УДК 004.932 (045)
DOI: 10.22213/2410-9304-2019-3-4-67-74

ПРОГРАММА ДЛЯ АВТОМАТИЧЕСКОЙ КЛАСТЕРИЗАЦИИ МНОГОЧИСЛЕННЫХ (СВЫШЕ 1000) ИЗОБРАЖЕНИЙ ЗЕРЕН ПЫЛЬЦЫ, ПОЛУЧЕННЫХ С ПОМОЩЬЮ ОПТИЧЕСКОГО МИКРОСКОПА, В PYTHON, ДЛЯ ОБУЧЕНИЯ СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ LENET

Ю. Б. Камалова, ИжГТУ имени М.Т. Калашникова, Ижевск, Россия

Рассматривается способ сортировки изображений пыльцевых зерен на этапе предобработки для формирования выборки для обучения сверточной нейронной сети LeNet. На python разработана программа кадрирования изображения пыльцы, полученного с помощью оптического микроскопа, ImageSplitter, с помощью которой было кадрировано изображение полифлорного меда на 1131 изображение размером 128 × 128 пикселей при шаге скользящего окна 32 пикселя. На python разработана программа первичной классификации, позволяющая отсортировывать пустые изображения от непустых посредством задачи условий и интервалов степени заполненности черным. Проведено исследование на возможность сортировки непустых изображений на три класса по проценту заполненности черным. Вычисляется доля черных точек среди всех Perimeter total (т. е. критерий занятой зерном площади кадра), доля черных точек среди точек на границах – критерий занятого зерном периметра кадра – Perimeter black и в процентном соотношении Perimeter percentage. Осуществлялась инвертированная пороговая бинаризация для выделения объектов. Были получены следующие результаты: в промежутке от 0 до 30 программа не вывела никаких изображений, т. е. папка output_result_0-30 оказалась пустой; в промежутке от 31–60 программа классифицировала 4 изображения, т. е. в папке output_result_31-60 оказалось 4 изображения; в промежутке 61–99 программа классифицировала 130 изображений.

Ключевые слова: предобработка изображений, изображения пыльцевых зерен, нейросеть LeNet, python, библиотека opencv в python, полифлорный мед.

Введение

Ранее был проведен ряд исследований по выявлению разнообразных статистических параметров пыльцевых зерен [1]. Также были проведены эксперименты со сверточной нейронной сетью LeNet [2]. Однако изображения для нее были классифицированы и разложены по каталогам вручную.

В данной статье разрабатывается способ для автоматической классификации [3, 4] подготовленных, точнее, кадрированных, изображений зерен пыльцы с применением python. Язык программирования python используется для распознавания изображений и машинного обучения [5–9].

Обучать сверточную нейронную сеть LeNet необходимо при помощи разделенных заранее, подготовленных, прошедших предобработку [10, 11] изображений пыльцы.

Цель работы на первоначальном этапе – автоматическое разделение выборки электрооптических изображений пыльцы на две подвыборки – пустые изображения и непустые изображения, на последующем этапе – классификация непустых с помощью нейронной сети LeNET.

Задачи:

1. Кадрирование изображения пыльцевых зерен, полученного с помощью оптического микроскопа.
2. Бинаризация.
3. Первоначальная автоматическая сортировка – отделение пустых от непустых изображений-кадров.
4. Автоматизированное сохранение в разные каталоги – пустых и непустых изображений-кадров.

Методика исследования

Кадрируем изображение (рис. 1) при помощи программы ImageSplitter на изображения размером 128×128 пикселей при шаге скользящего окна в 32 пикселя (рис. 2).

В результате работы программы появилась папка, в которой находится 1131 изображение.

При бинаризации абсолютно пустые изображения выглядят абсолютно черными. Программа должна делать следующее: есть текстовый файл result_black.txt на диске C, где находятся имена файлов. Есть папка source на диске C, где много различных файлов, в том числе и те, имена которых находятся в текстовом файле. Эта программа должна брать имена файлов из текстового файла и находить файлы с этими именами в папке source, затем копировать их и сохранять в папку output [12, 13].

Имеем каталог с 1131 изображением (рис. 3).

Бинаризированные изображения в данном каталоге представлены на рис. 4.

