DOI 10.37882/2223-2966.2023.11.20

РАЗРАБОТКА И ПРИМЕНЕНИЕ В ГРАФИЧЕСКОМ РЕДАКТОРЕ АЛГОРИТМА ДЛЯ РЕДАКТИРОВАНИЯ ЭЛЕМЕНТОВ ЛИЦА НА ФОТОГРАФИИ

DEVELOPMENT AND APPLICATION OF ALGORITHMS FOR MULTILAYER IMAGE PROCESSING IN A GRAPHICAL EDITOR

V. Monastyrev
S. Molodyakov

Summary. This article discusses a new algorithm for editing face elements in a photograph. The paper considers existing solutions, identifies their advantages and disadvantages. An algorithm for editing facial features is described, and an example of its implementation is given.

Keywords: image processing, face editing algorithm, image editor, machine learning.

Монастырев Виталий Викторович
Аспирант, Санкт-Петербургский политехнический университет Петра Великого
vit34-95@mail.ru
Молодяков Сергей Александрович
Д.т.н., профессор, Санкт-Петербургский политехнический университет Петра Великого
molodyakov_sa@spbstu.ru

Аннотация. В данной статье рассматривается новый алгоритм для редактирования элементов лица на фотографии. В работе рассматриваются существующие решения, выявляются их преимущества и недостатки. Описан алгоритм для редактирования черт лица, а также приводится пример его реализации.

Ключевые слова: обработка изображений, алгоритм редактирования лица, редактор изображений, машинное обучение.

Введение

Область обработки изображений развивается с каждым годом. Это во многом связано с развитием электронных устройств и увеличением их вычислительной мощности, что дает возможность применять достаточно сложные алгоритмы обработки изображений. Одним из наиболее интересных направлений являются нейронные сети. Они дают возможность создания таких алгоритмов, которые раньше бы были трудно реализуемыми или вовсе невозможными. Нейронные сети сейчас доступны как на компьютерах (PyTorch [1], TensorFlow [2], Keras [3]), так и на мобильных платформах (CoreML [4]).

Отдельный интерес может представлять возможность редактирования лица. Нейронные сети на сегодняшний день могут достаточно точно определять на изображении отдельные элементы лица (например, глаза, нос, рот и т.д.). Возможность автоматического определения элементов лица, позволяет применять по отношению к этим элементам различные механизмы обработки изображений.

Анализ существующих решений

На сегодняшний день представлено множество различных фоторедакторов, которые позволяют вносит правки на исходное изображение. Один из наиболее популярных примеров — Photoshop [5, 6], который позволяет вносить новые слои при редактировании изображений, создавать коллажи и прочее.

Другой пример многослойной обработки изображений, можно найти в статье «Разработка и применение в графическом редакторе алгоритмов многослойной обработки изображений» [7]. В данной статье рассматривается пример алгоритма, который позволяет реализовывать многослойную обработку изображений, совмещая вычислительные ресурсы на локальном устройстве и серверной части.

Стоит отметить, что в перечисленных выше примерах для редактирования элементов лица, необходимо вручную отмечать точки на фотографии и вносить новые слои на отмеченные участки. В данной статье будет предложен алгоритм, который позволит автоматизировать данный процесс.

В статье «Interlinked Convolutional Neural Networks for Face Parsing» [8] приводится алгоритм для реализации нейронной сети, которая способна определять элементы лица на изображении. Авторы предлагают использовать сверточную нейронную сеть (iCNN), которая в свою очередь состоит из нескольких нейронных сетей (CNN) [9, 10, 11]. Для возможности обмена CNN информацией, авторами был разработан специальный уровень, позволяющий эффективно интегрировать локальную и контекстную информацию. Отличительной чертой iCNN является широкое использование понижения и повышения частоты дискредитации во взаимосвязанных слоях, в то время как традиционные CNN обычно используют только понижение частоты дискретизации. Для разбора лиц авторами предлагается двухэтапный конвейер.

