Санкт-Петербургский Государственный университет
Математико-механический факультет
Кафедра системного программирования

Использование оптимизирующего алгоритма Левенберга-Марквардта в задаче поиска ракурса объекта по контурам

Курсовая работа студента 344 группы
Шубина Сергея Владимировича

Научный руководитель Вахитов А.Т.
доцент кафедры системного программирования

Санкт-Петербург
2014

Оглавление

Введение

Поиск ракурса объекта - это базовая вещь во многих приложениях компьютерного зрения, таких как сегментирование изображения, восстановление объектов, определение позиции объектов и тому подобных. На данный момент существует много решений для решения данной задачи, но они имеют свои существенные недостатки. Например, метод на базе оптического потока является вычислительно сложным и нестабильным, ICP основан на эвристиках и в отдельных случаях медленно сходится, методы, использующие особые точки, могут давать неверные результаты при неправильном сопоставлении особых точек и так далее. Все осложняется тем, что в рамках данной задачи объекты не имеют характерных точек, которые можно выделить с достаточным шансом распознавания и сопоставления (примером могут являться контуры машин разных марок, которые, вообще говоря, одинаковыми не являются, но являются очень похожими), то есть таких точек либо совсем нет, либо они сходны между собой. В связи с этим есть потребность в методе, который универсально хорошо обрабатывает контуры без особенностей и является устойчивым к шумам. Из различных методов оптимизации был выбран метод Левенберга-Марквардта как устойчивый, хорошо изученный и легко предсказуемый алгоритм. Его преимущества перед другими - простота, стабильная сходимость к минимуму и быстрая сходимость в окрестности минимума.

Какова постановка задачи? В качестве исходных данных имеются изображения двух контуров, которые различаются поворотом и сдвигом, и их надо сопоставить. Под сопоставлением подразумевается минимизация некоторого функционала, зависящего от трех параметров - сдвиг по координате x, по координате y, угол поворота относительно центра , и использующего изображения для высчитывания значения. Причем, когда параметры таковы, что при сдвиге и повороте второго изображения получается первое, то функционал должен достигать своего глобального минимума.

Обзор алгоритма Левенберга-Марквардта

Оптимизирующий алгоритм Левенберга-Марквардта - это локально оптимизирующий алгоритм, который можно описать как соединение двух методов - градиентного метода и метода Ньютона. Он, как правило, используется при решении задачи о наименьших квадратах, т.е. минимизации следующего функционала:

$$\sum_{i=0}^{n}(f(x))^2 \to \min$$

Его преимущество - соединение в одном сходимости к минимуму от градиентного метода и скорости сходимости в маленькой окрестности от метода Ньютона. Алгоритм является итеративным, и при наличии начального приближения x_0 каждое следующее получается по формуле (1):

$$x_{i+1} = x_i - (H + \lambda I)^{-1} \bigtriangledown f(x_i), \tag{1}$$

где H - это гессиан функции f. В отличие от других методов оптимизации присутствует неопределенный параметр λ, который задает, грубо говоря, как алгоритм будет вести себя: как градиентный или как метод Ньютона. При выборе слишком маленького значения будет плохая сходимость в окрестности точки (так как это является проблемой градиентного метода), при слишком большом - нельзя гарантировать, что предел вообще будет существовать. Для решения этой проблемы существуют различные эвристики, которых мы не будем касаться в рамках данной статьи, так как это выходит за пределы темы курсовой работы. Была использована реализация levmar, доступная по адресу [5]

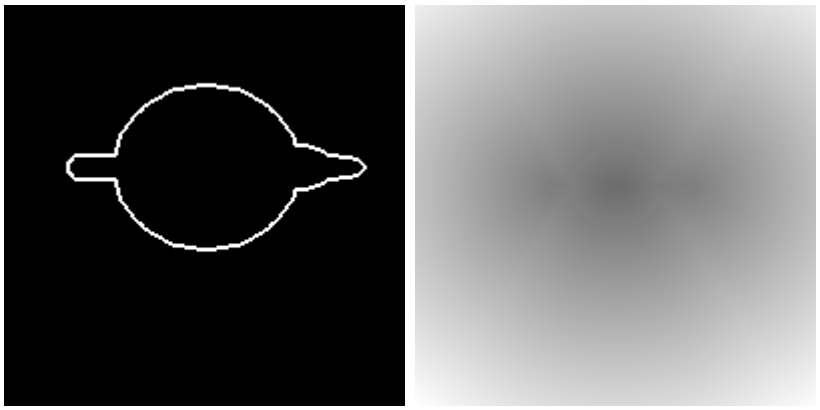
Применение

Описание

Как же применяется оптимизирующий алгоритм в задаче о поиске ракурса? В качестве исходных данных у нас выступают два контура на двух изображениях A и B:

Исходные контуры

Вначале в изображения добавляются отступы, что необходимо при последующей оптимизации, так как при сдвиге и повороте изображения B будет выход пикселей относительно изображения A. Далее вычисляются расстояния в целых точках от этой точки до контура с использованием алгоритма, описанного в [2]. Данный алгоритм работает за линейное от размеров картинки время, что более чем удовлетворяет требованиям задачи.



Исходный контур и получившиеся расстояния

Затем мы задаем функционал от полученных матриц расстояний, который нам надо минимизировать и который имеет следующий вид:

$$\sum_{x,y}(A_{x,y} - B_{x',y'})^2,$$

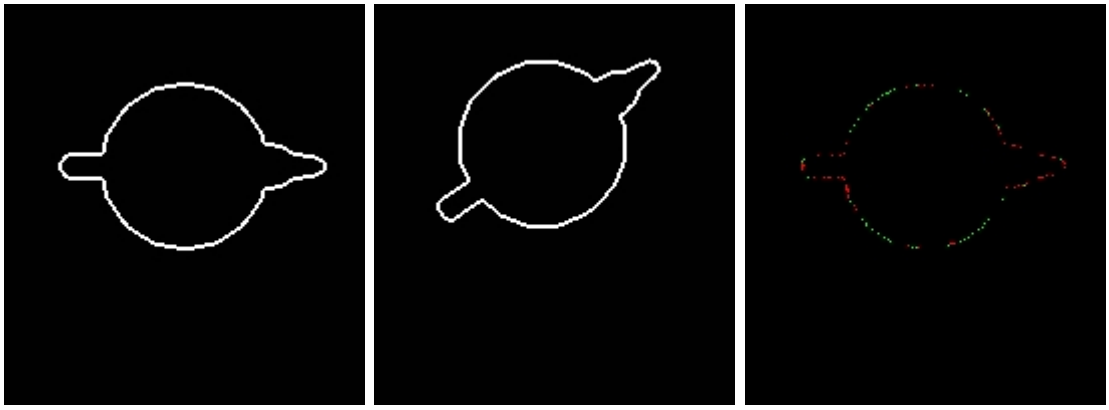
где $A_{x,y}$ - это значение расстояния на матрице A в точке (x, y), а $B_{x',y'}$ - значение расстояния на матрице B в точке (x', y'), полученной из точки (x, y) сдвигом на вектор (dx, dy) и поворотом на угол α относительно центра картинки. Понятно, что x' и y' не обязаны быть целыми, так что в этом случае для получения значения используется билинейная интерполяция. Видно, что функционал имеет квадратичный вид, на котором алгоритм работает лучше всего.

Функционал зависит от трех параметров, по которым и будет производиться оптимизация. Растяжение не было добавлено по причине плохой работы оптимизирующего алгоритма, так как, во-первых, при изменении масштаба приходится пересчитывать расстояния, что накладно вычислительно, и, во-вторых, при очень маленьком отступе по параметру скалирования значение функционала не изменится, так как изначально расстояния высчитываются в целых точках и для получения изменения нужен достаточно большой сдвиг. Причем, когда высчитываются градиенты, используются формулы приближенных производных, где изменение параметра очень мало, и в случае масштабирования производная всегда будет равна нулю. Центр поворота не был добавлен, так как данные три параметра полностью задают все возможные трансформации изображения для сдвига и поворота вокруг любой точки. Очевидно, что при совпадающих изображениях данный функционал будет равен нулю, так как все слагаемые будут равны нулю, и при небольших отклонениях функционал только возрастет. К сожалению, нельзя гарантировать строгой монотонности, как будет показано в следующем разделе.