© Камалова Ю. Б., 2019

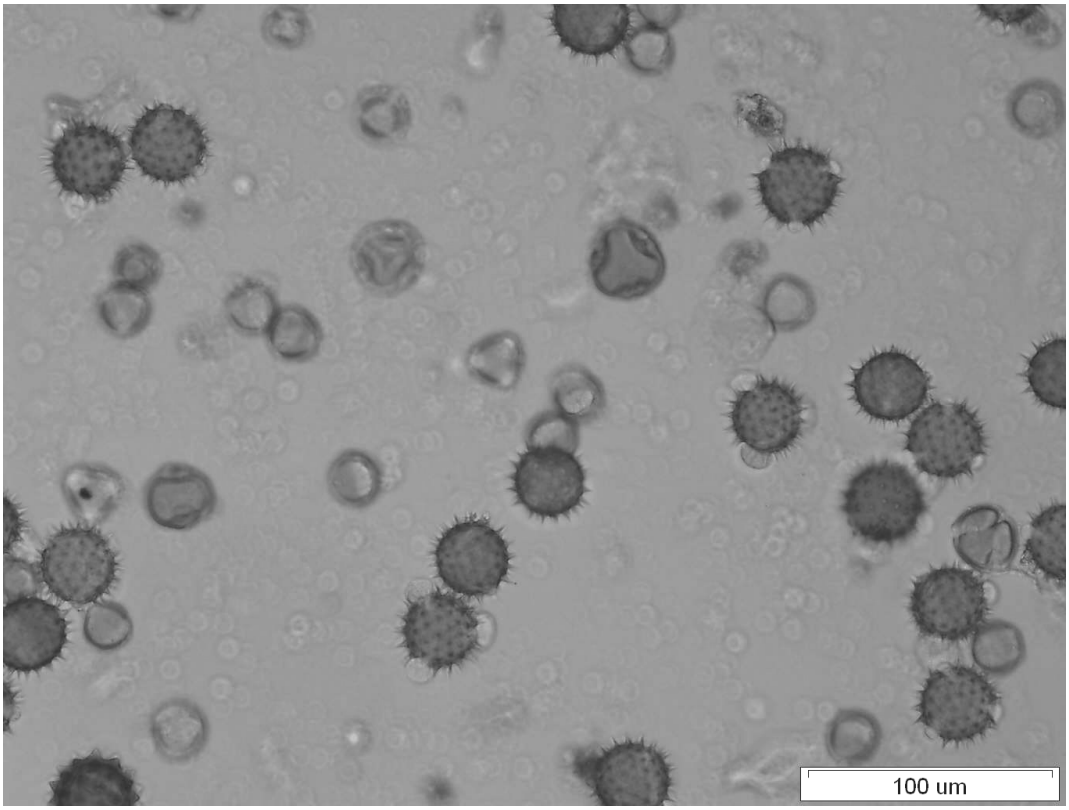


Рис. 1. Изображение полифлорного меда

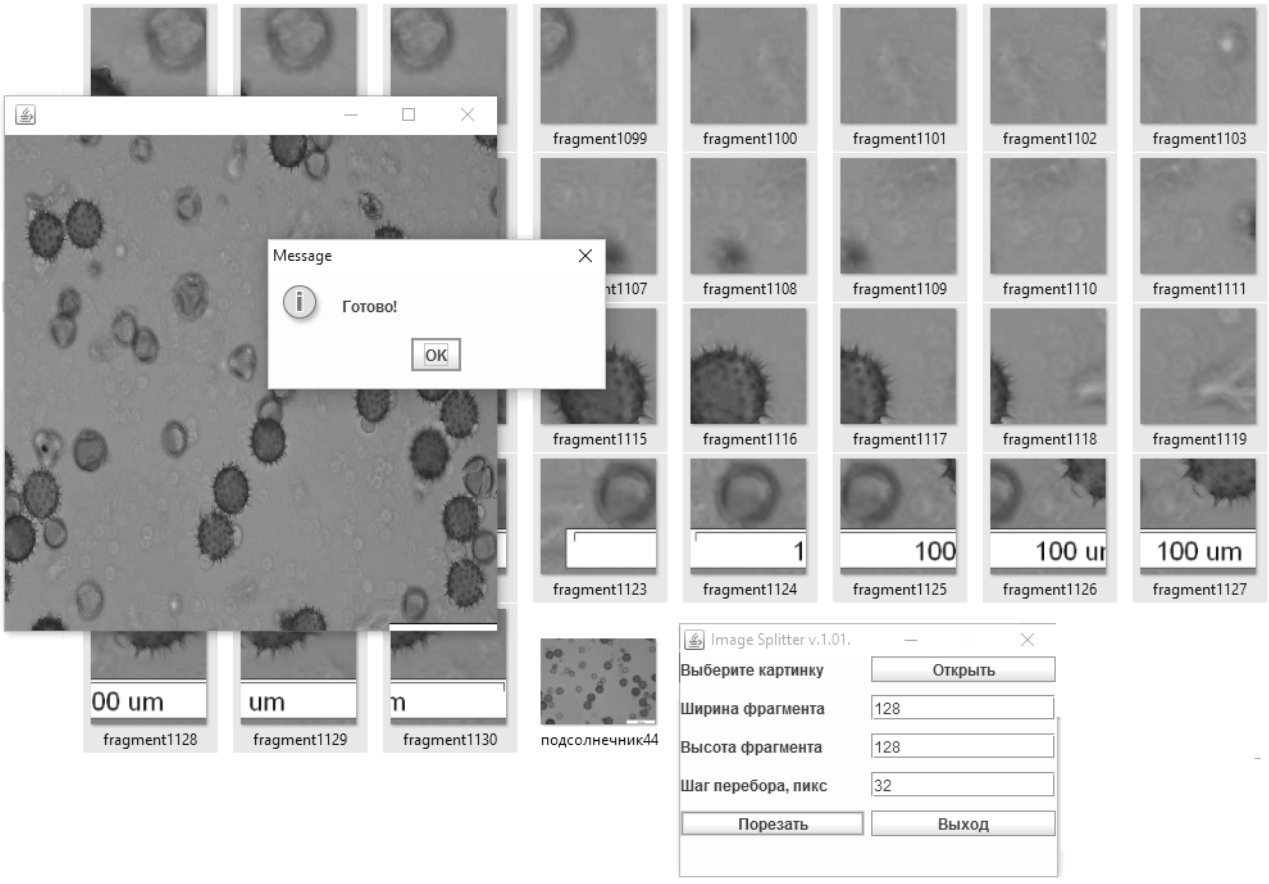


Рис. 2. Результат работы программы ImageSplitter

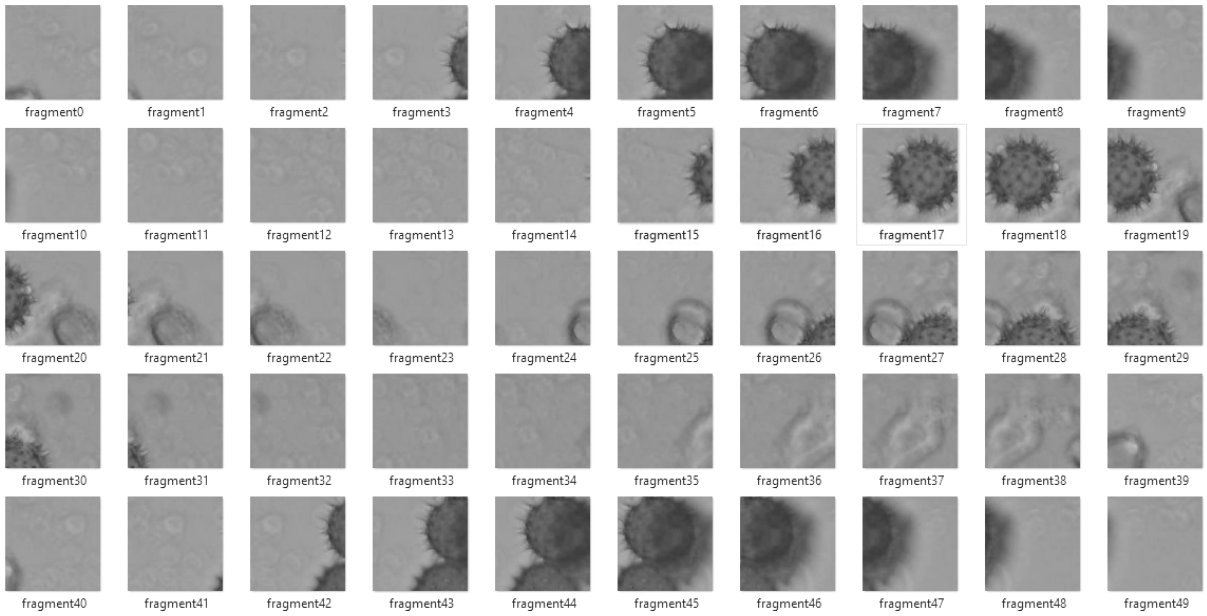


Рис. 3. Каталог с изображениями

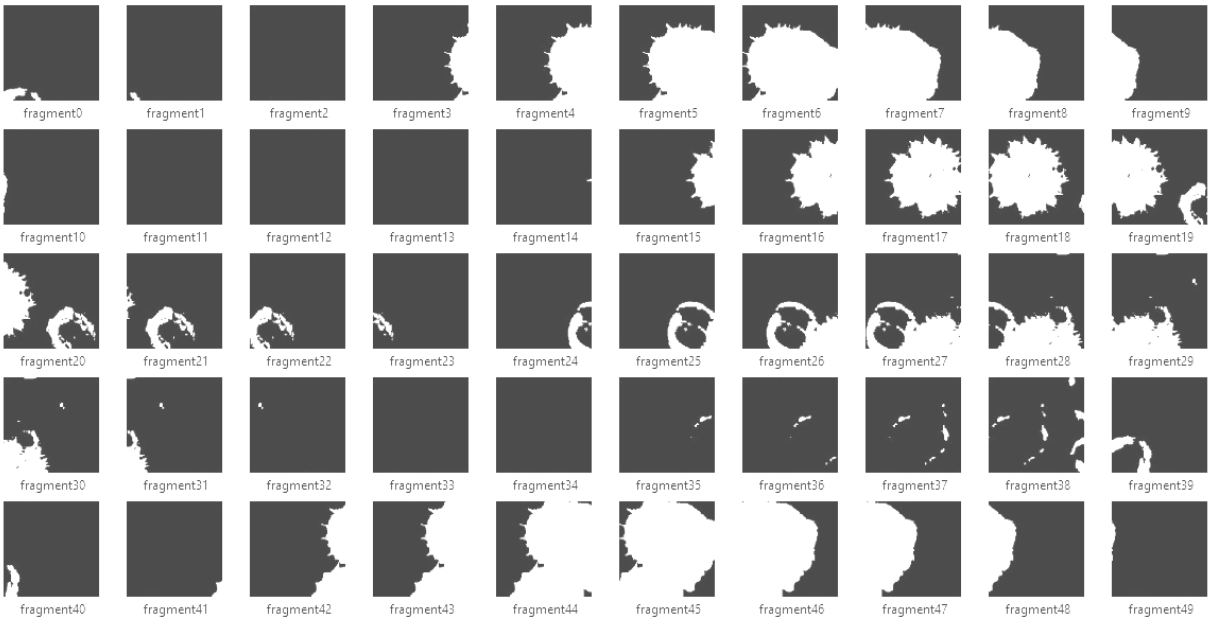


Рис. 4. Бинаризированные изображения в каталоге

Теперь надо по каждому бинаризированному кадру подсчитать:

1) долю черных точек среди всех – критерий занятой зерном площади кадра Perimeter total – количество точек периметра (две ширины, две высоты за вычетом углов);

2) долю черных точек среди точек на границах – критерий занятого зерном периметра кадра (нужны кадры с минимальной долей черных на периметре) – Perimeter black и в процентном соотношении Perimeter percentage.

Для нескольких картинок с пыльцевым зернами найдем для каждого долю черных точек на периметре по следующему алгоритму:

