Научно учебная группа
«Цифровые методы в неврологии»

Пермь,
2024

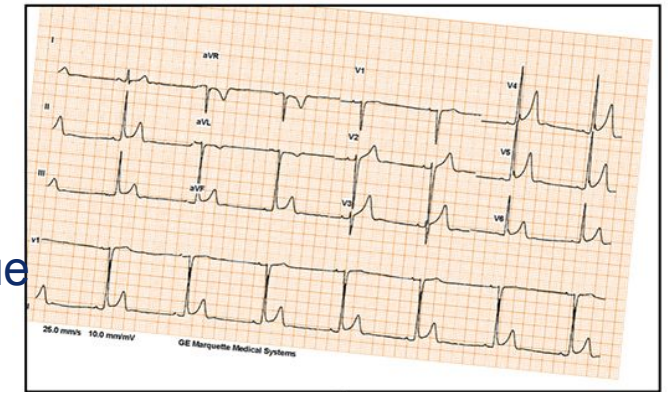
Методы коррекции перекоса изображений ЭКГ и детекции отведений

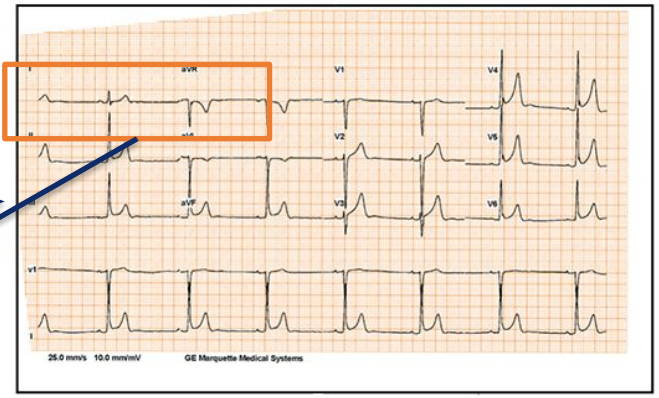
Лукин Семён Олегович
студент 4 курса ОП Программная инженерия

Процесс

1. Коррекция перекоса

Вручную или преобразование Хафа или Радона





2. Локализация отведений

Вручную или ML/DL

3. Бинаризация

Saulova/Otsu

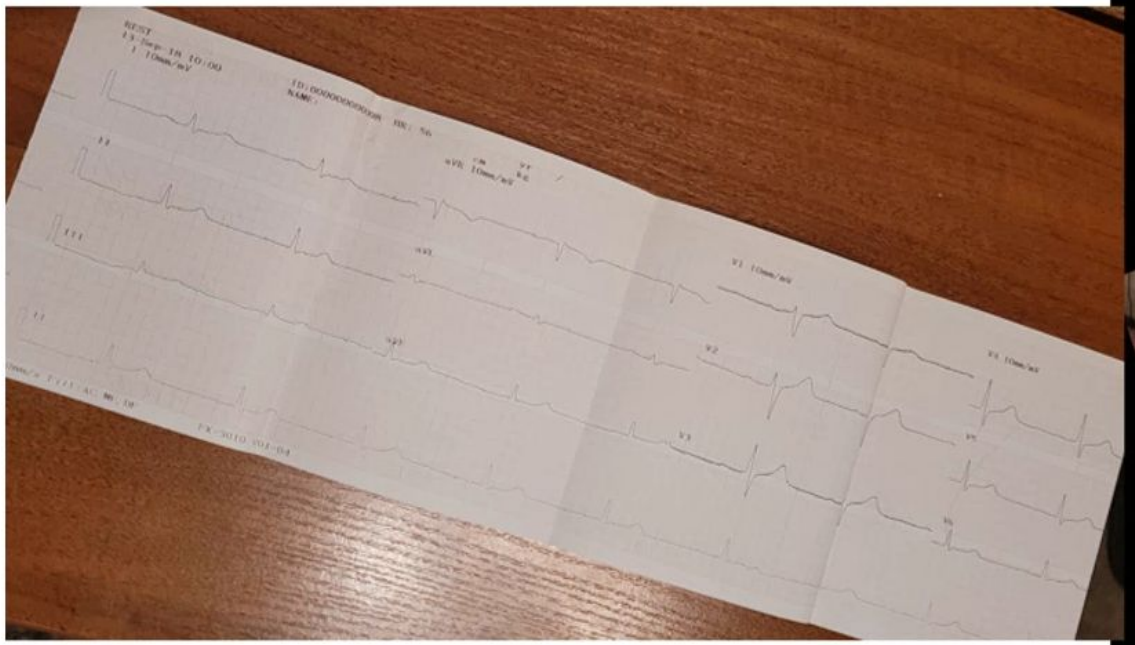
4. Извлечение сигнала

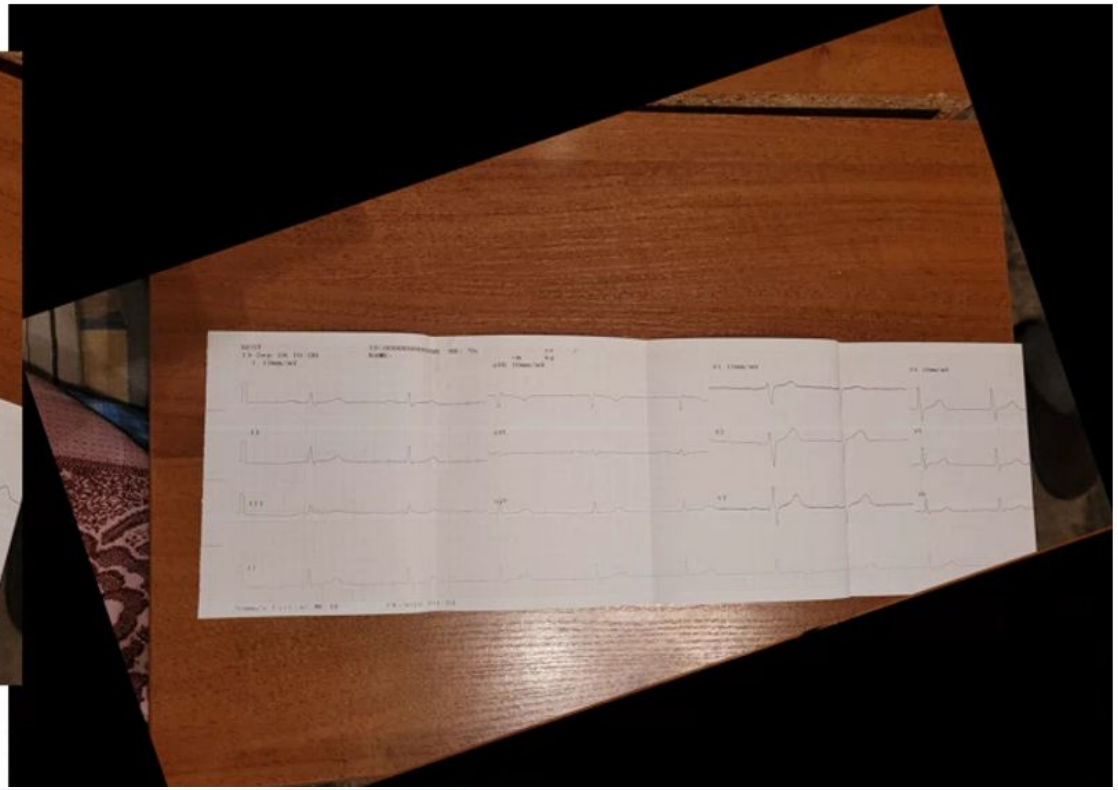
```
-0.0,-0.0,75.00000000000081,-0.0,nan,100.00000000000108,-100.00000000000108,50.00000000000054,75.00000000000081,275.00000000000296,75.000
-0.0,-0.0,75.00000000000081,-0.0,-75.00000000000081,100.00000000000108,-75.00000000000081,25.00000000000027,100.00000000000108,200.000000
-0.0,-25.00000000000027,75.00000000000081,-0.0,-100.00000000000108,112.50000000000122,-75.00000000000081,25.00000000000027,75.0000000000
-0.0,-25.00000000000027,75.00000000000081,-0.0,-100.00000000000108,125.00000000000135,-75.00000000000081,25.00000000000027,62.5000000000
-0.0,-25.00000000000027,75.00000000000081,-0.0,-100.00000000000108,125.00000000000135,-50.00000000000054,25.00000000000027,50.0000000000
```

