ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПОИСКА ОДИНАКОВЫХ ФОТОГРАФИЙ С ПОМОЩЬЮ АНАЛИЗА РАСПРЕДЕЛЕНИЯ ЯРКОСТИ И ПОСТРОЕНИЯ БИНАРНЫХ ИЗОБРАЖЕНИЙ

Б.Б. Чимитов
Научный руководитель: Н.А. Маркова, ст. преподаватель
ФГАОУ ВО «Национальный исследовательский Томский политехнический университет»
E-mail: bbc7@tpu.ru

Введение

Несомненно, на сегодняшний день фотографии стали одними из самых распространенных и удобных способов сохранения информации. То, насколько хороша камера у телефона, рассматривается как один из показателей качества современных смартфонов. Однако, очевидно, пользователь не может хранить фотографии долго на своем устройстве, рано или поздно могут возникнуть проблемы, связанные с недостатком памяти у телефона. Поэтому, вопрос о рациональном и эффективном использовании гаджетов на данный момент является актуальным.

Одним из решений данной проблемы является удаление ненужных, однотипных фотографий. Однако, достаточно часто, количество фотографий в смартфоне может превышать тысячи, и простое удаление фотографий становится очень затратной работой.

Поэтому, была поставлена цель работы – написать программное обеспечение, которое позволило бы пользователям находить и удалять одинаковые фотографии оптимально быстро и в том числе, для того, чтобы освободить место в памяти их электронных устройств.

Описание алгоритма

Для решения задачи поиска одинаковых фотографий в электронном устройстве было разработано приложение на базе языка программирования Python.

Для удобства, работа с изображениями основывалась на представлении самих изображений в виде массивов numpy сразу после их чтения в программе.

Каждый прочтенный файл-изображение был представлен в виде трехмерного массива, первые два измерения которого составляли ширину и высоту фотографий (в пикселях), а последний – каналы цветов (RGB). Далее, было решено сделать изображение полутоновым так, чтобы избавиться от работы с трехмерными объектами.

Перед переходом к третьему шагу программа разделяет все фотографии на три группы: светлые, средние и темные. Это сделано для того, чтобы сортировать данные по их среднему значению элементов массивов, и чтобы избежать сравнения очевидно противоположных фотографий: светлых и темных.

Также, было решено применить квадратичную функцию к каждому изображению для того, чтобы придать деталям фотографий более четкие формы.

На данном этапе программа работает с двумерными массивами, элементы которых находятся в диапазоне от 0 до 255 (оттенки серого). Для сравнения изображений такое разнообразие значений не очень удобно, поэтому следующий шаг – сделать изображения бинарными, диапазон которых составляют числа 0 и 1.

Пример получения бинарного изображения программой представлен на рисунке 1.

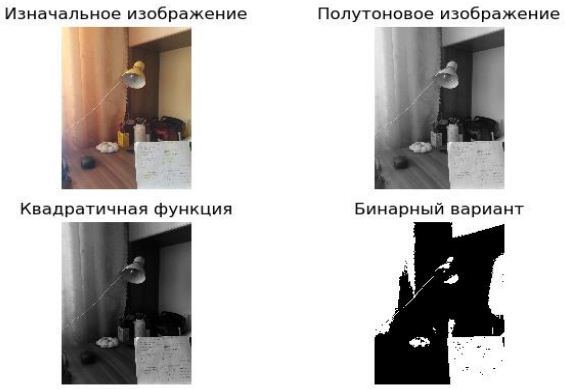

Рис. 1. Получение бинарного изображения

Непосредственно перед получением бинарного эквивалента изображений было принято решение размыть фотографии. Под влиянием шума две абсолютно одинаковые фотографии при дальнейшем сравнении могут быть приняты за разные. Также, есть вероятность, что пользователь поневоле перед фотографированием изменил угол поворота камеры. В итоге, получаются две фотографии, одна из которых немного наклонена. Во избежание этого, в программе использованы фильтры filters из модуля scipy.ndimage.

Пример использования фильтра Гаусса представлен на рисунке 2.

Рис. 2. Использование фильтра Гаусса.

Далее идет алгоритм сравнения массивов. Два массива-изображения сравниваются поэлементно. Инициализируются счетчик изначально равный нулю и величина шага. Разрешение фотографий может достигать высоких значений, поэтому время сравнения всех элементов возрастает с их количеством. Для того, чтобы сократить это время, значение шага можно регулировать, но с другой стороны, теряется качество сравнения. Так можно проводить сравнение не по всем элементам, а по чередующимся через некоторое число *n*.

Каждый раз, когда два элемента различны, счетчик увеличивается на величину квадрата шага. Как только значение достигает 25% от всего числа точек, программа прерывает сравнение и делает вывод, что фотографии различны.

Тестирование алгоритма поиска

Для оценки алгоритма поиска одинаковых изображений было проведено тестирование его работы на различных входных данных.

В процессе тестирования было выбрано несколько фотографий, сделанные со смартфона и характеризующиеся различными яркостями и содержанием. Для оценки быстроты работы программы был произведен замер времени.

Усредненное время работы алгоритма при 18 исходных фотографиях одинакового размера представлено в таблице 1.

Таблица 1. Показатели работы алгоритма при различном n чередовании элементов.

Чередование *N* кадров при поиске	Время выполнения алгоритма
n=1	57,32 с.
n=2	15,41 с.
n=4	4,75 с.
n=6	2,99 с.
n=8	2,28 с.
n=10	2,06 с.
n=20	1,6 с.
n=30	1,55 с.
n=40	Ответ отличается от предыдущих попыток

Наилучшие результаты при тестировании были получены при чередовании через каждые n=30 элементов. При n=40 вывод программы сильно отличался от тех тестов, когда значения шага были меньше.

Также, был проведен тест зависимости времени работы алгоритма от количества исходных фотографий.

Усредненное время работы алгоритма при различных количествах исходных фотографий представлено в таблице 2.

Таблица 2. Показатели работы алгоритма при различном количестве исходных фотографий (Величина шага равна 6).

Количество фотографий	Время выполнения алгоритма
n=2	0,16 с.
n=4	0,39 с.
n=6	0,49 с.
n=8	0,62 с.
n=10	0,79 с.
n=18	1,59 с.

Заключение

В результате проведения тестирования можно сделать вывод о том, что алгоритм поиска подобных фотографий является работоспособным. Определить точность алгоритма весьма трудно, так как с одной точки зрения фотографии могут быть одинаковы, с другой - разными.

Однако, можно и отметить минусы алгоритма:

1. Основной акцент идеи алгоритма основывается на отличиях в яркости двух фотографий. Если подобрать к одному изображению такой же эквивалент, сохраняя при этом игру света и тени, то программа покажет, что данные образцы подобны.
2. Программа работает только с теми изображениями, которые имеют одинаковые размеры (высоту и ширину), при изменении размера выдается ошибка. Если рассматривать только одно устройство, данная проблема может не проявить себя, так как в большинстве случаев пользователь сохраняет изображения в одном и том же разрешении.
3. Так как программа использует чередование элементов, некоторые детали могут быть пропущены. Данный факт можно не учитывать при малых значениях шага, однако, чтобы справиться с большими количествами исходных данных, столь малые значения n будут способствовать огромным затратам времени.

Список использованных источников

1. Солем Я. Э. Программирование компьютерного зрения: пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2016. – 312 с.
2. Mark Lutz. *Learnimg Python*. O'Reilly Media Inc., 2009.
3. Travis Oliphant. *Guide to Numpy*. [Электронный ресурс] – Режим доступа: https://web.mit.edu/dvp/Public/numpybook.pdf, 2006.