1. Надо вычислить для каждого изображения долю черных точек в периметре.

2. Ну и затем отсекаем часть изображений с «плохими» периметрами, т. е. те, у которых, например, доля черных точек выше 0,25.

3. С кадрами, у которых «хорошие» периметры, работаем дальше, т. е. сортируем вручную на кластеры.

В данном случае достаточно осуществить инвертированную пороговую бинаризацию для выделения объектов [14] (рис. 5):

$$B(x,y) = 0 \text{, если } A(x,y) > \text{порога, иначе 1,}$$

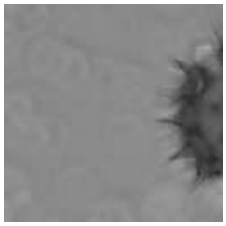
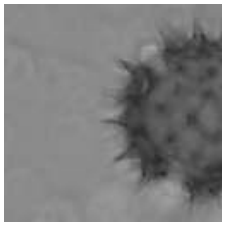
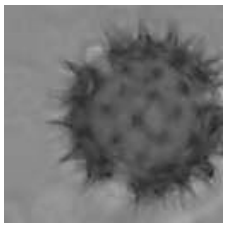
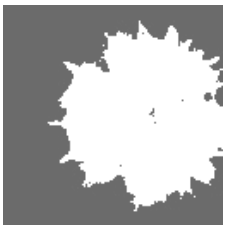
где A – исходное полутоновое изображение; B – выходное бинарное изображение.

```
Обработка изображения
  imgfile = cv2.imread(path_to_dir + '/' + img_file)
  LOWER_BOUND = 100
  img_gray = cv2.cvtColor(imgfile, cv2.COLOR_BGR2GRAY)
  ret, im_th = cv2.threshold(img_gray, LOWER_BOUND, 255, cv2.THRESH_BINARY_INV)
  total_pixels = im_th.shape[0]*im_th.shape[1]
  boundh = sum([1 if x==255 else 0 for x in im_th.flat])
  Percentage_black = (total_pixels - boundh)/total_pixels * 100
  perimeter_total = im_th.shape[1]*2 + im_th.shape[0]*2 - 4
```

Рис. 5. Обработка изображения на python и применение инверированной пороговой бинаризации

В таблице приведены результаты работы программы для трех изображений.

Результаты работы программы

Наименование и изображение	Бинаризированное изображение	Total pixels	More than bound	Less than bound	Percentage black	Perimeter total	Perimeter black	Perimeter percentage
 fragment15.jpg		16384	1835	14549	88.8	508	84	83.46
 fragment16.jpg		16384	4841	11543	70.45	508	94	81.50
 fragment17.jpg		16384	7219	9165	55.938720703125	508	31	93.9

В папке «source» находятся исходные файлы, в папке «output_result_black» – полностью пустые файлы (рис. 6).

В каталоге «output_result», таким образом, содержится все, что не попало в папку «output_result_black» (рис. 7).

