Лабораторная работа №5
Сегментация изображений

Цель работы

Освоение основных способов сегментации изображений на семантические области.

Методические рекомендации

До начала работы студенты должны ознакомиться с основными функциями среды MATLAB по преобразованию цветовых пространств изображений и способами определения порогов. Лабораторная работа рассчитана на 4 часа.

Теоретические сведения

Бинаризация изображений

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:

$$I_{new}(x,y) = \begin{cases} 0, I(x,y) \leqslant t, \\ 1, I(x,y) > t, \end{cases} \tag{5.1}$$

где I — исходное изображение, I_{new} — бинаризованное изображение, t — порог бинаризации. Бинаризация данным методом в среде MATLAB может быть выполнена с использованием функций `im2bw()` (устаревшая) или `imbinarize()`.

Листинг 5.1. Бинаризация.

```
1 I = imread('pic.jpg');
2 L = 255;
3 t = 127 / L; %norm to 0...1
4 Inew = im2bw(I, t);
```

Бинаризация по двойному порогу (диапазонная бинаризация):

$$I_{new}(x,y) = \begin{cases} 0, I(x,y) \leqslant t_1, \\ 1, t_1 < I(x,y) \leqslant t_2, \\ 0, I(x,y) > t_2, \end{cases} \tag{5.2}$$

где I — исходное изображение, I_{new} — бинаризованное изображение, t_1 и t_2 — верхний и нижний пороги бинаризации. Бинаризация данным методом в среде MATLAB может быть выполнена с использование функции `roicolor()`. Для преобразования полноцветного изображения в полутоновое можно предварительно воспользоваться функцией `rgb2gray()`.

Листинг 5.2. Бинаризация по двойному порогу.

```
1 I = imread('pic.jpg');
2 t1 = 110;
3 t2 = 200;
4 Igray = rgb2gray(I);
5 Inew = roicolor(Igray, t1, t2);
```

Пороги бинаризации t, t_1 и t_2 могут быть либо заданы вручную, либо вычислены с помощью специальных алгоритмов. В случае автоматического вычисления порога можно воспользоваться следующими алгоритмами.

1. Поиск максимального I_{max} и минимального I_{min} значений интенсивности исходного полутонового изображения и нахождение их среднего арифметического. Среднее арифметическое будет являться глобальным порогом бинаризации t:

$$t = \frac{I_{max} - I_{min}}{2}. \tag{5.3}$$

2. Поиск оптимального порога t на основе модуля градиента яркости каждого пикселя. Для этого сначала вычисляется модуль градиента в каждой точке (x,y):

$$G(x,y) = \max\left\{|I(x+1,y) - I(x-1,y)|, |I(x,y+1) - I(x,y-1)|\right\}, \tag{5.4}$$

затем вычисляется оптимальный порог t:

$$t = \frac{\sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} I(x,y)G(x,y)}{\sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} G(x,y)}. \tag{5.5}$$

3. Вычисление оптимального порога t статистическим методом Отсу (Оцу, англ. Otsu), разделяющим все пиксели на два класса 1 и 2, минимизируя дисперсию внутри каждого класса $\sigma_1^2(t)$ и $\sigma_2^2(t)$ и максимизируя дисперсию между классами.

Алгоритм вычисления порога методом Отсу:

1. Вычисление гистограммы интенсивностей изображения и вероятности $p_i = \dfrac{n_i}{N}$ для каждого уровня интенсивности, где n_i — число пикселей с уровнем интенсивности i, N — число пискселей в изображении.

2. Задание начального порога $t = 0$ и порога $k \in (0, L)$, разделяющего все пиксели на два класса, где L — максимальное значение интенсивности изображения. В цикле для каждого значения порога от $k = 1$ до $k = L - 1$:

   (a) Вычисление вероятностей двух классов $\omega_j(0)$ и средних арифметических $\mu_j(0)$, где $j = \overline{1,2}$:

$$\omega_1(k) = \sum_{s=0}^{k} p_s, \tag{5.6}$$

$$\omega_2(k) = \sum_{s=k+1}^{L} p_s = 1 - \omega_1(k), \tag{5.7}$$

$$\mu_1(k) = \sum_{s=0}^{k} \frac{s \cdot p_s}{\omega_1}, \tag{5.8}$$

$$\mu_2(k) = \sum_{s=k+1}^{L} \frac{s \cdot p_s}{\omega_2}. \tag{5.9}$$

   (b) Вычисление межклассовой дисперсии $\sigma_b^2(k)$:

$$\sigma_b^2(k) = \omega_1(k)\omega_2(k)(\mu_1(k) - \mu_2(k))^2. \tag{5.10}$$

(c) Если вычисленное значение $\sigma_b^2(k)$ больше текущего значения t, то присвоить порогу значение межклассовой дисперсии $t = \sigma_b^2(k)$.

