УДК 343.982.4
doi: 10.25724/VAMVD.A236

ПРИМЕНЕНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON ДЛЯ РЕШЕНИЯ ОТДЕЛЬНЫХ ЗАДАЧ ТЕХНИКО-КРИМИНАЛИСТИЧЕСКОЙ ЭКСПЕРТИЗЫ ДОКУМЕНТОВ

Алексей Федорович Купин*, *Александра Ивановна Дончук*
* Следственный комитет Российской Федерации, МГТУ им. Н. Э. Баумана, Москва, Россия, alexcrim@rambler.ru
** ООО «РусБИТех-Астра», МГТУ им. Н. Э. Баумана, Москва, Россия, sachadonchuk2001@mail.ru

Аннотация. В статье рассматриваются возможности решения задач технико-криминалистической экспертизы документов с помощью скриптов, написанных на языке программирования Python, реализующих свое назначение с помощью алгоритмов компьютерного зрения. По сравнению с графическими редакторами, такими как GIMP и KRITA, применяемыми в процессе изучения документов, разработанные инструменты более просты в эксплуатации и лишены избыточного функционала. Их применение к изображениям документов напрямую позволяет успешно решать следующие задачи: выявлять содержание рукописной записи, штрихи которой были частично утрачены в результате смывания; устанавливать содержание вдавленных неокрашенных записей; выявлять и фиксировать факт дописки; определять содержание зачеркнутых записей в ситуациях, когда первоначальные записи и записи, их покрывающие, выполняются разными по цвету материалами письма одного вида.

По результатам апробации предлагаемых инструментов на изученных объектах – рукописных реквизитах документов, подвергшихся разным видам внешнего воздействия, продемонстрирована эффективность использования разработанных скриптов при решении широкого круга задач технико-криминалистической экспертизы документов.

Ключевые слова: документ, языки программирования, методы экспертного исследования, судебная технико-криминалистическая экспертиза документов, заключение эксперта

Для цитирования: Купин А. Ф., Дончук А. И. Применение языка программирования Python для решения отдельных задач технико-криминалистической экспертизы документов // Судебная экспертиза. 2024. № 1 (77). С. 81–92. doi: 10.25724/VAMVD.A236

© Купин А. Ф., Дончук А. И., 2024

APPLICATION OF THE PYTHON PROGRAMMING LANGUAGE TO SOLVING SELECTED TASKS OF TECHNICAL AND FORENSIC EXAMINATION OF DOCUMENTS

Alexey Fedorovich Kupin*, Alexandra Ivanovna Donchuk**
*Investigative Committee of the Russian Federation,
BMSTU, Moscow, Russia, alexcrim@rambler.ru
** LLC "RusBITech-Astra", BMSTU, Moscow, Russia, sachadonchuk2001@mail.ru

Abstract. The article explores the possibilities of solving forensic document examination tasks using scripts written in the programming language "Python", implementing their purpose through computer vision algorithms. In comparison to graphic editors such as GIMP and KRITA, commonly used in document analysis, the developed tools are simpler to use and devoid of redundant functionality. Their application to document images directly enables the successful resolution of the following tasks: identifying the content of handwritten notes, the strokes of which have been partially lost due to smudging; determining the content of recessed unpainted entries; detecting and recording the fact of addition; determining the content of crossed-out entries in situations where the original entries and the entries covering them are made with materials of different colors in the same type of document.

Following the validation of the proposed tools on examined objects-handwritten details of documents subjected to diverse external factors the efficacy of employing the developed scripts in addressing a broad spectrum of forensic document examination tasks has been showcased.

Key words: document, programming languages, expert research methods, forensic technical and forensic examination of documents, expert opinion

For citation: Kupin A. F., Donchuk A. I. Application of the Python programming language to solving selected tasks of technical and forensic examination of documents. Forensic Examination, 81–92, 2024. (In Russ.). doi: 10.25724/VAMVD.A236

Для решения части задач технико-криминалистической экспертизы документов и почерковедческой экспертизы нередко используются компьютерные программы обработки изображений, в частности, растровые графические редакторы, обладающие большим функционалом для редактирования различных параметров изображения [1, с. 74–75]. К растровым графическим редакторам относятся такие компьютерные программы, как Adobe Photoshop, GIMP, KRITA и др., позволяющие выявлять факты дописки; устанавливать содержание записи, подвергшейся смыванию; первоначальное содержание рукописного текста, выполненного шариковой ручкой с пастой синего цвета и зачеркнутого шариковой ручкой с пастой черного цвета; содержание слабовидимой записи, образованной вдавленными неокрашенными штрихами [2, с. 29–31].