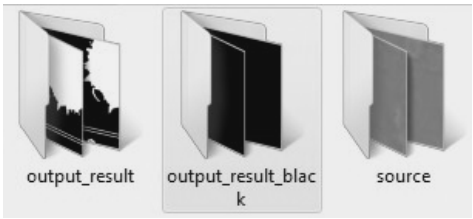


Рис. 6. Три сформированных в результате работы программы каталога

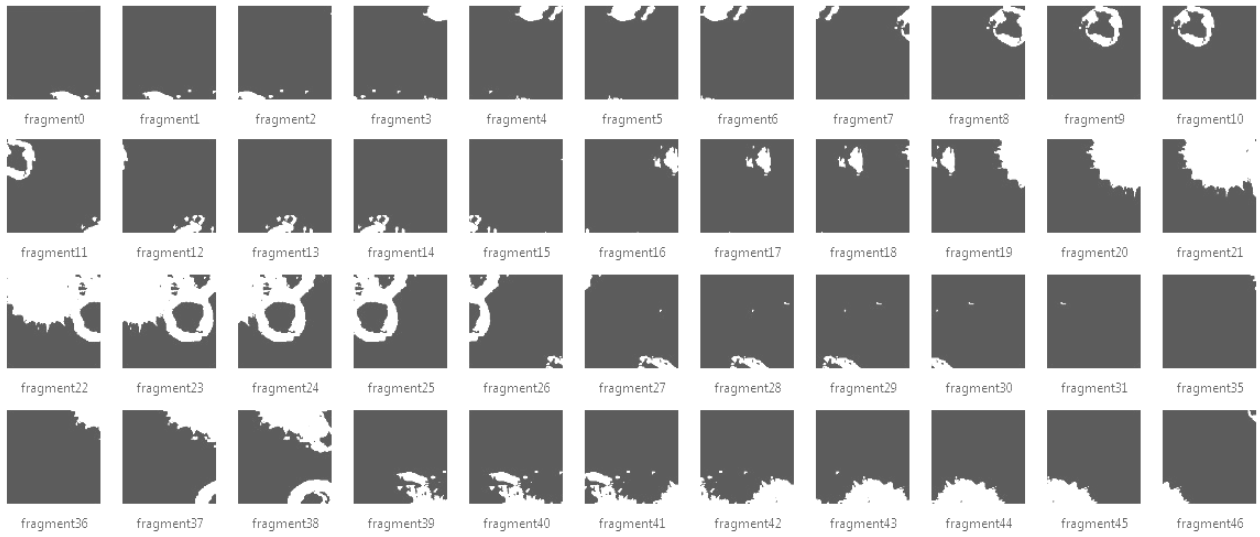


Рис. 7. Содержимое каталога output_result

Анализ результатов

Изначально было 1131 изображение в выборке, полученные с помощью программы для кадрирования.

Было задано три условия – при рассмотрении периметра менее 32 пикселей и различных интервалов степени заполненности черным (percentage black) (рис. 8).

```
if perimeter_sum < 32 and Percentage_black == 100 :
    cv2.imwrite(path_to_dir + '/output_result_black/' + img_file, im_th)
  if perimeter_sum < 32 and 99 > Percentage_black > 61 :
    cv2.imwrite(path_to_dir + '/output_result_61-99/' + img_file, im_th)
  if perimeter_sum < 32 and 60 > Percentage_black > 31 :
    cv2.imwrite(path_to_dir + '/output_result_31-60/' + img_file, im_th)
  if perimeter_sum < 32 and 30 > Percentage_black > 1 :
    cv2.imwrite(path_to_dir + '/output_result_0-30/' + img_file, im_th)
```

Рис. 8. Условия для сортировки бинаризированных изображений

Были получены следующие результаты:

1. В промежутке от 0 до 30 программа не вывела никаких изображений, т. е. папка output_result_0-30 оказалась пустой.

2. В промежутке от 31–60 программа классифицировала 4 изображения, т. е. в папке output_result_31-60 оказалось 4 изображения (рис. 9).

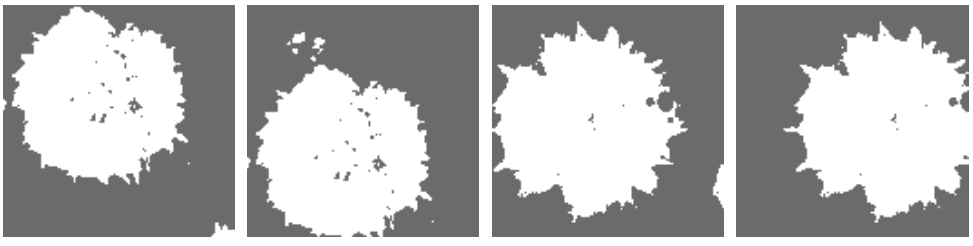


Рис. 9. Классифицированные изображения в промежутке 31–60

3. В промежутке 61–99 программа классифицировала 130 изображений.

Наиболее характерные целые приведены на рис. 10.

А также их части, которые оказались также в этом промежутке, приведены на рис. 11.

И более мелкие изображения приведены на рис. 12.