На первом этапе локализуются части лица на изображении с уменьшенным размером, а на втором этапе маркируются пиксели в идентифицированных частях лица на исходном изображении.

Сверточные слои такие же, как и в традиционной CNN, где используются локальные соединения и распределение веса. Для веса $w_{uvkq}^{(l)}$ выход единицы в (i, j) в l-м слое равен:

$$y_{ijq}^{(l)} = f\left(\sum_{k=1}^{C}\sum_{u=1}^{P_1}\sum_{v=1}^{P_2} w_{uvkq}^{(l)} y_{i+u,j+v,k}^{(l-1)} + b^{(l)}\right)$$

где P_1 и P_2 обозначают размер весового ядра на карте признаков, C обозначает количество каналов в $(l-1)$-уровне, $b^{(l)}$ обозначает предвзятость в l-уровне, $f(.)$— функция активации.

В данной статье будет использована предложенная авторами нейронная сеть в качестве первого модуля, который будет определять элементы лица на изображении. В качестве нововведения в данной статье предлагается расширение данной нейронной сети последующими модулями, которые позволят редактировать изображения.

Алгоритм редактирования элементов лица

Для возможности реализации подобного алгоритма, необходимо реализовать два модуля. Первый модуль — это нейронная сеть, которая определяет на входном изображении элементы лица и составляет маску элемента на ее основе. Маска элемента содержит в себе список пикселей, которые принадлежат определенному элементу лица. Например, список пикселей, которые находятся в области глаз. Эти данные необходимо сохранять в кэше устройства, либо на жестком диске, чтобы иметь возможность к ним обращаться из второго модуля.

Второй модуль — это алгоритм, который будет на основе полученных масок накладывать эффекты на искомое изображение. Накладываемые эффекты могут быть различными и зависят от конечной цели. В качестве примера, можно привести эффект «рыбьего глаза», который позволяет регулировать размер области изображения: увеличивать или уменьшать.

Теперь рассмотрим проблемы, которые возникнут при таком подходе. Как правило, нейронные сети подобного формата тренируют на портретных изображениях. Т.е. они не смогут определить корректно элементы лица, если человек изображен на фотографии во весь рост. Для решения этой проблемы, необходимо определять на фото непосредственно лицо и сохранять его в виде отдельного портретного изображения. Данное портретное изображение уже можно будет передавать в следующий модуль. Для определения лица на фотографии не обязательно использовать нейронные сети, можно применять и более простые алгоритмы, чтобы сэкономить время и ресурсы.

Следующая проблема — разрешение. Чем разрешение выше, тем дольше будет работать нейронная сеть и больше будет кэш с набором пикселей. Для решения этой проблемы, можно уменьшать разрешение исходного изображения и работать с ним. При этом, во 2 модуле необходимо учитывать смещение пикселей, вызванной уменьшением разрешения и применять масштабирование.

Наконец, последняя проблема, это групповые фото, когда на одном изображении находится одновременно несколько людей. Для решения данной проблемы, необходимо создавать список из нескольких лиц, которые будут поочередно подаваться в первый и второй модуль обработки. Таким образом, результирующий алгоритм, можно схематично изобразить следующим образом (рис. 1).

Реализация

В качестве платформы для реализации предложенного алгоритма, была выбрана платформа Apple. Реализация будет иметь возможность запуска на iOS, iPadOS и MacOS. В качестве языка был выбран Swift.

Изменения разрешения изображения было реализовано при помощи функции UIGraphicsBeginImageContextWithOptions, которая позволяет настраивать среду

