Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«Московский физико-технический институт

(государственный университет)»

Факультет управления и прикладной математики

Кафедра информатики

**Распознавание образов методом декомпозиции форм**

Выпускная квалификационная работа

(магистерская диссертация)

Направление подготовки:

03.04.01-Прикладная математика и физика (магистратура)

Выполнил:

студент 973 группы Бочаров Дмитрий Сергеевич

Научный руководитель:

к.ф. - м.н. Скороваров Константин Владимирович

Москва 2015

Оглавление

[Введение 2](#_Toc420924830)

[Основные понятия и постановка задачи 4](#_Toc420924831)

[Алгоритмы декомпозиции 6](#_Toc420924832)

[ACD (Approximate Convex Decomposition) – приблизительно выпуклая декомпозиция 6](#_Toc420924833)

[Метод связности 12](#_Toc420924834)

[Глубинный метод 17](#_Toc420924835)

[Алгоритм хеширования 21](#_Toc420924836)

[Выбор алгоритма декомпозиции 24](#_Toc420924837)

[Особенности реализации 26](#_Toc420924838)

[Заключение 27](#_Toc420924839)

[Список литературы 28](#_Toc420924840)

# Введение

Разработка систем машинного зрения включает в себя сложную задачу – научить машину выделять форму объектов в изображениях, сравнивать объекты по их форме и определять сходные и различные формы. Основной проблемой является то, что механизмы работы человеческого мозга не вполне понятны и не имеют четкой теоретической модели. В повседневной жизни мы на интуитивном уровне определяем понятие формы. Поэтому, казалось бы, такой легкий даже для ребенка вопрос как одногорбый или двугорбый верблюд нарисован на картинке, превращается в сложную задачу для компьютера.

Изображение в компьютерном представлении – это прямоугольная матрица точек, обладающих определенным цветом и яркостью. Поэтому перед создателями систем обработки изображений возникает проблема преобразований формы изображений, при которой «человеческие», интуитивно понятные геометрические способы анализа, сравнения, распознавания и преобразования формы можно было бы применить к изображениям, представляемым в компьютере в виде матриц. Существует два пути решения этой проблемы: «дискретный» и «непрерывный». В «дискретном» способе решения предпринимаются попытки представить человеческие понятия формы, как непрерывного «сплошного» объекта, в виде дискретного растрового изображения. В «непрерывном» же способе строятся непрерывные модели для объектов, выделенных в дискретной матрице изображений, и далее строятся алгоритмы в терминах непрерывных моделей, близких к человеческому восприятию понятия формы. На самом деле эти 2 способа являются крайними, и чаще всего используется их комбинация.

В данной работе предлагается инструмент математической бинарной морфологии. Основной идеей является стратегия "разделяй и властвуй", а именно будет рассматриваться декомпозиция форм. Также как и сегментация изображений в анализе изображений, декомпозиция форм является фундаментальной проблемой в таких областях как компьютерное зрение и визуализация данных. После декомпозиции многие операции, которые сложно применить к изначальному изображению, могут быть применены к декомпозированным частям легко и эффективно. Поэтому декомпозиция очень полезна при анализе форм, построении топологий, детектировании столкновений и в других геометрических методах использующих стратегию "разделяй и властвуй".

Во введении будет представлена постановка задачи и даны основные определения. Далее рассматриваются основные уже существующие алгоритмы декомпозиции и предложен свой новый. Приведено их сравнение и причина выбора ACD (Approximate Convex Decomposition) алгоритма. Далее предлагается метод хеширования простых декомпозированных частей для их быстрого сравнения друг с другом. В конце представлены особенности реализации Web-интерефейса для работы с данными алгоритма и примеры декомпозиции и хеширования.

# Основные понятия и постановка задачи

Бинарное изображение — это двухцветная картинка, на которой представлены один или несколько объектов одного цвета на фоне, имеющем другой цвет. Такие изображения представляются в компьютере в виде матрицы точек, каждая из которых может иметь лишь одно из двух значений цвета, условно обозначаемых через 0 или 1. Точки также называются пикселями (pixel — picture's element). He нарушая общности, будем считать бинарное изображение «черно-белым»: точки объекта являются черными (значение цвета равно 1), а точки фона — белыми (значение цвета равно 0).

