



**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина: ТЕХНИЧЕСКОЕ ЗРЕНИЕ**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

**“Сегментация изображений”**

Выполнили:

**Новичков Дмитрий, ТЕХ.ЗРЕНИЕ 1.1**

Преподаватель:

**Шаветов С. В.**

Санкт-Петербург, 2024

# Содержание

<b>1. Бинаризация</b>	<b>2</b>
1.1. Бинаризация по порогу . . . . .	2
1.2. Бинаризация по двойному порогу . . . . .	4
<b>2. Сегментация изображений</b>	<b>7</b>
2.1. На основе принципа Вебера . . . . .	7
2.2. Сегментация изображения в пространстве CIE Lab по методу k-средних . . . . .	9
<b>3. Текстурная сегментация</b>	<b>12</b>
3.1. Оценка параметров текстур . . . . .	16
<b>4. Ответы на вопросы</b>	<b>16</b>

Исходный код на [GitHub](#).

Для удобства отладки и сборки используется средство автоматизации сборки ПО CMake:

```
cmake_minimum_required(VERSION 2.8)

project( CV_LW4 )
find_package( OpenCV REQUIRED )

include_directories( ${OpenCV_INCLUDE_DIRS} )
add_executable( ${PROJECT_NAME} src/lab2.cpp )

target_link_libraries( ${PROJECT_NAME} ${OpenCV_LIBS} )
```

Листинг 1: CMakeLists.txt для сборки проекта

## 1. Бинаризация

### 1.1. Бинаризация по порогу

Простейшим способом сегментации изображения на два класса (фоновые пиксели и пиксели объекта) является бинаризация. Бинаризацию можно выполнить по порогу или по двойному порогу. В первом случае:

$$I_{new}(x, y) = \begin{cases} 0, & I(x, y) \leq t \\ 1, & I(x, y) > t, \end{cases} \quad (1)$$

где  $I$  — исходное изображение,  $I_{new}$  — бинаризованное изображение,  $t$  — порог бинаризации.

```
cv::threshold(image, image_new, 127, 255, cv::THRESH_BINARY);

cv::imwrite(path + "/outputs/image_binarization.png", image_new);
cv::imshow("image", image_new);
cv::waitKey();
```

Листинг 2: Исходный код для бинаризации



Рис. 1: Исходное изображение



Рис. 2: Бинаризованное изображение

## 1.2. Бинаризация по двойному порогу

$$I_{new}(x, y) = \begin{cases} 0, & I(x, y) \leq t_1 \\ 1, & t_1 < I(x, y) \leq t_2, \\ 1, & I(x, y) > t_2, \end{cases} \quad (2)$$

где  $I$  — исходное изображение,  $I_{new}$  — бинаризованное изображение,  $t_1$  и  $t_2$  — верхний и нижний пороги бинаризации.

```
cv::threshold(image, image_new, 127, 255, cv::THRESH_TOZERO);
cv::threshold(image_new, image_new, 200, 255, cv::THRESH_BINARY);

cv::imwrite(path + "/outputs/image_double_binarization.png", image_new);
cv::imshow("image", image_new);
cv::waitKey();
```

Листинг 3: Исходный код для бинаризации по двойному порогу



Рис. 3: Бинаризованное изображение по двойному порогу

```
cv::threshold(image, image_new, 0, 255, cv::THRESH_OTSU);  
  
cv::imwrite(path + "/outputs/image_otsu_binarization.png", image_new);  
cv::imshow("image", image_new);  
cv::waitKey();
```

Листинг 4: Исходный код для бинаризации по двойному порогу, вычисленному статистическим методом Отсу



Рис. 4: Бинаризованное изображение по двойному порогу, вычисленному по статистическому методу Отсу

```
cv::threshold(image, image_new, 127, 255, cv::THRESH_TOZERO);  
cv::threshold(image_new, image_new, 200, 255, cv::THRESH_BINARY);  
  
cv::imwrite(path + "/outputs/image_double_binarization.png", image_new);  
cv::imshow("image", image_new);  
cv::waitKey();
```

