Лабораторная работа №2
Гистограммы, профили и проекции

Цель работы

Освоение основных яркостных и геометрических характеристик изображений и их использование для анализа изображений.

Методические рекомендации

До начала работы студенты должны ознакомиться с основными функциями среды MATLAB или библиотеки OpenCV по работе с гистограммами, профилями и проекциями изображений. Лабораторная работа рассчитана на 4 часа.

Теоретические сведения

Пиксель цифрового изображения характеризуется тремя параметрами: (x,y,I), где пара целочисленных значений (x,y) описывает геометрическое положение пикселя в плоскости изображения, а значение I характеризует его яркость (интенсивность) в точке плоскости. Таким образом, в изображении можно выделить яркостную и геометрическую составляющие. В общем случае данные составляющие не связаны между собой (например, изменение в освещенности сцены не изменит геометрических параметров объектов на сцене). Из-за этого проще исследовать отдельно яркостные свойства изображения и отдельно — геометрические. Такой подход понижает порядок исследуемого изображения в случае геометрических свойств с $n = 3$ (x,y,I) до $n = 2$ (x,y), а в случае яркостных свойств — до $n = 1$ (I). Яркостная составляющая изображения характеризуется одномерным массивом гистограммы, из которого можно вычислить контраст.

Гистограмма — это распределение частоты встречаемости пикселей одинаковой яркости на изображении.

Яркость — это среднее значение интенсивности сигнала.

Контраст — это интервал значений между минимальной и максимальной яркостями изображения.

Для сведéния геометрических составляющих изображения к одномерному массиву данных $n = 1$ используются такие характеристики, как *«профили»* и *«проекции»* изображения.

Профиль вдоль линии — это функция интенсивности изображения, распределенного вдоль данной линии *(прорезки)*.

Проекция на ось — это сумма интенсивностей пикселей изображения взятая в направлении перпендикулярном данной оси.

Гистограмма изображения

Для 8-битного полутонового изображения гистограмма яркости представляет собой одномерный целочисленный массив Hist из 256 элементов $[0 \dots 255]$. Элементом гистограммы $\mathrm{Hist}[i]$ является сумма пикселей изображения с яркостью i. По визуальному отображению гистограммы можно оценить необходимость изменения яркости и контрастности изображения, оценить площадь, занимаемую светлыми и темными элементами, определить местоположение на плоскости изображения отдельных объектов, соответствующих некоторым диапазонам яркости. Для цветного RGB-изображения необходимо построить три гистограммы по каждому цвету.

Листинг 2.1. Построение гистограмм изображения в MATLAB.

```
[numRows numCols Layers] = size(I);
imhist(I(:,:,1)); %red
imhist(I(:,:,2)); %green
imhist(I(:,:,3)); %blue
```

Listing 2.2. Построение гистограмм изображения с использованием OpenCV и языка программирования C++

```
// I is an RGB-image
// Number of histogram bins
int histSize = 256;
// Histogram range
// The upper boundary is exclusive
float range[] = { 0, 256 };
const float* histRange[] = { range };
// Split an image into color layers
// OpenCV stores RGB image as BGR
vector<Mat> I_BGR;
```

```
split(I, I_BGR);
// Calculate a histogram for each layer
Mat bHist, gHist, rHist;
calcHist(&I_BGR[0], 1, 0, Mat(), bHist, 1,
         &histSize, histRange);
calcHist(&I_BGR[1], 1, 0, Mat(), gHist, 1,
         &histSize, histRange);
calcHist(&I_BGR[2], 1, 0, Mat(), rHist, 1,
         &histSize, histRange);
```

Listing 2.3. Построение гистограмм изображения с использованием OpenCV и языка программирования Python

```
# I is an RGB-image
# Number of histogram bins
histSize = 256
# Histogram range
# The upper boundary is exclusive
histRange = (0, 256)
# Split an image into color layers
# OpenCV stores RGB image as BGR
I_BGR = cv2.split(I)
# Calculate a histogram for each layer
bHist = cv2.calcHist(I_BGR, [0], None,
                     [histSize], histRange)
gHist = cv2.calcHist(I_BGR, [1], None,
                     [histSize], histRange)
rHist = cv2.calcHist(I_BGR, [2], None,
                     [histSize], histRange)
```

Если гистограмма неравномерна, то для улучшения изображения можно ее выровнять, причем выравнивание гистограммы в зависимости от решаемой задачи можно выполнять различным образом.

