|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  (МГУПИ) |

Институт КБСП направление 09.03.02

Кафедра КБ-4 «Автоматизированные системы управления»

Дисциплина «Технология программирования в среде Python»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

**Разработка Desktop приложения для распознавания текста на графических изображениях**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Романов П.С.

подпись, дата

Группа БСБО-01-15

Работа защищена на оценку\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев А.С.

подпись, дата

Члены комиссии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  (МГУПИ) |

Институт КБСП направление 09.03.02

Кафедра КБ4 «Автоматизированные системы управления»

Дисциплина «Технология программирования в среде Python»

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

Студент: 3 курса группа: БСБО-01-15

1 Тема:

**Разработка Desktop приложения для распознавания текста на графических изображениях**

2 Срок представления проекта (работы) к защите 23.12.2017 г.

3 Исходные данные для разработки

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4 Содержание пояснительной записки:

Титульный лист

Задание

Содержание

Введение

1 Цели

2 Задачи

3 Техническое задание

Заключение

Приложения

Список использованных источников

Руководитель работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев А.С.

подпись, дата

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Романов П.С.

подпись, дата

Содержание

[Цели 1](#_Toc501753975)

[Задачи 2](#_Toc501753976)

[Введение 2](#_Toc501753977)

[Составляющие проекта 5](#_Toc501753978)

[Техническое задание 6](#_Toc501753979)

[Основания для разработки 6](#_Toc501753980)

[Назначение разработки 6](#_Toc501753981)

[Требования к программе 6](#_Toc501753982)

[Стадии и этапы разработки. 8](#_Toc501753983)

[Заключение 9](#_Toc501753984)

[Приложение 1. Исходный код 10](#_Toc501753985)

[Main.py: 10](#_Toc501753986)

[tesseract.py: 13](#_Toc501753987)

[Список использованных источников 14](#_Toc501753988)

Цели

Разработать desktop-приложение, осуществляющее распознавание текста, присутствующего в выделенной области экрана, вывод распознанного текста в окно приложения и перевод данного текста на русский язык.

Задачи

Для реализации функционала приложения следует разработать:

* Пользовательский интерфейс
* Модуль захвата изображения
* Алгоритм предобработки изображения
* Модуль распознавания текста, присутствующего на изображении
* Модуль перевода распознанного текста

Введение

К 2017 году популярность онлайн-курсов, обучающих YouTube каналов и разнообразных видеоуроков достигла колоссальных размеров. Сегодня, все больше студентов предпочитают просмотр краткого видео поиску и чтению информации из учебника. Все больше крупных корпораций при обучении и повышении квалификации кадров переходит на eLearning – систему обучения с помощью интернета и мультимедиа.

Однако появление и быстрое развитие этих систем привело к возникновению необходимости в новых инструментах конспектирования и записи информации: текст лекции того или иного видеоурока, презентации и иные ресурсы не всегда бывают доступны.

В основе разработанного desktop-приложения лежит идея о создании быстрого, эффективного и удобного средства ведения конспектов, которое бы идеально дополняло систему электронного обучения инструментом для записи и, в случае отсутствия исходных ресурсов, извлечения необходимой информации посредством захвата изображения и распознавания текста напрямую с экрана.

Для реализации данной идеи была применена технология Оптического распознавания символов (OCR) — механического или электронного перевод изображений рукописного, машинописного или печатного текста в текстовые данные, использующихся для представления символов в компьютере (например, в текстовом редакторе).

В данной работе за основу был взят OCR-движок Tesseract от Google LLC. Система распознавания, реализованная в Tesseract Engine, работает следующим образом:

На вход системы распознавания поступает растровое изображение страницы документа. Для работы алгоритмов распознавания желательно, чтобы поступающее на вход изображение было как можно более высокого качества. Если изображение зашумлено, нерезкое, имеет низкую контрастность, то это усложнит задачу алгоритмов распознавания. Поэтому перед обработкой изображения алгоритмом распознавания проводится его предварительная обработка, направленная на улучшение качества изображения. Она включает фильтрацию изображения от шумов, повышение резкости и контрастности изображения, выравнивание и преобразование в используемый системой.

Подготовленное изображение попадает на вход модуля сегментации. Задачей этого модуля является выявление структурных единиц текста – строк, слов и символов. Выделение фрагментов высоких уровней, таких как строки и слова, может быть осуществлено на основе анализа промежутков между тёмными областями.