Листинг 5: Исходный код для адаптивной бинаризации

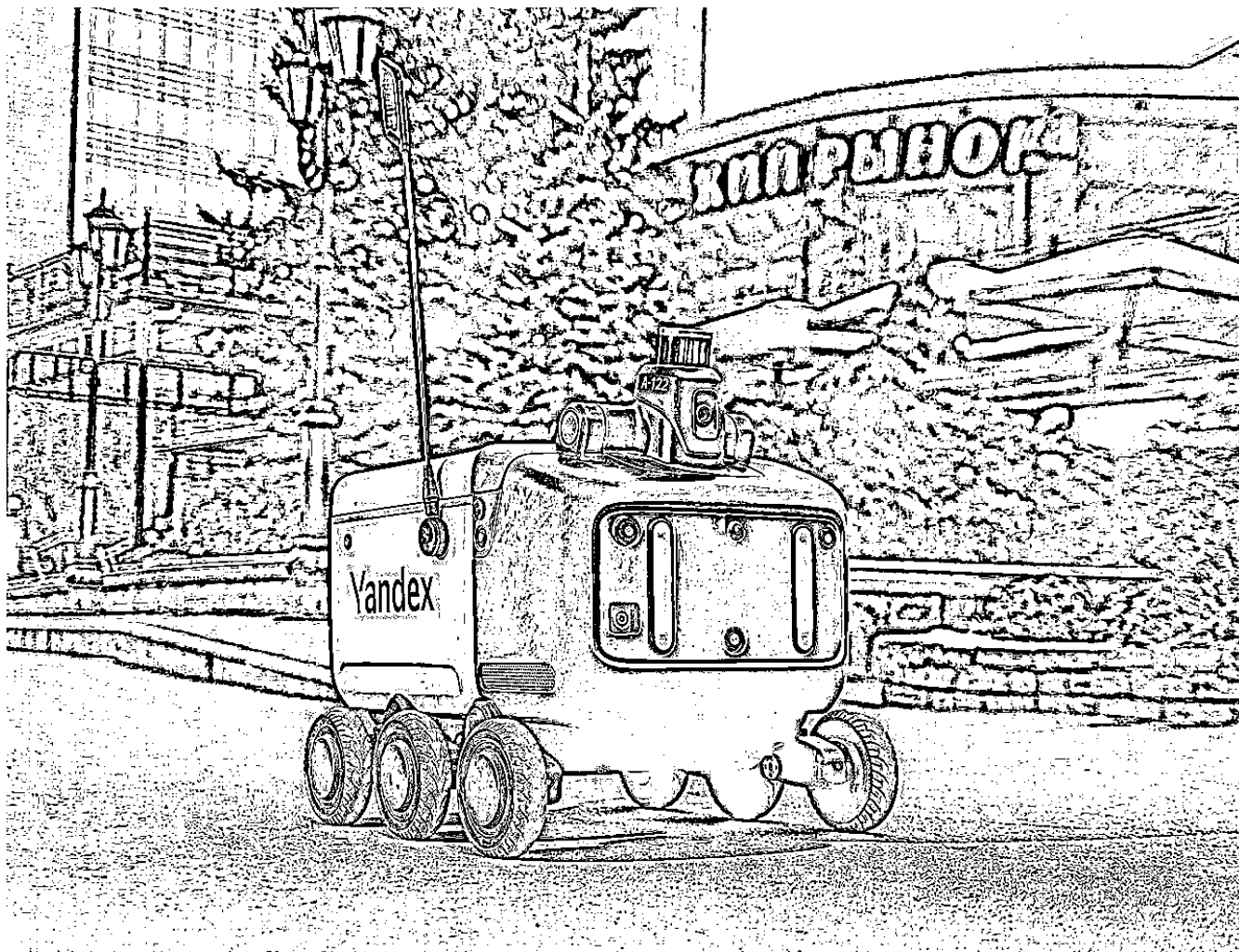


Рис. 5: Бинаризованное изображение адаптивным методом

## 2. Сегментация изображений

### 2.1. На основе принципа Вебера

Алгоритм предназначен для сегментации полутоновых изображений. Принцип Вебера подразумевает, что человеческий глаз плохо воспринимает разницу уровней серого между  $I(n)$  и  $I(n) + W(I(n))$ , где  $W(I(n))$  — функция Вебера,  $n$  — номер класса,  $I$  — кусочно-нелинейная функция градаций серого. Функция Вебера может быть вычислена по формуле:

$$W(I) = \begin{cases} 20 - \frac{12I}{88}, & 0 \leq I \leq 88, \\ 0.002(I - 88)^2, & 88 \leq I \leq 138, \\ \frac{7(I-138)}{117} + 13, & 138 \leq I \leq 255 \end{cases} \quad (3)$$





Рис. 6: Исходное изображение с лицом для сегментации

```
image = cv::imread(path + "/source/face.png", 0);
image_new = image.clone();

for(int i = 0; i < image_new.rows; ++i){
    for(int j = 0; j < image_new.cols; ++j){
        if(image_new.at<uchar>(i, j) <= 88)
            image_new.at<uchar>(i, j) = 20 - 12 * image_new.at<uchar>(i, j)
            / 88;

        else if(image_new.at<uchar>(i, j) <= 138)
            image_new.at<uchar>(i, j) = 0.002 *
            cv::pow(image_new.at<uchar>(i, j) - 88, 2);

        else if(image_new.at<uchar>(i, j) <= 255)
            image_new.at<uchar>(i, j) = 7 * (image_new.at<uchar>(i, j) -
            138) / 117;
    }
}

cv::imwrite(path + "/outputs/image_veber_segmentation.png", image_new);
cv::imshow("image", image_new);
```

```
cv::waitKey();
```

Листинг 6: Исходный код для сегментации по принципу Вебера



Рис. 7: Сегментация изображения по принципу Вебера

Вероятно, из-за оттенка фона и светлого цвета машины лицо не удалось сегментировать по выбранному алгоритму