Арифметические операции

Простейшими способами выравнивания гистограммы являются *арифметические операции* с изображениями. Например, в случае, если большинство значений гистограммы находятся слева, то изображение является темным. Для увеличения детализации тем-

ных областей можно сдвинуть гистограмму правее, в более светлую область, например, на 50 градаций для каждого цвета. В среде MATLAB программная реализация имеет вид:

```
Inew(i,j) = I(i,j) + 50 / 255;
```

Гистограмма исходного изображения сдвигается в среднюю часть диапазона, которая является более приемлемой с точки зрения визуального восприятия. Данный подход обладает следующим недостатком: повышение интенсивностей темных областей приводит к сдвигу светлых к максимуму, что может привести к потере информации в светлых областях.

Растяжение динамического диапазона

Если интенсивности пикселей областей интереса находятся в узком динамическом диапазоне, то можно растянуть этот диапазон. Подобные преобразования выполняются согласно следующему выражению:

$$I_{new} = \left(\frac{I - I_{min}}{I_{max} - I_{min}} \right)^{\alpha}, \tag{2.1}$$

где I и I_{new} — массивы значений интенсивностей исходного и нового изображений соответственно; I_{min} и I_{max} — минимальное и максимальное значения интенсивностей исходного изображения соответственно; α — коэффициент нелинейности.

Данное выражение является нелинейным из-за коэффициента α. В случае, если $\alpha = 1$, применение формулы (2.1) к исходному изображению не даст желаемого эффекта, поскольку гистограммы цветовых компонент изображения занимают весь возможный диапазон. Нелинейные преобразования проводятся для каждой цветовой составляющей.

Листинг 2.4. Нелинейное растяжение динамического диапазона при $\alpha = 0{,}5$ в MATLAB.

```
1 [numRows numCols Layers] = size(I);
2 for k=1:1:Layers;
3     Imin = min(min(I(:, :, k)));
4     Imax = max(max(I(:, :, k)));
5     for i=1:1:numRows;
```

```
6           for j=1:1:numCols;
7               Inew(i,j,k) = ...
8                   (((I(i,j,k) - Imin) / ...
9                       (Imax - Imin)))^0.5;
10              if Inew(i,j,k) > 1;
11                  Inew(i,j,k) = 1;
12              end
13              if Inew(i,j,k) < 0;
14                  Inew(i,j,k) = 0;
15              end
16          end
17      end
18  end
```

Библиотека OpenCV не производит автоматическое преобразование типов при выполнении деления, поэтому при делении двух целых чисел, находящихся в диапазоне от 0 до 255, будет получено также целое число 0 или 1. Поэтому при обработке изображений с использованием библиотеки OpenCV следует предварительно конвертировать изображение в вещественные числа с помощью функции `convertTo`. После завершения обработки изображения его следует сконвертировать к изначальной глубине цвета.