Автоматическая коррекция перекоса

Цель - устранить перекос путем поворота изображения на ту же величину, что и его перекос, но в противоположном направлении.

Результат - выровненное по горизонтали и вертикали изображение.





Автоматическая коррекция перекоса

- Вычисление угла перекоса с помощью фильтра Кэнни и преобразования Хафа
- ответ между -45° и 45°

scikit-image
image processing in python

Алгоритм коррекции перекоса

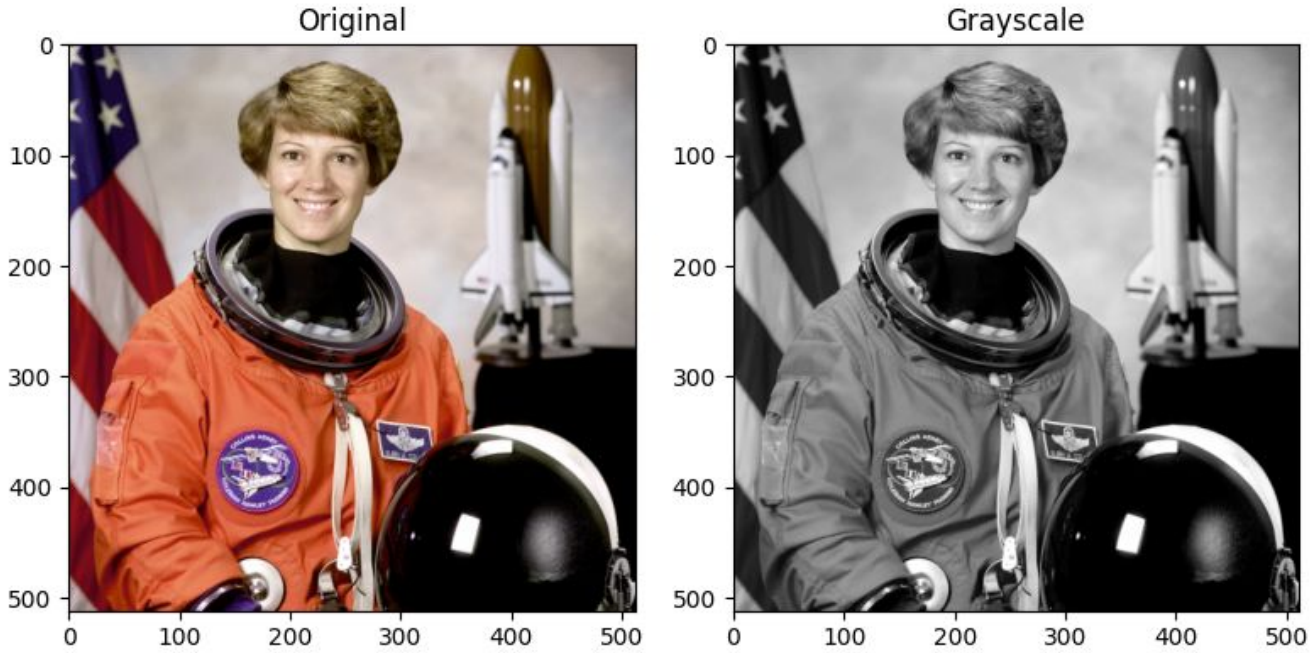
1. Преобразование изображения в градации серого
2. Выделение границ методом Кэнни
3. Преобразование Хафа для линий
4. Поиск пиков в преобразовании Хафа
5. Поиск угла

1. Преобразование изображения в градации серого

```
imagergb = rgba2rgb(image) if len(image.shape) == 3 and image.shape[2] == 4 else image
img = rgb2gray(imagergb) if len(imagergb.shape) == 3 else imagergb
```

- RGBA -> RGB
- RGB -> Grayscale

$$I = 0.299 \times p_r + 0.587 \times p_g + 0.114 \times p_b$$



2. Выделение границ методом Canny (Кэнни, Канни)

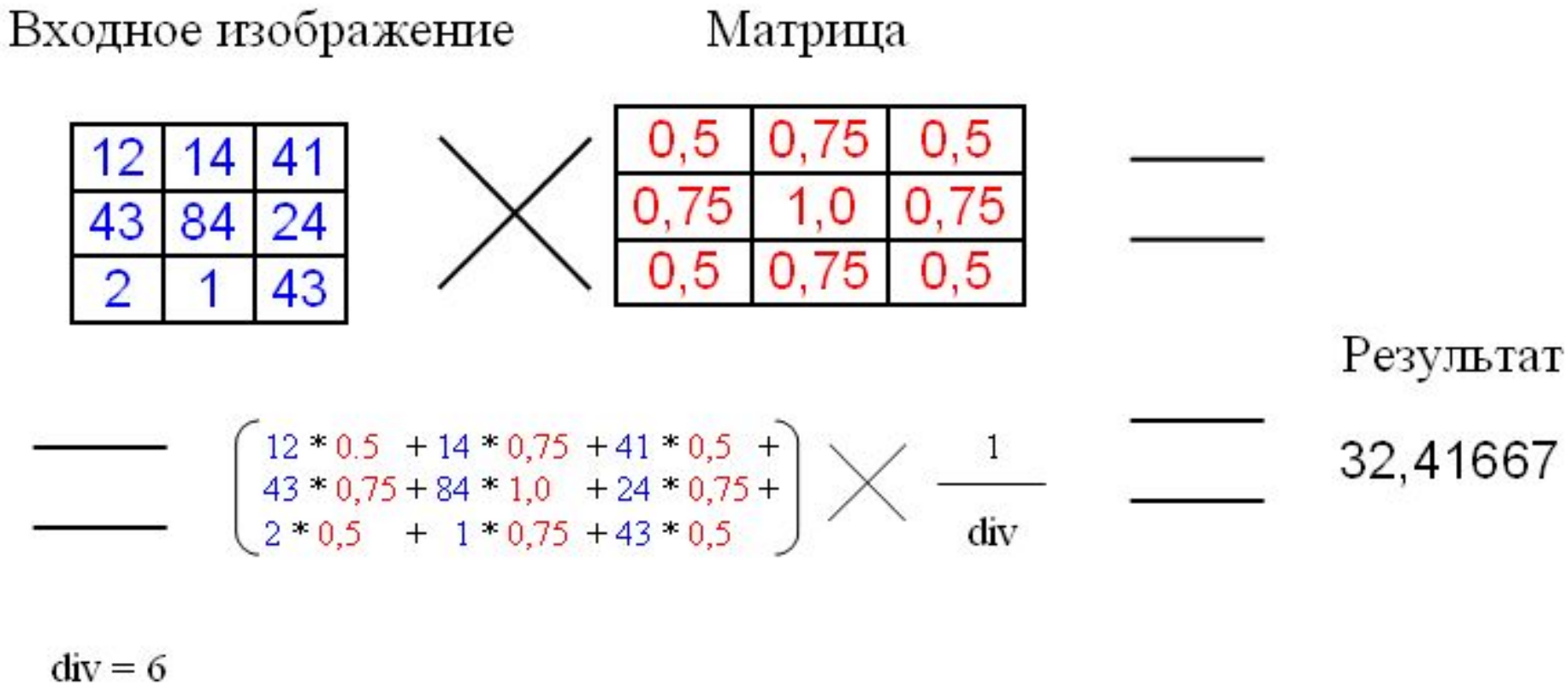
1. Фильтрация с помощью фильтра Гаусса
2. Поиск градиентов
3. Подавление немаксимумов
4. Двойная пороговая фильтрация
5. Трассировка области неоднозначности

```
edges = canny(img, sigma=sigma)

edges = cv.Canny(img,100,200)
```

