Санкт-Петербургский государственный университет

Кафедра технологии программирования

Волох Владислав Васильевич

Выпускная квалификационная работа бакалавра

Алгоритм автоматического сбора изображения типа «паззл» из фрагментов

Направление 010400

Прикладная математика и информатика

Научный руководитель,

Старший преподаватель

Малинин К. А.

Санкт-Петербург

2016

Содержание

Введение

Компьютерное зрение и, в частности, задачи обработки изображений и распознавания образов в настоящее время получают широкое распространение в различных сферах деятельности. Методы из этой области знаний находят применение в решении самых разнообразных задач. Повсеместно используются системы видеонаблюдения, способные отслеживать движущиеся объекты; автоматические системы обработки видеопотока осуществляют контроль качества на производствах; необходимость в методах обработки изображений возникает и в криминалистике, и в информационном поиске, и при разработке автономных автомобилей.

Несмотря на широкое распространение компьютерного зрения, существует довольно много нерешенных или имеющих потенциал для лучшего решения задач в этой области. Ярким примером стал DARPA Shredder Challenge, в котором всем желающим было предложено написать программу, позволяющую максимально восстановить документы из фрагментов, извлеченных из уничтожителей бумаг разного уровня безопасности. Соревнование, получив широкую огласку, привлекло большое число исследователей и энтузиастов. В результате, конкурсантам удалось решить задачу с достаточно высокой точностью, позволяющей извлечь достаточно большое количество информации из документов.

В настоящее время задачи восстановления изображения из фрагментов не получают достаточно широкого распространения. Большинство работ решают довольно узкую задачу восстановления цифрового изображения, разделенного на одинаковые прямоугольные(а чаще всего даже квадратные) фрагменты[1],[4],[5]. Но при такой постановке задачи авторы этих работ не сталкиваются с многими аспектами, как например: извлечение необходимых фрагментов из изображения, содержащего эти фрагменты; решение проблем с помехами и зашумленностью изображения; различие в размерах и форме

фрагментов; искажение формы фрагментов; влияние освещения на изображения и многие другие. Такой вариант постановки задачи практически неприменим к реальным проблемам восстановления изображений, поскольку не принимает во внимание указанные выше аспекты, которые всегда в той или иной степени будут присутствовать.

В процессе изучения данного вопроса, была замечена работа[2], авторы которой пытались собрать паззл, полученный из цифрового изображения, но по результатам тестов признали свой метод имеющим крайне низкую точность. По прежнему, им не приходилось иметь дело с описанными выше аспектами, за исключением различия формы фрагментов.

В данной работе рассматривается алгоритм сборки паззла, исходные изображения которого получены при помощи устройств захвата изображений, таких как сканер или фотоаппарат. Рассматриваемый алгоритм является комплексным решением, включающим в себя полный цикл действий от разбора исходных изображений до формирования итогового изображения.

Постановка задачи

Целью данной выпускной квалификационной работы является разработка алгоритма, решающего задачу автоматической, с возможностью корректировки ошибок человеком, сборки изображения типа “паззл” из фрагментов.

Для достижения поставленной задачи требуется:

- определить набор подзадач алгоритма;
- изучить существующие подходы для решения каждой подзадачи;
- изучить используемый инструментарий для разработки(библиотеки OpenCV, Qt5, GTEngine);
- реализовать этапы алгоритма сборки изображения;
- произвести тестирование и анализ полученных результатов.

Входные данные алгоритма:

В качестве входных данных выступает набор изображений, на каждом из которых изображены один или более фрагментов. Фрагменты паззла должны содержаться на изображениях полностью, иметь одинаковый масштаб, а также не должны соприкасаться друг с другом. Каждый фрагмент должен содержаться во входных данных только один раз. Входные изображения в совокупности должны содержать все фрагменты паззла.

Выходные данные алгоритма:

- присвоение каждому фрагменту уникального идентификатора
- указания по сборке фрагментов в единое изображение
- итоговое изображение

Глава 1. Теоретическая часть

В данной работе паззлы расматриваются, как частный случай задачи восстановления изображения из фрагментов. Исходными данными является серия необработанных изображений, полученных при помощи сканера и содержащих один или несколько фрагментов паззла. Ставя перед собой задачу сборки паззла, необходимо учесть все указанные выше аспекты, а также решить задачу получения отдельных фрагментов из входных данных.

Получение отдельных фрагментов является нетривиальной задачей и включает в себя очистку изображения от шумов, улучшение качества изображения и получение точных границ объекта. Каждая проблема по отдельности имеет множество возможных решений, каждое со своими нюансами. Для получения результата необходимо принять во внимание особенности всех применяемых алгоритмов, подобрать оптимальный набор инструментов и порядок их применения.

Решение поставленной задачи разбивается на несколько этапов:

- этап парсинга изображений
- этап постобработки
- этап сборки.

Начальные данные не включают в себя информации об отдельных фрагментах, об их расположении на исходных изображениях и т.п. Задача этапа парсинга изображений - максимально точно извлечь из исходных изображений информацию о:

- количестве фрагментов
- расположении фрагментов на исходных изображениях
- характерных точках фрагментов(углы)
- форме фрагментов(в виде границ, отделяющих фрагмент от фона)

Также, задачей этапа парсинга является представление извлеченной информации в удобном для обработки виде. Так, в дополнение к описанным выше целям добавляется:

- генерация масок, покрывающих только заданный фрагмент
- извлечение фрагмента с исходного изображения при помощи сгенерированных масок для использования в последующих этапах
- извлечение информации о форме отдельных сторон фрагмента

Первая задача, которую необходимо решить - извлечение информации о форме фрагментов. Классическим решением для этой задачи является применение детекторов границ. Значительная часть детекторов является чувствительной к зашумленности входных изображений, поэтому перед их применением необходимо произвести предварительную обработку изображений. Для снижения уровня шума на изображении чаще всего применяется пространственная фильтрация изображений с использованием сглаживающих фильтров, таких как:

- фильтр Гаусса
- медианный фильтр
- билатеральный фильтр

После предварительной обработки изображения можно применить детектор границ для получения информации о границах на исходных изображениях.

Полученную информацию можно использовать для определения точных границ фрагментов, их формы, количества; а также извлечь с ее помощью отдельные фрагменты с исходных изображений. Для этого был разработан алгоритм получения масок фрагментов, использующий данные о границах на изображении. В своей реализации алгоритм использует морфологические преобразования для повышения качества

границ(устранения незамкнутых ответвлений, а также замыкания небольших разрывов границы).

Маски дают возможность отделить фрагменты от исходных изображений, а также используются в качестве входных данных для детектора углов, позволяющего найти углы на изображении. Найденные углы, в свою очередь, послужат точками, разделяющими контур фрагмента на 4 отдельные стороны. Классическим решением задачи нахождения углов является детектор углов Харриса. Нахождение углов и разделение контура фрагмента на отдельные стороны завершают этап парсинга.

Получив из этапа парсинга отдельные стороны, необходимо подготовить данные для этапа сборки. Это общая задача этапа пост обработки; она включает в себя:

- стандартизацию сторон
- вычисление метрик для последующего сравнения сторон

Первым делом необходимо привести стороны к одному стандарту. Это откроет возможность сравнивать стороны, а также упростит вычисление метрик. Набор метрик, а также алгоритмы стандартизации сторон и вычисления метрик были разработаны в рамках данной работы.

Результаты работы этапа постобработки используются в этапе сборки для определения степени совпадения сторон и сборки финального изображения. Этап сборки также использует оригинальный алгоритм.

Далее подробно рассмотрим существующие решения и алгоритмы, использованные в данной работе.

1.1 Пространственная фильтрация

Представим изображение размером MxN пикселей как матрицу размерности MxN с целочисленными значениями. В случае многоканальных

изображений будем рассматривать каждый канал в отдельности по тому же принципу.

Для преобразований изображения будем применять обработку изображения с помощью скользящей маски, называемой также фильтром, ядром, окном или шаблоном, которая представляет собой некую квадратную матрицу, соответствующую указанной группе пикселей исходного изображения. Элементы матрицы принято называть коэффициентами. Оперирование такой матрицей в каких-либо локальных преобразованиях называется фильтрацией или пространственной фильтрацией.

f(x-1,y-1)	f(x,y-1)	f(x+1,y-1)
f(x-1,y)	f(x,y)	f(x+1,y)
f(x-1,y+1)	f(x,y+1)	f(x+1,y+1)

Элементы изображения под маской

w(-1,-1)	w(0,-1)	w(1,-1)
w(-1,0)	w(0,0)	w(1,0)
w(-1,1)	w(0,1)	w(1,1)

Коэффициенты маски

Результирующее изображение получается путем вычисления отклика фильтра в каждой точке исходного изображения. Отклик задается суммой произведения коэффициентов фильтра на соответствующие значения пикселей в области, покрытой маской фильтра.

$$f^*(x,y) = \sum_{i=-m}^{m} \sum_{j=-n}^{n} f(x+i, y+i) w(i,j)$$

Рассмотрим фильтры, которые часто применяются при предварительной обработке.

1.1.1 Фильтр Гаусса

Фильтр, применение которого визуально проявляется в размытии изображения; ядро фильтра вычисляется с использованием функции распределения Гаусса:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$ (одномерный случай),

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$ (двумерный случай),

где x - расстояние по оси X от начала координат, y - расстояние по оси Y от начала координат, σ - стандартное отклонение распределения Гаусса. Фильтр с таким ядром реализует идею вычисления взвешенного среднего значения каждого пикселя, где влияние каждого пикселя на результат зависит от его расстояния до рассматриваемого пикселя. Теоретически, в каждой точке изображения функция Гаусса принимает ненулевое значение и каждый пиксель должен влиять на результат вычислений для каждого пикселя изображения, но, поскольку значения функции Гаусса на удалении более 3σ от начала координат являются чрезвычайно маленькими, ими можно пренебречь.

Стоит отметить два полезных свойства фильтра Гаусса:

- применение нескольких фильтров Гаусса друг за другом имеют тот же эффект, что и применение одного фильтра с размером ядра, равным корню от суммы квадратов размеров ядер меньших фильтров
- применение фильтра в двумерном случае может быть разложено на 2 этапа(вертикальный и горизонтальный), что снижает вычислительную сложность алгоритма

Фильтр Гаусса хорошо зарекомендовал себя в задачах компьютерного зрения и часто применяется для сглаживания шума, а также в задачах downsampling’a(децимации) изображений.

При разработке детектора границ Канни фильтр Гаусса был выбран в качестве первого шага алгоритма. Его применение значительно снижает уровень мелкого, точечного шума. Минус фильтра состоит в размытии им изображения: незначительное размытие практически не влияет на нахождение границ, но при сильном воздействии края сильно теряют свою четкость и могут перестать распознаваться как граница изображения, либо распознаваться некорректно. Тем самым, фильтр Гаусса применим для избавления от небольших шумов и улучшения качества распознавания путем снижения реакции детектора на небольшие зоны и шум.

1.1.2 Медианный фильтр

Принцип работы медианного фильтра очень прост – значения внутри окна фильтра сортируются в порядке возрастания (убывания); и значение, находящееся в середине упорядоченного списка, поступает на выход фильтра. В случае четного числа отсчетов в окне выходное значение фильтра равно среднему значению двух отсчетов в середине упорядоченного списка.

Эффект медианного фильтра сильно зависит от размеров его ядра. Выбрав небольшой размер, например 3x3 или 5x5, можно добиться практически полного подавления мелких шумов(занимающих менее 50% от размера ядра), а также добиться большей однотонности зон с близким цветом. Ядра больших размеров усиливают эффект цветовой сегментации, что, с одной стороны, дает положительный эффект, так как исчезает реакция детектора на мелкие перепады внутри зон, но, с другой стороны, потенциально порождает нежелательный ответ детектора границ на границе цветовых зон. Также побочным эффектами применения медианного фильтра являются подавление тонких границ и слабых границ, а также сильная тенденция к скруглению углов.

1.1.3 Билатеральный фильтр

Билатеральный фильтр является нелинейным сглаживающим фильтром. Значение каждого пикселя изображения заменяется взвешенным средним соседних пикселей. Веса могут определяться, например, распределением Гаусса. Но, веса в билатеральном фильтре также зависят и от радиометрических показателей(таких, например, как разница в цвете или яркости рассматриваемых пикселей).