В первую очередь для решения перечисленных задач используются инструменты, которые дают возможность работать с цветом, яркостью и контрастом изображений (в графическом редакторе GIMP инструменты «Цвет-Экспозиция», «Цвет-Тон-Насыщенность», «Цвет-Кривые», «Цвет-Цветовой баланс»; в графическом редакторе «KRITA» инструменты «Фильтр-Коррекция-Коррекция HSV/HSL», «Фильтр-Коррекция-Коррекция цвета кривыми», «Фильтр-Коррекция-

Цветовой баланс»). Аналогичные инструменты имеют место в решении задач почерковедческой экспертизы, например, в случае необходимости улучшения качества представленного почеркового объекта путем повышения резкости и контрастности, выявления слабовидимых штрихов, удаления элементов, мешающих восприятию информативной составляющей [3, с. 90].

В основу функционирования рассматриваемых графических редакторов заложены определенные алгоритмы, позволяющие экспертам выполнять операции с изучаемыми изображениями. При этом нужно понимать, что любой графический редактор изначально является программой с заранее заданными функциональными возможностями, не предназначенными для решения задач, направленных на выявление признаков технической подделки документов. Принципы, по которым осуществляется непосредственная работа графических редакторов в части обработки изображений, могут быть установлены экспертами только в общих чертах, исходя из имеющихся у них знаний о формировании изображения, полученных во время обучения по конкретной экспертной специальности. Не стоит забывать еще и то, что доступность последних версий ряда графических редакторов в свободном обороте ограниченна, в ряде случаев для работы с ними следует закупить лицензию, а в некоторых ситуациях применение в работе иностранных программных продуктов противоречит требованиям безопасности государственной организации.

Альтернатива графическим редакторам для выявления признаков технической подделки документов – написание программного кода под конкретную задачу исходя из знания алгоритмов формирования изображения, наличия библиотек компьютерного зрения, а также принципов, по которым осуществляется обработка (изучение) исследуемых изображений документов. Одним из инструментов, позволяющих написать такой код, является язык программирования Python. Его преимущества над другими языками программирования заключаются в следующем [4, с. 49]:

1. Высокоуровневость, т. е. использование абстрактных структур для описания данных и операций над ними, упрощающее процесс написания и понимания программного кода.

2. Удобочитаемость и лаконичность. Синтаксис Python облегчает зрительное восприятие программного кода, сокращает требуемый объем его строк.

3. Интерпретируемость. Программный код, написанный на данном языке, выполняется сразу с помощью программы-интерпретатора без предварительного перевода в машинный код, что обеспечивает совместимость с разными аппаратными платформами.

4. Универсальность. Python предназначен для решения широкого круга задач за счет наличия большого количества доступных для подключения библиотек.

Для исследования изображений с помощью языка программирования Python может быть использована OpenCV – библиотека компьютерного зрения с открытым исходным кодом [5, с. 6]. Она содержит различные алгоритмы компьютерного зрения на основе машинного обучения, позволяющие работать с изображениями для идентификации лиц и иных объектов, улучшения качества изображений и внесения в них изменений (замены фона, цвета и др.).

Основные операции OpenCV для работы с изображениями:
1) cv2.imread() – считывание данных из файла;
2) cv2.imshow() – вывод содержимого файла на экран;
3) cv2.resize() – изменение размера изображения;
4) cv2.getRotationMatrix2D(), cv2.warpAffine() – поворот изображения;
5) cv2.flip() – зеркальное отражение изображения по осям;
6) cv2.imwrite() – сохранение изображения в файл и др. [5, с. 37–48]

Рассмотрим возможности применения библиотеки OpenCV в решении отдельных задач технико-криминалистической экспертизы документов на примерах, встречающихся в экспертной практике.