2. Выделение границ методом Canny (Кэнни, Канни)

1. Фильтрация с помощью фильтра Гаусса - свёртки



тодо

фильт


125 KZ2


125 KZ2

2. Выделение границ методом Canny (Кэнни, Канни)

2. Поиск градиентов - фильтр Собеля

$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A}$$

Градиенты можно получить с помощью:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

$$\mathbf{\Theta} = \operatorname{atan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

- матрица аппроксимации модуля градиента $G^{N \cdot M}$
- матрица аппроксимации направления градиента $\Theta^{N \cdot M}$

2. Выделение границ методом Canny (Кэнни, Канни)

2.

$$I = \begin{bmatrix} 1 & 9 & 4 \\ 6 & 3 & 2 \\ 7 & 8 & 6 \end{bmatrix} \quad Gx = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 9 & 4 \\ 6 & 3 & 2 \\ 7 & 8 & 6 \end{bmatrix} = 6$$

$\mathbf{G}_y$

$$\mathrm{Gy} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 9 & 4 \\ 6 & 3 & 2 \\ 7 & 8 & 6 \end{bmatrix} = -6$$

Гр

$$G = \sqrt{36 + 36} = 8{,}48$$

$$\theta = \tan^{-1}\frac{-6}{6} = 135°$$

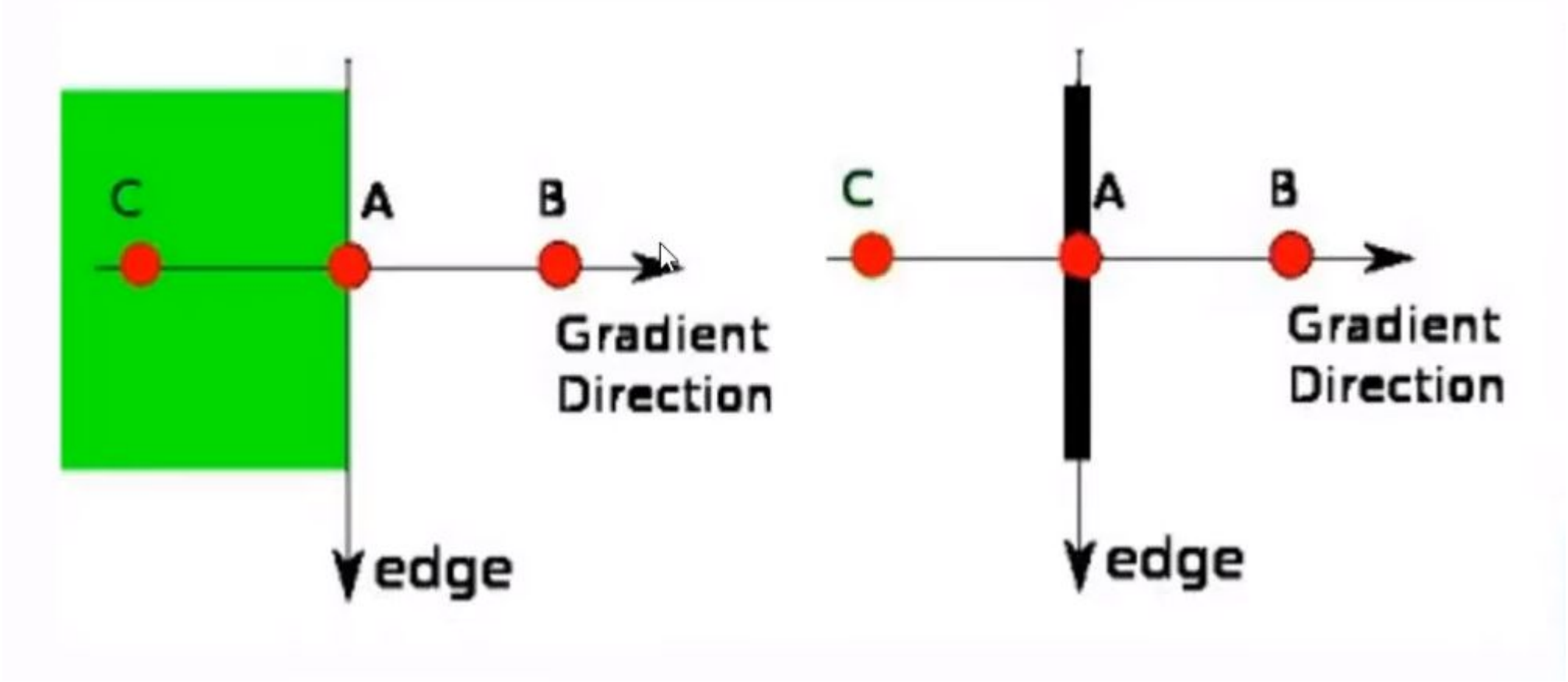
$$\begin{bmatrix} 1 & 9 & 4 \\ 6 & 8{,}48 & 2 \\ 7 & 8 & 6 \end{bmatrix}_M \quad \Theta^{N \cdot M}$$