Билатеральный фильтр определяется следующим выражением:

$$I^{filtered}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|),$$

где

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

нормализует отклик и удерживает его в допустимых пределах.

$I^{filtered}$ - результирующее изображение;

I - исходное изображение;

x - координаты пикселя, в котором вычисляется значение;

Ω - окно с центром в x;

f_r - ядро, сглаживающее разницу в интенсивности цветов; может определяться функцией Гаусса;

g_s - ядро, сглаживающее разницу в расстоянии между пикселями; может определяться функцией Гаусса;

Как было показано выше, W_p определяется разницей в расстоянии между пикселями и интенсивности цвета пикселей. Для примера рассмотрим пиксель, расположенный по координатам (i, j), для которого вычисляется отклик билатерального фильтра и один из пикселей, попадающих в ядро фильтра, расположен по координатам (k,l). Тогда вес, назначенный пикселю по координатам (k,l) при определении отклика фильтра в точке (i, j) определяется формулой:

$$w(i, j, k, l) = e^{\left(- \frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\left\| I(i,j) - I(k,l) \right\|^2}{2\sigma_r^2} \right)}$$

где σ_d и σ_r являются параметрами сглаживания, а $I(i, j)$ и $I(k,l)$ представляют собой интенсивность цвета пикселей и соответственно. После вычисления весов, нормализуем их:

$$I_D(i, j) = \frac{\sum_{k,l} I(k,l) w(i, j, k, l)}{\sum_{k,l} w(i, j, k, l)},$$

где I_D - новое значение пикселя по координатам (i, j).

Основным преимуществом билатерального фильтра перед медианным и гауссовским фильтром является сохранение границ – правильно примененный фильтр не приводит к видимым изменением в положении и

четкости границ, причем свобода выбора параметров довольно велика. Билатеральный фильтр с чрезмерно высокими параметрами же приведет к сильному размытию изображения. Результат его работы можно охарактеризовать как “более мягкий медианный фильтр”. Является более сложным в вычислительном плане по сравнению с медианным фильтром и требует настройки большего числа параметров, что может затруднить его применение в автоматическом режиме. Также, использование билатерального фильтра может дать побочные эффекты в виде порождения регионов с постоянным цветом, что, в свою очередь, создает ложные границы[9], а также инвертировать цвета[10]. Эффект изменяется достаточно плавно, и умеренное изменение параметров не приводит к значительным изменениям в результате.

1.2 Детекторы границ

Для обнаружения границ используются аналоги производных первого и второго порядка. Вычисление первой производной цифрового изображения основано на различных дискретных приближениях двумерного градиента. По определению, градиент изображения $f(x,y)$ в точке (x,y) — это вектор:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Важную роль при обнаружении контуров играют такие характеристики, как модуль этого вектора: $|\nabla f| = \sqrt{G_x^2 + G_y^2}$

и направление вектора градиента: $\alpha(x,y) = arctg\left(\frac{G_y}{G_x}\right)$

Рассмотрим одни из самых распространенных детекторов границ - операторы Робертса, Превитта, Собеля, Шарра и детектор границ Канни.

1.2.1 Оператор Робертса

Оператор Робертса является простейшим в реализации, наименее ресурсозатратным(по причине размера ядра 2x2, что уменьшает количество необходимых вычислений для каждого пикселя), считается дающим неплохие результаты и находит свое применение в задачах, требующих максимального быстродействия или накладывающих ограничения на производительность системы.

-1	0
0	1

0	-1
1	0

Маски оператора Робертса

1.2.2 Оператор Превитта

Оператор Превитта использует ядро размером 3x3, что является наиболее распространенным размером ядра. В своей реализации оператор Превитта приближает производную по оси X как разность между верхней и нижней строкой, по оси Y - между левым и правым столбцом.

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Маски оператора Превитта

1.2.3 Оператор Собеля

Оператор Собеля очень похож на оператор Превитта. Его отличие заключается в использовании весового коэффициента 2 для средних элементов, что на практике дает меньший эффект сглаживания.

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Маски оператора Собеля

1.2.4 Оператор Шарра

Оператор Шарра является модификацией оператора Собеля.

-3	-10	-3
0	0	0
3	10	3

-3	0	3
-10	0	10
-3	0	3

Маски оператора Шарра

1.2.5 Детектор границ Канни

Описанные выше операторы, по своей сути, только приближают градиент изображения, предоставляя свободу в интерпретации полученных результатов, но поскольку их применение визуально выделяет границы, эти операторы относятся к категории детекторов границ. Детектор границ Канни представляет собой несколько более комплексный алгоритм, включающий в себя предварительную обработку изображения для уменьшения шумов, которые крайне негативно влияют на результат применения любого детектора границ, а также обработку полученных результатов, в результате которой отсекаются "слабые" границы, а оставшиеся границы приводятся к толщине в 1 пиксель.

Основные этапы работы детектора границ Канни:

1. *Сглаживание*. На этом этапе применяется гауссовское размытие - фильтр который может быть хорошо приближен к первой производной гауссианы. Ядро может несколько варьироваться из-за выбранного способа получения целого числа из дробного, а также в зависимости от выбранного параметра.

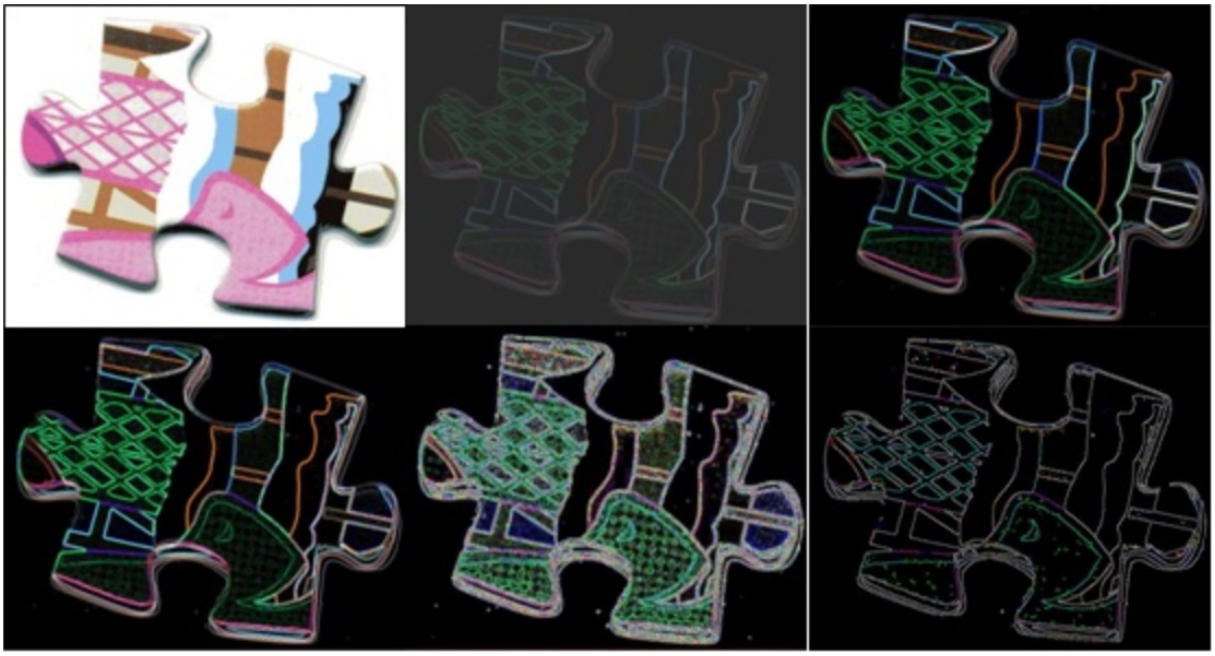
   Уравнение для нахождения элементов ядра:

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(- \frac{(i-k-1)^2 + (j-k-1)^2}{2\sigma^2} \right)$$

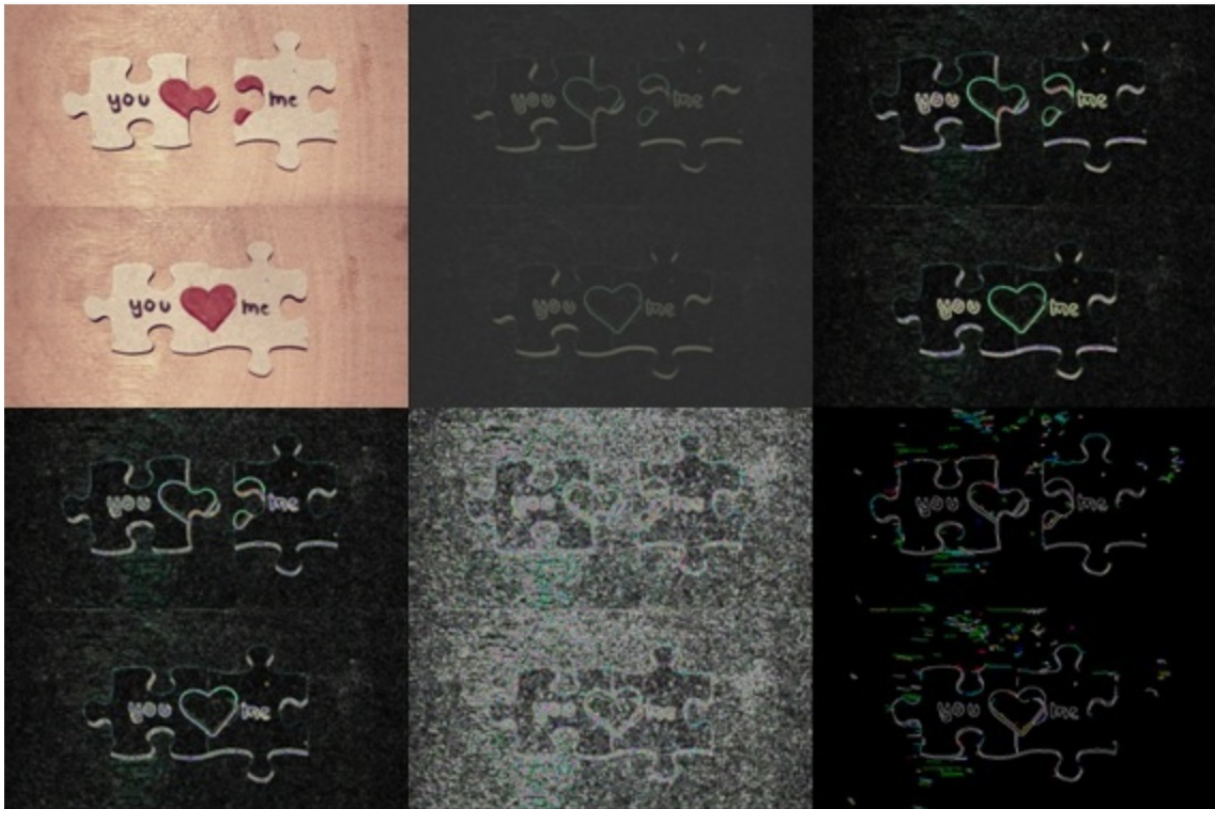
   Чаще всего на практике применяют ядро размерностью 5x5.

2. *Поиск градиентов*. Для каждой точки изображения рассчитывается значение и направление градиента. Угол направления вектора градиента округляется и может принимать такие значения: 0, 45, 90, 135.
3. *Подавление немаксимумов*. Значение градиента в каждом пикселе сравнивается со значением градиента в соседних пикселях в направлении градиента и антиградиента. Если значение в рассматриваемом пикселе оказывается локальным максимумом, то оно не изменяется, иначе заменяется на 0.
4. *Двойная пороговая фильтрация*. Детектор границ Канни имеет настраиваемый верхний и нижний порог. Значения в каждом пикселе сравниваются с пороговыми значениями: если значение выше верхнего порога – данный пиксель считается «сильной» границей; если значение выше нижнего порога и ниже верхнего – данный пиксель считается «слабой» границей. В пиксели, значение в которых меньше нижнего порога, записывается 0.
5. *Трассировка области неоднозначности*. На этом этапе принимается решение о том, какие пиксели, определенные как «слабая» граница, будут включены в итоговый результат. Чаще всего, критерием становится соприкосновение слабой и сильной границы.

Детектор границ Канни хорошо справляется с задачей нахождения границ, с достаточной точностью находя границы объекта, плюсом является и то, что результат его работы удобен для дальнейшей обработки. Как и любой другой подобный алгоритм, он сильно зависит от зашумленности изображения, а также не может обнаружить слабые границы, сохранив при этом высокую степень точности.



