|  |
| --- |
|  |
| 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 И.О.

подпись, дата

Группа БСБО-02-15

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

МОСКВА, 2017 г.

|  |
| --- |
|  |
| 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-02-15

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 И.О.

подпись, дата

МОСКВА, 2017 г.

Содержание

[Содержание 1](#_Toc501590458)

[Введение 1](#_Toc501590459)

[Исследовательский раздел 2](#_Toc501590460)

[Принципы работы OCR 3](#_Toc501590461)

[Составляющие проекта 5](#_Toc501590462)

[Программное обеспечение 5](#_Toc501590463)

[Техническое задание 6](#_Toc501590464)

[Введение. 6](#_Toc501590465)

[Основания для разработки. 6](#_Toc501590466)

[Назначение разработки. 6](#_Toc501590467)

[Требования к программе. 6](#_Toc501590468)

[Стадии и этапы разработки. 7](#_Toc501590469)

[Заключение 8](#_Toc501590470)

[Приложение 9](#_Toc501590471)

[Список использованной литературы 14](#_Toc501590472)

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.

Исследовательский раздел

Принципы работы OCR

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.

Программное обеспечение

Для разработки приложения была использована IDE PyCharm от JetBrains s.r.o.

Техническое задание

Введение.

Наименование программы.

Наименование программы: «MKE – Make Life Easy»

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.

Разработанное 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 разработанной 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.

1. Требования к составу и параметрам технических средств.
   1. В состав технических средств должен входить персональный компьютер. на базе операционной системы Windows 7 и далее, или семейства ОС UNIX.
   2. Оперативная память объемом от 1 Гб.
   3. Свободное место на SSD или HDD диске объемом не менее 80 Мб.
2. Требования к информационной и программной совместимости.

3.1 Требования к исходным кодам и языкам программирования.

Разработка данной программы должно вестись на языке Python. Дополнительных требований к средам разработки не предъявляется.

Стадии и этапы разработки.

1. Стадии разработки.

Разработка должна быть проведена в три стадии:   
1.1 Разработка технического задания.

* 1. Рабочее проектирование.
  2. Релиз.

1. Этапы разработки.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. разработка программы;

2. разработка программной документации;

3. испытания программы.

На стадии релиза должен быть выполнен этап разработки, подготовка и выпуск программы.

Заключение

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

При выполнении, для создания программы была использована интегрированная среда разработки PyCharm. В ходе работы был разработан понятный и удобный для использования пользовательский интерфейс. Готовое приложение обеспечивает комфортную работу и выполнение требуемых от ПО функций.

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

В процессе написания курсового проекта была использована разнообразная профильная литература и Интернет-ресурсы о разработке и алгоритмах работы OCR.

Приложение

Основное окно приложения:



Исходный код программы:

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)


125. self.pushButton\_2 = QtWidgets.QPushButton(self.widget\_3)
126. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)
127. sizePolicy.setHorizontalStretch(1)
128. sizePolicy.setVerticalStretch(0)
129. sizePolicy.setHeightForWidth(self.pushButton\_2.sizePolicy().hasHeightForWidth())
130. self.pushButton\_2.setSizePolicy(sizePolicy)
131. self.pushButton\_2.setText("")
132. icon = QtGui.QIcon()
133. icon.addPixmap(QtGui.QPixmap("Handwritten OCR\_16px.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
134. self.pushButton\_2.setIcon(icon)
135. self.pushButton\_2.clicked.connect(self.shoot)
136. self.pushButton\_2.setObjectName("pushButton\_2")
137. self.horizontalLayout.addWidget(self.pushButton\_2)

140. self.pushButton = QtWidgets.QPushButton(self.widget\_3)
141. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)
142. sizePolicy.setHorizontalStretch(1)
143. sizePolicy.setVerticalStretch(0)
144. sizePolicy.setHeightForWidth(self.pushButton.sizePolicy().hasHeightForWidth())
145. self.pushButton.setSizePolicy(sizePolicy)
146. self.pushButton.setText("")
147. icon1 = QtGui.QIcon()
148. icon1.addPixmap(QtGui.QPixmap("Delete\_16px.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
149. self.pushButton.setIcon(icon1)
150. self.pushButton.setObjectName("pushButton")
151. self.pushButton.clicked.connect(self.closeIt)
152. self.horizontalLayout.addWidget(self.pushButton)
153. self.horizontalLayout\_2.addLayout(self.horizontalLayout)
154. self.verticalLayout.addWidget(self.widget\_3)
155. self.widget\_2 = QtWidgets.QWidget(self.centralwidget)
156. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
157. sizePolicy.setHorizontalStretch(0)
158. sizePolicy.setVerticalStretch(21)
159. sizePolicy.setHeightForWidth(self.widget\_2.sizePolicy().hasHeightForWidth())
160. self.widget\_2.setSizePolicy(sizePolicy)
161. self.widget\_2.setStyleSheet("border:3px solid rgb(62, 102, 141); opacity: 1;")