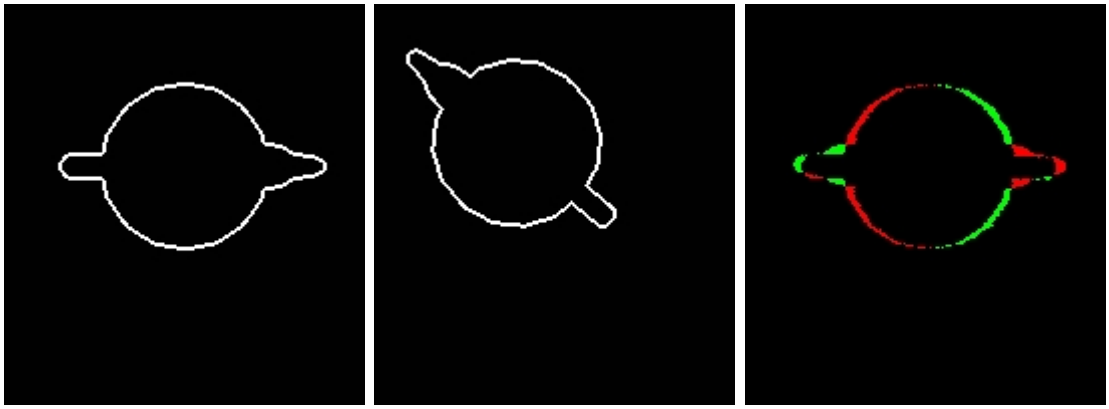
Таким образом, к введенному данным образом функционалу можно применять оптимизирующий алгоритм, получив на выходе параметры трансформации изображения B в A, а именно вектор сдвига и угол поворота вокруг центра.

Локальные минимумы

Сразу же встает проблема - так как алгоритм локально оптимизирующий, то вполне возможно получить неправильные параметры из-за наличия локальных минимумов. При применении были сразу замечены такие случаи, как показано на следующих рисунках:



Пример корректного сопоставления контуров, справа - разность исходного и трансформированного контуров



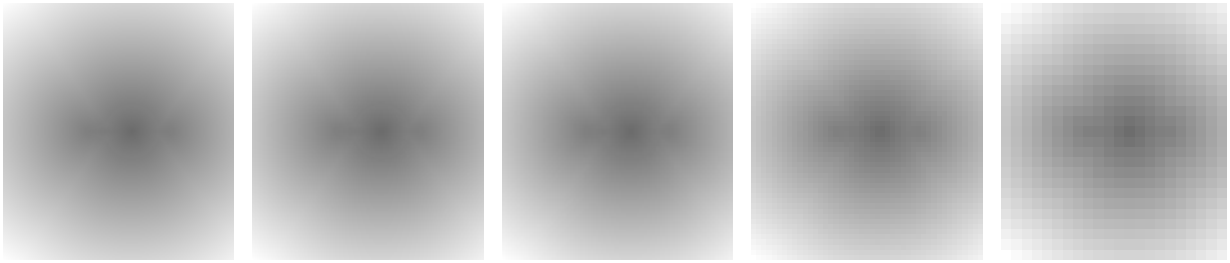
Пример некорректного сопоставления контуров

При оптимизации только по повороту при сопоставленных центрах проблема локальных минимумов встает в полный рост, так как в нашем образце мы имеем две области, сильно влияющие на нашу матрицу расстояний - это нос чайника и ручка, которые, в принципе, достаточно похожи друг на друга и задают сходные расстояния в своей окрестности. Как следствие, в случаях, когда носик чайника находится достаточно близко к ручке, то алгоритм их сопоставит, вместо того, чтобы развернуть на 180 градусов. Можно попытаться искать начальное приближение для угла, помня о том, что, во-первых, нас не интересует совсем точное значение угла (ибо в любом случае будет использоваться оптимизирующий алгоритм, а он дойдет до глобального минимума, если начальное приближение находится в в некой его окрестности), и, во-вторых, контур может иметь неточности.

Однако, как было замечено при исследовании, для оптимизации только по сдвигу проблема локальных минимумов не характерна, что объясняется большим количеством точек, не лежащих в выпуклом замыкании контура и при сопоставлении центров модуль разности расстояний в этих точках будет только уменьшаться, даже при увеличении значений ближе к центру, что дает возможность искать начальное приближение только для поворота, игнорируя сдвиг.