Демонстрация алгоритмов детектирования границ. Слева направо, сверху вниз: исходное изображение, результат применения оператора Робертса, оператора Превитта, оператора Собеля, оператора Шарра, детектора границ Канни



Демонстрация алгоритмов, объект близок по цвету и текстуре к фону, множество мелких шумов. Слева направо, сверху вниз: исходное изображение, результат применения оператора Робертса, оператора Превитта, оператора Собеля, оператора Шарра, детектора границ Канни

1.3 Морфологические преобразования

Морфологические преобразования в основном применяются только для бинарных изображений и позволяют, в зависимости от выбранных преобразований и их параметров, уменьшить уровень зашумленности(артефакты на границах для бинарных объектов), сгладить контуры объектов, избавиться от случайных вкраплений и эффективно заполнить небольшие пустоты.

Морфологические преобразования требуют задания двух изображений - исходного изображения и структурного элемента.

Исходное изображение представляет собой некоторое изображение; чаще всего, определяя морфологические операции, на него накладывают требование о бинарности исходного изображения. Некоторые допущения позволят определить морфологические операции для любых изображений, но целесообразность этого находится под вопросом. В данной работе, согласно традиционному подходу, на исходное изображение для морфологических операций накладывается требование о его бинарности.

Структурный элемент представляет собой некоторое двоичное изображение (геометрическую форму). Он может быть произвольного размера и произвольной структуры. Чаще всего используются симметричные элементы, как прямоугольник фиксированного размера или круг некоторого диаметра. В каждом элементе выделяется особая точка, называемая начальной. Она может быть расположена в любом месте элемента (и вне его). По умолчанию начальной точкой симметричных элементов является центральная точка.

Основные операции

В начале результирующая поверхность заполняется 0, образуя полностью черное изображение. Затем осуществляется зондирование (probing) или сканирование исходного изображения пиксель за пикселем структурным элементом. Для зондирования каждого пикселя на изображение «накладывается» структурный элемент так, чтобы совместились зондируемая

и начальные точки. Затем проверяется некоторое условие на соответствие пикселей структурного элемента и точек изображения «под ним». Если условие выполняется, то на результирующем изображении в соответствующем месте ставится 1 (в некоторых случаях будет добавляться не один единичный пиксель, а все единички из структурного элемента).

По рассмотренной выше схеме выполняются базовые операции. Такими операциями являются расширение и сужение. Производные операции — это некоторая комбинация базовых, выполняемых последовательно. Основными из них являются открытие и закрытие.

Базовые операции

Перенос

Операция переноса X_t множества пикселей X на вектор t задаётся в виде $X_t = \{x + t \mid x \in X\}$. Следовательно, перенос множества пикселей на бинарном изображении сдвигает все пиксели на заданное расстояние. Вектор переноса t может задаваться в виде упорядоченной пары $(\Delta r, \Delta c)$, где Δr — компонент вектора переноса в направлении строк, а Δc — компонент вектора переноса в направлении столбцов изображения.

Наращивание(дилатирование, dilate)

Наращивание бинарного изображения A структурным элементом B обозначается $A \oplus B$ и задается выражением:

$$A \oplus B = \bigcup_{b \in B} A_b \ .$$

В данном выражении оператор объединения можно считать оператором, применяемым в окрестности пикселей. Структурный элемент B применяется ко всем пикселям бинарного изображения. Каждый раз, когда начало координат структурного элемента совмещается с единичным бинарным пикселем, ко всему структурному элементу применяется перенос и последующее логическое сложение (логическое ИЛИ) с соответствующими пикселями бинарного изображения. Результаты логического сложения

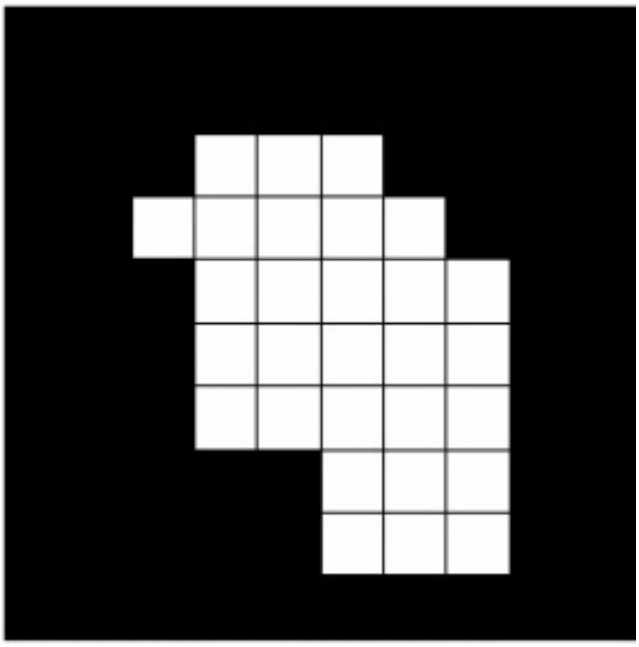
записываются в выходное бинарное изображение, которое изначально инициализируется нулевыми значениями.

Эрозия(erode)

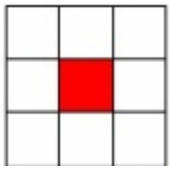
Эрозия бинарного изображения A структурным элементом B обозначается $A \ominus B$ и задается выражением:

$$A \ominus B = \left\{ z \in A \mid B_z \subseteq A \right\}.$$

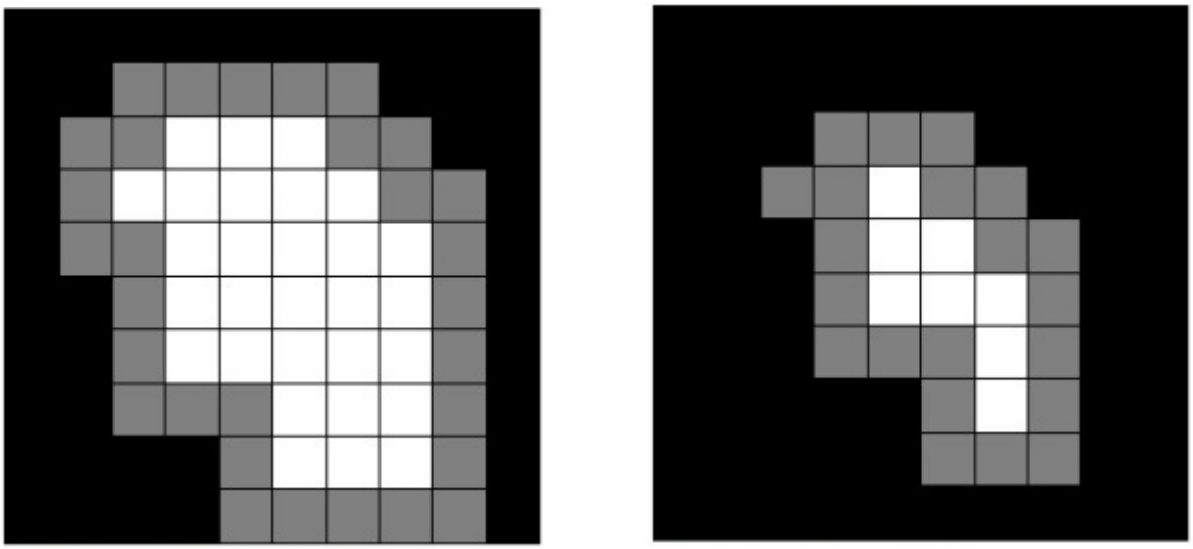
При выполнении операции эрозии структурный элемент тоже проходит по всем пикселям изображения. Если в некоторой позиции каждый единичный пиксель структурного элемента совпадет с единичным пикселем бинарного изображения, то выполняется логическое сложение центрального пикселя структурного элемента с соответствующим пикселем выходного изображения. В результате применения операции эрозии все объекты, меньшие чем структурный элемент, стираются, объекты, соединённые тонкими линиями становятся разъединёнными и размеры всех объектов уменьшаются.



Исходное изображение



Структурный элемент



Результаты применения операций дилатирования(слева) и эрозии(справа). Серым отмечены пиксели, которые будут закрашены белым(в случае дилатирования) или черным(в случае эрозии).

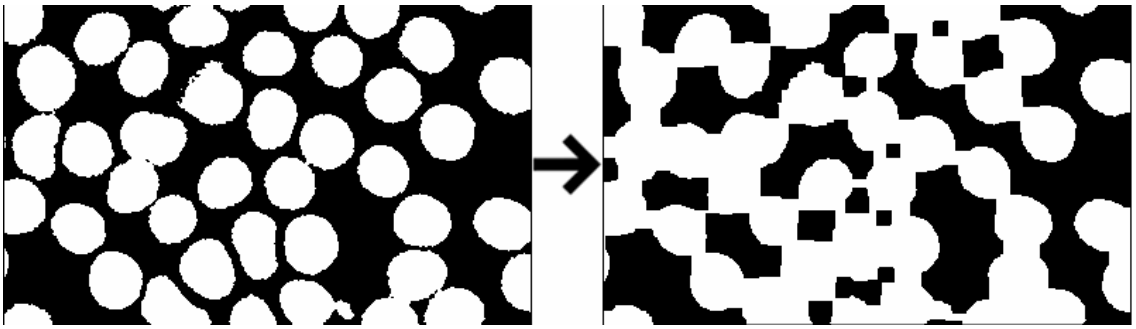
Производные операции

Замыкание(close)

Замыкание бинарного изображения A структурным элементом B обозначается $A \bullet B$ и задается выражением:

$$A \bullet B = \left(A \oplus B\right) \ominus B .$$

Если к изображению применить сначала операцию наращивания, то мы сможем избавиться от малых дыр и щелей, но при этом произойдёт увеличение контура объекта. Избежать этого увеличения позволяет операция эрозия, выполненная сразу после наращивания с тем же структурным элементом. Последовательное применение операций наращивания и эрозии называется замыканием. Операция замыкания позволяет избавиться от небольших пустых областей в изображении, и убрать углубления по краям областей.



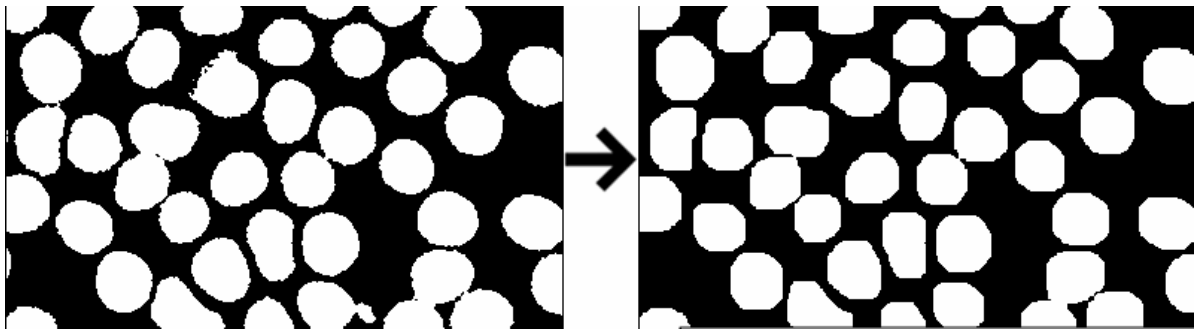
Пример замыкания изображения

Размыкание(open)

Размыканием бинарного изображения A структурным элементом B обозначается и задается выражением:

$$A \circ B = \left(A \ominus B\right) \oplus B .$$

Операция эрозии полезна для удаления малых объектов и различных шумов, но у этой операции есть недостаток — все объекты становятся меньше. Этого можно избежать, если после операции эрозии применить противоположную операцию - операцию наращивания с тем же структурным элементом. Последовательное применение операций эрозии и наращивания называется размыканием. Размыкание отсеивает все объекты, меньшие чем структурный элемент, но при этом помогает избежать сильного уменьшения размера объектов. Также размыкание идеально подходит для удаления линий, толщина которых меньше, чем диаметр структурного элемента. Также важно помнить, что после этой операции контуры объектов становятся более гладкими.