125 KZ2


125 KZ2

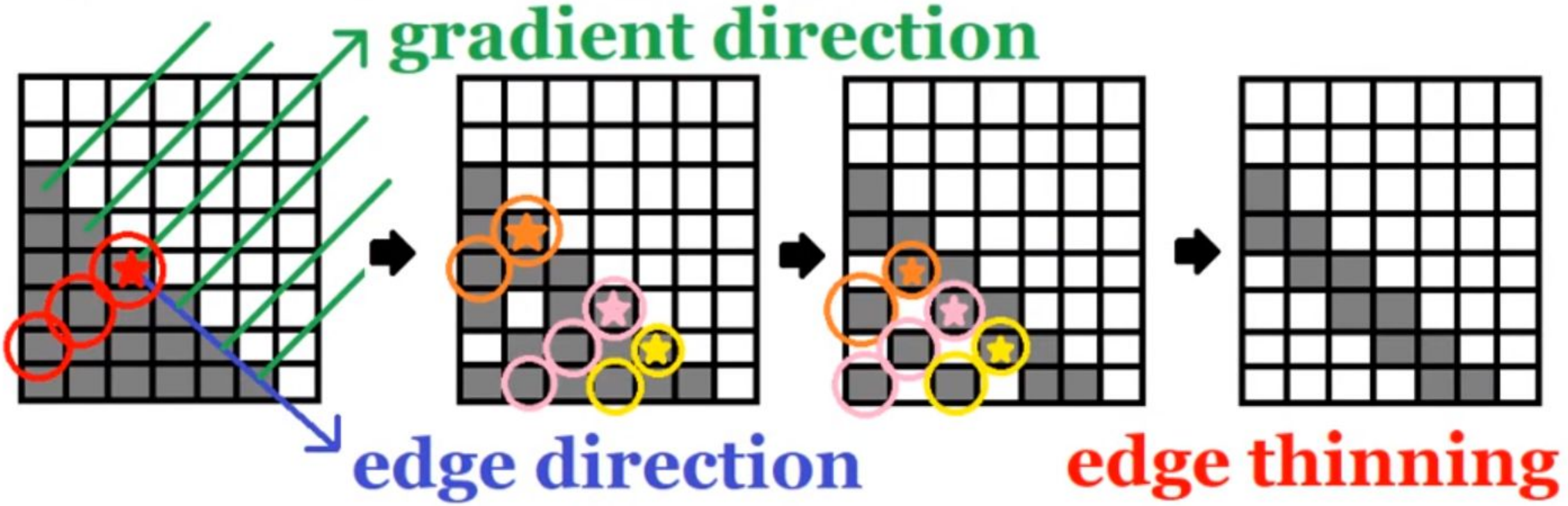
2. Выделение границ методом Canny (Кэнни, Канни)

3. Подавление немаксимумов

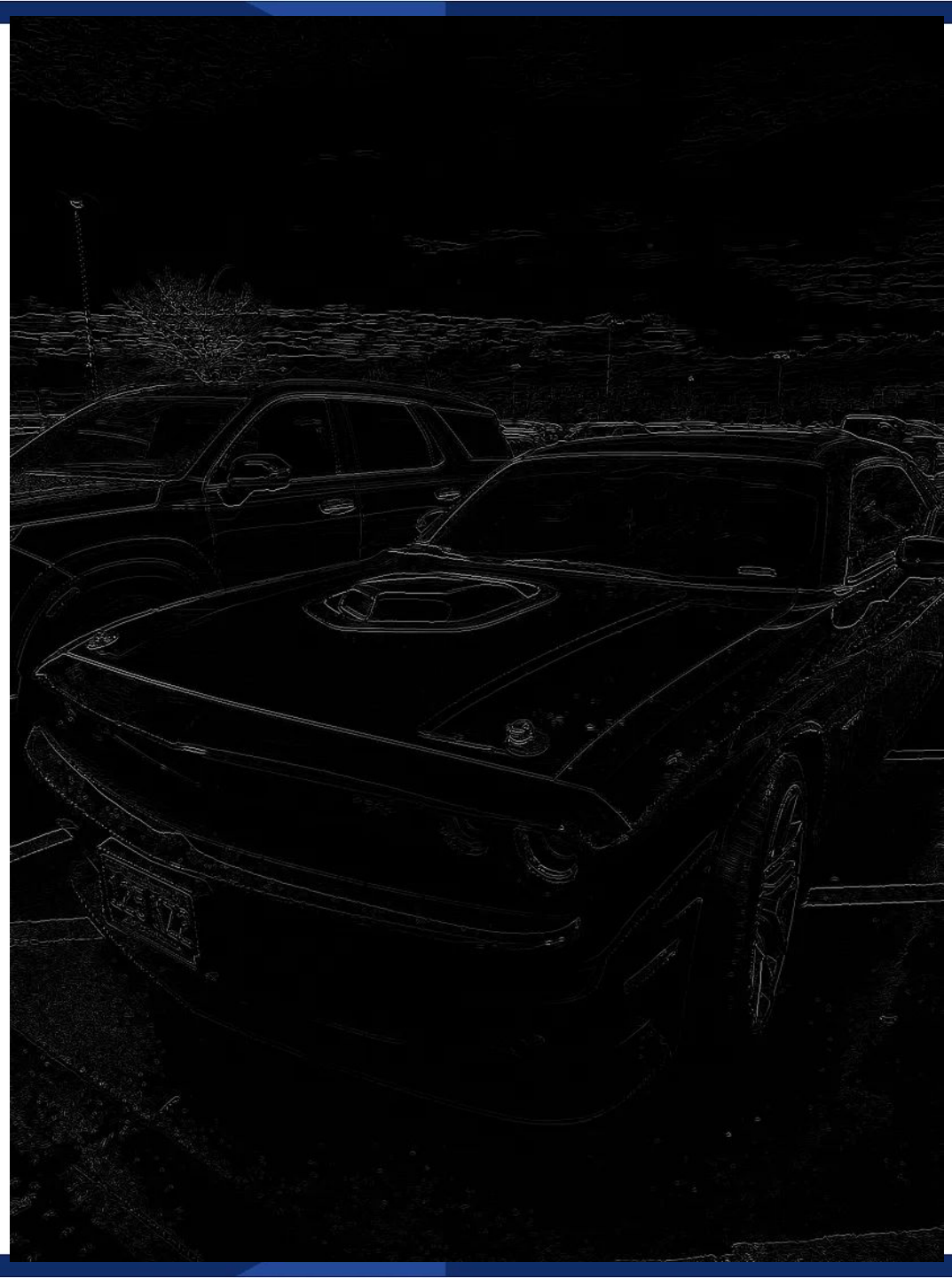


2. Выделение границ методом Canny (Кэнни, Канни)

3. Подавление немаксимумов (Non-max Suppression)