3. Оптимальный порог t соответствует максимуму $\sigma_b^2(k)$.

В среде MATLAB порог t методом Отсу может быть вычислен с использованием функции `graythresh()`:

Листинг 5.3. Бинаризация методом Отсу.

```
I = imread('pic.jpg');
t = graythresh(I);
Inew = im2bw(I, t);
```

либо с использованием функции `otsuthresh()` на основе гистограммы изображения:

Листинг 5.4. Бинаризация методом Отсу на основе гистограммы.

```
I = imread('pic.jpg');
Igray = rgb2gray(I);
[counts,x] = imhist(Igray);
t = otsuthresh(counts);
Inew = imbinarize(Igray, t);
```

4. Адаптивные методы, работающие не со всем изображением, а лишь с его фрагментами. Такие подходы зачастую используются при работе с изображениями, на которых представлены неоднородно освещенные объекты. В среде MATLAB порог t адаптивным методом может быть вычислен при помощи функции `adaptthresh()`:

Листинг 5.5. Бинаризация адаптивным методом.

```
I = imread('pic.jpg');
Igray = rgb2gray(I);
t = adaptthresh(Igray);
Inew = imbinarize(Igray, t);
```

Помимо рассмотренных методов существуют и многие другие, например методы Бернсена, Эйквела, Ниблэка, Яновица и Брукштейна и др.

Сегментация изображений

Рассмотрим несколько основных методов сегментации изображений.

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:

$$W(I) = \begin{cases} 20 - \dfrac{12I}{88}, 0 \leqslant I \leqslant 88, \\ 0{,}002(I - 88)^2, 88 < I \leqslant 138, \\ \dfrac{7(I - 138)}{117} + 13, 138 < I \leqslant 255. \end{cases} \tag{5.11}$$

Можно объединить уровни серого из диапазона $[I(n),I(n) + W(I(n))]$ заменив их одним значением интенсивности.

Алгоритм сегментации состоит из следующих шагов:

1. Инициализация начальных условий: номер первого класса $n = 1$, уровень серого $I(n) = 0$.

2. Вычисление значения $W(I(n))$ по формуле Вебера и присваивание значения $I(n)$ всем пикселям, интенсивность которых находится в диапазоне $[I(n),I(n) + W(I(n))]$.

3. Поиск пикселей с интенсивностью выше $G = I(n) + W(I(n)) + 1$. Если найдены, то увеличение номера класса $n = n + 1$, присваивание $I(n) = G$ и переход на второй шаг. В противном случае закончить работу. Изображение будет сегментировано на n классов интенсивностью $W(I(n))$.

Сегментация RGB-изображений по цвету кожи

Общим приницпом данного подхода является определение критерия близости интенсивности пикселей к оттенку кожи. Аналитически описать *оттенок кожи* довольно затруднительно, поскольку

его описание базируется на человеческом восприятии цвета, меняется при изменении освещения, отличается у разных народностей, и т.д.

Существует несколько аналитических описаний для изображений в цветовом пространстве RGB, позволяющих отнести пиксель к классу «кожа» при выполнении условий:

$$\begin{cases} R > 95, \\ G > 40, \\ B < 20, \\ \max R,G,B - \min R,G,B > 15, \\ |R - G| > 15, \\ R > G, \\ R > B, \end{cases} \tag{5.12}$$

или

$$\begin{cases} R > 220, \\ G > 210, \\ B > 170, \\ |R - G| \leqslant 15, \\ G > B, \\ R > B, \end{cases} \tag{5.13}$$

или

$$\begin{cases} r = \frac{R}{R+G+B}, \\ g = \frac{G}{R+G+B}, \\ b = \frac{B}{R+G+B}, \\ \frac{r}{g} > 1{,}185, \\ \frac{rb}{(r+g+b)^2} > 0{,}107, \\ \frac{rg}{(r+g+b)^2} > 0{,}112. \end{cases} \tag{5.14}$$

На основе цветового пространства CIE Lab

В цветовом пространстве **Lab** значение светлоты отделено от значения хроматической составляющей цвета (тон, насыщенность).