## 2.2. Сегментация изображения в пространстве CIE Lab по методу $k$ -средних

Идея метода заключается в определении центров  $k$ -кластеров и отнесении к каждому кластеру пикселей, наиболее близко относящихся к этим центрам. Все пиксели рассматриваются как векторы  $x_i$ ,  $i = \overline{1, p}$ . Алгоритм сегментации состоит из следующих шагов:

1. Определение случайным образом  $k$  векторов  $m_j$ ,  $j = \overline{1, k}$ , которые объявляются начальными центрами кластеров.
2. Обновление значений средних векторов  $m_j$  путем вычисления расстояний от каждого вектора  $x_i$  до каждого  $m_j$  и их классификации по критерию минимальности расстояния от вектора до кластера, пересчет средних значений  $m_j$  по всем кластерам.
3. Повторение шагов 2 и 3 до тех пор, пока центры кластеров не перестанут изменяться.



Рис. 8: Исходное изображение из KITTI Dataset

```
image = cv::imread(path + "/source/road.png", 1);

cv::Mat image_CIE_Lab, ab;
cv::cvtColor(image, image_CIE_Lab, cv::COLOR_BGR2Lab);

std::vector<cv::Mat> image_Lab_channels;
cv::split(image_CIE_Lab, image_Lab_channels);

cv::merge(&(image_Lab_channels[1]), 2, ab);
ab = ab.reshape(0, 1);
ab.convertTo(ab, CV_32F);

int k = 3;
cv::Mat labels;
cv::kmeans(ab, k, labels, cv::TermCriteria(cv::TermCriteria::EPS +
                                           cv::TermCriteria::COUNT, 10, 1.0), 10,
           cv::KMEANS_RANDOM_CENTERS);

labels = labels.reshape(0, image_Lab_channels[0].rows);

std::vector<cv::Mat> segmentedFrames;
for(int i = 0; i < k; ++i){
    cv::Mat image_tmp = cv::Mat::zeros(image.rows, image.cols,
    image.type());
    image.copyTo(image_tmp, labels == i);
    segmentedFrames.push_back(image_tmp);
}
```



```

}

cv::imwrite(path + "/outputs/image_CIELab_segmentation.png",
    segmentedFrames[2]);

cv::threshold(segmentedFrames[2], segmentedFrames[2], 90, 115,
    cv::THRESH_BINARY);
cv::imwrite(path + "/outputs/image_road_segmentation.png",
    segmentedFrames[2]);

cv::imshow("image", segmentedFrames[2]);
cv::waitKey();

