Обработка и совмещение каналов изображения

Антон Конушин, Максим Новиков, Александр Сергеев, Влад Шахуро, Петров Илья.

Срок сдачи: 23 сентября (23:59).

Graphics & Media Lab
Vision Group

1 Обзор задания

Первым цветным фотографом России является Михаил Сергеевич Прокудин-Горский, сделавший единственный цветной портет Льва Толстого. Каждый его снимок представляет из себя три чёрно-белых фотопластинки, соответствующие красному, зелёному и синему цветовым каналам. Сейчас коллекция его снимков находится в американской библиотеке конгресса, сканкопии фотопластинок доступны в интернете.

В данном задании мы предлагаем вам создать программу, которая будет совмещать изображения, полученные с фотопластинок Прокудина-Горского и, таким образом, познакомиться с базовыми операциями обработки изображений. Реализация базовой части программы выполняется в несколько этапов:

1. Загрузка изображения.
2. Разделение изображения на цветовые каналы.
3. Поиск наилучшего сдвига для совмещения каналов.
4. Сохранение результата.

Возможны улучшения базового алгоритма. Они описаны после базовой части.

2 Базовая часть задания

За базовую часть задания можно получить **5** баллов, однако эта часть должна быть написана полностью. Получение баллов за бонусную часть невозможно, если базовая часть не написана полностью.

2.1 Загрузка и разделение изображения на цветовые каналы

Входное изображение представляют из себя три пластинки, соответствующие (сверху вниз) каналам B, G и R. На данном этапе нужно загрузить изображение из файла и разделить его на три канала, по трети высоты изображения на один канал.

2.2 Поиск наилучшего сдвига для совмещения каналов

На этом этапе происходит совмещение каналов. Для того, чтобы совместить два изображения, будем сдвигать одно изображение относительно другого в некоторых пределах, например, от -15 до 15 пикселей. Далее, для перекрывающихся областей изображений посчитаем некоторую метрику. Оптимальным будет тот сдвиг, при котором метрика принимает наибольшее/наименьшее значение (в зависимости от метрики). Предлагается реализовать две метрики и выбрать ту, которая позволяет получить более качественный результат при совмещении:

1. Среднеквадратичное отклонение для изображений I_1 и I_2:

$$MSE(I_1, I_2) = \frac{1}{width \cdot height} \sum_{x,y} (I_1(x, y) - I_2(x, y))^2,$$

   где $width, height$— ширина и высота изображений соответственно. Для нахождения оптимального сдвига нужно взять минимум по всем сдвигам.

2. Кросс-корреляция для изображений I_1 и I_2:

$$I_1 \star I_2 = \sum_{x,y} I_1(x, y) I_2(x, y).$$

   Для нахождения оптимального сдвига нужно взять максимум по всем сдвигам.

3 Бонусная часть задания

При желании, вы можете выбрать и реализовать любые пункты из бонусной части, однако получить можно максимум **10** баллов.

3.1 Удаление рамок пленки (5 баллов)

Можно заметить, что во многих изображениях в каналах со всех четырёх краев есть обрамление плёнки. Это обрамление может существенно ухудшить качество совмещения каналов. На данном этапе нужно реализовать детектор границ Canny, а затем с его помощью найти рамки плёнки и кадрировать изображение по этим границам.

Описание алгоритма работы детектора Canny:

1. *Сглаживание изображения.* Для уменьшения влияния шума изображение изображение сглаживается с помощью фильтра Гаусса радиуса 2 (т.е. ядро является матрицей размера 5×5) и $\sigma = 1.4$. Поскольку изображение может быть довольно большого размера, для ускорения алгоритма кроме обычной фильтрации Гаусса нужно реализовать сепарабельную фильтрацию. Тестироваться будут оба вида фильтрации (см. параграф про аргументы командной строки). Обратите внимание, что сумма элементов в ядре фильтра Гаусса должна равняться 1 (т.е. ядро перед свёрткой нужно нормализовать).

2. *Вычисление градиентов.* Для сглаженного изображения I вычисляются производные по x и y — I_x и I_y. Это делается путём свертки I с ядрами Собеля K_x и K_y соответственно:

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, \quad K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

   Затем вычисляется модуль градиента по формуле

$$|G| = \sqrt{I_x^2 + I_y^2},$$

и направление градиента по формуле

$$\theta(x, y) = \text{atan2}(I_y, I_x),$$

где atan2 — знаковый арктангенс, принимающий значения от $-\pi$ до π (эта функция есть в стандартной библиотеке).