1. Установление содержания записи, подвергшейся смыванию. Решение данной задачи продемонстрируем на примере восстановления первоначального содержания слова «день», выполненного пастой для шариковой ручки синего цвета, последняя буква которого была смыта с помощью 30 %-го изопропилового спирта (рис. 1).

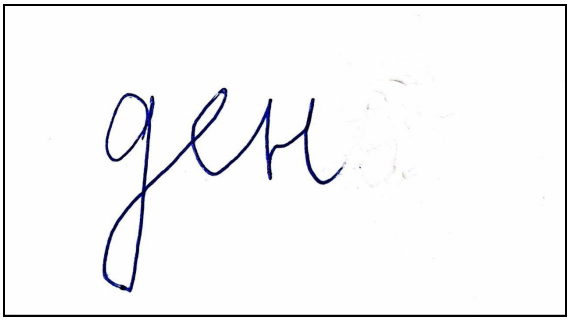

Рис. 1. Исходное изображение

Для решения данной задачи был написан следующий программный код[1]:

```
import cv2 // добавление библиотеки OpenCV
import numpy as np // добавление библиотеки NumPy и присваивание ей имени «np»
image = cv2.imread('image_1.jpg') // чтение файла «image_1.jpg» (исходного изображения)
lut_in = [0, 127, 255] // входной массив пикселей изображения
lut_out = [0, 0, 1000] // выходной массив пикселей изображения
lut_8u = np.interp(np.arange(0, 256), lut_in, lut_out).astype(np.uint8) // преобразование массива пикселей изображения с помощью функции LUT (таблиц поиска)
image_contrasted = cv2.LUT(image, lut_8u) // применение функции LUT к изображению
cv2.imwrite('test_1.jpg', image_contrasted) // сохранение результата выполнения функции LUT в файл «test_1.jpg»
```

[1] Здесь и далее сочетание символов «//» отделяет комментарий от кода.

При написании программного кода было принято во внимание следующее. Изображение представляет собой совокупность отдельных точек (пикселей) различных цветов. Интенсивность цвета пикселя определяется числом и находится в диапазоне от 0 (самый темный) до 255 (самый светлый) [6, с. 27]. Функция LUT позволяет преобразовать входной массив пикселей (lut_in) в новый массив с измененными числовыми значениями (lut_out). В рассматриваемом примере необходимо увеличить контраст на изображении и тем самым установить смытую часть слова. Для этого пиксели со значением 127 (среднего уровня интенсивности) были заменены пикселями со значением 0 (максимально темными пикселями), а пиксели со значением 255 (самые светлые) – пикселями со значением 1 000 (выход за границы диапазона нарушает цветопередачу, однако не препятствует осветлению).

Применение программного кода к исходному изображению позволило получить результат, представленный на рисунке 2. На изображении отобразились контуры смытой буквы «ь».

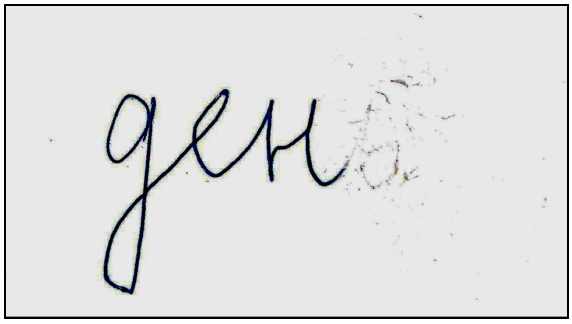

Рис. 2. Итоговое изображение

2. Установление содержания слабовидимой записи, образованной вдавленными неокрашенными штрихами, на примере слова «утро» (рис. 3).

Рис. 3. Исходное изображение

Для решения задачи был написан программный код, аналогичный предыдущему. Применение программного кода к исходному изображению способствовало получению результата, представленного на рисунке 4. На изображении удалось прочитать слово «утро».