К сожалению, такой подход не может быть применён для выделения отдельных букв, поскольку, в силу особенностей начертания или искажений, изображения соседних букв могут объединяться в одну компоненту связанности или наоборот — изображение одной буквы может распадаться на отдельные компоненты связанности. Во многих случаях для решения задачи сегментации на уровне букв используются сложные эвристические алгоритмы.

Полагаем, что для принятия окончательного решения о прохождении границы букв на таком раннем этапе обработки, системе распознавания недостаточно информации. Поэтому задачей модуля сегментации на уровне букв в используемом алгоритме является нахождение возможных границ символов внутри буквы, а окончательное решение о разбиении слова принимается на последнем этапе обработки, с учётом идентификации отдельных фрагментов изображения как букв. Дополнительным преимуществом такого подхода является возможность работы с начертаниями букв, состоящих из нескольких компонент связанности без специальной обработки таких случаев.

Результатом работы модуля сегментации является дерево сегментации – структура данных, организация которой отражает структуру текста на странице. Самому верхнему уровню соответствует объект страница. Он содержит массив объектов, описывающих строки. Каждая строка в свою очередь включает набор объектов слов. Слова являются листьями этого дерева. Информация о возможных местах разделения слова на буквы храниться в слове, однако отдельные объекты для букв не выделяются. В каждом объекте дерева хранится информация об области, занимаемой соответствующим объектом на изображении.

Выявленные фрагменты изображения подаются на вход классификатора, выходом которого является вектор возможности принадлежности изображения к классу той или иной буквы. В используемом алгоритме используется классификатор составной архитектуры, организованный в виде дерева, листьями которого являются простые классификаторы, а внутренние узлы соответствуют операциям комбинирования результатов нижележащих уровней.

Работа простого классификатора осуществляется в два шага. Сначала по исходному изображению вычисляются признаки. Значение каждого признака является функцией от яркостей некоторого подмножества пикселей изображения. В результате получается вектор значений признаков, который поступает на вход нейронной сети. Каждый выход сети соответствует одной из букв алфавита, а получаемое на выходе значение рассматривается как уровень принадлежности буквы нечёткому множеству.

Задачей алгоритма комбинирования является обобщение информации, поступающей в виде входных нечётких множеств и вычисление на их основе выходного нечёткого подмножества множества распознаваемых символов.

Результатом работы классификатора является нечёткое множество, полученное в результате комбинирования на самом верхнем уровне.

На последнем этапе принимается решение о наиболее правдоподобном варианте прочтения слова. Для этого используются уровни возможности прочтения отдельных букв, межбуквенной сегментации и частоты сочетаний букв в данном языке.

Составляющие проекта

Main.py – описание класса окна и основных функций приложения

Tesseract.py – предварительная обработка и непосредственно распознавание текста

Текст исходного кода приведен в приложении к пояснительной записке.

Техническое задание

Основания для разработки

Учебный план по дисциплине «Технология программирования в среде Python», направление бакалавриата «Информационные системы и Технологии» кафедры КБ4 «Автоматизированные системы управления», Московский технологический университет (МТУ), 2017 г.

Назначение разработки

Desktop-приложение предназначено для захвата выделенной области экрана, распознания текста (на основе Tesseract OCR Engine), присутствующего в выделенной области на любом языке и перевода данного текста на русский язык. Данное приложение предполагается как часть комплекса систем электронного обучения, предназначенная для увеличения эффективности обучения и повышения удобства работы с онлайн-ресурсами и eLearning-платформами.

Требования к программе

К программе предъявляются следующие требования:

1. Требования к функциональным характеристикам

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* 1. Захват изображения, находящегося в пределах окна программы.
  2. Динамическое изменение размера окна.
  3. Автоматическое распознавание текста, находящегося в вышеуказанных пределах.
  4. Вывод распознанного текста в специальное поле в окне программы.
  5. Перевод распознанного текста с любого данного языка на русский.
  6. Внешний вид окна приложения должен соответствовать следующему:



Рисунок 1. Внешний вид окна

1. Требования к составу и параметрам технических средств
   1. Оперативная память объемом от 1 Гб.
   2. Свободное место на SSD или HDD диске объемом не менее 80 Мб.
2. Требования к информационной и программной совместимости

3.1 Требования к техническим средствам

В состав технических средств должен входить персональный компьютер. на базе операционной системы Windows 7 и новее

3.2 Требования к версиям используемых библиотек

На персональном компьютере должны присутствовать следующие библиотеки следующих версий:

3.2.1 PyTesseract – wrapper для Tesseract OCR Engine.

Требуемая версия – 0.1.7

3.2.2 OpenCV – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения.

Требуемая версия – 3.3.1

3.2.3. Pillow (PIL fork) – Python Imaging Library.

Требуемая версия – 4.3.0

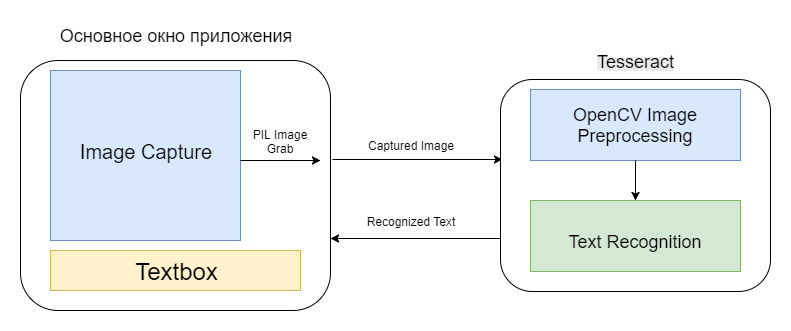
3.2.4 NumPy – предоставляет реализации вычислительных алгоритмов, оптимизированные для работы с многомерными массивами.

Требуемая версия – 1.11.1

3.2.5 PyQt5 – это набор Python библиотек для создания графического интерфейса на базе платформы Qt5 от компании Digia.

Требуемая версия – 5.9.2

3.3 Схема взаимодействия компонентов приложения



Стадии и этапы разработки.

Разработка должна быть проведена в 4 стадии:

1. Анализ имеющихся решений и существующих алгоритмов.
2. Разработка интерфейса и схемы взаимодействия компонентов приложения.
3. Разработка технического задания на разработку приложения.
4. Разработка программного кода, реализующего функциональные требования к приложению.

Разработка программного кода должна происходить на языке программирования Python версии 3.6.4 в среде разработки PyCharm от JetBrains s.r.o.

Заключение

В результате проделанной работы было создано desktop-приложение, согласно требованиям, описанным в техническом задании.

Разработан удобный и интуитивный пользовательский интерфейс при помощи библиотеки PyQt5 на базе платформы Qt5.

Разработан модуль захвата изображения с применением библиотеки PIL.

Разработан алгоритм предобработки изображения с применением библиотеки OpenCV.

Разработан модуль распознавания текста с применением технологии Tesseract от Google и библиотеки pytesseract.

Разработан модуль перевода распознанного текста с использованием Yandex.Translate API.

Для создания приложения была использована интегрированная среда разработки PyCharm.

При использовании данного приложения в процессе обучения, прогнозируется увеличение эффективности усвоения информации, облегчение конспектирования и извлечения информации.

Приложение 1. Исходный код

Main.py:

1. # -\*- coding: utf-8 -\*-
3. **from** PyQt5 **import** QtCore, QtGui, QtWidgets

6. **from** PyQt5 **import** QtCore, QtGui, QtWidgets
7. **from** PyQt5.QtCore **import** \*
8. **from** PyQt5 **import** QtGui
9. **import** sys
10. **from** PIL **import** ImageGrab
11. **from** tesseract **import** get\_string
13. **def** setMoveWindow(widget):
14. """
15. Позволяет перемещать окно ухватившись не только за заголовок, а за произвольный виджит (widget).
16. """
17. win = widget.window()
18. cursorShape = widget.cursor().shape()
19. moveSource = getattr(widget, "mouseMoveEvent")
20. pressSource = getattr(widget, "mousePressEvent")
21. releaseSource = getattr(widget, "mouseReleaseEvent")
23. **def** move(event):
24. **if** move.b\_move:
25. x = event.globalX() + move.x\_korr - move.lastPoint.x()
26. y = event.globalY() + move.y\_korr - move.lastPoint.y()
27. win.move(x, y)
28. widget.setCursor(QtCore.Qt.SizeAllCursor)
29. **return** moveSource(event)
31. **def** press(event):
32. **if** event.button() == QtCore.Qt.LeftButton:
33. # Корекция геометрии окна: учитываем размеры рамки и заголовока
34. x\_korr = win.frameGeometry().x() - win.geometry().x()
35. y\_korr = win.frameGeometry().y() - win.geometry().y()
36. # Корекция геометрии виджита: учитываем смещение относительно окна
37. parent = widget
38. **while** **not** parent == win:
39. x\_korr -= parent.x()
40. y\_korr -= parent.y()
41. parent = parent.parent()
42. move.\_\_dict\_\_.update({"lastPoint": event.pos(), "b\_move": True, "x\_korr": x\_korr, "y\_korr": y\_korr})
43. **else**:
44. move.\_\_dict\_\_.update({"b\_move": False})
45. widget.setCursor(cursorShape)
46. **return** pressSource(event)
48. **def** release(event):
49. move.\_\_dict\_\_.update({"b\_move": False})
50. widget.setCursor(cursorShape)
51. **return** releaseSource(event)
53. setattr(widget, "mouseMoveEvent", move)
54. setattr(widget, "mousePressEvent", press)
55. setattr(widget, "mouseReleaseEvent", release)
56. move.\_\_dict\_\_.update({"b\_move": False})
57. **return** widget