Пример размыкания изображения

1.4 Детектор углов

Характерная точка (точка интереса) — точка изображения, обладающая высокой локальной информативностью. Одними из самых информативных особенностей любого изображения являются углы, которые повсеместно встречаются на изображениях. Углы фрагментов паззла представляют большой интерес, поскольку обладая знанием о местоположении углов паззла можно с легкостью разбить контур фрагмента на контуры отдельных сторон фрагмента.

Исследования локальных точек интереса начались в 1981г. с работы по стерео-привязке с использованием детектора углов. Автор рассмотрел изменение яркости небольшого фрагмента вокруг интересующей точки при сдвиге фрагмента на один пиксель в восьми направлениях (горизонтальном, вертикальном и диагональном). В дальнейшем исследователи стали рассматривать производные яркости изображения для исследования изменений яркости по множеству направлений.

Рассмотрим фрагмент U изображения $I(x, y)$ с центром в точке (u, v), и его копии, сдвинутые на величину (x, y).

Для каждой точки фрагмента можно вычислить взвешенный квадрат разности между сдвинутым и исходным фрагментом, и рассмотреть функцию:

$$S(x, y) = \sum_{(u,v)\in U} w(u,v)(I(u+x, v+y) - I(u,v))^2$$

Функция $I(u+x, v+y)$ может быть разложена в ряд Тейлора в окрестности центра (u, v), что позволяет перейти к выражению:

$$S(x, y) \approx \sum_{(u,v)\in U} w(u,v)(I_x(u,v)x - I_y(u,v)y)^2$$

где: I_x и I_y — частные производные яркости в горизонтальном и вертикальном направлениях.

Последнее выражение можно записать в матричном виде:

$$S(x, y) \approx (xy)M\begin{pmatrix} x \\ y \end{pmatrix}$$

где: $M = \sum_{(u,v)\in U} w(u,v)\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$ — матрица локальной структуры.

В качестве весовой функции $w(u, v)$ обычно используется функция Гаусса. Угол характеризуется большими изменениями функции $S(x, y)$ по всем возможным направлениям (x, y), что эквивалентно большим по модулю собственным значениям матрицы M.

Отсюда следует ряд выводов:

1. Если собственные значения λ_1 и λ_2 близки к нулю, то пиксель с центром в (x, y) не является точкой интереса, поскольку он лежит в однородной области.

2. Если $\lambda_1 \approx 0$, а λ_2 принимает большое по модулю значение, то пиксель (x, y) принадлежит краю.

3. Если оба собственных значения велики и принимают положительные значения, то пиксель (x, y) является углом.

Большинство операторов детектирования углов основаны на свойствах матрицы М. Харрисом и Стефенсом было предложено использовать меру отклика угла:

$$z(x, y) = \det(M) - k \times tr(M)^2$$

где: k — эмпирически найденный параметр порядка 0,04-0,06,

а $\det(M)$ и $tr(M)$ — определитель и след матрицы.

При отрицательном отклике точка классифицируется как попавшая на край; при отклике, близком к нулю, точка считается попавшей в «плоскую» область. При больших положительных значениях $z(x, y)$ считается, что точка является углом, так как в ней яркость сильно меняется во всех направлениях. Детектор Харриса инвариантен к вращению и сдвигу изображения, а также к сдвигу и равномерному линейному изменению яркости.

Глава 2. Практическая часть

2.1 Описание алгоритма

2.1.1 Этап парсинга изображений

Первая задача, появляющаяся при решении задачи автоматической сборки - отделение информации о фрагментах паззла от исходных изображений. Классическое решение задачи поиска объектов на изображении, метод поиска шаблона, ищет уже известный фрагмент на изображении; он не подходит в данной ситуации по причине большого разнообразия искомых фрагментов как по цветовому наполнению, так и по форме. Методы, основывающиеся на сопоставлении характерных точек, также требуют заранее известный объект и отличаются от метода поиска шаблона устойчивостью к повороту и искажениям формы искомого объекта.

Для получения извлечения фрагментов паззла проанализируем информацию о границах на исходных изображениях. Идея заключается в том, что фрагмент паззла будет представлять из себя замкнутый контур достаточно большого размера, который может заключать внутри себя другие контуры в зависимости от того, что изображено на фрагменте; все же объекты, которые не являются фрагментами паззла, будут иметь значительно меньший размер и/или разомкнутый контур.

Предварительная обработка

Все методы нахождения границ очень зависимы от уровня зашумленности исходного изображения - пиксели, подвергшиеся влиянию шума, сильно отличаются от окружающих пикселей изображения, что может повлечь за собой ложный отклик детекторов границ. Другой причиной появления нежелательных границ может стать несовершенство оборудования, производившего захват изображения, или алгоритмов обработки, применявшихся к исходному изображению(при захвате изображения оборудованием, при сохранении на диск, сжатии и др.), в результате чего на изображении может появиться множество мелких зон

близкого цвета имеющих четкую границу, которых не было на исходном объекте.

Предварительная обработка изображения позволяет сгладить шумы и незначительные перепады цвета, значительно повысив точность определения границ на изображении.

Для данной задачи необходимо отобрать фильтры, помогающие достичь следующих результатов в автоматическом режиме:

- снизить зашумленность исходного изображения
- снизить количество ложных границ
- объединить смежные регионы с похожими цветами
- уменьшить или полностью ликвидировать артефакты
- максимально сохранить истинные границы объекта и форму объекта.

Исходя из этих требований были выбраны преобразования, позволяющие достичь необходимого эффекта:

- фильтр Гаусса
- билатеральный фильтр
- медианный фильтр

Для применения фильтров в автоматическом режиме необходимо подобрать наиболее эффективные значения параметров. Поскольку размеры изображений могут меняться в очень широком диапазоне, параметры могут принимать зависимость от этих размеров.

Правильное применение фильтра Гаусса к изображению положительно сказывается на качестве определения границ - размытие сглаживает границы между близкими по цвету зонами, уменьшая ложный отклик детектора границ. Но размытие также негативно сказывается и на четких границах, приближая цвет соседних пикселей к цвету границ. Поэтому, размер ядра следует выбирать достаточно небольшим. Экспериментально размер ядра для

фильтра Гаусса показал лучшие результаты при размерах ядра около 4-8% от меньшего измерения изображения.

Применение медианного фильтра на размытых изображениях дает ощутимо меньшую фильтрацию мелких шумов из-за усреднения цвета шума и фона размытием. Это объясняется принципом работы медианного фильтра - выбирая медиану среди рассматриваемых фильтром значений, подавляются шумы, сильно отличающиеся по цвету от большинства пикселей в рассматриваемой области. Размытие уменьшает эту разницу, делая переходы цветов более плавными. Поэтому, медианный фильтр следует применять до фильтров дающих эффект размытия, чтобы добиться максимального подавления мелких шумов. Но после его применения может остаться множество небольших областей с резкими краями, которые будут обнаруживаться детектором границ. Также нужно помнить о том, что медианный фильтр сильнее всего из представленных фильтров разрушает тонкие и слабые границы. Следовательно, нужно выбрать настолько маленький размер ядра, насколько это возможно. Эмпирическим путем было установлено, что размер ядра, равный 3-7% от меньшего измерения изображения(длина/ширина), дает оптимальный результат.

Билатеральный фильтр по своему визуальному эффекту напоминает медианный фильтр. Он позволяет избавиться от шумов, сгладить локальные перепады цветов. Оптимальными параметрами для него будут: размер ядра – 10-15% от меньшего измерения, σd и σr в пределах 80-130. Из-за высоких вычислительных затрат, а также необходимости ядер большего размера, по сравнению с медианным и гауссовским фильтрами, билатеральный фильтр не применяется в алгоритме (но имеет хороший потенциал для его применения).

Экспериментальным путем были установлены последовательности применения фильтров, дающие наилучший эффект(все фильтры

применяются с параметрами, отвечающим определенным выше интервалам значений параметров фильтров):

1. Билатеральный фильтр → медианный фильтр;
2. Фильтр Гаусса → медианный фильтр;

Изображения, получающиеся в результате применения этих последовательностей фильтров, очень похожи и зачастую не имеют значительных визуальных отличий. Поэтому, учитывая высокие вычислительные затраты билатерального фильтра, выбор был сделан в пользу последовательности 2. В описываемом в этой работе алгоритме фильтрация происходит в автоматическом режиме с параметрами:

- размер ядра фильтра Гаусса: 4% от меньшего измерения изображения;
- размер ядра медианного фильтра: 3% от меньшего измерения изображения.

Комбинация медианного фильтра и фильтра Гаусса с указанными выше параметрами показала хорошие результаты по удалению точечного шума и небольших артефактов. Но, хоть предварительная обработка изображений и позволяет значительно снизить уровень точечного шума, проблема крупных артефактов требует другого подхода. Ситуация усугубляется тем, что крупные артефакты в большинстве случаев получают сильный отклик от детекторов границ, чем сильно отличается от мелкого шума, который может искажать результат, но чаще всего или не распознаётся как граница на изображении, или же отбрасывается как незначительная граница в 1-2 пикселя.

В рассматриваемом в этой работе алгоритме происходит отсеивание крупных артефактов, не имеющих общих границ с фрагментами и имеющих площадь минимального обрамляющего соосного прямоугольника менее 60% от средней площади всех таких прямоугольников. Данный подход имеет очень малые вычислительные затраты, а также гарантированно избавляется от артефактов, соответствующих описанным выше критериям. Большинство

артефактов на практике имеют площадь не более 20% от площади рассматриваемых фрагментов, что позволяет успешно применять описанные выше критерии. Отсеивание артефактов, имеющих общие границы с фрагментом, является обширной задачей, требующей разработки сложных алгоритмов. Проблема осложняется многими факторами: отсутствие достоверной информации о изображении на фрагментах; как изображение на фрагменте, так и артефакт могут иметь одинаковые текстуры; полное отсутствие информации о закрытой артефактом части фрагмента. Все эти факторы негативно сказываются на алгоритмической и вычислительной сложности, точности определения и уровне искажений исходных данных.

Выделение границ

Произведя предварительную обработку изображения, можно перейти к непосредственному выделению границ на изображении. Существует большое количество методов обнаружения границ на изображениях, самые известные из них описаны в теоретической части. Детектор границ Канни, который в настоящее время является распространенным решением задачи обнаружения границ, представляет из себя комплексное решение, объединяющее элементарную предобработку, вычисление градиента и извлечение из него информации о расположении границ. Также, преимущество алгоритма Канни состоит в широких возможностях по его настройке, в то время как большинство популярных алгоритмов представляют собой лишь фиксированные ядра(в т.ч. фиксированного размера) для нахождения градиента.

По описанным выше причинам, детектор границ Канни нашел применение в этапе обнаружения границ алгоритма сборки.

В качестве входных данных детектор границ Канни принимает изображение в оттенках серого, характеризующееся равенством цветовых компонент в RGB представлении. Соответственно, имея цветные изображения, их необходимо преобразовать для использования в детекторе

границ Канни. Классический способ преобразования цветных изображений в изображения в градациях серого заключается в вычислении нового значения для каждого пикселя по заданной формуле. Единого мнения насчет выбора этой формулы нет, одними из самых распространенных являются:

- формула, основанная на особенностях восприятия цвета человеком

$$Y' = 0.2126R + 0.7152G + 0.0722B$$

- формула стандартов PAL и NTSC

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

- формула, приближенно учитывающая уровень освещенности

$$Y' = (\max(R,G,B) + \min(R,G,B))/2$$

- усреднение значений RGB

$$Y' = (R + G + B)/3$$

Объединяет эти формулы только одно - невозможность обратного преобразования, следовательно - потеря информации о цвете. В результате преобразований участки, имевшие ярко выраженную границу на цветном изображении, могут получить близкие значения интенсивности, что приведет к потере отклика детектора границ.

Решение этой проблемы состоит в разделении исходного изображения на несколько изображений в оттенках серого. На практике чаще всего применяется разделение на каналы RGB или HSV. После чего к каждому полученному изображению применяется детектор границ. В алгоритме, описанном в данной работе, применяется разделение на каналы RGB; причиной тому более широкое распространение формата RGB для хранения исходных данных(меньшие вычислительные затраты на приведение изображений в нужный формат), а также одинаковая природа описываемых каждым каналом свойств(возможность использовать подобранные параметры для детектора границ на всех трёх каналах, меньшая вероятность ошибок и снижения точности по причине неверно подобранных параметров).

После применения к каждому каналу детектора границ необходимо объединить полученные бинарные изображения; описываемый в данной работе алгоритм, например, использует для этого попиксельное ИЛИ.