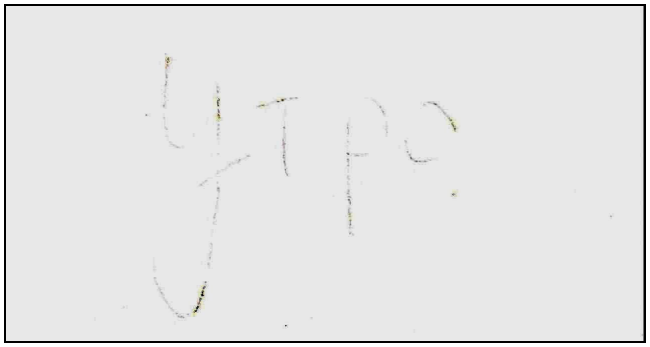

Рис. 4. Итоговое изображение

3. Выявление факта дописки. Решение задачи продемонстрировано на примере слова «рассказ», выполненного шариковой ручкой с пастой синего цвета, к которому было дописано сочетание букв «чик» другой шариковой ручкой с пастой синего цвета (рис. 5).

Рис. 5. Исходное изображение

Для ее решения был написан следующий программный код:

```
import cv2 // добавление библиотеки OpenCV
import numpy as np // добавление библиотеки NumPy и присваивание ей имени «np»
image = cv2.imread('image_3.jpg') // чтение файла «image_3.jpg» (исходного изображения)
b, g, r = cv2.split(image) // разложение изображения на три цветовых канала: синий, зеленый, красный
cv2.imwrite('test_3.jpg', b) // сохранение синего канала в файл «test_3.jpg»
new_image = cv2.imread('test_3.jpg') // чтение файла «test_3.jpg»
lut_in = [0, 50, 100, 150, 200, 255] // входной массив пикселей изображения
```

```
lut_out = [0, 0, 255, 255, 255, 255] // выходной массив пикселей изображения
lut_8u = np.interp(np.arange(0, 256), lut_in, lut_out).astype(np.uint8) // преобразование массива пикселей изображения с помощью функции LUT (таблиц поиска)
image_contrasted = cv2.LUT(new_image, lut_8u) // применение функции LUT к изображению
cv2.imwrite('test_3.jpg', image_contrasted) // сохранение результата выполнения функции LUT в файл «test_3.jpg»
```

При написании программного кода было принято во внимание следующее. В рассматриваемом примере обе части слова выполнены пастой синего цвета, в связи с этим для повышения контраста необходимо выделить синий канал изображения и в дальнейшем производить операции только над ним. Стандартным цветовым пространством для библиотеки OpenCV является BGR (то же, что и цветовое пространство RGB с каналами R (красным), G (зеленым), B (синим) [6, с. 27], но с другим порядком расположения каналов). При разложении изображения на цветовые каналы была использована функция cv2.split. Для более тонкой настройки входной массив пикселей был представлен расширенным набором числовых значений. В целях повышения контраста темные пиксели со значением 50 были заменены на самые темные (со значением 0); более светлые пиксели (со значениями 100, 150, 200) – на самые светлые (со значением 255).

Применение программного кода к исходному изображению позволило получить результат, представленный на рисунке 6. На изображении видно различие штрихов букв «рассказ» и «чик» по интенсивности окраски.

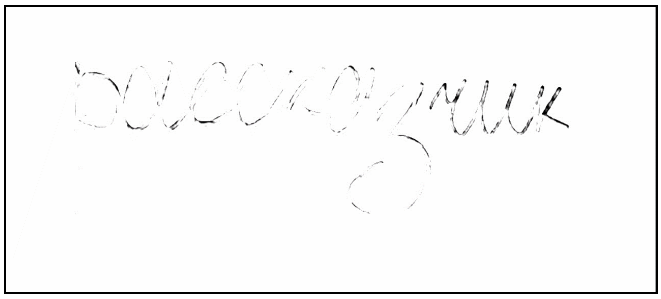

Рис. 6. Итоговое изображение

4. Установление первоначального содержания рукописного текста, выполненного шариковой ручкой с пастой черного цвета и зачеркнутого шариковой ручкой с пастой синего цвета. Решение данной задачи продемонстрировано на примере установления слова «лето» (рис. 7).

