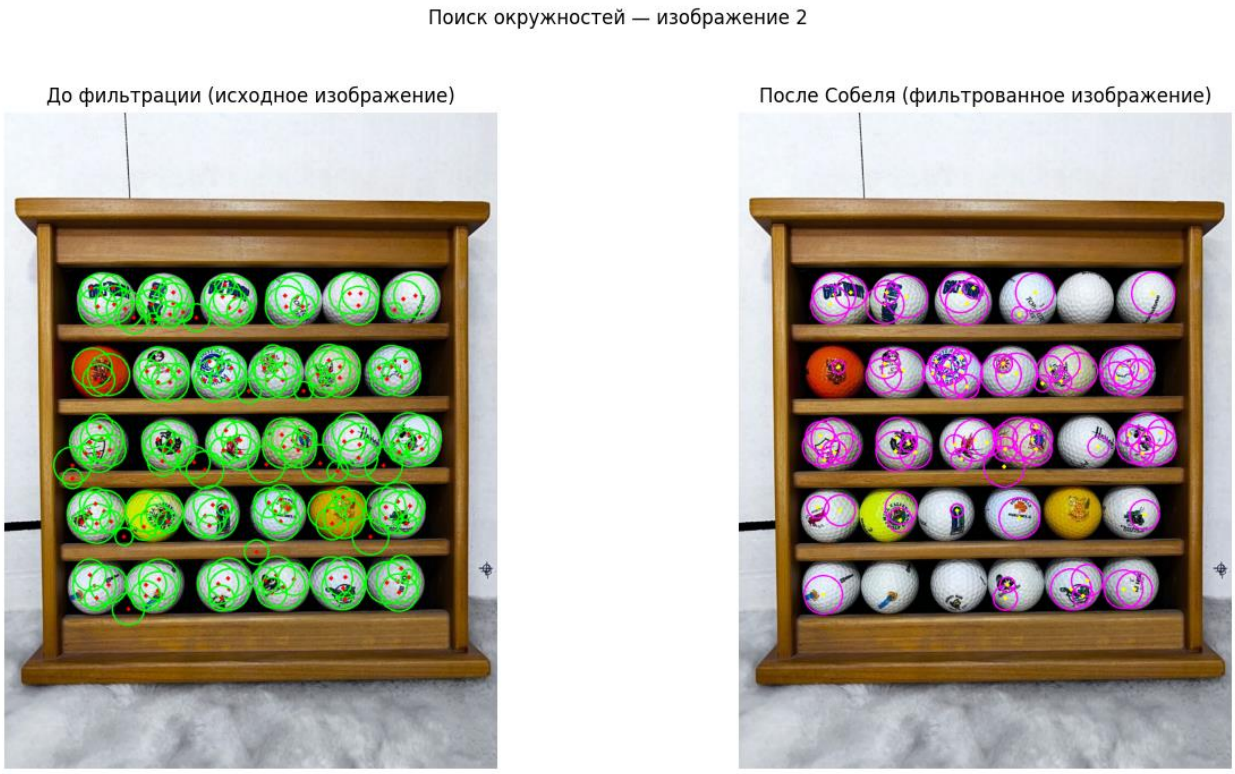

Рисунок 7 - Задание 2, изображение 1

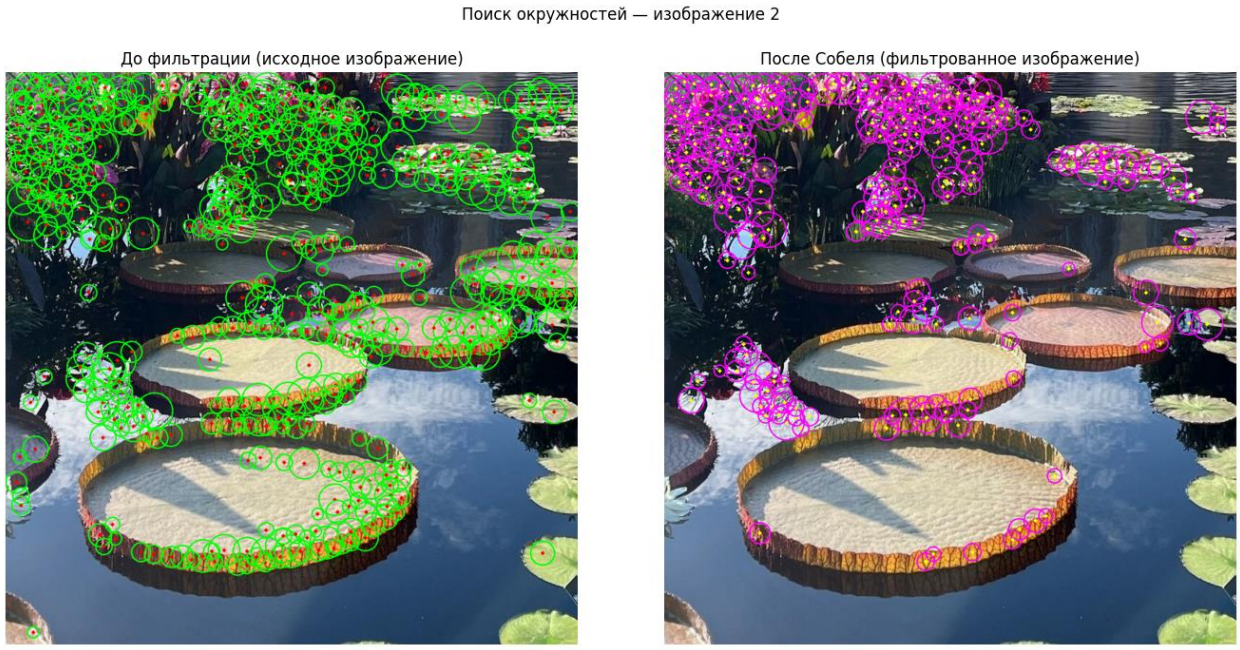

Рисунок 8 - Задание 2, изображение 1

Выводы о проделанной работе:

В ходе лабораторной работы №5 были изучены и реализованы методы преобразования Хафа для поиска геометрических примитивов: прямых и окружностей.

В первом задании были успешно найдены прямые на трёх изображениях как без предварительной обработки, так и после применения дифференциальных

операторов (Собеля и Лапласа). Показано, что фильтрация позволяет уменьшить количество ложных срабатываний и повысить точность обнаружения линий. Для каждой прямой определены координаты концов и рассчитана длина, определены минимальные и максимальные значения.

Во втором задании выполнен поиск окружностей как заданного радиуса, так и из диапазона. Использование операторов фильтрации (Собель) также улучшило качество детекции, позволив точнее выделить окружности и отфильтровать шум. Полученные результаты отображены графически с указанием центров и радиусов.

Метод Хафа подтвердил свою эффективность при наличии предварительной обработки и корректно подобранных параметров.