```

Листинг 7: Исходный код для сегментации изображения изображения в пространстве CIE Lab по методу k-средних

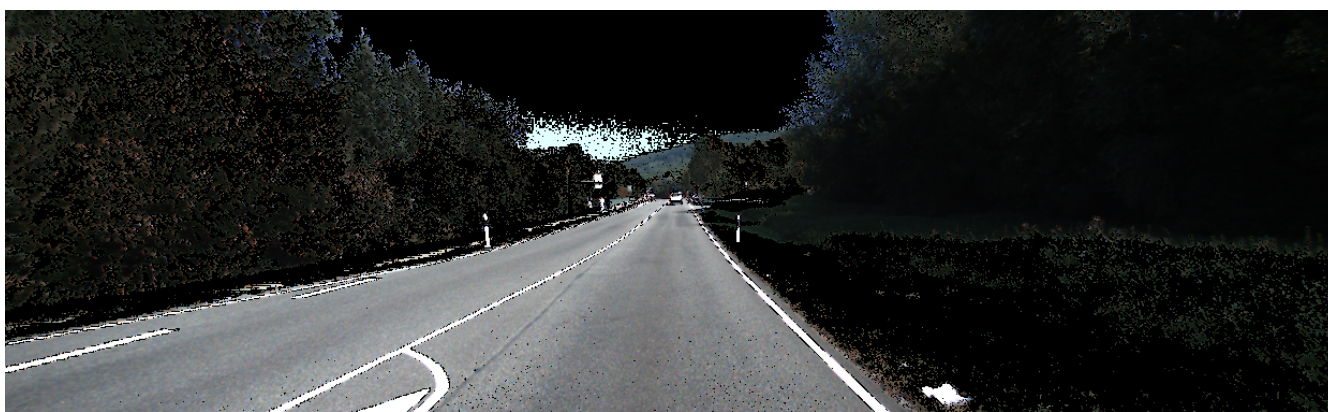


Рис. 9: Сегментация изображения в пространстве CIE Lab по методу k-средних (составляющая синего оттенка)