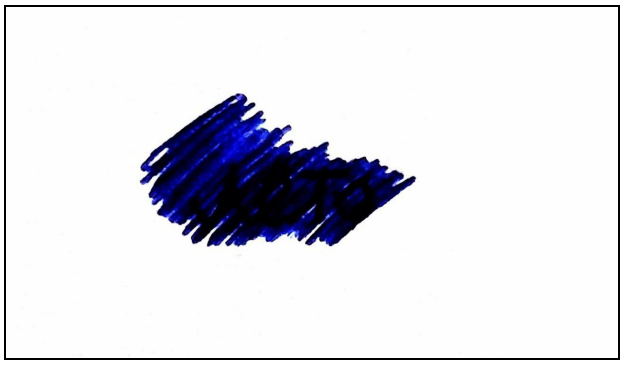

Рис. 7. Исходное изображение

Для решения данной задачи был написан следующий программный код:

```
import cv2 // добавление библиотеки OpenCV
import numpy as np // добавление библиотеки NumPy и присваивание ей имени «np»
img = cv2.imread('image_4.jpg') // чтение файла «image_4.jpg»
b, g, r = cv2.split(img) // разложение изображения на три цветовых канала: синий, зеленый, красный
cv2.imwrite('test_4.jpg', b) // сохранение синего канала в файл «test_4.jpg»
new_image = cv2.imread('test_4.jpg') // чтение файла «test_4.jpg»
lut_in = [0, 50, 100, 150, 200, 255] // входной массив пикселей изображения
lut_out = [0, 255, 255, 255, 255, 255] // выходной массив пикселей изображения
lut_8u = np.interp(np.arange(0, 256), lut_in, lut_out).astype(np.uint8) // преобразование массива пикселей изображения с помощью функции LUT (таблиц поиска)
image_contrasted = cv2.LUT(new_image, lut_8u) // применение функции LUT к изображению
cv2.imwrite('test_4.jpg', image_contrasted) // сохранение результата выполнения функции LUT в файл «test_4.jpg»
```

При написании программного кода было принято во внимание следующее. Программный код примера аналогичен предыдущему, однако в силу того, что штрихи одного цвета были нанесены поверх штрихов другого цвета, для повышения контраста пиксели со всеми значениями, кроме 0, были заменены самыми светлыми пикселями (со значением 255).

Применение программного кода к исходному изображению позволило получить результат, представленный на рисунке 8: на изображении удалось прочитать слово «лето».

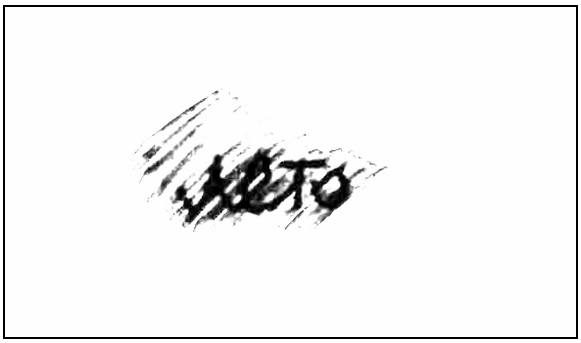

Рис. 8. Итоговое изображение

5. Установление первоначального содержания рукописного текста, выполненного шариковой ручкой с пастой синего цвета и зачеркнутого шариковой ручкой с пастой черного цвета. Решение данной задачи продемонстрировано на примере установления слова «зима» (рис. 9).