3. *Подавление немаксимумов.* Для каждого пикселя выберем двух соседей (соседями считаются 8 пикселей вокруг данного пикселя): на первого показывает градиент (это можно определить по θ, будем считать, что на каждый соседний пиксель приходится по $\frac{\pi}{4}$ радиан, 0 — направление строго вправо), на второго показывает вектор, противоположный градиенту. Если модуль градиента текущего пикселя больше модулей градиентов соседей, то он остаётся без изменений. В противном случае модуль градиента пикселя подавляется (обнуляется).

4. *Отсечение по двум порогам.* На данном этапе производится сравнение модулей градиента с двумя заранее заданными числами (порогами), первое число меньше второго. Градиенты, меньшие по модулю первого порога, считаются незначительными и подавляются (обнуляются); градиенты, большие по модулю второго порога, считаются сильными, соответствующие им пиксели считаются границами и сохраняются в финальной карте границ; остальные градиенты считаются слабыми, вопрос включения соответствующих им пикселей в карту границ решается на следующем шаге алгоритма.

5. *Отслеживание границ по гистерезису.* Пиксель w со слабым градиентом объявляется границей и включается в финальную карту границ только в том случае, когда он лежит в одной связной компоненте с некоторым пикселем s, имеющим сильный градиент. Другими словами, w и s должны соединяться некоторой цепочкой пикселей-соседей (соседями считаются 8 пикселей вокруг данного пикселя) со слабыми градиентами.

   Предлагается придумать и реализовать *однопроходный* алгоритм (т.е. такой алгоритм, который обходит каждый пиксель карты градиентов по одному разу), выделяющий связные компоненты. После этого нужно решить, какие пиксели со слабыми градиентами будут включаться в финальную карту границ. Это нужно сделать за ещё один проход.

Рассмотрим теперь поиск строк и столбцов пикселей, по которым будет обрезаться рамка изображения.

В случае верхней границы будем искать строку в некоторой близости от верхнего края изображения (например, 5% от его высоты). Для каждой такой строки посчитаем количество принадлежащих ей пикселей границ, полученных с помощью детектора Canny. Среди этих чисел выберем два максимума. Этим максимумам будут соответствовать две строки. Из них выберем строку, наиболее удалённую от верхнего края изображения. Это делается потому, что в рамке изображения есть два перехода: от светлой подложки сканера к темной незасвеченной пленке, а потом уже к самому изображению. По выбранной строке и нужно обрезать изображение.

Для того, чтобы оба максимума не нашлись в непосредственной близости друг от друга (края пленки не соответствуют строго строкам изображения), необходимо реализовать подавление немаксимумов в некоторой окрестности: после обнаружения первой строки-максимума соседние к ней строки обнуляются, и после этого ищется второй максимум.

3.2 Построение и работа с пирамидой изображений (до 3 баллов)

Построение пирамиды (2 балла) Для ускорения совмещения предлагается реализовать пирамиду изображений. В пирамиде изображений исходное изображение последовательно уменьшается в k раз, например $k = 1.5$ или $k = 2$, до некоторого минимального размера (например, чтобы меньшая сторона была не меньше 400 пикелей в длину). Поиск оптимального сдвига начинается с самого

маленького изображения, а затем на пути к исходному изображению уточняется на уменьшенных копиях изображения. Таким образом, оригинальное изображение совмещается не в диапазоне $-15 \dots 15$ пикселей, а в меньшем, уточненном с помощью уменьшенных копий изображения.

Для составления пирамиды необходимо реализовать функцию изменения размера изображения. Для вычисления промежуточных значений при масштабировании изображения нужно использовать билинейную интерполяцию.

Бикубическая интерполяция (1 балл) Реализация бикубической интерполяции для вычисления промежуточных значений при масштабировании изображения.

3.3 Постпроцессинг: визуальное улучшение совмещенного изображения (до 6 баллов)

На данном этапе к совмещенному изображению можно применить различные преобразования, которые могут визуально улучшить изображение. Можно реализовать четыре преобразования.