125 KZ2

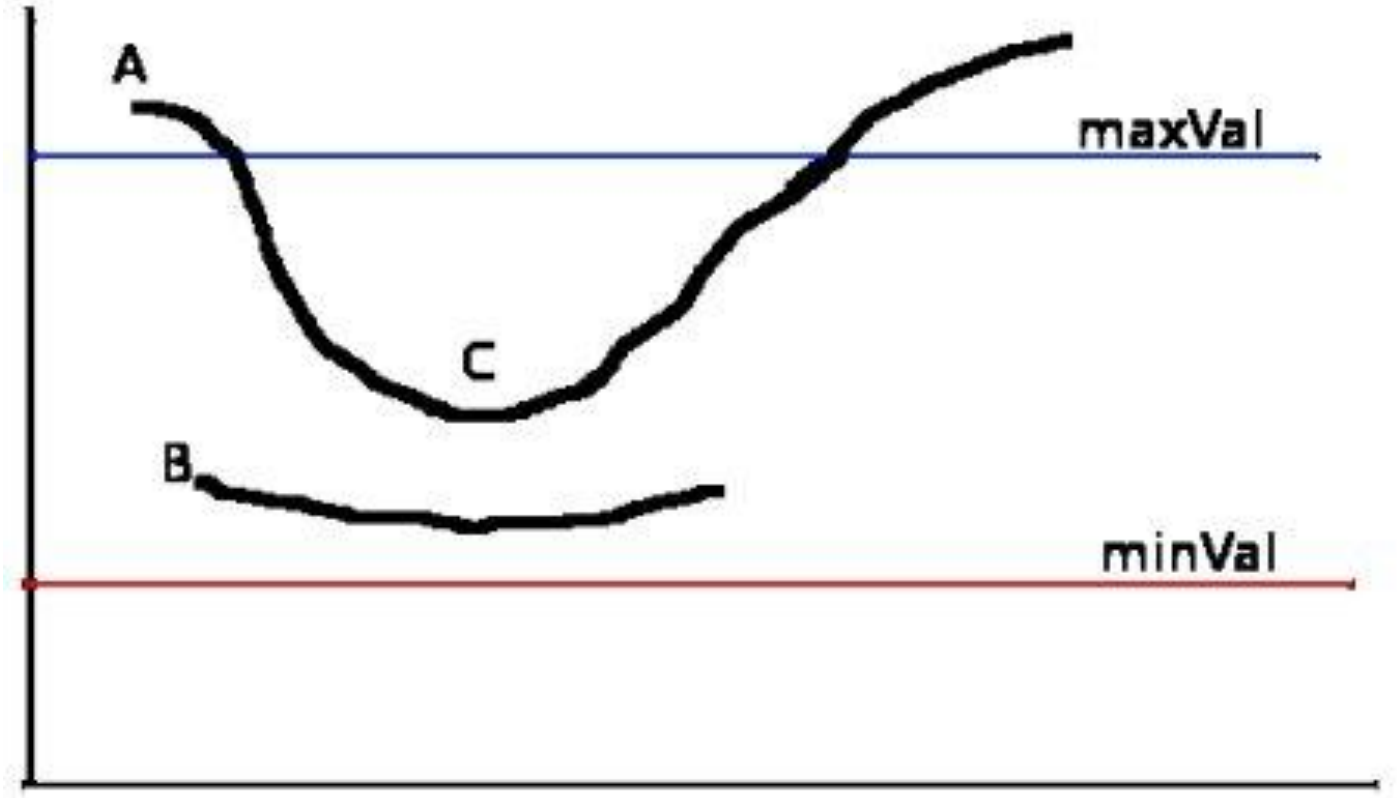


2. Выделение границ методом Canny (Кэнни, Канни)

4. Двойная пороговая фильтрация

Правила:

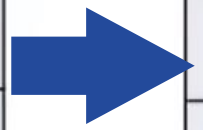
1. >maxVal -> 255
2. <minVal -> 0
3. точка соединена с сильными пикселями -> 255
4. точка соединена со слабыми пикселями -> 0



2. Выделение границ методом Canny (Кэнни, Канни)

4. Двойная пороговая фильтрация (minVal=50, maxVal=100)

20	60	58	120	158	80
48	214	15	18	67	49
59	5	11	12	32	25
53	26	31	59	36	55
46	38	15	45	18	40
55	98	59	89	89	56



0	60	58	255	255	80
0	255	0	0	67	0
59	0	0	0	0	0
53	0	0	59	0	55
0	0	0	0	0	0
55	98	59	89	0	56

2. Выделение границ методом Canny (Кэнни, Канни)