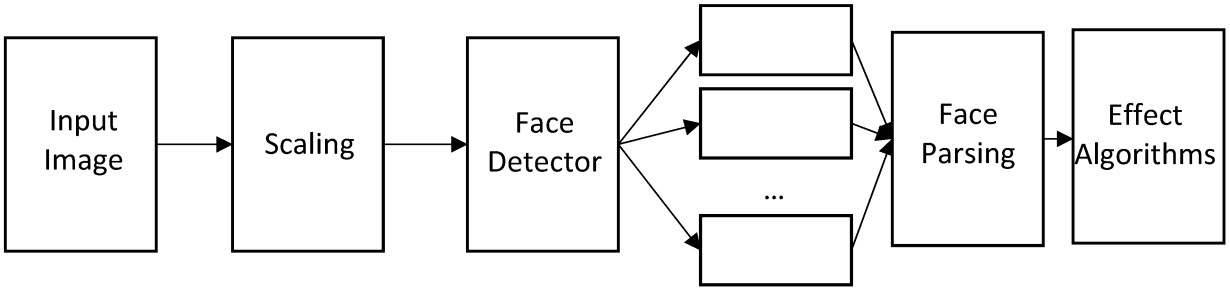

Рис. 1. Алгоритм редактора

рисования для рендеринга в растровое изображение. Формат растрового изображения — это формат 32-разрядных целых пикселей ARGB с использованием порядка байтов хоста. Таким образом, можно перерисовать исходное изображение в новое с использованием меньшего количества пикселей.

Для возможности запуска нейронной сети было использовано решение на PyTorch, которое было оптимизировано под запуск на платформах Apple при помощи фреймворка CoreML. Таким образом был реализован модуль «Face detector».

Для поиска лиц был использован класс CIDetector [12]. Объект CIDetector использует обработку изображения для поиска и идентификации заметных элементов (лиц, прямоугольников и штрих-кодов) в неподвижном изображении или видео. При помощи данных из CIDetector генерируются новые изображения, которые содержат только лицо и передаются в модуль определения элементов лица.

На основе данных об элементах лица, с учетом измененного разрешения, производится сохранение координат элементов лица в оперативную память устройства.

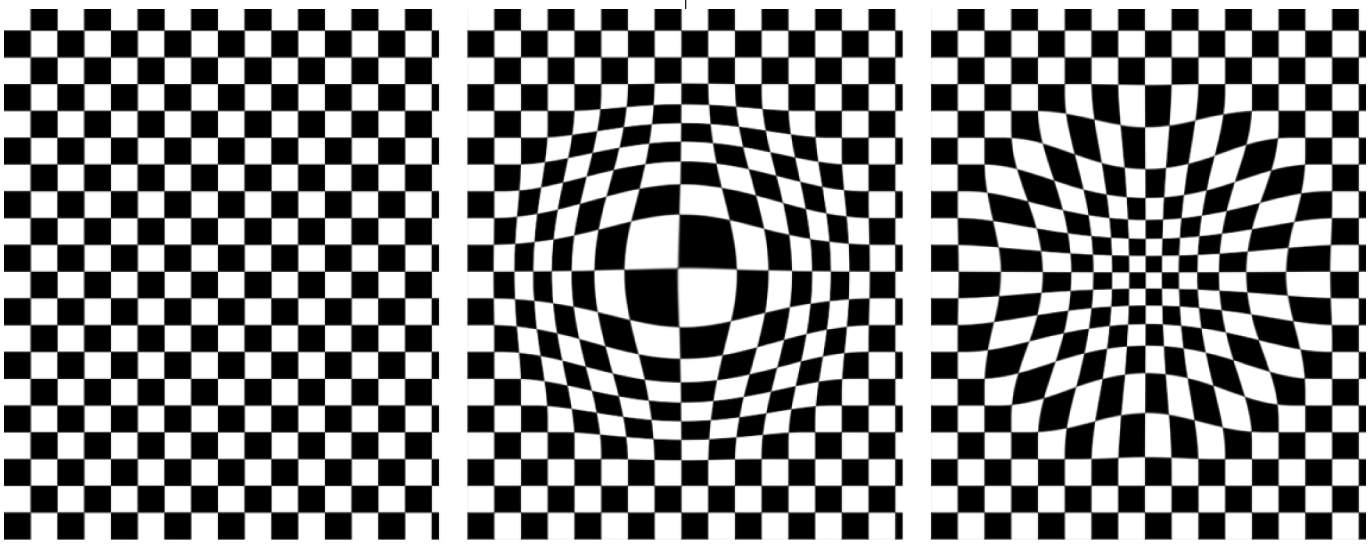

Рис. 2. Пример работы фильтра, который вносит искривления на изображения

Рис. 3. Пример фото до и после редактирования, при котором были увеличены размеры глаз, рта и носа

Рис. 4. Алгоритм зависимости времени обработки от количества лиц на изображении

Согласно координатам пикселей производится накладывание эффекта на исходное изображение.