Светлота задается координатой $\boldsymbol{L}$, которая может находиться в диапазоне от 0 (темный) до 100 (светлый). Хроматическая составляющая цвета задается двумя декартовыми координатами $\boldsymbol{a}$ (означает положение цвета в диапазоне от *зеленого* (-128) до *красного* (127)) и $\boldsymbol{b}$ (означает положение цвета в диапазоне от *синего* (-128) до *желтого* (127)). Бинарное изображение получается при нулевых значениях координат $\boldsymbol{a}$ и $\boldsymbol{b}$. Идея алгоритма состоит в разбиении цветного изображения на сегменты доминирующих цветов.

В качестве исходных данных выберем следующее цветное изображение:

Рис. 5.1 — Исходное цветное изображение.

В первую очередь, чтобы уменьшить влияние освещенности на результат сегментации, преобразуем полноцветное изображение из цветового пространства **RGB** в пространство **Lab**. Для этого в среде MATLAB используется функция `rgb2lab()`.

Листинг 5.6. Сегментация на основе цветового пространства **Lab**.

```
1 I = imread('pic2.jpg');
2 Ilab = rgb2lab(I);
3 L = Ilab(:,:,1);
4 a = Ilab(:,:,2);
5 b = Ilab(:,:,3);
```

На следующем шаге необходимо определить количество цветов, на которые будет сегментировано изображение, и задать области, содержащие пиксели примерно одного цвета. Области можно задать для каждого цвета интерактивно в виде многоугольников при помощи функции `roipoly()`:

```
6 numColors = 3;
7 sampleAreas = false([size(I, 1)
```

```
 8      size(I, 2) numColors]);
 9 for i=1:1:numColors
10      [BW, xi, yi] = roipoly(I);
11      sampleAreas(:,:,i) = roipoly(I, xi, yi);
12 end
```

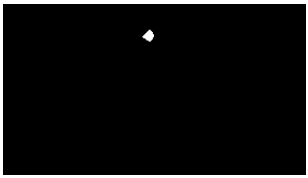


Рис. 5.2 — Пример выделенной красной области из четырех точек.

После этого требуется определить цветовые метки для каждого из сегментов путем расчета среднего значения цветности в каждой выделенной области. Средние значения можно вычислить при помощи функции `mean2()`:

```
13 colorMarks = zeros([numColors, 2]);
14 for i=1:1:numColors
15      colorMarks(i,1) = ...
16          mean2(a(sampleAreas(:,:,i)));
17      colorMarks(i,2) = ...
18          mean2(b(sampleAreas(:,:,i)));
19 end
```

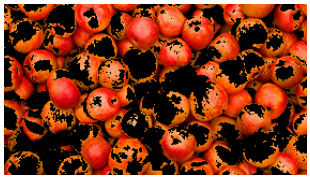
Затем используем принцип ближайшей окрестности для классификации пикселей путем вычисления евклидовых метрик между пикселями и метками: чем меньше расстояние до метки, тем лучше пиксель соответствует данному сегменту. Евклидова метрика по двум цветовым координатам расчитывается по формуле: $\sqrt{(a(x,y) - a_{mark})^2 + (b(x,y) - b_{mark})^2}^2$. Для поиска минимального расстояния будем использовать функцию `min()`. Приведем листинг для поиска меток сегментов `label` для каждого пикселя:

```
20 distance = zeros([size(a), numColors]);
21 for i=1:1:numColors
22      distance(:,:,i) = ...
```

```
23          ((a-colorMarks(i,1)).^2 + ...
24          (b-colorMarks(i,2)).^2).^0.5;
25      colorLabels = i;
26  end
27  [~, label] = min(distance, [], 3);
28  label = colorLabels(label);
```

Таким образом, матрица `label` рамерности равном исходному изображению будет содержать идентификаторы классов для каждого пикселя. Для сегментации изображения на фрагменты `segmentedFrames` используем следующий листинг:

```
29  rgbLabel = repmat(label, [1 1 3]);
30  segmentedFrames = ...
31      zeros([size(I), numColors], 'uint8');
32  for i=1:1:numColors
33      color = I;
34      color(rgbLabel ~= colorLabels(i)) = 0;
35      segmentedFrames(:,:,:,i) = color;
36  end
```



а)

б)

Рис. 5.3 — Сегментированные области: а) красные, б) желтые.

