

Контекстно-зависимое масштабирование изображений

Даниил Киреев, Владимир Гузов, Влад Шахуро



Обзор задания

В данном задании предлагается реализовать алгоритм, применяющийся для контекстно-зависимого масштабирования изображений. При стандартном подходе изображение равномерно деформируется по всей длине при изменении размера (объекты на изображении уменьшаются вместе со всем изображением). Данный же алгоритм учитывает контекст, и деформация происходит так, что объекты сохраняют свои размеры. Кроме того, если с помощью маски выделить какой-нибудь объект, то его можно удалить из изображения или наоборот оставить неизменным.



Описание задания

Вертикальное и горизонтальное сжатие изображения (3 балла)

Идея алгоритма заключается в том, чтобы удалять *швы* с наименьшей *энергией* из изображения. Шов — это связанный кривая, соединяющая первую и последнюю строчки/столбцы изображения (на втором изображении швы выделены красным). Энергией же каждой точки мы будем называть модуль градиента яркости в данной точке. Яркость изображения — компонента Y в цветовой модели YUV. Конверсия из RGB осуществляется по следующей формуле:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B$$

Если шов проходит через малое количество перепадов яркости, это означает, что он имеет малую энергию. Заметим, что объекты обычно имеют более сложную структуру, чем фон, а значит швы, которые будут проходить через них, будут иметь более высокую энергию, следовательно мы их не удалим. Вам необходимо реализовать функцию, которая бы уменьшила размеры изображения на 1 пиксель по заданному направлению.

Алгоритм удаления шва (для случая горизонтального уменьшения) выглядит следующим образом:

1. Считаем энергию изображения (находим норму градиента в каждой точке). Чтобы найти градиент, необходимо найти частные производные по каждому из направлений (по X и по Y). Для того, чтобы найти производную по X, нужно для каждого пикселя из правого соседа вычесть левый (получим разностную производную). Аналогично можно найти частную производную по Y (из нижнего пикселя вычитаем верхний). Производные на границе изображения считаем равными нулю. Теперь для того, чтобы найти норму градиента, необходимо для каждого пикселя извлечь корень из суммы квадратов частных производных.



2. Находим шов с минимальной энергией:

- (a) Создаем матрицу такого же размера, как исходное изображение, и инициализируем первую строку этой матрицы энергией соответствующих точек.
- (b) Построчно заполняем нашу матрицу. Для каждой точки мы находим минимального из трех верхних соседей (у крайних пикселей два соседа; в случае, если среди соседей есть одинаковые значения, то при формировании шва мы возьмем левого), складываем его значение, значение энергии данной точки и записываем результат в матрицу.

3	4	3	5
5	4	5	6
1	1	1	1

3	4	3	5
8	7	8	9
8	8	8	9

На данном примере: в первой матрице — значения энергии; во второй — матрица, которую требуется построить.

- (c) Теперь мы выбираем минимальное значение в последней строчке (если их несколько, то берем самое левое). Эта точка принадлежит минимальному шву. Теперь мы можем проследить все точки, которые принадлежат минимальному шву.

3. Удаляем этот шов из изображения.

Работа с маской + расширение изображения (2 балла)

Далее описание будет приведено для случая горизонтального изменения изображения (для вертикального всё аналогично).

С помощью маски мы можем контролировать формирование швов. Мы можем либо увеличить энергию соответствующих точек (тогда эти точки останутся на итоговом изображении), либо наоборот уменьшить (тогда эти точки будут удалены из изображения). Увеличивать или уменьшать маску мы будем на заведомо большую величину: произведение количества строк изображения на количество

столбцов изображения на 256 (верхняя оценка значения градиента в точке). С помощью такого метода мы можем, например, выделить с помощью маски лицо человека, и оно в результате работы нашего алгоритма не будет деформировано. В рамках задания вам необходимо будет реализовать возможность удалять объекты, которые выделены маской (уменьшать энергию соответствующих точек), и защищать их от деформации (увеличивать энергию соответствующих точек)



Для увеличения изображения необходимо найти минимальный шов и вставить справа от него новый, который бы являлся усреднением минимального шва и следующего за ним (для каждого пикселя минимального шва берется его сосед справа). Заметим, что если мы будем проводить несколько итераций нашего алгоритма, то минимальный шов каждый раз будет одним и тем же. Чтобы этого избежать необходимо увеличивать значение маски в точках данного шва (именно таким образом мы используем маску в случае расширения).