Рис. 10. Пыльцевые зерна в промежутке 61–99

Рис. 11. Части пыльцевых зерен в промежутке 61–99

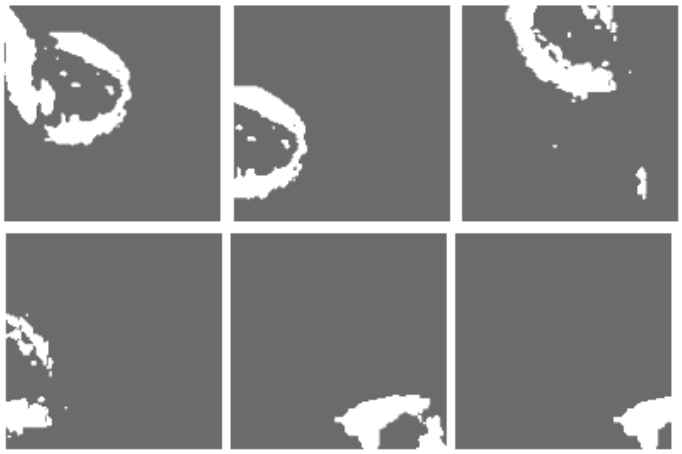


Рис. 12. Более мелкие изображения пыльцевых зерен в промежутке 61–99

Среди полностью пустых или полностью черных в бинаризированном варианте программа классифицировала 255 объектов: в папке output_result_black оказалось 255 абсолютно черных изображений.

Выводы

Таким образом, существуют различные варианты классификации кадрированных бинаризированных изображений пыльцевых зерен в python – по степени заполненности черным с помощью условий, а также при помощи использования библиотеки PIL можно проверить изображение на предмет наличия пикселей, отличных от черных. В данной работе применялся первый вариант.

Однако все данные методы должны быть предварены установкой библиотеки opencv [15] в python, а также библиотеки cv2, os и sys [16] и установкой pip [17]. На данном этапе возникали определенные трудности, связанные в основном с версиями python, а также с местом и способом установки данных библиотек (в данном случае использовался pip install opencv-contrib-python).

Задача копирования определенных файлов из нескольких каталогов с помощью python была решена при помощи функции Os.listdir [18], которая выводит список файлов директории. Данная задача в приведенном исследовании решалась при помощи возвращения результата True False, то есть картинка подходит под условие или нет, а результат image_handler сохраняли в переменную, где затем проверяли, если True, то сохраняли исходный файл в необходимом каталоге.

Было выявлено, что при классификации изображений при помощи условий и задания интервалов классифицируются объекты незамкнутой формы, т. е. части пыльцевых зерен, а не полные зерна, либо на фотографиях с полными зернами присутствуют их части. Однако программа справляется с отделением пустых изображений от непустых. Дальнейшее развитие работы – в классификации непустых изображений.

Автор благодарит за предоставленные фотографии пыльцевых зерен коллектив и д.б.н., проф. кафедры ботаники и генетики растений, заведующую лабораторией цитогенетики и генетических ресурсов растений Пермского государственного университета Л. В. Новоселову.

Библиографические ссылки

1. *Камалова Ю. Б.* Вычисление статистических параметров в распознавании изображений зерен пыльцы, полученных с помощью растрового электронного микроскопа // Интеллектуальные системы в производстве. 2014. № 1 (23). С. 120–123.

2. *Коробейников А. В., Камалова Ю. Б.* Применение сверточной нейронной сети LeNet для классификации изображений зерен пыльцы // Приборостроение в XXI веке – 2017. Интеграция науки, образования и производства [Электронный ресурс]: сб. материалов XIII Междунар. науч.-техн. конф. (Ижевск, 22–24 нояб. 2017 г.). Ижевск : Изд-во ИжГТУ имени М. Т. Калашникова, 2018. С. 228–233. 25,9 Мб (PDF). Систем. требования: Acrobat Reader 6.0 и выше.

3. Аппроксимация цветных изображений на основе кластеризации цветовой палитры и сглаживания границ сплайнами и дугами / А. В. Кучуганов, В. Н. Кучуганов, П. П. Осколков, Д. Р. Касимов // Программирование. 2018. № 5. С. 3–11.

4. *Кучуганов М. В., Кучуганов А. В.* Дескрипционная логика на графах изображений // Вестник удмуртского университета. Математика. Механика. Компьютерные науки. 2018. Т. 28, № 4. С. 582–594.

5. Полное руководство параллельного программирования на Python. Куан Нгуен. Глава 8. Параллельная обработка изображений Python в качестве инструмента обработки изображения [Onreader]. URL: http://onreader.mdl.ru/MasteringConcurrencyInPython/content/Ch08.html#0201 (дата обращения: 11.03.2019).

6. Простой классификатор изображений на Python с помощью библиотеки TensorFlow: пошаговое руководство [TPRoger]. URL: https://tproger.ru/translations/image-classifier-tensorflow/ (дата обращения: 12.03.2019).