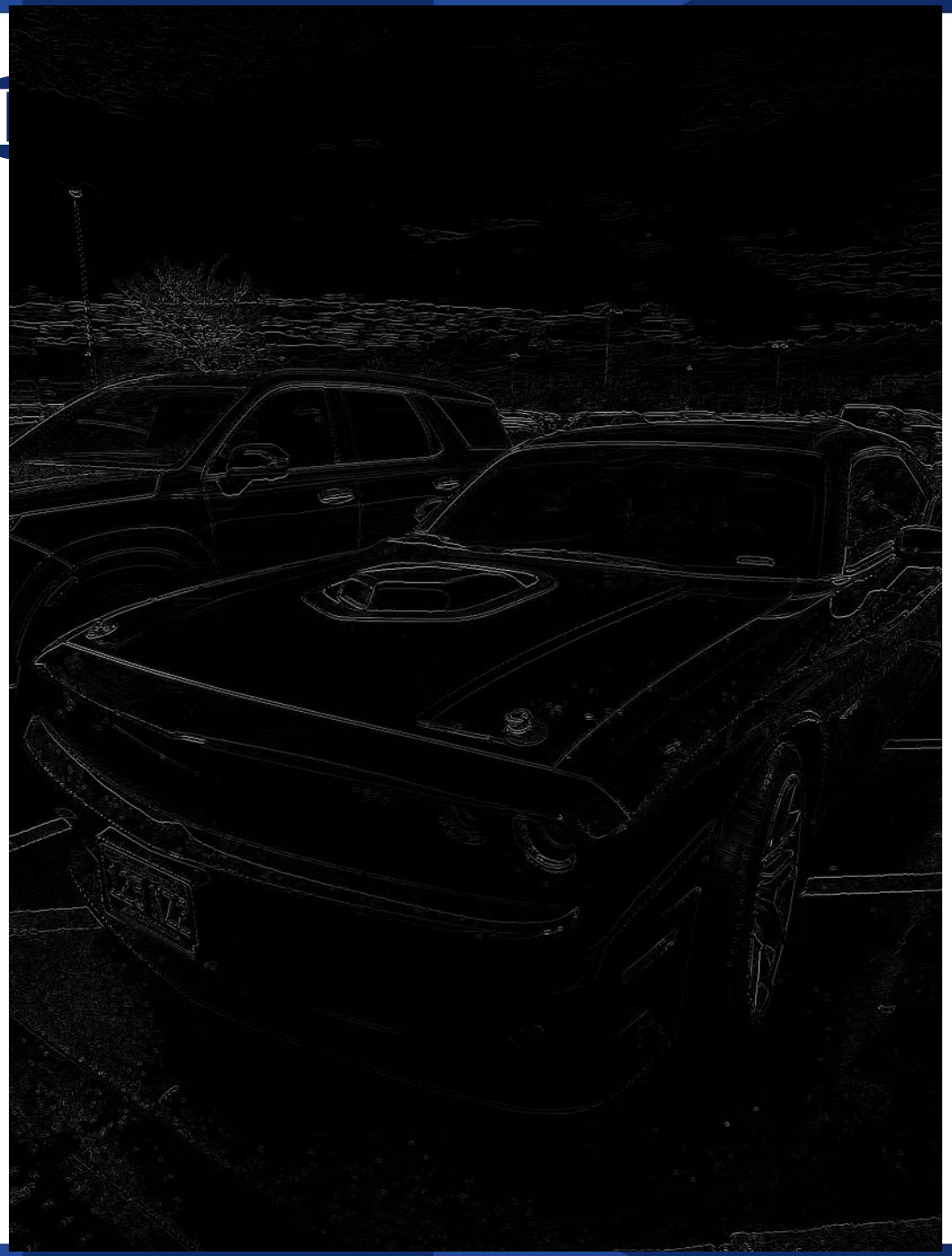
5. Трассировка области неоднозначности

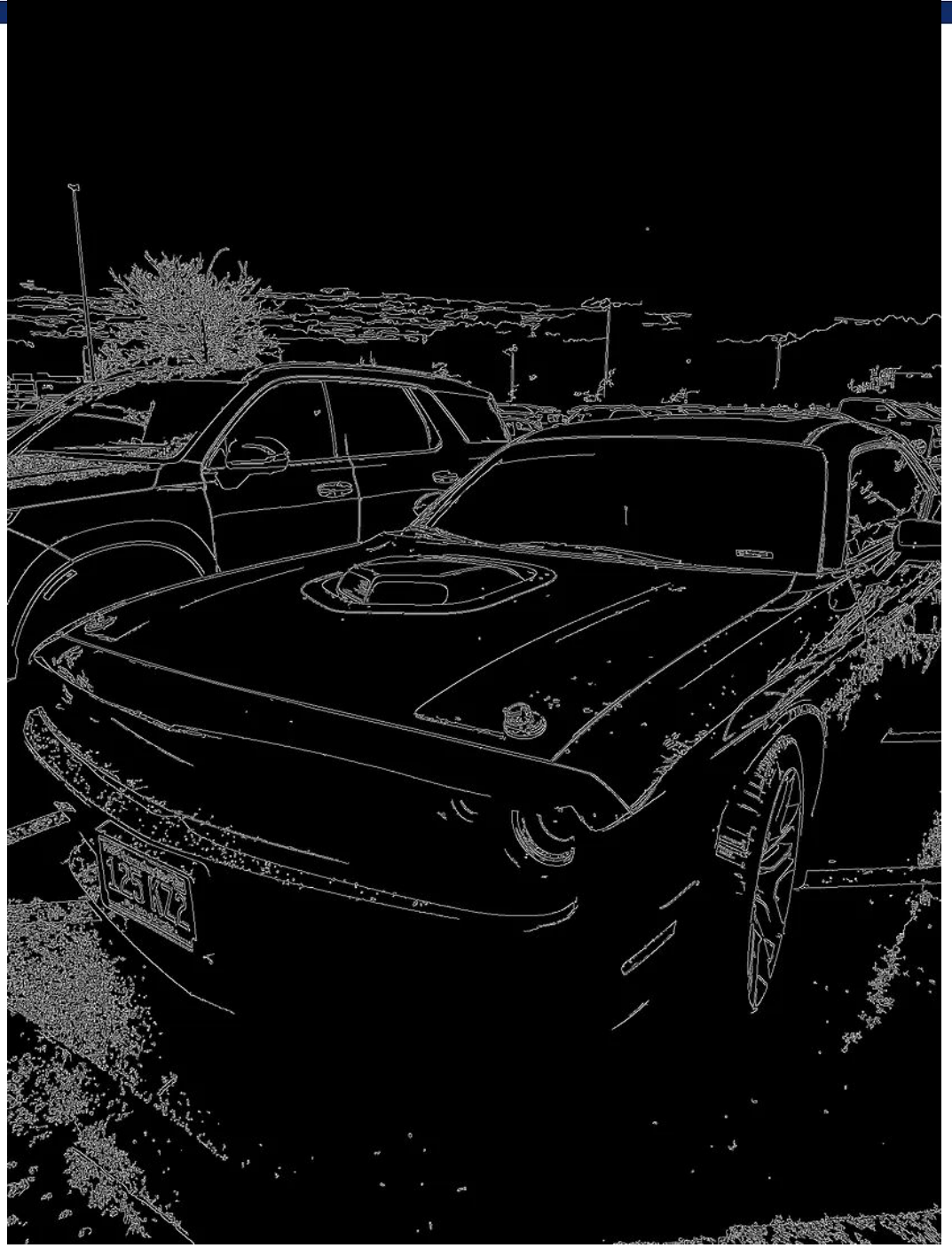
0	60	58	255	255	80
0	255	0	0	67	0
59	0	0	0	0	0
53	0	0	59	0	55
0	0	0	0	0	0
55	98	59	89	0	56

2. Выделение границ методом Canny (Кэнни, Канни)

5. Трассир

0	255	255	255	255	255
0	255	0	0	255	0
255	0	0	0	0	0
255	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