60. **class** Ui\_MainWindow(object):
61. MainWindow = ""
62. **def** setupUi(self, MainWindow):
63. self.MainWindow = MainWindow
64. MainWindow.setObjectName("MainWindow")
65. MainWindow.resize(800, 600)
66. MainWindow.setWindowFlags(Qt.FramelessWindowHint)
67. MainWindow.setAttribute(Qt.WA\_NoSystemBackground, True)
68. MainWindow.setAttribute(Qt.WA\_TranslucentBackground, True)
69. self.centralwidget = QtWidgets.QWidget(MainWindow)
70. self.centralwidget.setObjectName("centralwidget")
71. self.verticalLayout\_2 = QtWidgets.QVBoxLayout(self.centralwidget)
72. self.verticalLayout\_2.setContentsMargins(0, 0, 0, 0)
73. self.verticalLayout\_2.setSpacing(0)
74. self.verticalLayout\_2.setObjectName("verticalLayout\_2")
75. self.verticalLayout = QtWidgets.QVBoxLayout()
76. self.verticalLayout.setSpacing(0)
77. self.verticalLayout.setObjectName("verticalLayout")
78. self.widget\_3 = QtWidgets.QWidget(self.centralwidget)
79. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
80. sizePolicy.setHorizontalStretch(0)
81. sizePolicy.setVerticalStretch(1)
82. sizePolicy.setHeightForWidth(self.widget\_3.sizePolicy().hasHeightForWidth())
83. self.widget\_3.setSizePolicy(sizePolicy)
84. self.widget\_3.setStyleSheet("background-color: rgb(38, 44, 52);")
85. self.widget\_3.setObjectName("widget\_3")
86. self.horizontalLayout\_2 = QtWidgets.QHBoxLayout(self.widget\_3)
87. self.horizontalLayout\_2.setContentsMargins(0, 0, 0, 0)
88. self.horizontalLayout\_2.setSpacing(0)
89. self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")
90. self.horizontalLayout = QtWidgets.QHBoxLayout()
91. self.horizontalLayout.setSpacing(0)
92. self.horizontalLayout.setObjectName("horizontalLayout")
93. self.label = QtWidgets.QLabel(self.widget\_3)
94. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
95. sizePolicy.setHorizontalStretch(20)
96. sizePolicy.setVerticalStretch(0)
97. sizePolicy.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())
98. self.label.setSizePolicy(sizePolicy)
99. font = QtGui.QFont()
100. font.setPointSize(8)
101. font.setItalic(False)
102. self.label.setFont(font)
103. self.label.setStyleSheet("color: rgb(255, 255, 255);")
104. self.label.setText("")
105. self.label.setObjectName("label")
106. self.horizontalLayout.addWidget(self.label)
108. self.pushButton\_3 = QtWidgets.QPushButton(self.widget\_3)
109. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)
110. sizePolicy.setHorizontalStretch(1)
111. sizePolicy.setVerticalStretch(0)
112. sizePolicy.setHeightForWidth(self.pushButton\_3.sizePolicy().hasHeightForWidth())
113. self.pushButton\_3.setSizePolicy(sizePolicy)
114. self.pushButton\_3.setText("Tr")
115. self.pushButton\_3.setStyleSheet("color: rgb(255, 255, 255);")
116. # icon = QtGui.QIcon()
117. # icon.addPixmap(QtGui.QPixmap("Handwritten OCR\_16px.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
118. # self.pushButton\_2.setIcon(icon)
119. # self.pushButton\_2.clicked.connect(self.shoot)
120. self.pushButton\_3.setObjectName("pushButton\_3")
121. self.horizontalLayout.addWidget(self.pushButton\_3)
123. self.pushButton\_2 = QtWidgets.QPushButton(self.widget\_3)
124. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)
125. sizePolicy.setHorizontalStretch(1)
126. sizePolicy.setVerticalStretch(0)
127. sizePolicy.setHeightForWidth(self.pushButton\_2.sizePolicy().hasHeightForWidth())
128. self.pushButton\_2.setSizePolicy(sizePolicy)
129. self.pushButton\_2.setText("")
130. icon = QtGui.QIcon()
131. icon.addPixmap(QtGui.QPixmap("Handwritten OCR\_16px.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
132. self.pushButton\_2.setIcon(icon)
133. self.pushButton\_2.clicked.connect(self.shoot)
134. self.pushButton\_2.setObjectName("pushButton\_2")
135. self.horizontalLayout.addWidget(self.pushButton\_2)
137. self.pushButton = QtWidgets.QPushButton(self.widget\_3)
138. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)
139. sizePolicy.setHorizontalStretch(1)
140. sizePolicy.setVerticalStretch(0)
141. sizePolicy.setHeightForWidth(self.pushButton.sizePolicy().hasHeightForWidth())
142. self.pushButton.setSizePolicy(sizePolicy)