7. Python: распознавание объектов в реальном времени [ProgLib]. URL: https://proglib.io/p/real-time-object-detection/ (дата обращения: 24.03.2019)

8. Полное руководство параллельного программирования на Python. Куан Нгуен, Packt Publishing, ноябрь 2018. URL: http://onreader.mdl.ru/MasteringConcurrencyInPython/content/index.html (дата обращения: 25.03.2019).

9. *Рябов П. И., Вологдин С. В., Максимова В. В.* Алгоритм распознавания изображений с приборов учета электроэнергии // Интеллектуальные системы в производстве. 2017. Т. 15, № 4. С. 42–48.

10. *Рябов П. И., Вологдин С. В., Максимова В. В.* Алгоритм распознавания изображений с приборов учета электроэнергии (Image Recognition Algorithm from the Electricity Metering Devices) // Интеллектуальные системы в производстве. 2017. Т. 15, № 4. С. 42–48.

11. Алгоритм позиционирования БПЛА по последовательности кадров видеопотока / И. О. Архипов, М. А. Мишенков, А. А. Шутов, М. О. Еланцев // Интеллектуальные системы в производстве. 2017. Т. 16, № 3. С. 66–69.

12. Файлы. Работа с файлами [Python 3 для начинающих]. URL: https://pythonworld.ru/tipy-dannyx-v-python/fajly-rabota-s-fajlami.html (дата обращения: 25.04.2019).

13. Модуль os [Python 3 для начинающих]. URL: https://pythonworld.ru/moduli/modul-os.html (дата обращения: 25.04.2019).

14. Интуит. Национальный открытый университет. Академия Intel: Введение в разработку мультимедийных приложений с использованием библиотек OpenCV и IPP. Лекция 1: Основные цветовые модели, представление изображения, базовые операции над изображениями. URL: https://www.intuit.ru/studies/courses/10621/1105/lecture/17979 (дата обращения: 25.04.2019).

15. Библиотека OpenCV [OpenCV]. URL: https://opencv.org/ (дата обращения: 25.04.2019).

16. Модуль cv2 при использовании OpenCV [QaruSite]. URL: http://qaru.site/questions/52086/opencv-cannot-find-module-cv2 (дата обращения: 25.04.2019).

17. Описание и установка pip [Pypi.org]. URL: https://pypi.org/project/opencv-contrib-python/ (дата обращения: 25.04.2019).

18. Модуль os [Python 3 для начинающих]. URL: https://pythonworld.ru/moduli/modul-os.html (дата обращения: 25.04.2019).

References

1. Kamalova Yu.B. [Calculation of statistical parameters in the recognition of images of pollen grains obtained using a scanning electron microscope]. *Intellektual'nye sistemy v proizvodstve.* 2014, no. 1, pp. 120-123 (in Russ.).

2. Korobeinikov A.V., Kamalova Yu.B. *Primenenie svertochnoi neironnoi seti LeNet dlya klassifikatsii izobrazhenii zeren pyl'tsy* [Application of the LeNet convolutional neural network to classify images of pollen grains]. *Priborostroenie v XXI veke – 2017. Integratsiya nauki, obrazovaniya i proizvodstva.* [Proc. Instrument engineering in the XXI century - 2017. Integration of science, education and production [Electronic resource]: Sat. XIII International scientific technology conf. (Izhevsk, Nov. 22-24, 2017)]. Izhevsk, Izd-vo IzhGTU imeni M. T. Kalashnikova, 2018. Pp. 228-233. 25.9 Mb (PDF) (in Russ.).

3. Kuchuganov A.V., Kuchuganov V.N., Oskolkov P.P., Kasimov D.R. [Approximation of color images based on clustering of the color palette and smoothing of borders with splines and arcs]. *Programmirovanie*, 2018, no. 5, pp. 3-11.

4. Kuchuganov M.V., Kuchuganov A.V. [Descriptive logic on image graphs]. *Vestnik udmurtskogo universiteta. Matematika. Mekhanika. Komp'yuternye nauki.* 2018. Vol. 28. No. 4. Pp. 582-594 (in Russ.).

5. *Polnoe rukovodstvo parallel'nogo programmirovaniya na Python. Kuan Nguen. Glava 8. Parallel'naya obrabotka izobrazhenii Python v kachestve instrumenta obrabotki izobrazheniya [Onreader]*. [A comprehensive guide to concurrent programming in Python. Kuan Nguyen. Chapter 8. Parallel processing of Python images as an image processing tool [Onreader]]. Available at: http://onreader.mdl.ru/MasteringConcurrencyInPython/content/Ch08.html#0201 (accessed March 11, 2019) (in Russ.).

6. *Prostoi klassifikator izobrazhenii na Python s pomoshch'yu biblioteki TensorFlow: poshagovoe rukovodstvo [TPRoger]*. [A simple image classifier in Python using the TensorFlow library: a step-by-step guide [TPRoger]]. Available at: https://tproger.ru/translations/ image-classifier-tensorflow (accessed March 12, 2019) (in Russ.).