Серый мир (1 балл) Преобразование уравнивает средние яркости каналов R, G и B. Сначала подсчитываются средние яркости каналов S_R, S_G, S_B, затем каждый пиксель каналов R, G, B домножается на число

$$\frac{S}{S_R}, \frac{S}{S_G}, \frac{S}{S_B}$$

соответственно. Здесь $S = \frac{S_R+S_G+S_B}{3}$.

Повышение резкости (1 балл) Преобразование повышает резкость изображения. Достаточно применить ядро повышения резкости к изображению:

$$\begin{pmatrix} -\frac{1}{6} & -\frac{2}{3} & -\frac{1}{6} \\ -\frac{2}{3} & 4\frac{1}{3} & -\frac{2}{3} \\ -\frac{1}{6} & -\frac{2}{3} & -\frac{1}{6} \end{pmatrix}.$$

Автоконтраст (1 балл) Преобразование состоит в линейном растяжении яркости изображения. Для начала считается яркость изображения Y по формуле

$$Y = 0.2125 \cdot R + 0.7154 \cdot G + 0.0721 \cdot B.$$

Затем вычисляются параметры линейного растяжения Y и к каждому из каналов R, G и B применяется линейное растяжение с вычисленными параметрами. Искомое линейное растяжение канала Y должно быть устойчивым (т.е. при подсчёте параметров для линейного растяжения заданная доля пикселей в начале и в конце гистограммы должна не учитываться).

Выравнивание баланса белого (3 балла) Добавьте в этап постпроцессинга алгоритм выравнивания баланса белого, описанный в параграфе 4 статьи Chikane, Fuh: Automatic White Balance for Digital Still Cameras.

3.4 Медианная фильтрация и быстрая медианная фильтрация (до 3 баллов)

Медианная фильтрация полезна для экспериментов с удалением импульсного шума. В данном задании нужно её реализовать. Это можно сделать тремя способами:

1. Обычная медианная фильтрация за $O(r^2)$ времени, где r — радиус окрестности, по которой берётся медиана. Фильтрация выполняется путём сортировки окрестности заданного радиуса каждого пикселя и выбора медианы в качестве нового значения пикселя.

2. Быстрая медианная фильтрация за $O(r)$ времени.

3. Быстрая медианная фильтрация за $O(1)$ времени.

Две последних фильтрации подробно и понятно описаны в статье Perreault et al: Median Filtering in Constant Time. Прочитайте статью и разберитесь в алгоритмах. Достаточно посмотреть описание линейного алгоритма в конце параграфа I и константного алгоритма, ему посвящен параграф II (мы не описываем его тут, разобраться — это часть задания).

За реализованный только пункт 1 начисляется **1** балл. За реализацию алгоритмов 1 и 2 или алгоритмов 1 и 3 можно получить **2** балла. За реализацию трёх алгоритмов можно получить **3** балла. Обращаем ваше внимание на то, что реализации алгоритмов должны удовлетворять указанным сложностям, будет проводиться проверка корректности и скорости алгоритмов.

3.5 Зеркальное дополнение изображения при фильтрации (2 балла)

В каркасе фильтруются только те пиксели, окрестности которых не выходят за границу изображения. Предлагается реализовать дополнение изображения по принципу зеркалирования. Баллы будут даваться только тем решениям, которые выполняют дополнение изображения до прохода фильтром по изображению.

3.6 Поиск смещения с субпиксельной точностью (1 балл)

Поиск сдвига с субпиксельной точностью означает, что каждому пикселю соответствует k субпикселей (например, $k = 2$) и поиск сдвига проводится с точностью до субпикселя. На практике достаточно увеличить масштабировать изображение в k раз и произвести обычный поиск сдвига, а затем совмещенное изображение уменьшить обратно в k раз.

4 Среда программирования и устройство каркаса

Для упрощения выполнения задания под Windows вам предлагаются файл cygwin.exe. Это самораспаковывающийся архив, в нём находится cygwin — среда программирования с gcc. Разархивируйте cygwin.exe и запустите сценарий Cygwin.bat. Откроется консоль и вы окажетесь в папке /home/<username>. Скопируйте в эту папку архив align_project.tar.gz с каркасом задания и разархивируйте командой `tar xvf align_project.tar.gz`; Зайдите в папку проекта, введя в консоли команду `cd align_project`. Запустите сценарий компиляции командой `make all`. В папке build/bin появятся исполняемые файлы. Их можно запустить из консоли командой `./build/bin/<имя файла>` . Удалить все скомпилированные файлы можно командой `make clean` .