143. self.pushButton.setText("")
144. icon1 = QtGui.QIcon()
145. icon1.addPixmap(QtGui.QPixmap("Delete\_16px.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
146. self.pushButton.setIcon(icon1)
147. self.pushButton.setObjectName("pushButton")
148. self.pushButton.clicked.connect(self.closeIt)
149. self.horizontalLayout.addWidget(self.pushButton)
150. self.horizontalLayout\_2.addLayout(self.horizontalLayout)
151. self.verticalLayout.addWidget(self.widget\_3)
152. self.widget\_2 = QtWidgets.QWidget(self.centralwidget)
153. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,QtWidgets.QSizePolicy.Preferred)
154. sizePolicy.setHorizontalStretch(0)
155. sizePolicy.setVerticalStretch(21)
156. sizePolicy.setHeightForWidth(self.widget\_2.sizePolicy().hasHeightForWidth())
157. self.widget\_2.setSizePolicy(sizePolicy)
158. self.widget\_2.setStyleSheet("border:3px solid rgb(62, 102, 141); opacity: 1;")
160. self.widget\_2.setObjectName("widget\_2")
161. self.verticalLayout.addWidget(self.widget\_2)
162. self.widget = QtWidgets.QWidget(self.centralwidget)
163. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,QtWidgets.QSizePolicy.Preferred)
164. sizePolicy.setHorizontalStretch(0)
165. sizePolicy.setVerticalStretch(1)
166. sizePolicy.setHeightForWidth(self.widget.sizePolicy().hasHeightForWidth())
167. self.widget.setSizePolicy(sizePolicy)
168. self.widget.setStyleSheet("background-color: rgb(38, 44, 52);")
169. self.widget.setObjectName("widget")
171. self.widget\_2.resizeEvent = self.resiz
172. self.sizegrip = QtWidgets.QSizeGrip(self.widget\_2)
173. self.sizegrip.setStyleSheet(
174. "border:0px solid rgb(62, 102, 141); opacity: 1; alignment: 'AlignRight | AlignBottom';")
175. width = self.widget\_2.frameGeometry().width()
176. height = self.widget\_2.frameGeometry().height()
177. self.label.setText(str(width) + "x" + str(height))
179. self.verticalLayout\_3 = QtWidgets.QVBoxLayout(self.widget)
180. self.verticalLayout\_3.setContentsMargins(0, 0, 0, 0)
181. self.verticalLayout\_3.setSpacing(0)
182. self.verticalLayout\_3.setObjectName("verticalLayout\_3")
183. self.frame = QtWidgets.QFrame(self.widget)
184. self.frame.setStyleSheet("background-color: rgb(38, 44, 52);")
185. self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
186. self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
187. self.frame.setObjectName("frame")
188. self.verticalLayout\_4 = QtWidgets.QVBoxLayout(self.frame)
189. self.verticalLayout\_4.setObjectName("verticalLayout\_4")
190. self.textBrowser = QtWidgets.QTextBrowser(self.frame)
191. self.textBrowser.setMaximumSize(QtCore.QSize(16777215, 65))
192. self.textBrowser.setStyleSheet("background-color: rgb(38, 44, 52);\n""color: rgb(0, 0, 0);")
193. self.textBrowser.setObjectName("textBrowser")
194. self.verticalLayout\_4.addWidget(self.textBrowser)
195. self.verticalLayout\_3.addWidget(self.frame)
196. self.verticalLayout.addWidget(self.widget)
197. self.verticalLayout\_2.addLayout(self.verticalLayout)
198. MainWindow.setCentralWidget(self.centralwidget)
200. self.retranslateUi(MainWindow)
201. QtCore.QMetaObject.connectSlotsByName(MainWindow)
203. **def** retranslateUi(self, MainWindow):
204. \_translate = QtCore.QCoreApplication.translate
205. MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))
207. **def** resiz(self, event):
208. width = self.widget\_2.frameGeometry().width()
209. height = self.widget\_2.frameGeometry().height()
210. self.label.setText("  " + str(width) + "x" + str(height))
212. **def** shoot(self):
213. **try**:
214. name = "buff.jpg"
216. x\_buff = self.widget\_2.x()
217. y\_buff = self.widget\_2.y()
218. x\_b = self.widget\_3.x()
219. y\_b = self.widget\_3.y()
220. x = self.MainWindow.x()
221. y = self.MainWindow.y()
222. width = self.widget\_2.width()
223. height = self.widget\_2.height()
225. path = name
226. ImageGrab.grab(bbox=(x + x\_buff, y + y\_buff, x + x\_buff + width, y + y\_buff + height)).save(path, "JPEG")
227. result = get\_string(path)
228. # print(result)
229. self.textBrowser.setText(result)
230. **except** Exception as e:
231. **print**(e)