Listing 2.5. Нелинейное растяжение динамического диапазона при $\alpha = 0{,}5$ с использованием OpenCV и языка программирования C++.

```
1   alfa = 0.5;
2   Mat Inew;
3   // Convert to floating points
4   if (I.depth() == CV_8U)
5     I.convertTo(Inew, CV_32F, 1.0 / 255);
6   else
7     Inew = I;
8   // We need to process layers separately
9   vector<Mat> I_BGR;
10  cv::split(Inew, I_BGR);
11  for (int k = 0; k < I_BGR.size(); k++)
12  {
13    // Create a new image with same size and
```

```
14      // one floating point layer
15      Inew = Mat(I_BGR[k].rows, I_BGR[k].cols,
16        I_BGR[k].type());
17      // Calculate min and max values
18      double Imin, Imax;
19      minMaxLoc(I_BGR[k], &Imin, &Imax);
20      // Change intensity of each image pixel
21      for (int i = 0; i < I_BGR[k].rows; i++)
22      {
23        for (int j = 0; j < I_BGR[k].cols; j++)
24          Inew.at<float>(i, j) =
25            float(pow((I_BGR[k].at<uchar>(i, j)
26            - Imin) / (Imax - Imin), alfa));
27      }
28      I_BGR[k] = Inew;
29    }
30    // Merge back
31    cv::merge(I_BGR, Inew);
32    // Convert back to uint if needed
33    if (I.depth() == CV_8U)
34      Inew.convertTo(Inew, CV_8U, 255);
```

Listing 2.5. Нелинейное растяжение динамического диапазона при $\alpha = 0{,}5$ с использованием OpenCV и языка программирования Python.

```
1   alfa = 0.5
2   # Convert to floating point
3   if I.dtype == np.uint8:
4     Inew = I.astype(np.float32) / 255
5   else:
6     Inew = I
7   # We need to process layers separately
8   I_BGR = cv2.split(Inew)
9   Inew_BGR = []
10  for layer in I_BGR:
11    Imin = layer.min()
12    Imax = layer.max()
13    Inew = np.clip((((layer - Imin) /
```

```
14        (Imax - Imin)) ** alfa), 0, 1)
15      Inew_BGR.append(Inew)
16    # Merge back
17    Inew = cv2.merge(Inew_BGR)
18    # Convert back to uint if needed
19    if (I.dtype == np.uint8):
20      Inew = (255 * Inew).clip(0, 255). \
21        astype(np.uint8)
```

Равномерное преобразование

Осуществляется по следующей формуле:

$$I_{new} = (I_{max} - I_{min}) \cdot P(I) + I_{min}, \tag{2.2}$$

где I_{min}, I_{max} — минимальное и максимальное значения интенсивностей исходного изображения I; $P(I)$ — функция распределения вероятностей исходного изображения, которая аппроксимируется *кумулятивной гистограммой*:

$$P(I) \approx \sum_{m=0}^{i} \mathrm{Hist}(m). \tag{2.3}$$

Кумулятивная гистограмма исходного изображения в среде MATLAB может быть вычислена с помощью функции `cumsum()`:

```
CH = cumsum(H) ./ (numRows * numCols);
```

где `H` — гистограмма исходного изображения, `numRows` и `numCols` — число строк и столбцов исходного изображения соответственно.

Согласно формуле (2.2) можно вычислить значения интенсивностей пикселей результирующего изображения. В среде MATLAB программная реализация имеет вид:

```
Inew(i,j) = (Imax - Imin) * CH(ceil(I(i,j) + eps)) + ...
  Imin / 255.0;
```

Параметр `eps` необходим для защиты программы от присваивания индексам кумулятивной гистограммы нулевых значений.

При использовании библиотеки OpenCV и языка программирования C++ вычисление кумулятивной гистограммы выполняется следующим способом:

```
Mat CH = H.clone();
for (int i = 1; i < CH.size[0]; i++)
  CH.at<float>(i) += CH.at<float>(i - 1);
CH /= numRows * numCols;
```

Далее, согласно формуле (2.2) можно вычислить значения интенсивностей пикселей результирующего изображения обращением к соответствующим элементам массива:

```
Inew<float>(i,j) = (Imax - Imin) *
  CH.at<float>(I.at<uchar>(i, j)) + Imin;
```

Поскольку в данном примере исходное изображение задано целыми числами в диапазоне от 0 до 255, то дополнительная коррекция на `eps` не требуется. Однако, если исходное изображение определено значениями с плавающей запятой, коррекцию следует учитывать.