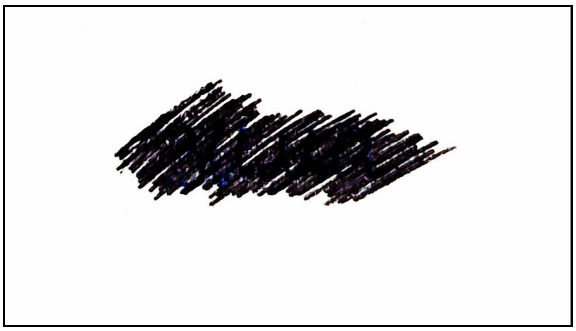

Рис. 9. Исходное изображение

Для решения данной задачи был написан следующий программный код:

```
import cv2 // добавление библиотеки OpenCV
import numpy as np // добавление библиотеки NumPy и присваивание ей имени «np»
img = cv2.imread('image_5.jpg') // чтение файла «image_5.jpg»
lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB) // преобразование цветового пространства изображения в Lab
l_channel, a, b = cv2.split(lab) // разложение изображения на три канала: яркость, диапазон цветов от красного до зеленого, диапазон цветов от желтого до синего
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8)) // инициализация алгоритма CLAHE (адаптивного выравнивания гистограммы с ограниченным контрастом)
cl = clahe.apply(l_channel) // применение алгоритма CLAHE к каналу яркости
l_img = cv2.merge((cl, a, b)) // объединение каналов изображения
```

```
enhanced_img = cv2.cvtColor(l_img, cv2.COLOR_LAB2BGR) // преобразование цветового пространства изображения в BGR
cv2.imwrite('test_5.jpg', enhanced_img) // сохранение результата преобразования в файл «test_5.jpg»
lut_in = [0, 50, 100, 150, 200, 255] // входной массив пикселей изображения
lut_out = [0, 255, 255, 255, 255, 255] // выходной массив пикселей изображения
lut_8u = np.interp(np.arange(0, 256), lut_in, lut_out).astype(np.uint8) // преобразование массива пикселей изображения с помощью функции LUT (таблиц поиска)
image_contrasted = cv2.LUT(enhanced_img, lut_8u) // применение функции LUT к изображению
cv2.imwrite('test_5.jpg', image_contrasted) // сохранение результата выполнения функции LUT в файл «test_5.jpg»
```

При написании данного программного кода было принято во внимание следующее. В силу того что штрихи более темного цвета нанесены поверх штрихов более светлого цвета, для повышения контраста было необходимо выполнить ряд дополнительных действий. Сначала изображение было преобразовано в цветовое пространство Lab и разложено на каналы L (яркость), a (диапазон цветов от красного до зеленого), b (диапазон цветов от желтого до синего) [7, с. 171–172]. Затем к каналу яркости был применен алгоритм CLAHE – адаптивное выравнивание гистограммы с ограничением контраста. В качестве порога ограничения контраста (clipLimit) было задано значение 2.0, а в качестве количества блоков, на которое разбивается изображение для осуществления выравнивания (tileGridSize), – значение 8,8 (8 блоков в строке и 8 блоков в столбце). Далее каналы изображения были объединены с помощью функции cv2.merge, и получившееся изображение преобразовано в цветовое пространство BGR. Преобразование пикселей изображения с помощью функции LUT было выполнено аналогично предыдущему примеру.

Применение программного кода к исходному изображению позволило получить результат, представленный на рисунке 10. На изображении удалось прочитать слово «зима».

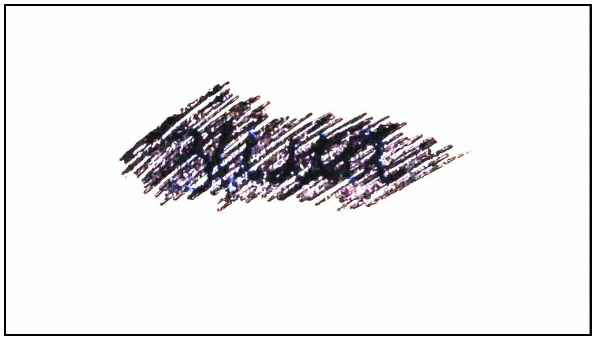

Рис. 10. Итоговое изображение

Таким образом, анализируя результаты эксперимента, следует заключить, что язык программирования Python является эффективным инструментом для решения отдельных задач технико-криминалистической экспертизы документов. С его помощью, благодаря применению библиотеки OpenCV, могут быть решены такие задачи, как выявление факта дописки; установление содержания записи, подвергшейся смыванию; содержания слабовидимой записи, образованной вдавленными неокрашенными штрихами; первоначального содержания рукописного текста, выполненного шариковой ручкой с пастой черного цвета и зачеркнутого шариковой ручкой с пастой синего цвета; первоначального содержания рукописного текста, выполненного шариковой ручкой с пастой синего цвета и зачеркнутого шариковой ручкой с пастой черного цвета.