Интерфейс программы, данные и скрипт для тестирования

Необходимо реализовать функцию `seam_carve` со следующими аргументами:

1. Входное изображение.
2. Режим работы алгоритма, одна из четырех строк:

```
'horizontal shrink' — сжатие по горизонтали,
'vertical shrink' — сжатие по вертикали,
'horizontal expand' — расширение по горизонтали,
'vertical expand' — расширение по вертикали.
```

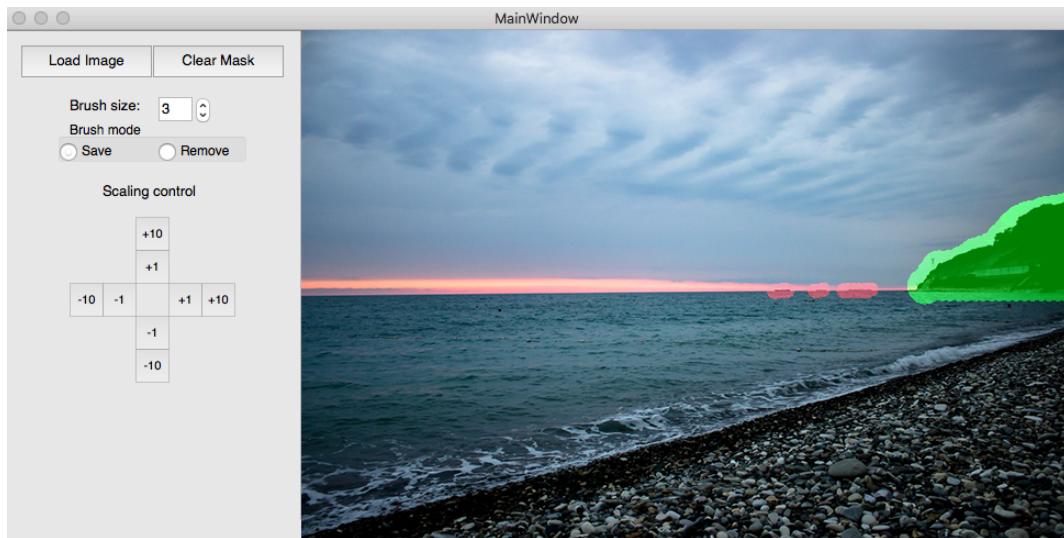
3. (Опциональный аргумент). Мaska изображения — одноканальное изображение, совпадающее по размерам со входным изображением. Мaska состоит из элементов $\{-1, 0, +1\}$. -1 означает пиксели, подлежащие удалению, $+1$ — сохранению. 0 означает, что энергию пикселей менять не надо.

Функция возвращает тройку — измененные изображение и маска, а также маска шва (1 — пиксели, которые принадлежат шву; 0 — пиксели, которые не принадлежат шву). В маске тоже необходимо удалить или добавить соответствующий шов.

Скрипт для тестирования `seam_carve_test` получает на вход строку с режимом работы и путь до директории с данными. Режимы работы: '--base' для базового задания (вертикальное и горизонтальное сжатие изображений) и '--full' для полного задания (+ работа с маской + расширение изображения). В результате работы тестового скрипта вы получите долю верных ответов работы вашего алгоритма.

Данные для тестирования — директория, в которой расположены изображения в формате png. Для каждого изображения в данной директории присутствуют два файла: `pic_xx_mask.png` — маска и `pic_xx_seams` — сериализованные с помощью модуля pickle координаты швов для всех возможных режимов работы (правильные ответы для каждого изображения).

Работу алгоритма можно визуализировать с помощью скрипта `gui.py` (может помочь вам с отладкой). У него есть optionalный аргумент — путь до изображения. Для работы скрипта требуется PyQt4. Попробуйте запустить свой алгоритм для разных вариантов изображений и масок.



Полезные ресурсы

[Видео-демонстрация работы алгоритма](#)

[Seam carving for content-aware image resizing](#) — оригинальная статья с большим количеством поясняющих примеров, рекомендуется для подробного знакомства с алгоритмом.

[Статья на википедии про seam carving](#)