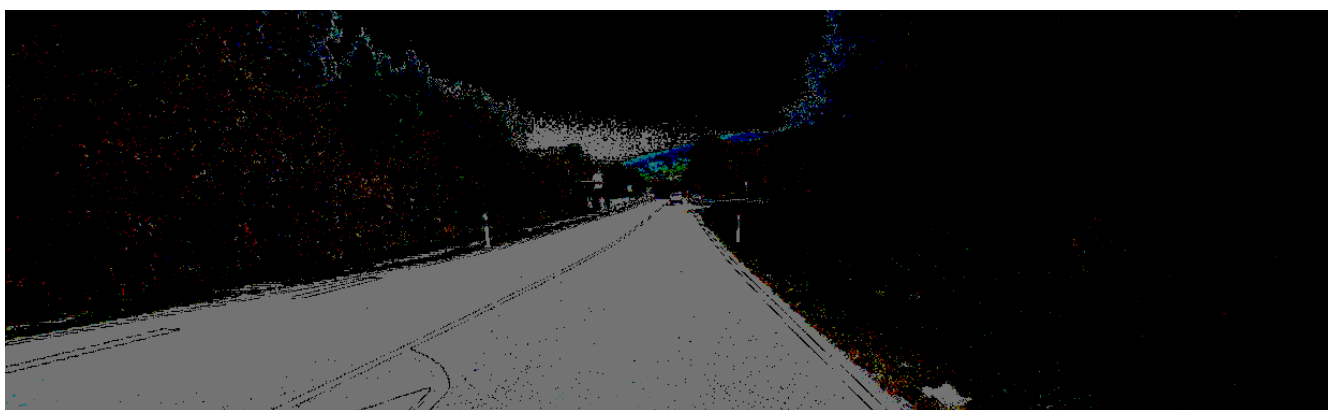


Рис. 10: Бинаризация предыдущего изображения для выделения дороги

Если убрать шумы, то будет очень хорошо сегментированная дорога. В случае, когда она пустая, никакие нейронки для сегментации и не нужны.