Отметим распредление сегментированных пикселей на координатной плоскости (a,b):

```
37  figure
38  for i=1:1:numColors
39      plot(a(label == i), b(label==i), ...
40          '.', 'MarkerEdgeColor', ...
41          plotColors{i}, ...
42          'MarkerFaceColor', plotColors{i});
```

```
43      hold on
44  end
```

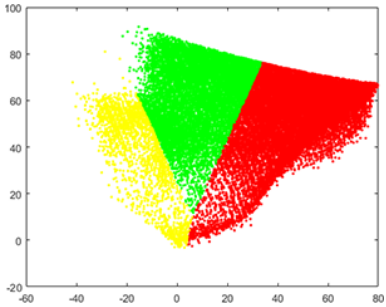


Рис. 5.4 — Распределение сегментированных пикселей на координатной плоскости (a,b).

На основе кластеризации методом 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 пора, пока центры кластеров не перестанут изменяться.

Реализация метода очень похожа на предыдущий подход и содержит ряд аналогичных действий (используем исходное изображение рис. 4.1). Будем работать в цветовом пространстве **Lab**, поэтому первым шагом перейдем из пространства **RGB** в **Lab**:

Листинг 5.7. Сегментация на основе кластеризации методом k-средних.

```
I = imread('pic2.jpg');
Ilab = rgb2lab(I);
L = Ilab(:,:,1);
a = Ilab(:,:,2);
b = Ilab(:,:,3);
```

Рассмотрим координатную плоскость (a,b). Для этого сформируем трехмерный массив `ab`, а затем функцией `reshape()` превратим его в двумерный вектор, содержащий все пиксели изображения:

```
ab(:,:,1) = a;
ab(:,:,2) = b;
nrows = size(I, 1);
ncols = size(I, 2);
ab = reshape(ab, nrows * ncols, 2);
```

Кластеризация методом k-средних в среде MATLAB осуществляется функцией `kmeans()`. Аналогично предыдущему способу разобьем изображение на три области соответствующих цветов. Используем евклидову метрику (параметр `'distance'` со значением `'sqEuclidean'` и для повышения точности повторим процедуру кластеризации три раза (параметр `'Replicates'` со значением `3`)):

```
k = 3;
[ids, centers] = kmeans(ab, k, 'distance',...
    'sqEuclidean', 'Replicates', 3);
label = reshape(ids, nrows, ncols);
```

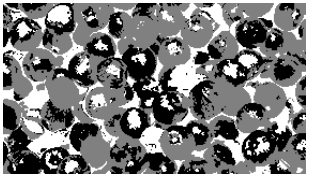
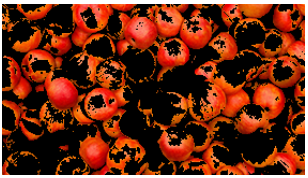
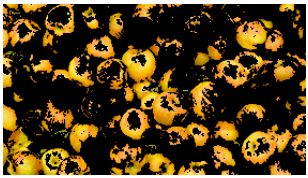


Рис. 5.5 — Метки классов.

Матрица `label` размера равном исходному изображению будет содержать идентификаторы классов для каждого пикселя. Для сегментации изображения на фрагменты `segmentedFrames` используем следующий листинг:



а)



б)

Рис. 4.6 — Сегментированные области: а) красные, б) желтые.

```
15 segmentedFrames = cell(1, 3);
16 rgbLabel = repmat(label, [1 1 3]);
17 for i = 1:1:k
18     color = I;
19     color(rgbLabel ~= i) = 0;
20     segmentedFrames{i} = color;
21     figure, imshow(segmentedFrames{i});
22 end
```

Массив L содержит значение о светлоте изображения. Используя эти данные можно, например, сегментированные красные области разделить на светло-красные и темно-красные сегменты.