7. *Python: raspoznavanie ob"ektov v real'nom vremeni [ProgLib]*. [Python: real-time object recognition [ProgLib]]. Available at: https://proglib.io/p/real-time-object-detection/ (accessed: 24.03.2019) (in Russ.).

8. *Polnoe rukovodstvo parallel'nogo programmirovaniya na Python. Kuan Nguen, Packt Publishing, noyabr' 2018.* [A comprehensive guide to concurrent programming in Python. Kuan Nguyen, Packt Publishing, November 2018]. Available at: http://onreader.mdl.ru/MasteringConcurrencyInPython/content/index.html (accessed: 03/25/2019) (in Russ.).

9. Ryabov P.I., Vologdin S.V., Maksimova V.V. [Algorithm for the recognition of images from electricity meters] *Intellektual'nye sistemy v proizvodstve*. 2017. Vol. 15, no. 4, pp. 42-48 (in Russ.).

10. Ryabov P.I., Vologdin S.V., Maksimova V.V.. [Image Recognition Algorithm from the Electricity Metering Devices]. *Intelligent systems in production*. 2017. Vol. 15, no. 4, pp. 42-48 (in Russ.).

11. Arkhipov I.O., Mishenkov M.A., Shutov A.A., Elantsev M.O. [UAV positioning algorithm according to the sequence of frames of the video stream]. *Intelligent systems in production.* 2017. Vol. 16, no. 3, pp. 66-69 (in Russ.).

12. *Faily. Rabota s failami [Python 3 dlya nachinayushchikh]*. [Files. Working with files [Python 3 for beginners]]. Available at: https://pythonworld.ru/tipy-dannyx-v-python/fajly-rabota-s-fajlami.html (accessed: 04/25/2019) (in Russ.).

13. *Modul' os [Python 3 dlya nachinayushchikh]*. [The os module [Python 3 for beginners]]. Available at: https://pythonworld.ru/moduli/modul-os.html (accessed: 04/25/2019) (in Russ.).

14. *Intuit. Natsional'nyi otkrytyi universitet. Akademiya Intel: Vvedenie v razrabotku mul'timediinykh prilozhenii s ispol'zovaniem bibliotek OpenCV i IPP. Lektsiya 1: Osnovnye tsvetovye modeli, predstavlenie izobrazheniya, bazovye operatsii nad izobrazheniyami*. [Intuition. National Open University. Intel Academy: An Introduction to Developing Multimedia Applications Using the OpenCV and IPP Libraries. Lecture 1: Basic color models, image representation, basic operations on images]. Available at: https://www.intuit.ru/studies/courses/10621/1105/lecture/17979 (accessed: 04.25.2019) (in Russ.).

15. *Biblioteka OpenCV [OpenCV]*. [OpenCV library [OpenCV]]. Available at: https://opencv.org/ (accessed: 04/25/2019) (in Russ.).

16. *Modul' cv2 pri ispol'zovanii OpenCV [QaruSite]*. [The cv2 module when using OpenCV [QaruSite]]. Available at: http://qaru.site/questions/52086/opencv-cannot-find-module-cv2 (accessed: 04/25/2019) (in Russ.).

17. *Opisanie i ustanovka pip [Pypi.org]*. [Description and installation of pip [Pypi.org]]. Available at: https://pypi.org/project/opencv-contrib-python/ (accessed: 04/25/2019) (in Russ.).

18. *Modul' os [Python 3 dlya nachinayushchikh]*. [The os module [Python 3 for beginners]]. Available at: https://pythonworld.ru/moduli/modul-os.html (accessed: 04/25/2019) (in Russ.).

* * *

Program for automatic clustering of numerous (over 1000) images of pollen grains obtained using an optical microscope in Python, for learning the LeNet convolutional neural network

Yu. B. Kamalova, Assistant, Kalashnikov ISTU, Izhevsk, Russia

*A method of images classification of pollen grains at the preprocessing sampling stage for learning of the LeNet convolutional neural network is considered. The Python program was developed for cropping an image of pollen obtained using an optical microscope, ImageSplitter by which an image of polyfluoric honey was framed at 1131 images 128 pixels * 128 pixels in size with a sliding window step of 32 pixels. The primary Python classification program was developed which allows sorting out empty images from non-empty ones through the task conditions and intervals of the degree of fullness in black. A study on the possibility of sorting non-empty images into three classes according to the percentage of fullness in black was conducted. The proportion of black points among all Perimeter total (i.e., the criterion of the area occupied by a grain of a frame) is calculated; the criterion of the frame's perimeter grain is Perimeter black and in percentage the Perimeter percentage. The inverted threshold binarization was performed to select objects. The following results were obtained: in the interval from 0 to 30, the program did not display any images, i.e., the folder output_result_0-30 was empty; in the interval from 31-60, the program has classified 4 images, i.e., there were 4 images in the folder output_result_31-60; between 61-99, the program has classified 130 images.*

Keywords: image preprocessing, pollen grain images, LeNet neural network, Python, opencv library in Python, polyflory honey.

Получено: 21.05.19