При использовании языка программирования Python можно использовать встроенную функцию `cumsum()` для вычисления кумулятивной гистограммы:

```
CH = np.cumsum(H) / (numRows * numCols)
```

Затем, для вычисления значения интенсивности пикселей изображения, можно обратиться к соответствующему элементу этого массива для одного пикселя:

```
Inew[i, j] = (Imax - Imin) * CH[I[i, j]] + Imin
```

Или для слоя изображения:

```
Inew[:, :, k] = (Imax - Imin) * CH[I[:, :, k]] + Imin
```

Экспоненциальное преобразование

Осуществляется по следующей формуле:

$$I_{new} = I_{min} - \frac{1}{\alpha} \cdot \ln(1 - P(I)), \tag{2.4}$$

где α — постоянная, характеризующая крутизну преобразования. Согласно формуле (2.4) можно вычислить значения интенсивностей пикселей результирующего изображения. В среде MATLAB программная реализация имеет вид:

```
Inew(i,j) = double(Imin) - ...
  (1 / alfa) * log(1 - CH(index));
```

При использовании библиотеки OpenCV и языка программирования C++ программная реализация примет вид:

```
Inew<float>(i,j) = Imin - 1 / alfa *
  log(1 - CH.at<float>(I.at<uchar>(i, j)));
```

При использовании библиотеки OpenCV и языка программирования Python программная реализация примет вид:

```
Inew[i, j] = Imin - 1 / alfa * math.log(1 - CH[I[i, j]])
```

Преобразование по закону Рэлея

Осуществляется по следующей формуле:

$$I_{new} = I_{min} + \left(2\alpha^2 \ln\left(\frac{1}{1 - P(I)}\right)\right)^{1/2}, \qquad (2.5)$$

где α — постоянная, характеризующая гистограмму распределения интенсивностей элементов результирующего изображения. В среде MATLAB программная реализация имеет вид:

```
Inew(i,j) = double(Imin) + sqrt(2 * alfa^2 * ...
 log(1 / (1 - CH(ceil(I(i,j) + eps)))));
```

При использовании библиотеки OpenCV и языка программирования C++ программная реализация примет вид:

```
Inew<float>(i,j) = Imin + sqrt(2 * alfa * alfa *
  log(1 / (1 - CH.at<float>(I.at<uchar>(i, j)))));
```

При использовании библиотеки OpenCV и языка программирования Python программная реализация примет вид:

```
Inew[i, j] = Imin + sqrt(2 * alfa ** 2 *
  math.log(1 / 1 - CH[I[i, j]]))
```

Преобразование по закону степени $2/3$

Осуществляется по следующей формуле:

$$I_{new} = P(I)^{2/3}. \tag{2.6}$$

В среде MATLAB программная реализация имеет вид:

```
Inew(i,j) = (CH(ceil(I(i,j) + eps)))^(2/3);
```

При использовании библиотеки OpenCV и языка программирования C++ программная реализация примет вид:

```
Inew<float>(i,j) =
  pow(CH.at<float>(I.at<uchar>(i, j)), 2/3);
```

При использовании библиотеки OpenCV и языка программирования Python программная реализация примет вид:

```
Inew[i, j] = CH[I[i, j]] ** (2 / 3)
```

Гиперболическое преобразование

Осуществляется по следующей формуле:

$$I_{new} = \alpha^{P(I)}, \tag{2.7}$$

где α — постоянная, относительно которой осуществляется преобразование и, как правило, равная минимальному значению интенсивности элементов исходного изображения $\alpha = I_{min}$.

В среде MATLAB программная реализация имеет вид:

```
alfa = 0.04;
Inew(i,j) = alfa^CH(ceil(I(i,j) + eps));
```

При использовании библиотеки OpenCV и языка программирования C++ программная реализация примет вид:

```
alfa = 0.04;
Inew<float>(i,j) =
  pow(alfa, CH.at<float>(I.at<uchar>(i, j)));
```

При использовании библиотеки OpenCV и языка программирования Python программная реализация примет вид:

```
alfa = 0.04
Inew[i, j] = alfa ** CH[I[i, j]]
```

Рассмотренные методы преобразования гистограмм могут применяться для устранения искажений при передаче уровней квантования, которым были подвергнуты изображения на этапе формирования, передачи или обработки данных. Кроме того, данные методы могут применяться не только ко всему изображению, но использоваться локально в *скользящем окне*, что позволит повысить детализированность отдельных областей.