Текстурная сегментация

При текстурной сегментации для описания текстуры применяются три основных метода: статистический, структурный и спектральный. В лабораторной работе будем рассматривать статистический подход, который описывает текстуру сегмента как гладкую, грубую или зернистую. Характеристики соответствующих текстурам параметров приведены в табл. 5.1. Рассмотрим пример изображения, представленного на рис. 5.7, на котором имеется два типа текстур. Их разделение в общем случае невозможно выполнить с использованием только лишь простой бинаризации.

Будем рассматривать интенсивность изображения I как случайную величину z, которой соответствует вероятность распределения $p(z)$, вычисляемая из гистограммы изображения. *Централь-*

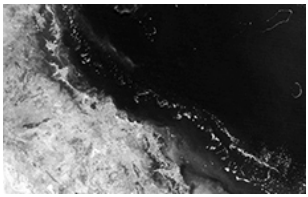


Рис. 5.7 — Исходное полутоновое изображение.

ным моментом порядка n случайной величины z называется параметр $\mu_n(z)$, вычисляемый по формуле:

$$\mu_n(z) = \sum_{i=0}^{L-1} (z_i - m)^n p(z_i), \tag{5.15}$$

где L — число уровней интенсивностей, m — среднее значение случайной величины z:

$$m = \sum_{i=0}^{L-1} z_i p(z_i). \tag{5.16}$$

Из 5.15 следует, что $\mu_0 = 1$ и $\mu_1 = 0$. Для описания тексту-ры важна *дисперсия* случайной величины, равная второму моменту $\sigma^2(z) = \mu_2(z)$ и являющаяся мерой яркостного контраста, которую можно использовать для вычисления признаков *гладкости*. Введем меру относительной гладкости R:

$$R = 1 - \frac{1}{1 + \sigma^2(z)}, \tag{5.17}$$

которая равна нулю для областей с постоянной интенсивностью (нулевой дисперсией) и приближается к единице для больших значений дисперсий $\sigma^2(z)$. Для полутоновых изображений с интервалом интенсивностей [0,255] необходимо нормировать дисперсию до интервала [0,1], поскольку для исходного диапазона значения дисперсий будут слишком велики. Нормирование осуществляется де-

Таблица 5.1 — Значения параметров текстур.

Текстура	m	s	$R \in [0,1]$
Гладкая	82,64	11,79	0,0020
Грубая	143,56	74,63	0,0079
Периодическая	99,72	33,73	0,0170

Текстура	$\mu_3(z)$	U	E
Гладкая	−0,105	0,026	5,434
Грубая	−0,151	0,005	7,783
Периодическая	0,750	0,013	6,674

лением дисперсии $\sigma^2(z)$ на $(L-1)^2$. В качестве характеристики текстуры также зачастую используется *стандартное отклонение*:

$$s = \sigma(z). \tag{5.18}$$

Третий момент является *характеристикой симметрии гистограммы*. Для оценки текстурных особенностей используется функция *энтропии* E, определяющая разброс интенсивностей соседних пикселей:

$$E = -\sum_{i=0}^{L-1} p(z) \log_2 p(z_i). \tag{5.19}$$

Еще одной важной характеристикой, описывающей текстуру, является *мера однородности* U, оценивающая равномерность гистограммы:

$$U = \sum_{i=0}^{L-1} p^2(z_i). \tag{5.20}$$

После вычисления какого-либо признака или набора признаков необходимо построить бинарную маску, на основе которой и будет производится сегментация изображения. Например, можно использовать энтропию E в окрестности каждого пикселя (x,y). Для этого в среде MATLAB используется функция `entropyfilt()`, по умолчанию у которой используется окрестность размера 9×9. Для нор-

мирования функции энтропии в диапазоне от 0 до 1 используем функцию `mat2gray()`, а для построения маски бинаризуем полученный нормированный массив `Eim` методом Отсу.

Листинг 5.8. Текстурная сегментация.

```
I = imread('pic3.jpg');
E = entropyfilt(I);
Eim = mat2gray(E);
BW1 = imbinarize(Eim,graythresh(Eim));
```

а) б)

Рис. 5.8 — а) Энтропия исходного изображения, б) бинаризованное изображение.