Результатом применения большинства детекторов границ является изображение в оттенках серого, описывающих выраженность границ в каждой точке изображения; детектор Канни на этом не останавливается и содержит в себе алгоритм извлечения информации о границах, включающий в себя подавление слабых границ и извлечение сильных границ. При этом детектор Канни приводит все границы к толщине в 1 пиксель и возвращает в качестве результата бинарное изображение, каждый пиксель которого содержит информацию о том, является ли соответствующий пиксель исходного изображения граничным.

Получение масок фрагментов

После применения детектора границ Канни из результирующего изображения необходимо извлечь информацию об отдельных контурах, определить принадлежность границ к тем или иным объектам. На данном этапе, информация о границах одного фрагмента представляет из себя неупорядоченный набор границ, без указания их вложенности, без упорядочивания точек для упрощения представления и обхода. Для получения маски фрагмента необходимо получить только самую внешнюю границу. Простой обход по самой внешней границе не дал необходимого результата из-за невозможности гарантировать замкнутость контура - часто присутствуют разрывы в 1-3 пикселя. Пакет OpenCV поддерживает извлечение информации об иерархии границ с множеством уровней вложенности, но на практике показал неудовлетворительные результаты - он аналогично простым методам зависим от разрывов в обрабатываемых границах, а механизм иерархий показывает удовлетворительные результаты лишь тогда, когда вложенные границы являются замкнутыми, что не может быть гарантировано. Для решения поставленной задачи было решено использовать подход, не включающий в себя обход границ в любом виде. К

изображению применяется операция dilate для утолщения границ, закрытия разрывов и сглаживания дефектов. Необходимо сохранить копию полученного изображения. Затем, производится внешняя заливка изображения цветом, совпадающим с цветом контура. Заливка не пройдет через внешнюю границу, закрасив лишь области, не принадлежащие фрагменту. Для получения маски необходимо лишь убрать внутренние границы. Для этого из полученного изображения вычитаем сохраненную копию. В результате, получаем изображение, на котором изображена маска фрагмента в черном цвете. Инвертируем изображение для получения маски в белом цвете на черном фоне, требуемой для дальнейшей обработки, а также применим операцию erode для компенсации сделанной в начале обработки операции dilate. К полученной маске можно применять методы, основанные на обходе границ, поскольку границы такой маски всегда будут неразрывными.

Детектирование углов

Полученная маска изображения используется в качестве входных данных для детектора углов. Маска является лучшим выбором в данном случае, так как содержит лишь информацию о форме внешних границ объекта, исключая возможность ложных срабатываний внутри объекта. Тем не менее, остаётся вероятность ложного(относительно задачи нахождения граничных точек сторон фрагмента) срабатывания детектора. Детектор углов Харриса основывается на анализе производных яркости изображения и, по сути, реагирует на любые области, так или иначе похожие на угол. Решение этой проблемы достаточно простое - наложим ограничения на минимальное расстояние между углами, тем самым выбирая отклики, расположенные не ближе заданного расстояния; также будем отбирать только 4 угла, критерием для их отбора будет служить максимизация суммарного значения отклика 4 рассматриваемых углов. Такой критерий выбран по причине того, что искомые углы имеют сильный отклик из-за максимального сходства с прямым углом, отклик детектора углов в большинстве неподходящих точек

значительно слабее. Эмпирическим путём было установлено, что наиболее эффективным является ограничение минимального расстояния в 70% от наименьшего измерения минимального соосного обрамляющего прямоугольника фрагмента. Параметр k детектора границ Харриса по умолчанию принимает значение 0.04.

Оценка правильности детектирования углов

Стандартный набор параметров для детектора углов не всегда дает правильные результаты, особенно это проявляется на фрагментах, имеющих нестандартные относительно других фрагментов размеры(удлиненные, трапециевидные). Для таких деталей могут потребоваться другие значения параметров для достижения лучших результатов.

В первую очередь после применения детектора углов со стандартными параметрами проверяется количество найденных углов: если их менее 4, сразу переходим к поискам более подходящих параметров, иначе оцениваем правильность нахождения углов.

Для того, чтобы оценить правильность нахождения углов, используется мера прямоугольности. Мера прямоугольности представляет из себя отношение площади четырехугольника, построенного на 4 точках, найденных детектором границ, и площади минимального обрамляющего прямоугольника, построенного для тех же 4 точек. Идея заключается в том, что четырехугольник. построенный на правильно найденных углах, будет иметь форму, близкую к прямоугольнику; в то же время, если один из углов расположен неправильно, мера прямоугольности будет иметь низкое значение. Определенное экспериментальным путем пороговое значение меры прямоугольности составило 0.85.

Если значение меры прямоугольности оказывается меньше порогового значения, необходимо произвести поиск более подходящих параметров. Для этого, определим допустимые интервалы значений каждого из параметров, затем выберем на этом интервале некоторое количество равноотстоящих друг от друга и от границ интервала значений. Затем, произведем

детектирование углов для всех возможных комбинаций значений, полученных на предыдущем шаге, т.н. поиск по сетке. Результат с лучшей мерой прямоугольности используем для дальнейшей обработки. Если значение меры прямоугольности в процессе поиска не превысит порогового значения, отметим данный фрагмент как потенциально ошибочный и предоставим пользователю возможность вручную разметить углы или же подтвердить правильность работы детектора.

Интервалы значений параметров, используемые в алгоритме:

минимальное расстояние: 60-80%

k: 0.04-0.06

2.1.2 Алгоритм этапа обнаружения границ

Для каждого исходного изображения:

1. применить медианный фильтр
2. применить фильтр Гаусса
3. разделить исходное изображение на 3 изображения в градациях серого, каждое из которых содержит информацию об одном из каналов исходного изображения
4. применить к каждому полученному изображению детектор границ Канни
5. совместить 3 изображения, полученных на шаге 4
6. применить к полученному изображению операцию dilate
7. произвести внешнюю заливку изображения цветом, совпадающим с цветом контура
8. вычесть из изображения, полученного на шаге 7, изображение шага 6
9. применить к полученному изображению операцию dilate
10. применить к полученному изображению детектор границ Канни
11. извлечь из полученного изображения информацию о границах
12. вычислить среднюю площадь минимальных соосных обрамляющих прямоугольников для каждого полученного контура
13. для каждого контура:

13.1. если площадь минимального соосного обрамляющего прямоугольника меньше чем половина средней площади – не обрабатывать контур; иначе продолжить
13.2. создать изображение размером, немного большим по размеру, чем соответствующий минимальный соосный обрамляющий прямоугольник
13.3. залить его белым цветом
13.4. отрисовать черным цветом соответствующий контур
13.5. произвести внешнюю заливку изображения черным цветом
13.6. применить операцию erode к полученной маске
13.7. применить детектор углов Харриса
13.8. вычислить меру прямоугольности для полученных углов
13.9. если значение, полученное на шаге 13.8, меньше, чем 0.85 – произвести поиск по сетке значений параметров детектора углов Харриса, использовать лучший результат
13.10. отсортировать углы по порядку встречаемости на изображении
13.11. используя отсортированные углы, извлечь информацию о сторонах детали
13.12. найти центр масс для контура

2.1.3 Этап постобработки

На данном этапе необходимо обработать полученные из этапа парсинга данные о форме сторон фрагмента. Обработка включает в себя приведение границ сторон к единому стандарту, их подготовка для последующего сравнения, классификация, а также вычисление метрик, используемых на этапе сборки.

Стандартизация сторон

Для начала, приведем все стороны к единому виду. Это упростит вычисление метрик, а также позволит быть уверенным в том, что все

значения были получены в одинаковых условиях. Помимо этого, станет возможным поточечное сравнение границ.

Потребуем такой ориентации границ, чтобы прямая, проходящая через крайние точки границы, была параллельна оси X, а отрезок минимальной длины, соединяющий центр масс фрагмента с прямой, находился ниже этой прямой. Такие требования дадут одинаковую ориентацию сторон относительно центра масс фрагмента. Найдем угол между описанной выше прямой и осью X, а также, построим отрезок минимальной длины. Повернем сторону и построенный отрезок вокруг центра масс фрагмента на найденный выше угол и проверим ориентацию отрезка. В случае, если отрезок лежит выше прямой, повернем границу на 180 градусов. Первый поворот гарантирует выполнение условия параллельности, второй, при необходимости, корректирует ориентацию стороны относительно центра масс.

Затем, произведя параллельный перенос, совместим крайнюю точку каждой стороны, имеющую наименьшее значение координаты X, с точкой начала отсчёта.

Описание и вычисление метрик

Приведя все стороны к единым требованиям их расположения и ориентации в пространстве, вычислим метрики для сравнения сторон. Метрики делятся между тремя этапами по порядку их применения в этапе сборки.

1) *«Быстрый» этап*

“Быстрый” этап включает в себя такие метрики как:

- длина стороны в пикселях

Благодаря требованию параллельности оси X прямой, соединяющей крайние точки, а также приведению одной из крайних точек к началу координат, в результате чего прямая совпадет с осью X, вычисление метрики сводится к определению того, какая из двух крайних точек не расположена в

начале координат. Значение метрики будет равняться значению координаты X этой точки.

- тип стороны(выпуклая, вогнутая, плоская)

Определяется максимальным отклонением границы от оси X по оси Y. Если отклонение невелико, в пределах ~3% от длины стороны, можно с уверенностью говорить о том, что эта сторона является плоской и принадлежит рамке паззла, иначе - можем определить является сторона выпуклой или вогнутой по направлению отклонения(выше/ниже оси X)

- расстояние по оси X до начала соединительной части фрагмента(для выпуклых/вогнутых сторон)

Для вычисления этой метрики пройдем последовательно по всем точкам границы, начиная с крайней точки, расположенной в начале координат. Координата X первой точки, значение координаты Y которой превысит пороговое значение в 5% от длины стороны, будет являться значением метрики.

- расстояние по оси X до конца соединительной части фрагмента(для выпуклых/вогнутых сторон)

Для вычисления этой метрики пройдем последовательно по всем точкам границы, начиная с точки, определённой при вычислении предыдущей метрики. Координата X первой точки, значение координаты Y которой упадёт ниже порогового значения в 4.5% от длины стороны, будет являться значением метрики.

2) *«Медленный этап»*

“Медленный” этап включает в себя попиксельное сравнение двух границ. Для каждой точки одной границы вычисляется минимальное расстояние от рассматриваемой точки до точек другой границы. Среднее значение всех найденных таким образом расстояний является значением метрики.

Вычисления “медленного” этапа производятся только на этапе сборки, после предварительного отбора кандидатов при помощи метрик “быстрого” этапа из-за крайне высоких вычислительных затрат. Но, ограниченное применение такой затратной метрики оправдывает себя высокой точностью её результатов - даже незначительные отличия в форме фрагментов заметно увеличивают значение метрики. Во время практического тестирования наблюдалась разница в 10-25% между значениями метрики у подходящих друг другу сторон и сторон, неправильное соединение которых человек был способен различить только рассмотрев место соединения с использованием источника яркого света.

3) *Этап сравнения цветов*

Третий этап использует цветовые характеристики сторон для их сравнения.

Для вычисления цветовой метрики отступим от границы внутрь фрагмента на небольшое расстояние, чтобы избежать захвата шумов/теней/искажений. Для этого воспользуемся имеющейся маской фрагмента. Применим к маске операцию эрозии; в результате, границы фрагмента сожмутся в сторону центра фрагмента. Найдем контур новой маски. Затем, найдем новые значения углов фрагмента. Новыми значениями углов будут ближайшие к старым углам точки на новом контуре. Теперь, разделив новый контур по новым углам на отдельные стороны, получим стороны фрагмента с учетом отступа внутрь фрагмента.

Для каждой полученной в результате отступа новой границы, возьмем некоторое количество точек, расположенных на одинаковом друг от друга расстоянии и вычислим среднее значение цвета в окрестности каждой точки. Последовательность значений цвета, полученная таким образом, будет являться цветовой метрикой и использоваться для сравнения сторон.

Вычисление описанных выше метрик заканчивает этап постобработки.