### 3. Текстурная сегментация

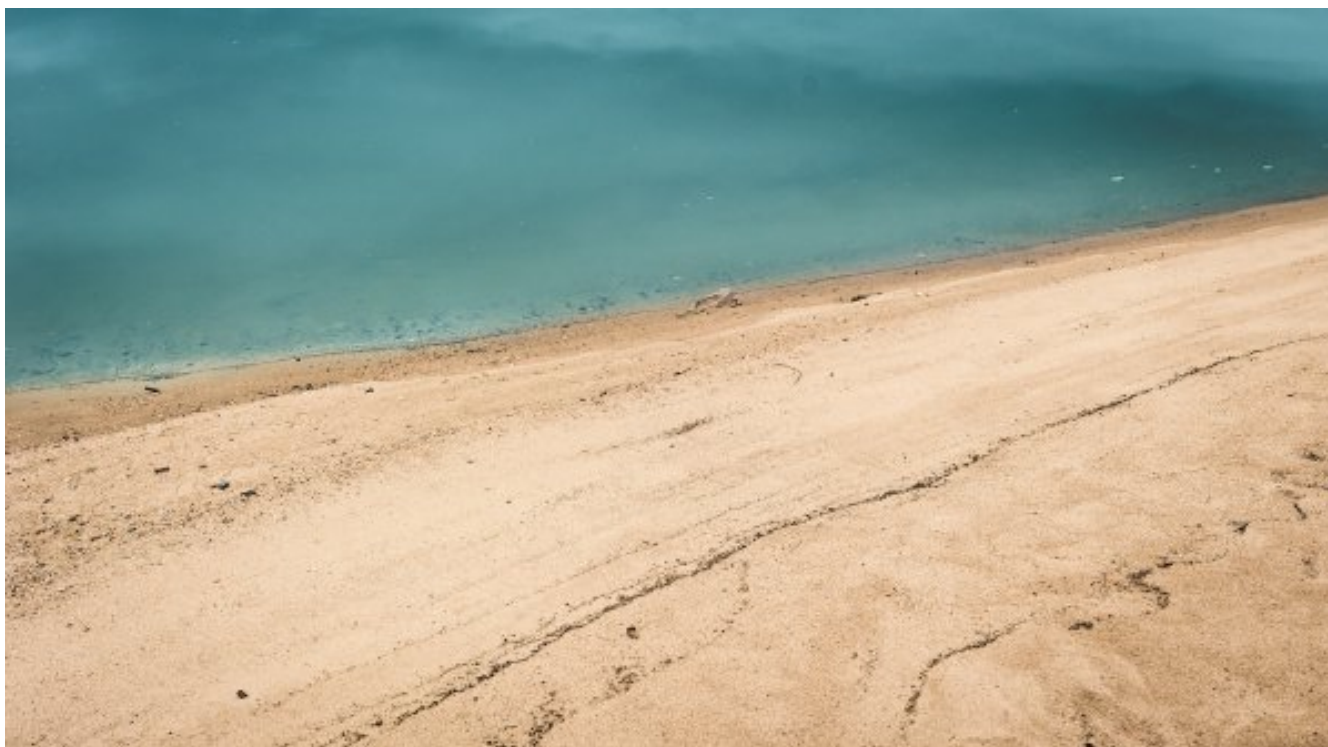


Рис. 11: Исходное изображение

```

image = cv::imread(path + "/source/sea.png", 0);

cv::Mat E, Eim;
cv::Mat el = cv::getStructuringElement(cv::MORPH_RECT, cv::Size(9, 9));
entropy(image, E, el);

double Emin, Emax;
cv::minMaxLoc(E, &Emin, &Emax);
Eim = (E - Emin) / (Emax - Emin);
Eim.convertTo(Eim, CV_8U, 255);
cv::Mat BW1;
cv::threshold(Eim, BW1, 0, 255, cv::THRESH_OTSU);

cv::imwrite(path + "/outputs/image_entropy_segmentation.png", BW1);

cv::Mat BWao, closeBWao, Mask1;
bwareaopen(BW1, BWao, 50000);
cv::imwrite(path + "/outputs/image_bwareoopen_segmentation.png", BWao);

cv::Mat nhood = cv::getStructuringElement(cv::MORPH_RECT, cv::Size(9, 9));
cv::morphologyEx(BWao, closeBWao, cv::MORPH_CLOSE, nhood);

imfillholes(closeBWao, Mask1);
cv::imwrite(path + "/outputs/image_imclose_segmentation.png", Mask1);

std::vector<std::vector<cv::Point>> contours;
cv::findContours(Mask1, contours, cv::RETR_TREE, cv::CHAIN_APPROX_NONE);
cv::Mat boundary = cv::Mat::zeros(Mask1.rows, Mask1.cols, CV_8UC1);

cv::drawContours(boundary, contours, -1, 255, 1);
cv::imwrite(path + "/outputs/image_countours_segmentation.png", boundary);

cv::Mat segmentResults = image.clone();
segmentResults.setTo(cv::Scalar(255), boundary != 0);
cv::imwrite(path + "/outputs/image_segmentResults_segmentation.png",
    segmentResults);

cv::Mat I2 = image.clone();
I2.setTo(0, Mask1 != 0);