164. self.widget\_2.setObjectName("widget\_2")
165. self.verticalLayout.addWidget(self.widget\_2)
166. self.widget = QtWidgets.QWidget(self.centralwidget)
167. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
168. sizePolicy.setHorizontalStretch(0)
169. sizePolicy.setVerticalStretch(1)
170. sizePolicy.setHeightForWidth(self.widget.sizePolicy().hasHeightForWidth())
171. self.widget.setSizePolicy(sizePolicy)
172. self.widget.setStyleSheet("background-color: rgb(38, 44, 52);")
173. self.widget.setObjectName("widget")
175. self.widget\_2.resizeEvent = self.resiz
176. self.sizegrip = QtWidgets.QSizeGrip(self.widget\_2)
177. self.sizegrip.setStyleSheet(
178. "border:0px solid rgb(62, 102, 141); opacity: 1; alignment: 'AlignRight | AlignBottom';")
179. width = self.widget\_2.frameGeometry().width()
180. height = self.widget\_2.frameGeometry().height()
181. self.label.setText(str(width) + "x" + str(height))
183. self.verticalLayout\_3 = QtWidgets.QVBoxLayout(self.widget)
184. self.verticalLayout\_3.setContentsMargins(0, 0, 0, 0)
185. self.verticalLayout\_3.setSpacing(0)
186. self.verticalLayout\_3.setObjectName("verticalLayout\_3")
187. self.frame = QtWidgets.QFrame(self.widget)
188. self.frame.setStyleSheet("background-color: rgb(38, 44, 52);")
189. self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
190. self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
191. self.frame.setObjectName("frame")
192. self.verticalLayout\_4 = QtWidgets.QVBoxLayout(self.frame)
193. self.verticalLayout\_4.setObjectName("verticalLayout\_4")
194. self.textBrowser = QtWidgets.QTextBrowser(self.frame)
195. self.textBrowser.setMaximumSize(QtCore.QSize(16777215, 65))
196. self.textBrowser.setStyleSheet("background-color: rgb(38, 44, 52);\n""color: rgb(0, 0, 0);")
197. self.textBrowser.setObjectName("textBrowser")
198. self.verticalLayout\_4.addWidget(self.textBrowser)
199. self.verticalLayout\_3.addWidget(self.frame)
200. self.verticalLayout.addWidget(self.widget)
201. self.verticalLayout\_2.addLayout(self.verticalLayout)
202. MainWindow.setCentralWidget(self.centralwidget)
204. self.retranslateUi(MainWindow)
205. QtCore.QMetaObject.connectSlotsByName(MainWindow)
207. **def** retranslateUi(self, MainWindow):
208. \_translate = QtCore.QCoreApplication.translate
209. MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))
211. **def** resiz(self, event):
212. width = self.widget\_2.frameGeometry().width()
213. height = self.widget\_2.frameGeometry().height()
214. self.label.setText("  " + str(width) + "x" + str(height))
216. **def** shoot(self):
217. **try**:
218. name = "buff.jpg"
220. x\_buff = self.widget\_2.x()
221. y\_buff = self.widget\_2.y()
222. x\_b = self.widget\_3.x()
223. y\_b = self.widget\_3.y()
224. x = self.MainWindow.x()
225. y = self.MainWindow.y()
226. width = self.widget\_2.width()
227. height = self.widget\_2.height()
229. path = name
230. ImageGrab.grab(bbox=(x + x\_buff, y + y\_buff, x + x\_buff + width, y + y\_buff + height)).save(path, "JPEG")
231. result = get\_string(path)
232. # print(result)
233. self.textBrowser.setText(result)
234. **except** Exception as e:
235. **print**(e)

238. **def** closeIt(self):
239. self.close()


243. **if** \_\_name\_\_ == "\_\_main\_\_":
244. app = QtWidgets.QApplication(sys.argv)
245. ex = Ui\_MainWindow()
246. w = QtWidgets.QMainWindow()
247. setMoveWindow(w)
248. ex.setupUi(w)
249. w.show()
250. 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 использованной литературы

|  |  |
| --- | --- |
| [1] | R. Smith, «The Extraction and Recognition of Text from,» University of Bristol, 1987. |
| [2] | R. Smith, «An Overview of the Tesseract OCR Engine,» 2007. |
| [3] | I. Marosi, «Industrial OCR approaches: architecture,algorithms and adaptation techniques,» SPIE, 2007. |
| [4] | R. Shillman, «Character Recognition Based on Phenomenological Attributes: Theory and Methods, PhD Thesis,» Massachusetts Institute of Technology, 1974. |
| [5] | G. N. T. N. S.V. Rice, Optical Character Recognition: An Illustrated Guide to the Frontier, Kluwer Academic Publishers, 1999. |
| [6] | Y. X. G. Nagy, «Automatic Prototype Extraction for Adaptive OCR,» 1997. |
| [7] | C. E. Shannon, «A Mathematical Theory of Communication,» 1948. |