2.1.4 Алгоритм этапа постобработки

Для каждого <фрагмента>:

Для каждой <стороны>:

1. найти на линии, соединяющей два угла(начало и конец <стороны>), точку, ближайшую к центру масс
2. найти угол между прямой, соединяющей центр масс и точку, найденную в шаге 1, и вектором (0,1)
3. повернуть <сторону> на найденный угол
4. параллельным переносом сместить <сторону> так, чтобы точка с минимальным значением X оказалась в точке (0,0)
5. найти длину <стороны> в пикселях: учитывая шаг 4, искомое значение – координата X последней точки <стороны>
6. определить тип <стороны>: найти максимальное отклонение от оси Y; если величина отклонения составляет менее 3% от длины <стороны> - сторона принадлежит рамке, иначе тип зависит от знака величины: вогнутая сторона при знаке «+», выпуклая при знаке «-»
7. если <сторона> выпуклая – повернуть границу на 180 градусов и выполнить аналогичные шагу 4 действия
8. если <сторона> не принадлежит рамке, необходимо вычислить два признака – расстояние по оси X до начала и до конца соединительной части фрагмента; для всех точек <стороны>, начиная с (0,0):
    8.1. если отклонение по оси Y составляет более 5% от длины детали и расстояние до начала соединительной части еще не найдено, установить расстояние до начала соединительной части равное координате X рассматриваемой точки
    8.2. если отклонение от оси Y составляет менее 4,5% от длины детали и расстояние до начала соединительной части

найдено, установить расстояние до конца соединительной части равное координате X рассматриваемой точки

9. Извлечь информацию о цвете <фрагмента> вдоль <стороны>:
    9.1. разделить <сторону> на 10 равных частей(равных по количеству точек)
    9.2. в каждой точке, разделяющей части, высчитать средний цвет в ядре размера 5x5

2.1.5 Этап сборки

Задача этапа сборки состоит в использовании полученной ранее информации для нахождения подходящих друг другу фрагментов.

Этап состоит из двух частей: первая находит подходящие друг другу элементы и генерирует инструкцию по сборке, вторая использует полученную инструкцию для генерации результирующего изображения.

Критерии сравнения сторон фрагментов

То, насколько хорошо стороны подходят друг к другу, оценивается с помощью метрик этапа постобработки. Как уже было сказано, оценка состоит из трех этапов.

Задача «быстрого» этапа состоит в том, чтобы отбросить как можно больше заведомо неподходящих сторон, затратив при этом минимальное количество времени. Стороны различной длины, стороны, не подходящие по типу, стороны с различным расположением соединительных элементов - все такие стороны быстро и эффективно выходят из рассмотрения после «быстрого» этапа. Сначала сравниваются типы - вогнутой стороне подойдет только выпуклая и наоборот. Если пройдена проверка типа, критерием для принятия решения служит сравнение суммарного штрафа с пороговым значением. Для вычисления штрафа необходимо разность между одинаковыми метриками возвести в квадрат и суммировать все штрафы в текущем сравнении. Возведение в квадрат позволяет значительно увеличить

штраф для больших расхождений. Пороговое значение вычисляется как $4 \cdot (0.03 \cdot L_{MAX})^2$, где L_{MAX} - максимальная длина стороны среди всех рассматриваемых сторон.

Задача «медленного» этапа - произвести попиксельное сравнение границ, дающее хорошую оценку схожести формы двух сторон, но требующее значительных вычислительных затрат. Отсеивание заведомо неподходящих сторон позволяет значительно сократить затраты. Критерием отбора на данном этапе является минимальное значение метрики.

Если находятся две и более стороны, у которых значения, вычисленные на медленном этапе отличаются менее, чем на 10%, то невозможно выбрать какую-либо из них с достаточной уверенностью. В таком случае, происходит переход к третьему этапу - сравнению цветовой метрики. Этот этап позволяет оценить, насколько согласованы цвета вдоль рассматриваемых сторон. Критерий выбора - минимальное значение штрафа. Штраф вычисляется как суммарное евклидово расстояние между элементами массивов цветовой метрики с одинаковыми индексами; элементы массива, представляющие собой упорядоченный набор чисел, необходимо рассматривать как точки в пространстве соответствующей размерности.

Также, стоит отметить что использование информации о цвете не показало на практике достаточной точности для использования в качестве первичных признаков для принятия решения о совместимости сторон. Информация о цвете оказалась бесполезна на однотипных участках; также, цветовые метрики при своем использовании в качестве первичных отбирали слишком большое количество кандидатов и часто отбрасывали верные решения в пользу неправильных на участках с большим числом небольших цветных зон.

Генерация инструкции по сборке

Рассмотрев по каким критериям происходит принятие решения о том, подходят ли фрагменты друг другу, перейдем к непосредственной сборке паззла, а именно к “текстовой” сборке.

Порядок сборки изображения во многом схож с принципом, по которому люди собирают паззл вручную.

Сначала собирается рамка изображения(крайние фрагменты в каждом столбце/строке). Определить фрагменты, относящиеся к рамке можно без труда - одна или две их стороны определяются как “плоские” метрикой типа стороны. Из набора фрагментов выбирается фрагмент, имеющий две плоские стороны, затем он располагается в вернем левом углу. Справа от него будем последовательно располагать подходящие друг другу элементы до тех пор, пока очередным подходящим фрагментом не станет фрагмент с двумя плоскими сторонами. Установка этого элемента закончит верхнюю строку. Аналогичным образом собирается остальная рамка.

После сборки рамки приступим к сборке остального изображения. Для этого последовательно будем проходить по всем местам, для которых фрагменты ещё не найдены и при этом они граничат с уже найденными фрагментами сверху/снизу/слева/справа. Таким образом, под рассмотрение попадут пустые места, к которым мы можем попытаться найти подходящий фрагмент, используя границы фрагментов, соприкасающихся с этим пустым местом.

Сравнивая совместимость соприкасающихся с пустым местом границ и границ оставшихся фрагментов-кандидатов, найдём подходящие фрагменты. Если после всех трёх этапов сравнения значения метрик не позволяют уверенно выделить подходящий фрагмент из кандидатов, следует пропустить рассматриваемое пустое место и вернуться к нему позже, после рассмотрения других пустых мест. Спорные моменты могут быть разрешены в результате получения дополнительных возможностей для сравнения, когда вокруг спорного места будут установлены дополнительные фрагменты или же в результате сокращения множества кандидатов как следствие установки фрагментов в другие пустые места.

В результате, для каждого фрагмента будет найдено соответствующее ему местоположение. Сценарий, в котором на некоторые пустые места не находятся подходящие фрагменты, является аварийным и предусматривает два варианта решения:

- завершение работы приложения с выводом всех сгенерированных данных(в т.ч. незавершённая инструкция по сборке)
- выбор фрагментов с лучшим показателем метрики вне зависимости от того, выполняется ли условие различия значений метрик более чем на 10%

Если приложение не было аварийно остановлено на предыдущем шаге, это означает, что приложение сгенерировало текстовую инструкцию по сборке; происходит переход к сборке итогового изображения.

Сборка итогового изображения

Генерируется новое изображение, залитое чёрным цветом, на нем будут располагаться фрагменты паззла. Затем, последовательно проходим по инструкции по сборке, извлекая указанные в ней фрагменты. Для каждого фрагмента:

- извлекаем изображение этого фрагмента
- получаем информацию о каком-либо соседнем фрагменте, уже установленном на финальное изображение; исключение составляют фрагменты, относящиеся к рамке, ориентиром для них становятся координатные оси
- определяем угол поворота как угол между прямыми, проходящими через крайние точки сторон, которые были определены как подходящие друг другу. Для элементов рамки - угол между плоской стороной и осью X или Y.
- поворачиваем фрагмент на найденный угол

- проверяем ориентацию фрагмента - вычислим координаты центра масс позиционируемого фрагмента при условии совмещения крайних точек сторон, которые были определены как подходящие друг другу. Если центр масс рассматриваемого фрагмента и центр масс фрагмента, относительно которого происходит позиционирование, располагаются по разные стороны прямой, соединяющей крайние точки сторон - фрагмент повернут правильно, иначе необходим поворот на 180 градусов
- производим фактический перенос изображения фрагмента на результирующее изображение, совместив при этом крайние точки сторон, которые были определены как подходящие друг другу

2.1.6 Алгоритм этапа сборки

1. Для каждого <фрагмента>
   1.1. проверить, имеет ли фрагмент 2 плоские стороны, если да - записать идентификатор фрагмента и номер стороны, которая будет ориентирована вверх, в верхний левый угол результирующей матрицы и перейти к шагу 2
2. Пока установленный фрагмент имеет менее 2 плоских сторон
   2.1. Найти сторону, которая будет ориентирована вправо, у последнего расположенного фрагмента
   2.2. Для каждого <фрагмента>
      2.2.1. Для каждой <стороны>
         2.2.1.1. Если <сторона> плоская - сравнить тип стороны, следующей за <стороной> по направлению против часовой стрелки, с типом <стороны>. Если они различны - перейти к следующему шагу, иначе немедленно перейти к следующей <стороне>

2.2.1.2. Вычислить штрафы “быстрого” шага. Если суммарный штраф меньше чем $4 \cdot (0.03 \cdot L_{MAX})^2$, где L_{MAX} - максимальная длина стороны среди всех рассматриваемых сторон - добавить информацию о стороне и содержащем ее фрагменте в массив кандидатов

2.2.2. Если кандидатов ровно 1 - перейти у шагу 2.2.9, иначе к шагу 2.2.3

2.2.3. Для всех кандидатов вычислить критерий “медленного” шага.

2.2.4. Отсортировать кандидатов по возрастанию значения критерия.

2.2.5. Удалить из массива кандидатов, значение критерия у которых превышает значение первого кандидата более чем на 10%. Если после этого останется лишь один кандидат - перейти к шагу 2.2.9

2.2.6. Для всех кандидатов вычислить критерий шага сравнения цвета по каждой смежной стороне

2.2.7. Отсортировать кандидатов по возрастанию значения критерия.

2.2.8. Удалить из массива кандидатов, значение критерия у которых превышает значение первого кандидата более чем на 10%. Если после этого останется более одного кандидата - удалить всех кандидатов из массива, кроме кандидата с наименьшим значением параметра

2.2.9. Информацию об оставшемся в массиве кандидате поместить в результирующую матрицу правее последнего записанного результата

3. Повторить пункт 2 с направлением движения вниз, формируя столбец рамки

Изменения:

п. 2.1 - Найти сторону, которая будет ориентирована вниз

п. 2.2.1.1 - следующей за <стороной> по направлению по часовой стрелке

п. 2.2.9 - поместить в результирующую матрицу ниже

4. Повторить пункт 2 с направлением движения вниз, начиная с верхнего правого элемента, формируя столбец рамки

   Изменения:

   п. 2.1 - Найти сторону, которая будет ориентирована вниз

   п. 2.2.9 - поместить в результирующую матрицу ниже

5. Повторить пункт 2 с направлением движения вправо, начиная с нижнего левого элемента, формируя столбец рамки

   Изменения:

   Условие остановки - достижение нижнего правого элемента, установленного на шаге 4

6. Пока не заполнена результирующая матрица

   6.1. Для каждого незаполненного <элемента> результирующей матрицы

   6.1.1. Если сверху/снизу/слева/справа от <элемента> есть заполненный элемент

   6.1.1.1. Для каждого заполненного элемента

   6.1.1.1.1. Для каждого <фрагмента>

   6.1.1.1.1.1. По типам сторон заполненных элементов выяснить, возможно ли разместить на рассматриваемом месте <фрагмент>. Если невозможно, перейти к рассмотрению следующего <фрагмента>.

   6.1.1.1.1.2. Вычислить штрафы “быстрого” шага. Если суммарный штраф меньше чем количество известных сторон, смежных с незаполненным <элементом>, умноженное на $4 \cdot (0.03 \cdot L_{MAX})^2$, где L_{MAX} - максимальная длина стороны среди всех рассматриваемых сторон - добавить информацию о стороне и содержащем ее фрагменте в массив кандидатов

6.1.1.1.1.3. Если кандидатов ровно 1 - перейти у шагу 2.2.9, иначе к шагу 2.2.3

6.1.1.1.1.4. Для всех кандидатов вычислить критерий “медленного” шага по каждой смежной стороне и просуммировать.

6.1.1.1.1.5. Отсортировать кандидатов по возрастанию значения критерия.