```

Листинг 8: Текстурная сегментация



Рис. 12: Бинаризованное изображение



Рис. 13: Результат выполнения функции `bwareaopen()`



Рис. 14: Результат выполнения функции `imclose()`

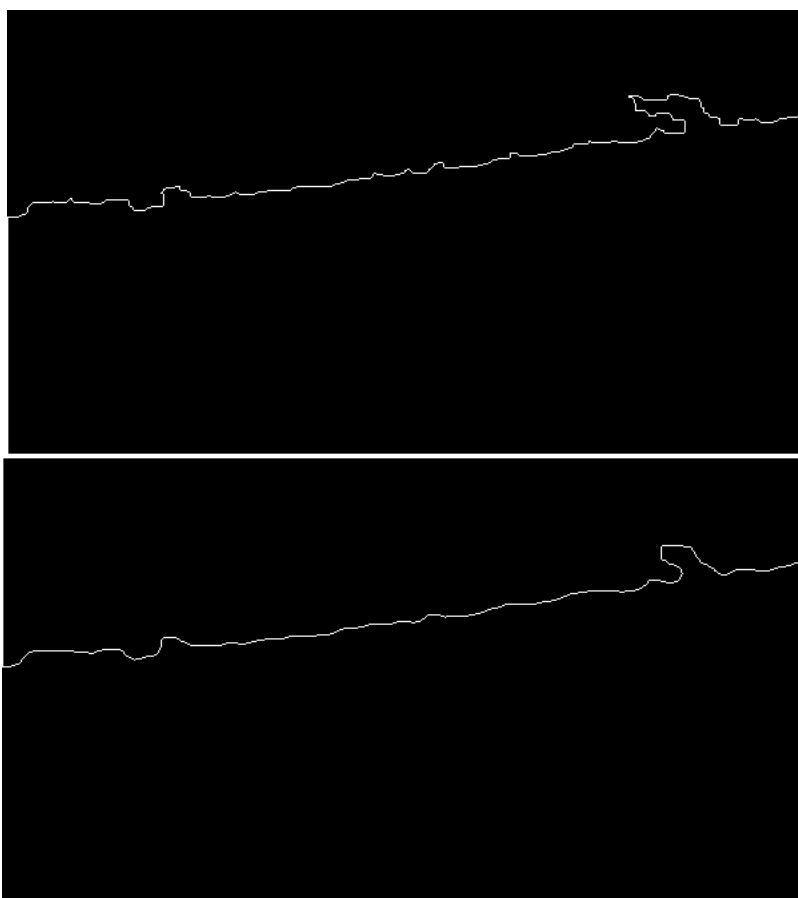


Рис. 15: Выделенные границы текстур относительно воды и суши соответственно



### 3.1. Оценка параметров текстур

#### 1. Текстура воды

Среднее значение случайной величины = 167.7429504395

Стандартное отклонение = 26.3628025055

R - относительная гладкость = 0.0106576710

Энтропия гистограммы = 19.4059734344

#### 2. Текстура суши

Среднее значение случайной величины = 163.2591247559

Стандартное отклонение = 32.4539321091

R - относительная гладкость = 0.0142642801

Энтропия гистограммы = 21.3457216344

## 4. Ответы на вопросы

Q1. В каких случаях целесообразно использовать сегментацию по принципу Вебера?

A1. Сегментация изображения по принципу Вебера целесообразно использовать в случаях, когда необходимо выделить объекты с малыми различиями в яркости или цвете от фона. Например, в медицинской диагностике для выделения сосудов на рентгеновских изображениях или для распознавания сложных текстур и узоров на изображениях в области компьютерного зрения. Этот метод эффективно работает при анализе изображений с низким контрастом или сложной структурой.

Q2. Какие значения имеют цветовые координаты a и b цветового пространства CIE Lab в полутоновом изображении?

A2. Значения цветовых координат a и b цветового пространства CIE Lab представляют собой две координаты, определяющие оттенок цвета. Значения этих координат могут изменяться от -128 до 127, где нулевое значение обозначает нейтральный серый цвет. Положительные значения указывают на красные и зеленые оттенки, а отрицательные - на голубые и желтые оттенки. В полутоновых изображениях, которые имеют только один канал интенсивности, координаты a и b всегда равны нулю, что указывает

на отсутствие цвета. Это потому, что полутоновые изображения представляют собой градации серого, а не настоящий цвет.

Q3. Зачем производить сегментацию в цветовом пространстве CIE Lab, а не в исходном RGB?

A3. Сегментация в цветовом пространстве CIE Lab предпочтительнее, чем в RGB, по следующим причинам:

CIE Lab - это универсальное цветовое пространство, которое более близко соответствует восприятию цвета человеческим глазом. Оно разделяет цвет и яркость, что делает его более подходящим для анализа и обработки цвета.

2. В CIE Lab цвета линейно распределены, что упрощает вычисления и обработку цветов. Также это пространство является независимым от устройства, что позволяет достичь более надежных результатов при анализе цвета.

3. CIE Lab позволяет более точно разделять различные цвета и текстуры в изображении, что делает сегментацию более эффективной и точной.

Таким образом, проведение сегментации в цветовом пространстве CIE Lab позволяет добиться более качественных и точных результатов, чем при использовании исходного RGB.

Q4. Что такое цветовое пространство и цветовой охват?

A4. Цветовое пространство – это система координат, которая определяет цвет каждой точки в изображении. Цветовой охват, или цветовая гамма, обозначает диапазон цветов, которые могут быть воспроизведены или отображены в данном цветовом пространстве. Например, RGB (красный, зеленый, синий) – одно из наиболее популярных цветовых пространств, которое включает в себя определенный диапазон цветов, которые могут быть созданы путем комбинирования этих трех основных цветов.