После этого используем морфологические фильтры (будут рассмотрены подробнее в лабораторной работе №6) сначала для удаления связных областей, содержащих менее заданного количества пикселей (функция `bwareaopen()`), а затем для удаления внутренних *дефектов формы* или «дырок» (функция `imclose()` со структурным элементом размера 9×9). Оставшиеся крупные «дырки» заполним при помощи функции `imfill()`. Таким образом, получим маску:

```
BWao = bwareaopen(BW1,2000);
nhood = true(9);
closeBWao = imclose(BWao,nhood);
Mask1 = imfill(closeBWao,'holes');
```

Применив полученную маску к исходному изображению выделим сегменты воды и суши.

Границу между текстурами рассчитаем с использованием функции определения периметра `bwperim()`:

а) б)

Рис. 4.9 — а) Результат выполнения функции `bwareaopen()`; б) результат выполнения функции `imclose()`.

а) б)

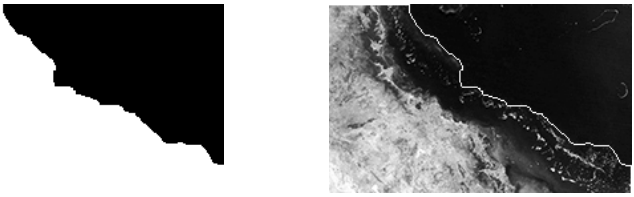
Рис. 4.10 — а) Текстура суши, б) текстура воды.

```
 9 boundary = bwperim(Mask1);
10 segmentResults = I;
11 segmentResults(boundary) = 255;
```

Аналогичный подход можно применить для построения маски относительно суши:

```
12 I2 = I;
13 I2(Mask1) = 0;
14 E2 = entropyfilt(I2);
15 E2im = mat2gray(E2);
16 BW2 = imbinarize(E2im, graythresh(E2im));
17 Mask2 = bwareaopen(BW2,2000);
18 boundary = bwperim(Mask2);
19 segmentResults = I;
20 segmentResults(boundary) = 255;
```

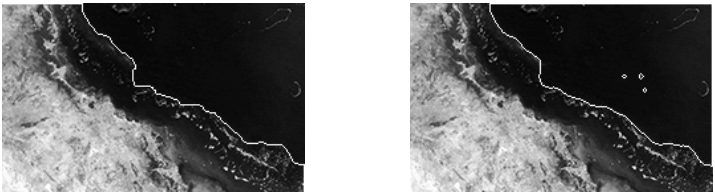
Найдем текстуры суши и воды:



а) б)

Рис. 4.11 — а) Результат выполнения функции `imfill()`, б) выделенная граница функцией `bwperim()`.

```
21 texture1 = I;
22 texture1(~Mask2) = 0;
23 texture2 = I;
24 texture2(Mask2) = 0;
```



а) б)

Рис. 4.12 — а) Результат сегментации относительно воды, б) результат сегментации относительно суши.

Порядок выполнения работы

1. *Бинаризация.* Выбрать произвольное изображение. Выполнить бинаризацию изображения при помощи рассмотренных методов. В зависимости от изображения использовать бинаризацию по верхнему или нижнему порогу.

2. *Сегментация 1.* Выбрать произвольное изображение, содержащее лицо(-а). Выполнить сегментацию изображения либо по принципу Вебера, либо на основе цвета кожи (на выбор).

3. *Сегментация 2.* Выбрать произвольное изображение, содержащее ограниченное количество цветных объектов. Выполнить сегментацию изображения в пространстве **CIE Lab** либо по методу ближайших соседей, либо по методу k-средних (на выбор).

4. *Сегментация 3.* Выбрать произвольное изображение, содержащее две разнородные текстуры. Выполнить текстурную сегментацию изображения, оценить не менее трех параметров выделенных текстур, определить к какому классу относятся текстуры.

Содержание отчета

1. Цель работы.

2. Теоретическое обоснование применяемых методов и функций сегментации изображений.

3. Ход выполнения работы:

   (a) Исходные изображения;

   (b) Листинги программных реализаций;

   (c) Комментарии;

   (d) Результирующие изображения.

4. Выводы о проделанной работе.

Вопросы к защите лабораторной работы

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

2. Какие значения имеют цветовые координаты $\boldsymbol{a}$ и $\boldsymbol{b}$ цветового пространства **CIE Lab** в полутновом изображении?

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

4. Что такое *цветовое пространство* и *цветовой охват*?