6.1.1.1.1.6. Удалить из массива кандидатов, значение критерия у которых превышает значение первого кандидата более чем на 10%. Если после этого останется лишь один кандидат - перейти к шагу 2.2.9

6.1.1.1.1.7. Для всех кандидатов вычислить критерий шага сравнения цвета по каждой смежной стороне и просуммировать

6.1.1.1.1.8. Отсортировать кандидатов по возрастанию значения критерия.

6.1.1.1.1.9. Удалить из массива кандидатов, значение критерия у которых превышает значение первого кандидата более чем на 10%. Если после этого останется более одного кандидата - удалить всех кандидатов из массива, кроме кандидата с наименьшим значением параметра

6.1.1.1.1.10. Информацию об оставшемся в массиве кандидате поместить в результирующую матрицу

7. Сгенерировать пустое изображение для размещения итогового изображения
8. Для всех <элементов> верхней строки результирующей матрицы
   8.1. Получить фрагмент, соответствующий идентификатору, записанному в <элементе> результирующей матрицы

8.2. Вычислить угол между прямой, проведенной через крайние точки стороны, ориентированной вверх, и осью X.

8.3. Повернуть фрагмент на найденный угол.

8.4. Проверить, располагается ли центр масс фрагмента ниже стороны, ориентированной вверх, если нет - повернуть фрагмент на 180 градусов

8.5. Поместить фрагмент на итоговое изображение, совместив крайнюю левую точку размещаемого фрагмента с крайней правой точкой последнего размещенного фрагмента

9. Для всех строк результирующей матрицы, начиная со второй

9.1. Для всех <элементов> строк результирующей матрицы

9.1.1. Получить <фрагмент>, соответствующий идентификатору, записанному в <элементе> результирующей матрицы

9.1.2. Вычислить угол между прямой, проведенной через крайние точки стороны <фрагмента>, ориентированной вверх, и прямой, проведенной через крайние точки стороны, ориентированной вниз, фрагмента, расположенного выше

9.1.3. Повернуть <фрагмент> на найденный угол.

9.1.4. Проверить, располагается ли центр масс <фрагмента> ниже стороны, ориентированной вверх, если нет - повернуть фрагмент на 180 градусов

9.1.5. Поместить <фрагмент> на итоговое изображение, совместив крайнюю левую точку размещаемого фрагмента с крайней левой точкой фрагмента, расположенного выше

10. Сохранить результирующую матрицу и итоговое изображение в выходные файлы

2.2 Реализация алгоритма

2.2.1 Используемые технологии

Для решения задачи был выбран язык программирования C++. Причинами такого выбора стали:

- скорость работы - C++ является одним из лучших языков по скорости своей работы, благодаря предоставлению разработчикам доступа к низкоуровневым возможностям, а также минимальному количеству ограничений для разработчиков;
- работа с памятью - большинство современных языков, отличных от C++, не позволяют разработчику(либо позволяют с ограничениями) получить прямой доступ к областям оперативной памяти, в которой хранятся данные, используемые программой; прямой доступ к памяти, а также возможность управления процессами чтения и записи позволяют разработчику оптимизировать работу с массивными данными(в т.ч. изображениями)
- широкий выбор библиотек - C++ получил широкое распространение среди разработчиков и развивается на протяжении более 20 лет, следствием этого является большое количество наработок и доступных библиотек

Основными библиотеками, используемыми в процессе разработки являются GTEngine 2.1, Qt5 и OpenCV 2.4.9.

Qt5 - хорошо зарекомендовавшая себя в профессиональной разработке библиотека для отрисовки пользовательских интерфейсов.

GTEngine предоставляет множество вспомогательных функций, производящих геометрические преобразования, реализацию алгебраических алгоритмов, а также структуры данных для поддержки этих операций.

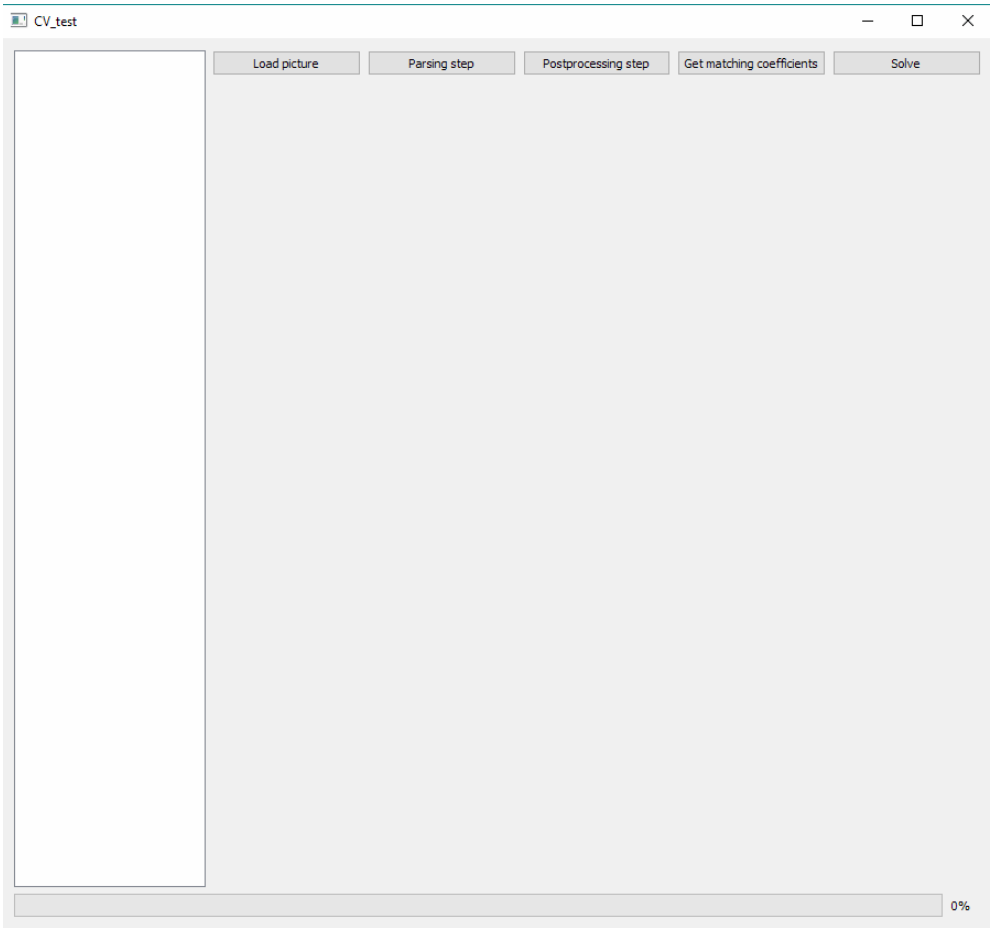
OpenCV является широко распространенной библиотекой алгоритмов компьютерного зрения и обработки изображений. Содержит в себе реализации детекторов характерных точек, детекторов границ, часто применяемых фильтров, а также инструментов для создания своих фильтров, инструментов обработки видеопотоков, инструментов для обучения

нейронных сетей и т.д. Также, OpenCV активно поддерживается сообществом, получая регулярные обновления, а также поддерживается многими крупными компаниями, позволяя произвести низкоуровневую оптимизацию библиотеки.

2.2.2 Реализация

Описанный в данной главе алгоритм был реализован в приложении, интерфейс которого можно увидеть на изображении ниже.

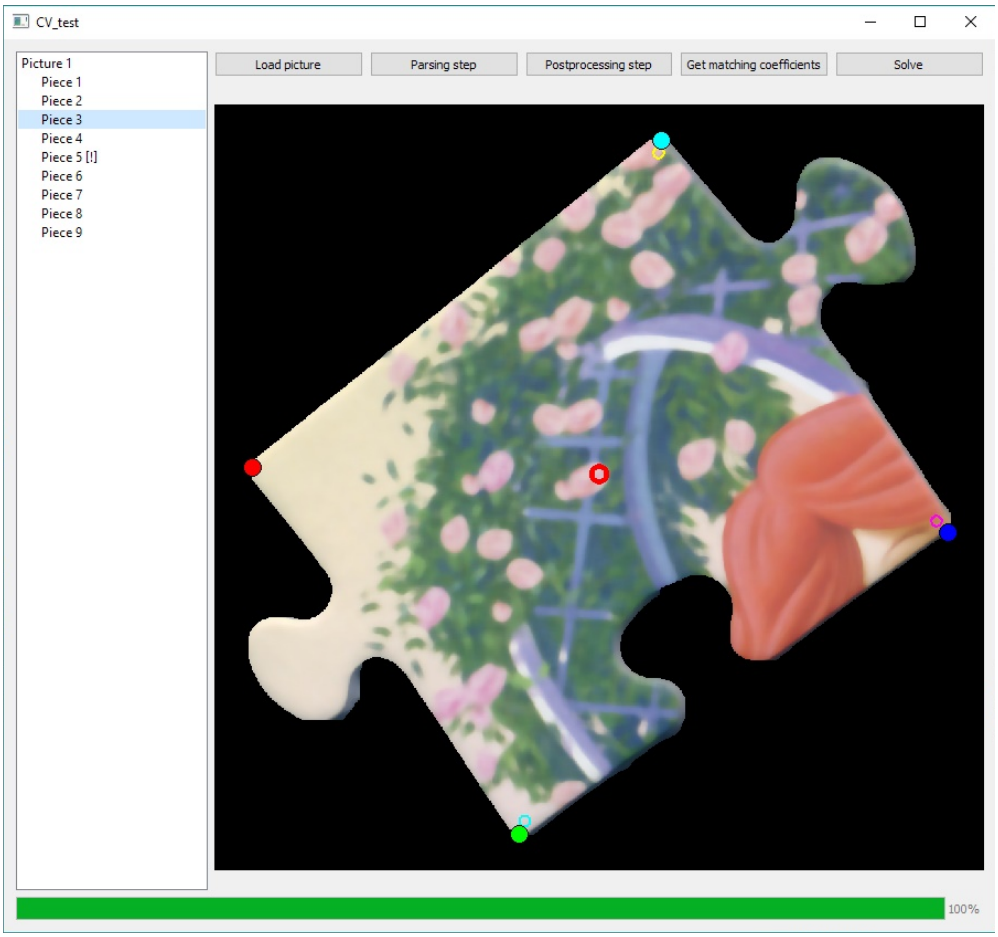
Интерфейс приложения состоит из списка в левой части (в нем отображается список загруженных исходных изображений, а также список извлеченных фрагментов для каждого исходного изображения), кнопок управления сверху, индикатора процесса выполнения снизу. Остальное пространство занимает рабочая область, где отображаются выбранные в списке исходные изображения и фрагменты.