В среде MATLAB реализовано несколько функций, автоматически выравнивающих гистограммы полутонового изображения:

1. `imadjust()` — повышает контрастность изображения, изменяя диапазон интенсивностей исходного изображения;
2. `histeq()` — эквализирует (выравнивает) гистограмму методом распределения значений интенсивностей элементов исходного изображения;
3. `adapthisteq()` — выполняет контрастно-ограниченное адаптивное выравнивание гистограммы методом анализа и эквализации гистограмм локальных окрестностей изображения.

Для обработки цветного изображения данные функции можно применять поочередно к каждому цвету.

В библиотеке OpenCV также реализовано несколько функций для автоматического выравнивания гистограммы полутонового изображения :

1. `equalizeHist()` — эквализирует (выравнивает) гистограмму методом распределения значений интенсивностей элементов исходного изображения;
2. `createCLAHE()` и `CLAHE.apply()` — выполняет контрастно-ограниченное адаптивное выравнивание гистограммы методом анализа и эквализации гистограмм локальных окрестностей изображения.

Аналогично с MATLAB, данные функции можно применять только к одному слою изображения. Поэтому для обработки цветного изображения данные функции можно применять поочередно к каждому цвету.

Таблица поиска

Таблица поиска или *LUT (Lookup table)* — удобный инструмент для преобразования интенсивностей точек всего изображения. Поскольку большинство изображений определяется с использованием ограниченного набора возможных дискретных значений цвета (чаще всего цвет задается с использованием целых чисел, находящихся в диапазоне [0, 255]), то можно заранее вычислить преобразование цвета для каждого потенциально-возможного значения цвета, сохранить их в таблице поиска, а затем преобразовать цвет исходного изображения, используя формулу:

$$I_{new} = LUT[I_{old}] \tag{2.8}$$

В библиотеке OpenCV есть встроенная функция для применения преобразования, заданного с использованием таблицы поиска, к изображению. Таблица поиска может быть определена либо одинаковой для всех слоев изображения, либо для каждого слоя отдельно, в зависимости от количества слоев таблицы. В языке программирования C++ программная реализация растяжения динамического диапазона с использованием таблиц поиска имеет вид:

```
alfa = 0.5;
Mat lut(1, 256, CV_8U);
uchar *lut_ptr = lut.ptr();
for (int i = 0; i < 256; i++)
{
  double var = (i - Imin) / (Imax - Imin);
  if (var < 0)
    lut_ptr[i] = 0;
  else
    lut_ptr[i] = saturate_cast<uchar>(
      255 * pow(var, alfa));
}
LUT(I, lut, Inew);
```

Аналогичным образом реализуется использование таблиц поиска в языке программирования Python:

```
alfa = 0.5
```

```
lut = np.arange(256, dtype = np.uint8)
lut = (lut - Imin) / (Imax - Imin)
lut = np.where(lut > 0, lut, 0)
lut = np.clip(255 * np.power(lut, alfa), 0, 255)
Inew = cv2.LUT(I, lut)
```

Профиль изображения

Профилем изображения вдоль некоторой линии называется функция интенсивности изображения, распределенного вдоль данной линии *(прорезки)*. Простейшим случаем профиля изображения является профиль строки:

$$\text{Profile}\, i(x) = I(x,i), \tag{2.9}$$

где i — номер строки изображения I.

Профиль столбца изображения:

$$\text{Profile}\, j(y) = I(j,y), \tag{2.10}$$

где j — номер столбца изображения I.