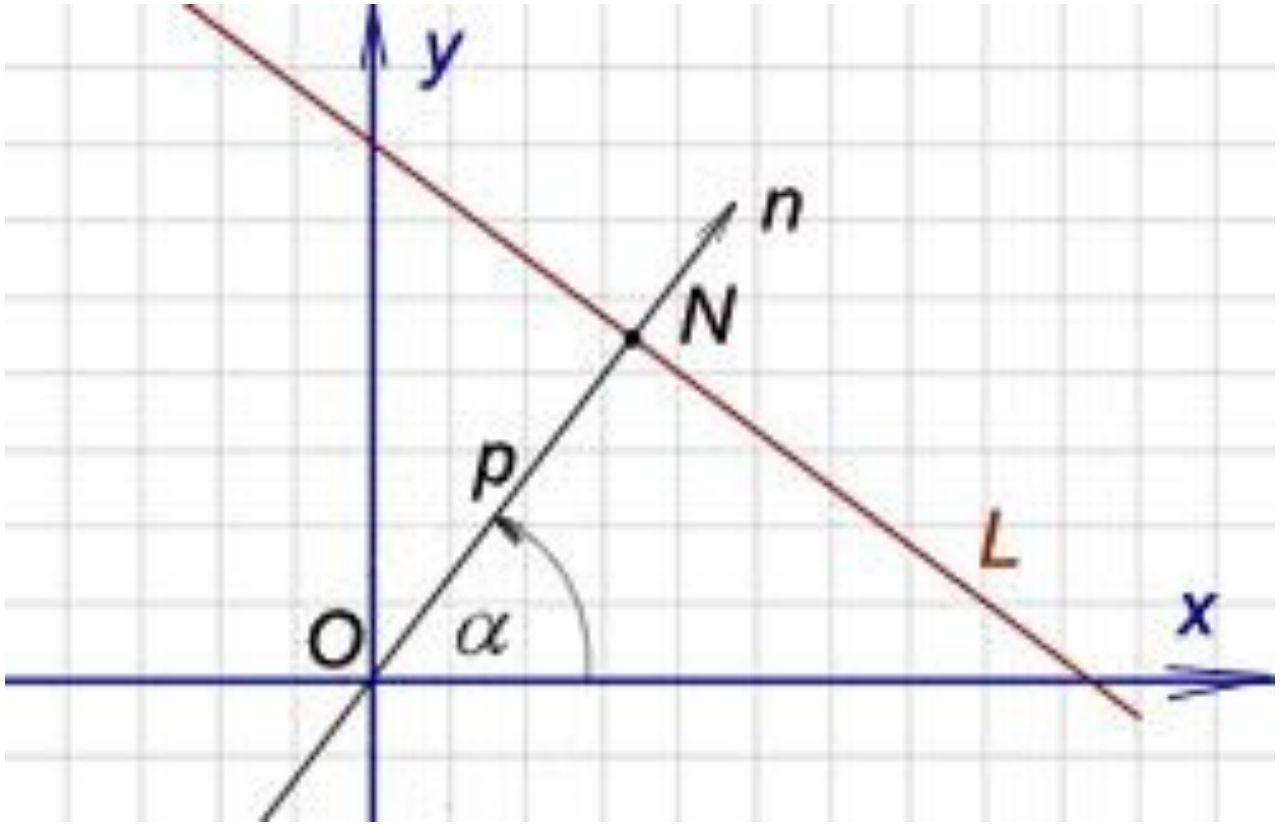




3. Преобразование Хафа для линий

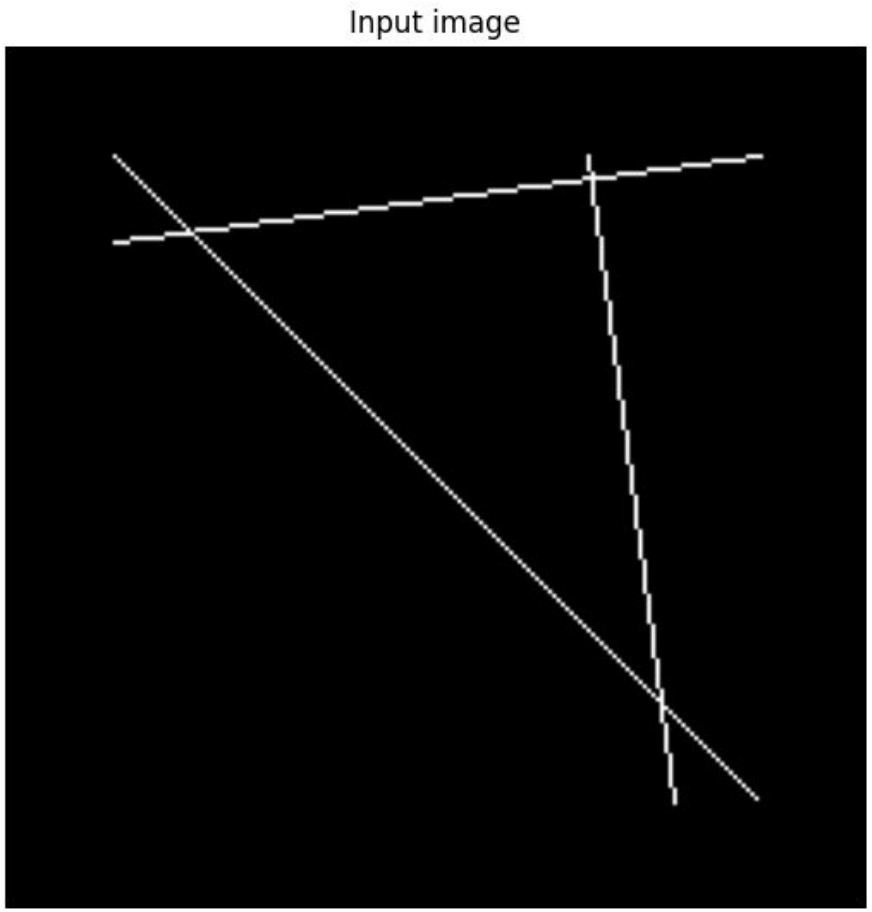
Нормальное уравнение прямой:

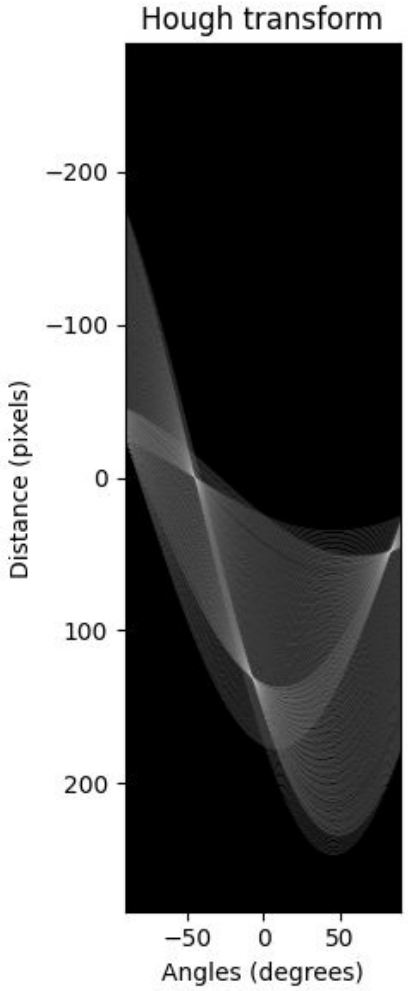
$x * \cos(a) + y * \sin(a) = p$

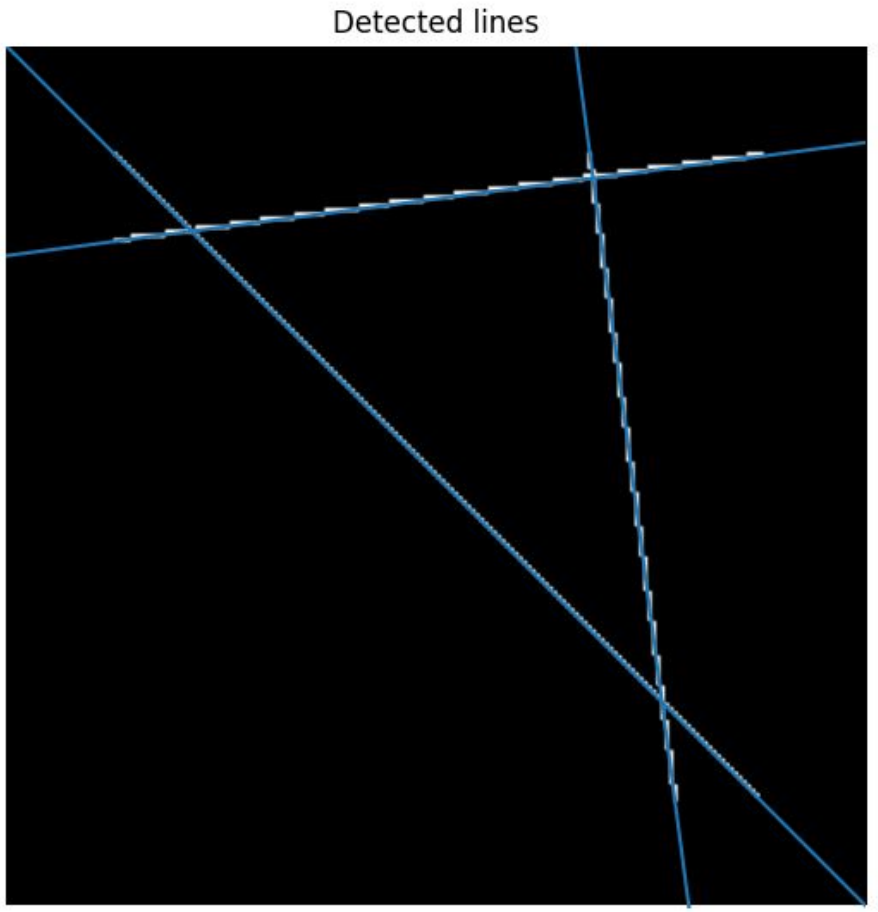


3. Преобразование Хафа для линий

4. Поиск пиков в преобразовании Хафа







5. Поиск угла

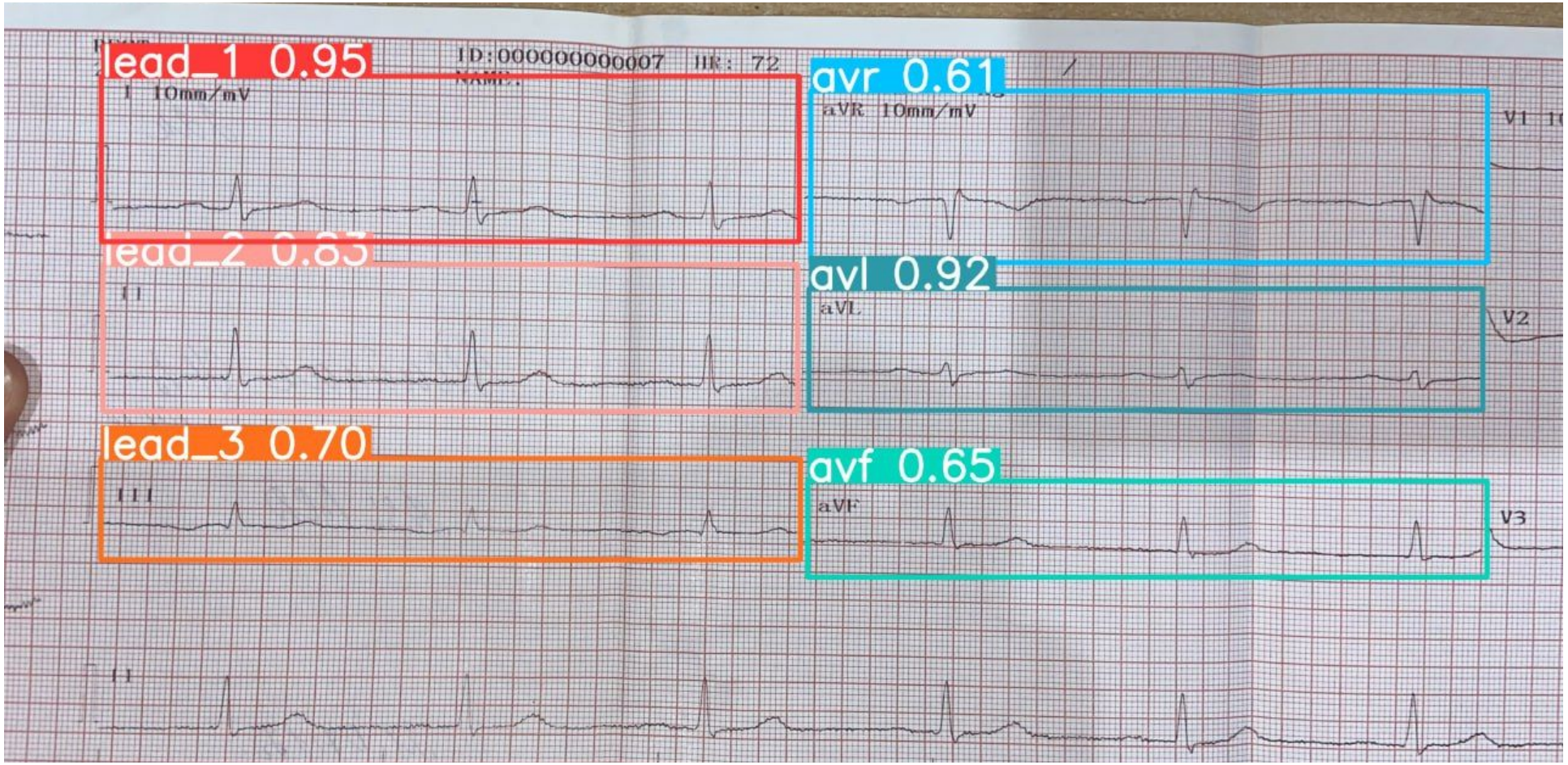
1. Перевести все углы в градусы
2. Формирование частотного словаря для каждого угла
3. Выбор наиболее часто встречающегося

Детекция

Гипотезы:

1. Достаточно ли YOLOv8n?
2. Улучшится ли качество, если добавить в рамку название класса?

Детекция


lead_1 0.95
avr 0.61
avl 0.92
lead_3 0.70
avf 0.65
ID:000000000007
HR: 72
I 10mm/mV
aVR 10mm/mV
aVL
aVF
II
III
V1
V2
V3

```
) PS C:\fouth year\ecg\ECGyolo train> yolo task=detect mode=val data=data.yaml model
ytics YOLOv8.2.28 🚀 Python-3.11.7 torch-2.3.1+cu121 CUDA:0 (NVIDIA GeForce RTX 3060
 summary (fused): 168 layers, 3007988 parameters, 0 gradients, 8.1 GFLOPs
canning C:\fouth year\ecg\ECGyolo train\datasets\ecg_dataset\my_yolo_2\valid\labels.
               Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                 all          5         27      0.912      0.872      0.967      0.698
              lead_1          2          2      0.797          1      0.995      0.846
              lead_2          2          2      0.838          1      0.995      0.746
              lead_3          2          2      0.766        0.5      0.745      0.472
                  v1          3          3          1      0.844      0.995      0.698
                  v2          3          3          1      0.834      0.995      0.697
                  v3          3          3      0.877      0.667      0.913      0.723
                  v4          2          2      0.778          1      0.995      0.821
                  v5          2          2      0.933          1      0.995      0.846
                  v6          2          2          1      0.975      0.995      0.657
                 avf          2          2          1      0.694      0.995      0.452
                 avl          2          2          1      0.955      0.995      0.622
                 avr          2          2      0.954          1      0.995      0.796
 1.6ms preprocess, 33.9ms inference, 0.0ms loss, 20.8ms postprocess per image
```

```
tralytics YOLOv8.2.28 🚀 Python-3.11.7 torch-2.3.1+cu121 CUDA:0 (NVIDIA GeForce RTX 3060, 12288MiB)
del summary (fused): 168 layers, 3007988 parameters, 0 gradients, 8.1 GFLOPs
  Scanning C:\fouth year\ecg\ECGyolo train\datasets\ecg_dataset\my_yolo_merged\test\labels... 17 ima
  New cache created: C:\fouth year\ecg\ECGyolo train\datasets\ecg_dataset\my_yolo_merged\test\labels
               Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                 all         17         99      0.865      0.809      0.863      0.489
              lead_1          7          7      0.763      0.857      0.735      0.356
              lead_2          7          7        0.8      0.857      0.856      0.368
              lead_3          7          7      0.643      0.714      0.758      0.209
                  v1         10         10      0.938        0.9      0.986      0.567
                  v2         10         10      0.943        0.9      0.986       0.64
                  v3         10         10          1        0.8      0.945      0.689
                  v4          7          7      0.966      0.857      0.978      0.594
                  v5          7          7      0.922          1      0.995      0.554
                  v6          7          7          1      0.858      0.995      0.578
                 avf          9          9          1      0.521      0.963      0.531
                 avl          9          9      0.654      0.667      0.483      0.363
                 avr          9          9      0.748      0.778      0.681      0.417
eed: 13.5ms preprocess, 29.2ms inference, 0.0ms loss, 6.6ms postprocess per image
```

Источники

- Canny - edge detection https://www.youtube.com/watch?v=PtSgA19sC5g
- Skimage docs https://scikit-image.org/docs/stable/auto_examples/edges/plot_line_hough_transform.html
- OpenCV docs
- https://www.youtube.com/watch?v=Tz0SSOVF59A