Окно приложения после запуска

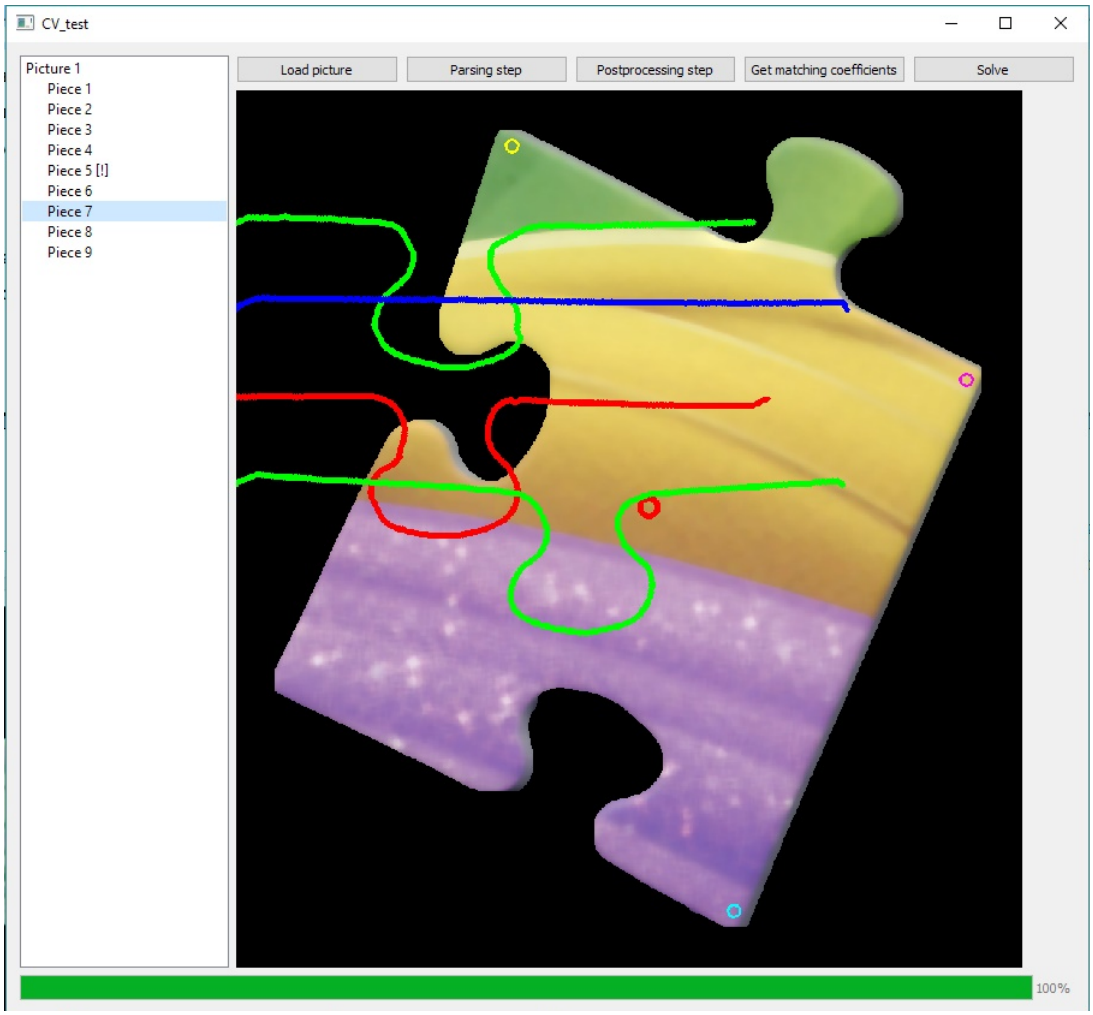
Приложение позволяет вручную запустить каждый этап алгоритма, предоставляя пользователю промежуточные результаты выполнения каждого этапа.

Этап парсинга предоставляет в качестве доступных промежуточных результатов изображения отдельных фрагментов, размеченные на фрагментах центры масс (красная окружность), найденные углы (круги различных цветов), а также предупреждения о срабатывании меры прямоугольности (пометка «[!]» в списке фрагментов), что сигнализирует о вероятных ошибках в детектировании углов. После завершения этапа парсинга пользователю предоставляется возможность отредактировать положение найденных алгоритмом углов. В приложении реализовано «прилипание» маркеров углов к границам фрагмента для исключения возможности передачи некорректных данных и удобства пользования.



Приложение после завершения этапа парсинга

Этап постобработки в качестве промежуточного результата отображает границы сторон фрагмента в стандартизированном виде, обозначая цветом тип границы (красный – выпуклая, зеленый – вогнутая, синий – плоская). Выделение цветом необходимо, поскольку и вогнутые, и выпуклые стороны ориентированы своим соединительным элементом в одну и ту же сторону, что не позволяет отличить их визуально.



Приложение после этапа постобработки

Этап сборки не добавляет никаких новых данных в приложение, результатом его работы, в случае его успешного завершения, являются файлы с инструкцией по сборке и итоговым изображением.

Список в левой части приложения позволяет выбрать для просмотра любое исходное изображение или найденный приложением фрагмент. Также, нумерация «номер исходного изображения - номер фрагмента» совпадает с

нумерацией фрагментов в инструкции по сборке, генерируемой приложением, что позволяет быстро оценивать полученные результаты.

Дополнительно, для тестирования метрик сравнения сторон, была реализована функция вывода результата сравнения всех сторон всех фрагментов друг с другом; эта функция доступна по нажатию на кнопку «Get matching coefficients».

Итоговый объем исходного кода приложения (без учета автоматически сгенерированного кода Qt5) составил 2000 строк

2.2.3 Тестирование приложения

Тестирование алгоритма производилось на трёх тестовых множествах(тестовых паззлах). Все исходные изображения для тестового множества были получены сканированием фрагментов с разрешением 400 DPI с выполнением всех требований из постановки задачи.

Алгоритм был испытан на имеющихся тестовых множествах; была собрана статистика качества работы шагов алгоритма, для которых есть возможность составить строгие критерии оценивания.

В качестве критериев для тестирования были выбраны:

- количество найденных фрагментов

  Для вычисления значения критерия производится просмотр всех найденных алгоритмом фрагментов и учет количества фрагментов, извлеченных алгоритмом.

- количество фрагментов, извлеченных без дефектов

  Для вычисления значения критерия производится просмотр всех найденных алгоритмом фрагментов, визуальное сравнение извлеченных фрагментов с исходным изображением и учет количества фрагментов, извлеченных алгоритмом без значительных дефектов границ и нарушения целостности фрагмента. Дефект считается значительным, если его площадь превышает 1% от площади фрагмента и/или одно из его измерений превышает 2% от минимального измерения фрагмента.

- количество правильно определенных углов

Для вычисления значения критерия производится просмотр всех найденных алгоритмом углов у всех фрагментов, визуальная оценка правильности расположения угла и учет всех правильно определенных углов.

- количество фрагментов без ошибок в определении углов

Для вычисления значения критерия производится оценивание точности расположения углов каждого фрагмента и учет всех фрагментов, не имеющих неправильно определенных углов.

- данные о точности определения ошибок детектирования углов

Данный критерий включает в себя количество истинно-положительных, истинно-отрицательных, ложно-положительных и ложно-отрицательных срабатываний критерия правильности определения углов.

Ниже приведена таблица с результатами тестирования:

Номер тестового множества	**Количество фрагментов**	**Количество найденных фрагментов**	**Количество фрагментов, извлеченных без дефектов**	**Количество правильно определенных углов**	**Количество фрагментов без ошибок в определении углов**	**Количество истинно-положительных правильности нахождения углов**	**Количество истинно-отрицательных правильности нахождения углов**	**Количество ложно-положительных правильности нахождения углов**	**Количество ложно-отрицательных правильности нахождения углов**
1	**30**	**30**	**30(100%)**	**120(100%)**	**30(100%)**	**0**	**29**	**1**	**0**
2	**54**	**54**	**48(89%)**	**181(84%)**	**41(76%)**	**13**	**41**	**0**	**0**
3	**104**	**104**	**95(91%)**	**329(79%)**	**62(60%)**	**32**	**62**	**0**	**10**

Программа успешно произвела сборку на тестовом множестве номер 1 в полностью автоматическом режиме(без внесения изменений в процессе работы). Для успешной сборки на тестовых множествах 2 и 3 потребовались корректировки в исходных данных – необходимо провести границу в местах, где части фрагмента сливаются с фоном или же получить изображение

фрагмента на другом фоне. Также, потребовались корректировки положения найденных углов.

Выводы

Анализ результатов тестирования обнаружил закономерность в ошибках, совершенных алгоритмом.

В частности, дефекты при извлечении фрагментов с исходного изображения вызваны тем, что часть фрагмента имела одинаковый цвет с фоном. В большинстве таких случаев после предварительной обработки граница между фрагментом и фоном становилась неразличима, в том числе человеком. Точность извлечения фрагментов составила 100% на тестовом множестве, имеющем только фрагменты с четко различимой границей и 89-91% на остальных тестовых множествах.

Также, часть ошибок в определении углов вызвана дефектами, описанными выше. Другая обширная категория ошибок в определении углов – ошибки на вытянутых фрагментах.

Легко заметить, что слабым местом алгоритма является точность определения углов. Точность в рамках отдельной задачи нахождения углов является вполне удовлетворительной, но для работы алгоритма в полностью автоматическом режиме точность должна быть близка к 100%. В настоящий момент алгоритм нахождения углов показал меньшую точность - 79-100% в зависимости от тестового множества. Таким образом, алгоритм может быть улучшен в части разделения фрагмента на отдельные стороны (улучшив алгоритм нахождения углов или же реализовав другой способ разделения фрагментов).

Проанализировав алгоритм, можно выявить другое слабое место - алгоритм не допускает возможности того, что при сборке рамки возникнет неопределенность в выборе следующего фрагмента. Это допущение сделано из-за низкой вероятности возникновения такой ситуации на практике, а также потому, что используемая методика сборки не позволит разрешить

такую неопределенность в случае её возникновения. Существует возможность улучшить устойчивость алгоритма заменой алгоритма сборки на алгоритм, не зависящий от сборки рамки изображения.

Заключение

Разработанный алгоритм показал хорошую точность работы в случаях, когда фрагменты паззла имеют четко различимую на исходных изображениях границу и способен произвести сборку с минимальным вмешательством в процесс или вовсе без вмешательства. В противном случае, алгоритм показывает удовлетворительные промежуточные результаты, но неспособен завершить сборку без дополнительной информации о границах.

Список литературы

[1]: Gallagher A.C. Jigsaw Puzzles with Pieces of Unknown Orientation / IEEE CVPR, 2012, P. 382-389;

[2]: Liang L. A Jigsaw Puzzle Solving Guide on Mobile Devices / Liang Liang, Zhongkai Liu // In EE368 Projects, 2010;

[3]: Kosiba D.A. An Automatic Jigsaw Puzzle Solver / David A. Kosiba, Pierre M. Devaux, Sanjay Balasubramanian, Tarak L. Gandhi, Rangachar Kasturi // 12th IAPR International Conference, 1994, vol. 1, P. 616-618

[4]: Sholomon D. A Genetic Algorithm-Based Solver for Very Large Jigsaw Puzzles / Dror Sholomon, Omid David, Nathan S. Netanyahu // IEEE CVPR, 2013, P. 1767-1774;

[5]: Taeg Sang Cho. A probabilistic image jigsaw puzzle solver / Taeg Sang Cho, Shai Avidan, William T. Freeman // IEEE CVPR, 2010, P. 183-190;

[6]: Bradski G. Learning OpenCV / Gary Bradski, Adrian Kaehler – O'Reilly Media Inc., 2008/ - 576 p., ISBN: 978-0-5965-1613-0;

[7]: Dawson-Howe K. A Practical Introduction to Computer Vision with OpenCV / Kenneth Dawson-Howe – Wiley, 2014 – 234 p., ISBN: 978-1-1188-4845-6;

[8]: Forsyth D.A. Computer Vision: A Modern Approach, 2nd Edition / David A. Forsyth, Jean Ponce – 2nd Edition – Pearson, 2011. – 792 p., ISBN: 978-0-1360-8592-8;

[9]: Nixon M.S. Feature Extraction and Image Processing / Nixon M.S., Aguado A.S. – Replika Press Pvt Ltd, 2002. – 360 p., ISBN: 0-7506-5078-8;

[10]: Ritter G.X. Handbook of Computer Vision Algorithms in Image Algebra / Gerhard X. Ritter, Joseph N. Wilson – 2nd Edition – CRC Press LLC, 2001. – 448 p., ISBN: 978-0-8493-0075-2, 978-1-4200-4238-2;

[11]: Shapiro L.G. Computer Vision / Linda G. Shapiro, George C. Stockman – Pearson, 2001. – 608 p., ISBN: 978-0-1303-0796-5;

[12]: Ложные границы, создаваемые билатеральным фильтром: http://people.csail.mit.edu/sparis/bf_course/slides/09_limitations.pdf

[13]: Инвертирование цветов билатеральным фильтром:
http://research.microsoft.com/en-us/um/people/kahe/eccv10/eccv10ppt.pdf
[14]: Обзор методов нахождения характерных точек:
http://www.racurs.ru/wiki/index.php/Обзор_методов_обнаружения_характерных_точек
[15]: Исходный код тестового приложения:
https://github.com/Chillintano/puzzle-algo
[16]: Библиотека OpenCV: http://opencv.org/
[17]: Библиотека Qt5; http://www.qt.io/
[18]: Библиотека GTEngine: https://www.geometrictools.com/
[19]: Материалы по OpenCV: http://robocraft.ru/page/opencv/
[20]: Фурман Я.А. Введение в контурный анализ; приложения к обработке изображений и сигналов / Фурман Я.А., Кревецкий В.А., Передреев А.К., Роженцов А.А. и др. – Под ред. Фурмана Я.А. – 2-е изд., испр. – М.:ФИЗМАТЛИТ, 2003. – 592 с. – ISBN 5-9221-0374-1;
[21]: Сборка OpenCV для использования с Qt:
http://recog.ru/blog/opencv/4.html