В общем случае профиль можно рассматривать вдоль любой прямой, ломаной или кривой линии, пересекающей изображение. После формирования массива профиля изображения проводится его анализ стандартными средствами. Анализ позволяет автоматически выделять особые точки функции профиля, соответствующие контурам изображения, пересекаемым данной линией. Например, на рис. 2.1 представлен профиль изображения штрих-кода, взятого вдоль оси Ox. Данный профиль содержит всю необходимую информацию для считывания штрих-кода, поскольку позволяет определить последовательность чередования «толстых» и «тонких» штрихов и пробелов различной ширины. В среде MATLAB профиль изображения можно найти при помощи фунции `improfile()`.

Листинг 2.6. Поиск профиля штрих-кода вдоль оси Ox в среде MATLAB.

```
1 I = imread('code.jpg');
2 [numRows, numCols, Layers] = size(I);
3 x = [1 numCols];
4 y = [ceil(numRows/2) ceil(numRows/2)];
```

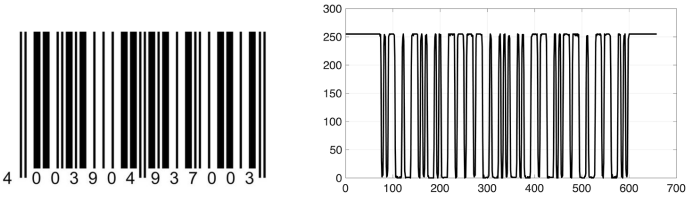

Рис. 2.1 — Слева — штрих-код,
справа — его профиль вдоль оси Ox.

```
5 figure
6 improfile(I,x,y),grid on;
```

Для интерактивного указания линии (ломаной), вдоль которой следует построить профиль, необходимо использовать функцию `improfile` без параметров.

Листинг 2.7. Интерактивное задание линии профиля в среде MATLAB.

```
1 I = imread('code.jpg');
2 imshow(I);
3 improfile
```

По сравнению с MATLAB в библиотеке OpenCV нет такого удобного инструмента для вычисления профиля изображения вдоль произвольной оси. Однако можно получить горизонтальный или вертикальный профиль с использованием функций библиотеки для быстрого доступа к отделенной строке или столбцу изображения.

Listing 2.8. Поиск профиля штрих-кода вдоль оси Ox с использованием библиотеки OpenCV и языка программирования C++.

```
1   Mat I = imread("code.jpg", IMREAD_COLOR);
2   Mat profile = I.row(I.rows / 2);
```

Listing 2.9. Поиск профиля штрих-кода вдоль оси Ox с использованием библиотеки OpenCV и языка программирования Python.

```
1   I = cv2.imread("code.jpg", cv2.IMREAD_COLOR)
2   profile = I[round(I.shape[0] / 2), :]
```

Проекция изображения

Проекцией изображения на некоторую ось называется сумма интенсивностей пикселей изображения в направлении, перпендикулярном данной оси. Простейшим случаем проекции двумерного изображения являются вертикальная проекция на ось Ox, представляющая собой сумму интенсивностей пикселей *по столбцам* изображения:

$$\mathrm{Proj}\, X(y) = \sum_{y=0}^{\dim Y - 1} I(x,y), \tag{2.11}$$

и горизонтальная проекция на ось Oy, представляющая собой сумму интенсивностей пикселей *по строкам* изображения:

$$\mathrm{Proj}\, Y(x) = \sum_{x=0}^{\dim X - 1} I(x,y). \tag{2.12}$$

Запишем выражение для проекции на произвольную ось. Допустим, что направление оси задано единичным вектором с координатами (e_x,e_y). Тогда проекция изображения на ось Oe определяется следующим выражением:

$$\mathrm{Proj}\, E(t) = \sum_{xe_x+ye_y=t} I(x,y). \tag{2.13}$$

Анализ массива проекции позволяет выделять характерные точки функции проекции, которые соответствуют контурам объектов на изображении. Например, если на изображении имеются контрастные объекты, то в проекции будут видны перепады или экстремумы функции, соответствующие положению каждого из объектов.

На рис. 2.2 показан пример проекции на ось Oy машиночитаемого документа. Видно, что две машиночитаемые текстовые строки порождают два существенных экстремума функции проекции. Подобные проекции могут быть использованы в алгоритмах обнаружения и сегментации текстовых строк в системах распознавания текста.