В качестве накладываемого эффекта был выбран эффект CIBumpDistortion. Данный фильтр позволяет задавать точку накладывания фильтра и его радиус. Пример работы фильтра представлен на рисунке (рис. 2).

Таким образом были реализованы модули предложенного алгоритма. Пример работы представлен на рисунке (увеличены размеры глаз, носа и рта, рис. 3).

Чтобы оценить время обработки от количества лиц на фото, были отобраны фотографии, где изображено от 1 до 10 людей. Был построен график зависимости времени обработки (с) от количества лиц на фото (рис. 4).

На графике можно увидеть, что одно лицо распознавалось дольше, чем два и три лица. Это связано с тем, что человек был изображен крупным планом и количество пикселей элементов лица было больше, чем на 2 и 3 фото. В целом можно увидеть, что сложность алгоритма растет практически линейно. Тестирование проводилось на MacBook Pro 13 2020 с процессором Intel Core i5 и 16 Гб ОЗУ.

Заключение

В статье был рассмотрен алгоритм для редактирования элементов лица на фотографии. Были рассмотрены существующие решения и предложено нововведение для существующего алгоритма распознавания элементов лица на фото. Была приведена реализация предложенного алгоритма.

ЛИТЕРАТУРА

1. Stevens, E., & Antiga, L., Deep learning with PyTorch. New York, NY: Manning Publications, 2020.
2. Hope, T., Resheff, Y.S., & Lieder, I., Learning TensorFlow. Sebastopol, CA: O'Reilly Media, 2017.
3. Gulli, A., & Pal, S., Deep learning with keras. Birmingham, England: Packt Publishing, 2017.
4. Newnham J., Machine Learning with Core ML: An iOS developer's guide to implementing machine learning in mobile apps. Birmingham, England: Packt Publishing, 2018.
5. Kelby S., The Adobe Photoshop Book for Digital Photographers. Pearson Education, 2023.
6. Chavez C., Faulkner A., Adobe Photoshop Classroom in a Book. Adobe Press, 2020.
7. Монастырев В.В., Молодяков С.А. Разработка и применение в графическом редакторе алгоритмов многослойной обработки изображений // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и Технические Науки. — 2023. — №03. — С. 81–86 DOI 10.37882/2223–2966.2023.03.26

8. Zhou Y., Hu X., Zhang B., Interlinked Convolutional Neural Networks for Face Parsing. International Symposium on Neural Networks. Springer, Cham, 2015, DOI 10.1007/978-3-319-25393-0_25.
9. Karthi M., Niroshini Infantia C., Subhashini G., Shyam Sundar V., Comparative Studies with Random Datasets Using Enhanced Faster R-CNN, Mask R-CNN, and Single Shot Detector. Springer, Singapore, 2020, DOI 10.1007/978-981-19-9989-5_19.
10. Muñoz-Martínez F., L. Abellán J., E. Acacio M., CNN-SIM: A Detailed Arquitectural Simulator of CNN Accelerators. In: Schwardmann, U., et al. Euro-Par 2019: Parallel Processing Workshops. Euro-Par 2019. Lecture Notes in Computer Science(), vol 11997. Springer, Cham, DOI 10.1007/978-3-030-48340-1_56.
11. Liu X., Zhang R., Meng Z., Hong R., Liu G., On fusing the latent deep CNN feature for image classification. World Wide Web 22, 423–436, 2019, DOI 10.1007/s11280-018-0600-3.
12. Marques O., Image Processing and Computer Vision in iOS. SpringerBriefs in Computer Science. Springer, Cham, 2020, DOI 10.1007/978-3-030-54032-6_3.