234. **def** closeIt(self):
235. self.close()


239. **if** \_\_name\_\_ == "\_\_main\_\_":
240. app = QtWidgets.QApplication(sys.argv)
241. ex = Ui\_MainWindow()
242. w = QtWidgets.QMainWindow()
243. setMoveWindow(w)
244. ex.setupUi(w)
245. w.show()
246. sys.exit(app.exec\_())

tesseract.py:

1. **import** cv2
2. **import** numpy as np
3. **import** pytesseract
4. **try**:
5. **import** Image
6. **except** ImportError:
7. **from** PIL **import** Image
8. **import** pytesseract

11. **def** get\_string(path):
12. **try**:
13. img = cv2.imread(path)
14. img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)
15. img = cv2.fastNlMeansDenoising(img,None,10,7,21 )
16. kernel = np.ones((1, 1), np.uint8)
17. img = cv2.dilate(img, kernel, iterations=1)
18. img = cv2.erode(img, kernel, iterations=1)
19. img = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY, 31, 2)
20. # cv2.imshow("Output", img)
21. cv2.imwrite("buff.jpg", img)
22. pytesseract.pytesseract.tesseract\_cmd = "C:\\Program Files (x86)\\Tesseract-OCR\\tesseract.exe"
23. **except** Exception as e:
24. **print**(e)
25. **return**  pytesseract.image\_to\_string(Image.open(path), lang='eng+rus')

Список использованных источников

[1] G. Nagy, Y. Xu, “Automatic Prototype Extraction for Adaptive OCR”, *Proc. of the 4th Int. Conf. on Document Analysis and Recognition*, IEEE, Aug 1997, pp 278-282.

[2] I. Marosi, “Industrial OCR approaches: architecture, algorithms and adaptation techniques”, *Document Recognition and Retrieval XIV*, SPIE Jan 2007, 6500-01.

[3] S.V. Rice, G. Nagy, T.A. Nartker, *Optical Character Recognition: An Illustrated Guide to the Frontier*, Kluwer Academic Publishers, USA 1999, pp. 57-60.

[4] Shannon C. E., Weaver W., Burks A. W. *The mathematical theory of communication*. – 1951.

[5] R.J. Shillman, *Character Recognition Based on Phenomenological Attributes: Theory and Methods*, PhD. Thesis, Massachusetts Institute of Technology. 1974.

[6] Smith R. An overview of the Tesseract OCR engine //*Document Analysis and Recognition*, *2007. ICDAR 2007. Ninth International Conference on*. – IEEE, 2007. – Т. 2. – С. 629-633.

[7] R.W. Smith, T*he Extraction and Recognition of Text from Multimedia Document Images*, PhD Thesis, University of Bristol, November 1987.