Если вы используете Linux, то вам будет достаточно поставить gcc и GNU make.

- src — директория с исходным кодом в формате .cpp. Здесь особенно интересен файл matrix_example.cpp, в котором показан пример свертки с фильтром, суммирующим окрестность пикселя.

- include — директория с заголовочными файлами. Здесь интересны файлы matrix.h — объявление класса матрицы и matrix.hpp — шаблонная реализация этого класса.

- externals — директория с исходными кодами библиотек. Здесь же библиотеки компилируются. В каркасе есть только одна библиотека, с помощью которой загружаются изображения в формате BMP.

- bridge — директория, в которую добавляются заголовочные файлы и скомпилированные библиотеки для импорта в основной проект.
- build/bin — директория, в которой сохраняются выполняемые файлы компиляции (т.е. после того, как отработала команда `make all`).

Вы можете выполнить задание в Visual Studio, но при посылке в проверяющую систему ваш проект должен иметь сценарий компиляций на языке GNU make, а также без ошибок собираться с помощью gcc.

Ваша программа не должна выполнять никаких сетевых взаимодействий, в противном случае она будет дисквалифицирована без права пересдачи. Использовать параллелизм в рамках базового задания не рекомендуется — программа будет запускаться с выделением одного логического процессора. Это необходимо для того, чтобы мы могли сравнивать производительность ваших программ с точки зрения эффективности реализованных алгоритмов.

5 Аргументы командной строки

В каркасе прописаны все необходимые опции программы, которые будут использоваться при проверке. Для проверки базовой части команда выглядит следующим образом (при вызове из папки build/bin/):

./align <путь к входному изображению> <путь к результирующему изображению> `--align`
Для проверки заданий бонусной части добавлены, например, следующие опции:

`--bicubic-interp` после опции `--align`, которая будет включать бикубическую интерполяцию;

`--subpixel <k>` после опции `--align`, которая будет включать субпиксельное совмещение изображений;

`--white-balance` в двух вариантах: после опции `--align` и отдельно для обработки обычных трехканальных изображений;

`--median <radius>`, `--median-linear <radius>`, `--median-const <radius>` — опции для сглаживания изображения медианным фильтром (используются только для обработки изображения).

Более подробную документацию и список всех опций можно получить, вызвав ./align --help из build/bin/.

6 Сдача и проверка заданий

Для сдачи задания выполните команду `make clean` в папке проекта, перейдите на папку выше командой `cd ..` и заархивирует папку с проектом командой `tar cvzf solution.tar.gz align_project`. Полученный архив solution.tar.gz нужно послать в проверяющую систему. Исполняемые программы или скомпилированные библиотеки прикладывать в архив не нужно, сборка будет производится на нашей стороне.

Срок сдачи: 23 сентября (23:59), после этой даты начинается начисление штрафных баллов (за первые двое суток по **-0.5** баллов, далее по **-1** баллу за сутки). Программы, присланные после 30 сентября (23:59) проверяться не будут. Такие сроки сдачи нужны для того, чтобы мы могли оперативно проверить ваши программы. Проверка проходит в два этапа:

1. Автоматизированный прогон вашей программы на тестах. Занимает от 1 дня до недели.

2. Личное собеседование по коду вашей программы и обсуждение реализованных алгоритмов. Время и аудитория, в которой будут проходить собеседования, будут сообщены отдельно.

Все задания проверяются только на нашем компьютере. Мы *не* будем смотреть выполненное задание на вашем ноутбуке.

Данные для тестирования — 6 картинок в двух вариантах: small и big для тестирования базовой реализации и реализации с пирамидой соответственно. Время работы пирамиды на одной картинке не должно превышать 20 секунд. Кроме того, совмещение должно проходить достаточно качественно — не должно быть явно видимых сдвигов (т.е. сдвигов величиной больше 10 пикселей).

7 Полезные ресурсы

Выставка о Прокудине-Горском на сайте библиотеки конгресса.