Список источников

1. Баринова О. А., Купин А. Ф., Титаренко В. А. Установление факта изменения первоначального содержания документа методами цифровой обработки изображений // Судебная экспертиза. 2017. № 1 (49). С. 74–86.

2. Купин А. Ф., Дончук А. И. Использование растровых графических редакторов в технико-криминалистической экспертизе документов // Эксперт-криминалист. 2023. № 3. С. 29–31.

3. Жижина М. В. Возможности использования графических редакторов при работе с изображениями цифровых почерковых объектов, представленных в виде электронных файлов // Вестник Московского университета МВД России. 2023. № 3. С. 89–91.

4. Любанович Б. Простой Python. Современный стиль программирования. 2-е изд. Санкт-Петербург: Питер, 2021. 592 с.

5. Шакирьянов Э. Д. Компьютерное зрение на Python. Первые шаги. Москва: Лаборатория знаний, 2021. 163 с.

6. Демин А. Ю. Основы компьютерной графики: учеб. пособие. Томск: Изд-во Том. политехн. ун-та, 2011. 191 с.

7. Лютов В. П., Четверкин П. А., Головастиков Г. Ю. Цветоведение и основы колориметрии: учеб. и практикум для вузов. 3-е изд., перераб. и доп. Москва: Юрайт, 2022. 224 с.

References

1. Barinova O. A., Kupin A. F., Titarenko V. A. Establishing the fact of a change in the original content of a document using digital image processing methods. Forensic examination, 74–86, 2017. (In Russ.).

2. Kupin A. F., Donchuk A. I. The use of raster graphic editors in the technical and forensic examination of documents. Forensic expert, 29–31, 2023. (In Russ.).

3. Zhizhina M. V. Possibilities of using graphic editors when working with images of digital handwriting objects presented in the form of electronic files. Bulletin of the Moscow University of the Ministry of Internal Affairs of Russia, 89–91, 2023. (In Russ.).

4. Lyubanovich B. Simple Python. Modern programming style. 2nd ed. Saint Petersburg: Peter; 2021: 592. (In Russ.).

5. Shakiryanov E. D. Computer vision in Python. First steps. Moscow: Laboratory of Knowledge; 2021: 163. (In Russ.).

6. Demin A. Yu. Fundamentals of computer graphics. Textbook. Tomsk: Tomsk Polytechnic University Publishing House; 2011: 191. (In Russ.).

7. Lyutov V. P., Chetverkin P. A., Golovastikov G. Yu. Color science and the basics of colorimetry. Textbook and workshop for universities. 3rd ed., rev. and add. Moscow: Yurait; 2022: 224. (In Russ.).

Купин Алексей Федорович,
старший инспектор управления научно-исследовательской деятельности
(научно-исследовательского института криминалистики)
Главного управления криминалистики (Криминалистического центра)
Следственного комитета Российской Федерации,
доцент кафедры безопасности в цифровом мире МГТУ им. Н. Э. Баумана,
кандидат юридических наук, доцент; alexcrim@rambler.ru

Дончук Александра Ивановна,
младший инженер отдела технологической совместимости
департамента развития технологического сотрудничества ДВиС
ООО «РусБИТех-Астра», лаборант кафедры безопасности в цифровом мире
МГТУ им. Н. Э. Баумана; sachadonchuk2001@mail.ru

Kupin Alexey Fedorovich,
senior inspector of the research directorate
(research institute of criminalistics)
of the Chief criminalistic directorate (criminalistic center)
of the Investigative Committee of the Russian Federation,
associate professor of the department of security
in the digital world of the BMSTU,
candidate of juridical sciences, associate professor; alexcrim@rambler.ru

Donchuk Alexandra Ivanovna,
junior engineer of the technological compatibility department
of the department for the development of technological cooperation
of DIaM LLC "RusBITech-Astra", laboratory assistant
of the department of security in the digital world of the BMSTU;
sachadonchuk2001@mail.ru

Статья поступила в редакцию 11.12.2023; одобрена после рецензирования 20.12.2023; принята к публикации 25.01.2024.

The article was submitted 11.12.2023; approved after reviewing 20.12.2023; accepted for publication 25.01.2024.

* * *