Метод пирамид

Первой идеей было использовать так называемую пирамиду расстояний. Этот метод заключается в следующем - берется матрица расстояний и уменьшается сначала в два раза, потом в четыре, потом в восемь и так далее определенное количество раз, причем значения в точках вычисляются как среднее четырех склеиваемых точек. Далее, оптимизирующий алгоритм применяется сначала для пары самых маленьких изображений, после он применяется для пары картинок на уровне выше, причем используется начальное приближение из предыдущего шага, и так далее до верхушки пирамиды. Идея этого метода заключается в том, что при уменьшении маленькие особенности должны либо полностью, либо частично пропасть, тогда как большие останутся и оптимизация пойдет именно по большим особенностям.



Пример полученной пирамиды расстояний.

Однако, как показала практика, этот метод работает либо практически никак (если пирамида недостаточно высокая), либо же начальные приближения совсем оторваны от реальности и поведение оптимизирующего алгоритма непредсказуемо:

Метод	Идентичные	Неидентичные
Высота пирамиды 7	12.5%	12.5%
Высота пирамиды 5	19.2%	19.2%
Высота пирамиды 3	19.2%	19.2%
Без пирамиды	19.2%	19.2%

Поэтому от этого метода было решено отказаться.

Метод дескрипторов

В самом общем случае метод дескрипторов заключается в том, что мы берем что-то от нашего объекта, строим по нему некий объект, описывающий его, и потом их каким-то образом сопоставляем и сравниваем. Вначале использовались дескрипторы особых точек, которые сопоставлялись между собой и на основе этого высчитывался угол. Как именно? Предположив, что точки x_1 и x_2 были сопоставлены, и центры первого и второго контуров обозначены как c_1 и c_2 соответственно, искомая аппроксимация угла будет равна углу между векторами $x_1 - c_1$ и $x_2 - c_2$. При наличии большого количества сопоставленных пар берется среднее всех аппроксимаций и получается финальное значение угла, которое используется при оптимизации. В качестве дескриптора был выбран дескриптор Малика, описанный в [4], который

заключается в следующем - выделяется особая точка, задается окружность радиуса R с центром в этой точке, эта окружность разбивается на 8 секторов по углам и каждый сектор делится на 3 части по радиусу, и после в каждом сегменте подсчитывается количество точек, попавших в него. К сожалению, его нельзя было использовать как есть, так как он зависит от угла, и в одной и той же точке при разных поворотах контура будут разные дескрипторы. Как решение, было решено направить вектор, от которого высчитывается угол в центр фигуры, в таком случае дескриптор становится инвариантным относительно угла. Как результат, появилась возможность корректно сопоставлять особые точки при разных углах поворота.

Если исходные контуры были одинаковы и без шумов, то метод давал точность до градуса в 99% случаев, что является очень хорошим результатом, но при добавлении шумов в контур метод резко сдавал позиции и процент ошибок повышался на порядки:

Метод	Идентичные	Неидентичные
Особые точки	99%	43.3%

Легко понять, почему такое случалось. Так как метод основан на особых точках, то при добавлении шумов особые точки могут как детектироваться с ошибкой, так и сопоставляться неправильно, что приводило к неверным результатам вычисления угла. В связи с этим данные методы были отброшены.

Следующим шагом было решение высчитывать дескриптор контура с применением метода Фурье, описанном в [3]. Для применения из контура вычленяется цепная линия, параметризуется как (x_i, y_i, s_i), где $s_i = \sum_{j<i} s_j + ||p_{i-1} - p_i||$ и $s_0 = 0$, где p_i - это i-я точка. Далее строится другая последовательность (x'_i, y'_i, s'_i), где s'_i - целые числа и $s'_i < S$, причем точки вычисляются линейной интерполяцией по s. После этого высчитывается центр, вычитается из всех точек, дескриптор нормируется по S и считается преобразование Фурье от получившегося массива, при этом точка (x, y) рассматривается как комплексное число $x + iy$. Если контуры были одинаковы с точностью до сдвига и поворота, то получившиеся преобразования Фурье будут отличаться в фазе на константу ϕ из-за поворота и линейно из-за разных начальных точек. Выглядит это следующим образом:

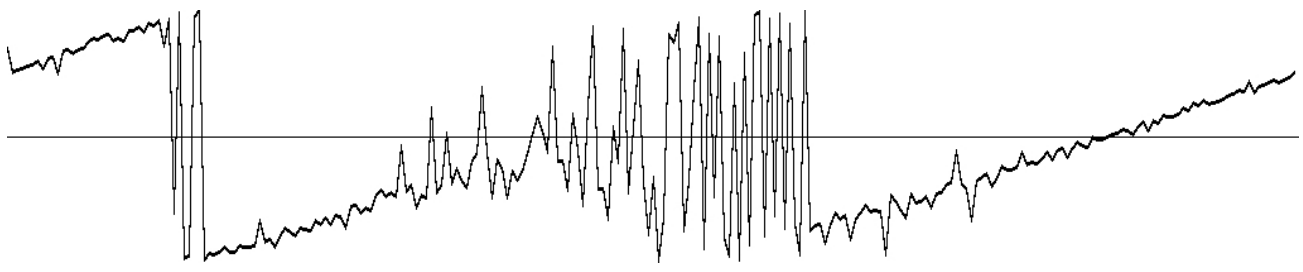


График разностей фаз у соответствующих элементах преобразований Фурье

Здесь изображены аргументы от почленного деления первого преобразования Фурье на второе, то есть насколько различны аргументы соответствующих комплексных чисел первого и второго преобразований. При этом во второй контур были добавлены шумы, и видно, что

ближе к центру колебания достигают очень высоких значений, но ближе к краям все достаточно линейно, и, выкинув центр, при высчитывании угла как среднее разностей аргументов получим достаточно близкое к реальному углу значение. Естественно, нельзя говорить о точности до градуса, но это не нужно для корректной работы оптимизатора, и при данном методе разброс достаточно маленький, чтобы метод сработал успешно:

Точность	Ошибка до 5	Ошибка до 20	Ошибка до 50
3 градуса	91%	56%	13%
10 градуса	96%	85%	25%
20 градусов	99%	87%	50%

Методика проверки - брался случайный контур, где координаты точек распределены равномерно от 0 до 100, добавлялась ошибка с равномерным распределением от нуля до указанного в таблице и замерялось, на сколько градусов отличались реальный и полученный углы. Как видно, метод Фурье дает достаточную для нашей задачи точность в вычислении угла при не очень высоких ошибках.

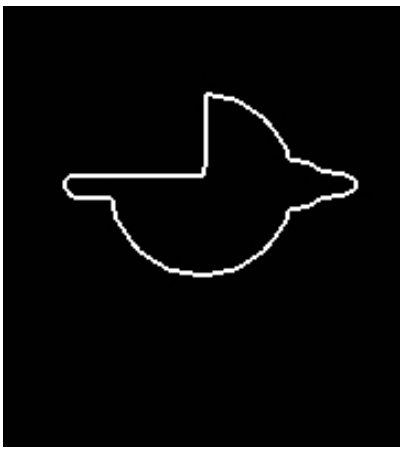
Заключение

Результаты

Таким образом, в рамках данной курсовой работы был применен оптимизирующий алгоритм Левенберга-Марквардта и исследована его работа при поиске ракурса объекта. Были исследованы метод пирамид, чья несостоятельность для решения данной задачи была показана, и различные дескрипторы, из которых был выбран метод на базе преобразования Фурье. Также была сгенерирована тестовая база, на которой проверялись описанные методы.

Дальнейшие исследования

Метод Фурье не дает правильных ответов в случаях, когда в контуре присутствует большая ошибка. Например, объект может быть закрыт чем-то, как показано на рисунке:



Пример перекрытого контура

В качестве решения могут быть использованы другие методы, как, например, построение дескриптора контура как последовательности дескрипторов точек и последующее сопоставление множества дескрипторов друг другу. Этот метод отличается от поточечного сравнения тем, что мы знаем, в каком порядке идут дескрипторы, и мы можем при сопоставлении узнать, где идет разрыв (т.е. та самая перекрытая область), убрать его и сопоставить уже по одинаковым частям контура.

Литература

1. Bodo Rosenhahn, Thomas Brox, Daniel Cremers, Hans-Peter Seidel - A Comparison of Shape Matching Methods for Contour Based Pose Estimation
2. Pedro F. Felzenszwalb Daniel P. Huttenlocher - Distance Transforms of Sampled Functions
3. Richard Szeliski - Computer Vision: Algorithms and Applications
4. Serge Belonge, Jitendra Malik, Jan Puzicha - Shape Matching and Object Recognition Using Shape Contexts
5. http://users.ics.forth.gr/ lourakis/levmar/ (дата обращения: 25.05.2014)