Листинг 2.10. Определение положения текста в среде MATLAB.

```
for i=1:1:numRows
    Proj(i,1) = (round(sum(I(i,:,1))) + ...
    round(sum(I(i,:,2))) + ...
    round(sum(I(i,:,3)))) / (256 * 3);
end
```

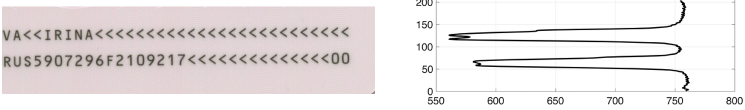

Рис. 2.2 — Слева — фрагмент заграничного паспорта, справа — его проекция на ось Oy.

Listing 2.11. Определение положения текста с использованием библиотеки OpenCV и языка программирования C++.

```
// An array to store projection
Mat Proj(I.rows, 1, CV_32F);
bool bw = (I.channels() == 1);
for (int i = 0; i < I.rows; i++)
{
  double sum = 0;
  for (int j = 0; j < I.cols; j++)
  {
    if (bw)
      sum += I.at<uchar>(i, j);
    else
    {
      const Vec3b &pix = I.at<Vec3b>(i, j);
      sum += pix[0] + pix[1] + pix[2];
    }
  }
  Proj.at<float>(i) = float(sum);
}
Proj /= 256 * I.channels();
// Calculate min and max values
```

```
21  double ProjMin, ProjMax;
22  minMaxLoc(Proj, &ProjMin, &ProjMax);
23  // Create graph image
24  Mat ProjI(256, img.rows, CV_8UC3,
25    Scalar(255, 255, 255));
26  DrawGraph<float>(ProjI, Proj,
27    Scalar(0, 0, 0), ProjMax);
28  transpose(ProjI, ProjI);
29  flip(ProjI, ProjI, 1);
30  // And display it
31  imshow("Projection␣to␣Oy", ProjI);
32  waitKey();
```

Listing 2.12. Определение положения текста с использованием библиотеки OpenCV и языка программирования Python.

```
1   # Calculate projection to Oy
2   if I.ndim == 2:
3     ProjI = np.sum(I, 1) / 255
4   else:
5     ProjI = np.sum(I, (1, 2)) / 255 / \
6       I.shape[2]
7   # Create graph image
8   ProjI = np.full((256, Proj.shape[0], 3),
9                   255, dtype = np.uint8)
10  DrawGraph(ProjI, Proj, (0, 0, 0),
11            Proj.max())
12  ProjI = cv2.transpose(ProjI)
13  ProjI = cv2.flip(ProjI, 1)
14  # And display it
15  cv2.imshow('Projection␣to␣Oy', ProjI)
16  cv2.waitKey()
```

Порядок выполнения работы

1. *Гистограммы.* Выбрать произвольное слабоконтрастное изображение. Выполнить выравнивание гистограммы и растяжение контраста, использовать рассмотренные преобразования

и встроенные функции пакета MATLAB, с использованием библиотеки OpenCV и языков программирования C++, PYTHON. Сравнить полученные результаты.

2. *Проекции.* Выбрать произвольное изображение, содержащее монотонные области и выделяющиеся объекты. Произвести построение проекций изображения на вертикальную и горизонтальную оси. Определить границы областей объектов.

3. *Профили.* Выбрать произвольное изображение, содержащие штрих-код. Выполнить построение профиля изображения вдоль штрих-кода.

Содержание отчета

1. Титульный лист.

2. Цель работы.

3. Теоретическое обоснование применяемых методов и функций геометрических преобразований.

4. Ход выполнения работы:

   (a) Исходные изображения;

   (b) Листинги программных реализаций;

   (c) Комментарии;

   (d) Результирующие изображения.

5. Выводы о проделанной работе.

Вопросы к защите лабораторной работы

1. Что такое контрастность изображения и как её можно изменить?

2. Чем эффективно использование профилей и проекций изображения?

3. Каким образом можно найти объект на равномерном фоне?