Теперь дадим понятие формы. Так как мы работаем с двухмерным изображением, то все определения дадим применительно к евклидовой плоскости *R*2.

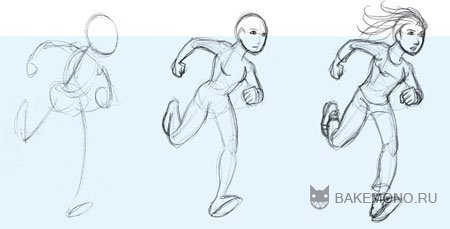
**Определение 1**. Жордановой кривой называется образ окружности при непрерывном инъективном ее отображении в евклидову плоскость. Отображение инъективно, если любые две различные точки прообраза отображаются в две различные точки образа.

**Определение 2**. Фигурой называется связная замкнутая область на плоскости, ограниченная конечным числом непересекающихся жордановых кривых.

Таким образом, фигура в нашем понимании — это связная часть плоскости, ограниченная одной или несколькими замкнутыми линиями. Под это определение подходят сплошные объекты с конечным числом дыр.

Перейдем к рассмотрению понятия декомпозиции. Однако сложно дать его четкое определение, потому что нашей целью является смоделировать интуитивную декомпозицию формы человеком. Существуют психофизические исследования, результатами которых являются модели человеческого представления формы как нечто составного. К примеру, иерархические модели путем последовательного разбиения формы на всё более мелкие части строят дерево вхождения этих частей.



Альтернативная модель основана на наблюдении за художниками, которые все значимые части представляют в виде эллипсов, в то время как параболические связки этих эллипсов считаются незначительными частями.

Таким образом, не существует единого определения части формы. Поэтому далее при рассмотрении конкретного метода декомпозиции каждый раз будет описываться определение декомпозированных частей свойственного в данном методе.

Второй целью работы была реализация алгоритма хранения формы в реляционной базе данных, на основании которого можно реализовать распознавание образа формы. Реляционные базы данных являются самым распространённым способом хранения информации. Поиск, вставка и изменение информации осуществляются в них с огромной скоростью. Для этого существуют различные механизмы ускорения. Один из самых базовых и эффективных – это индексация.

Просуммировав всё вышесказанное, задачей данной работы является реализация алгоритма хранения формы в реляционной базе данных, путем её декомпозиции.

# Алгоритмы декомпозиции

В данном разделе рассмотрены существующие алгоритмы декомпозиции форм, а также предложен свой.

## ACD (Approximate Convex Decomposition) – приблизительно выпуклая декомпозиция

Целью данной декомпозиции является разбиение формы на приблизительно выпуклые части. Для начала введем некоторые определения.

**Определение 3.** Для объекта *О* размерности *n* **разрез** – это пересечение (*n - 1*) –мерной гиперплоскости с *O.*

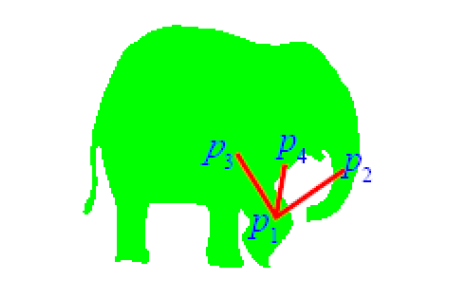
Таким образом, разрез удовлетворяет 2 критериям: 1) он лежит внутри объекта; 2) концы разреза лежат на границе формы.

Введем характеристики разреза и части формы:

* Стоимость разреза *C*, Cost(*C*), – действительное положительное число. Для его определения могут вводиться различные метрики, однако мы рассмотрим простой и интуитивно понятный вариант Евклидовой метрики. Cost(*C*) – длина разреза.
* Вогнутость части *P*, Concavity(*P*), – действительное положительное число, которое характеризует величину вогнутости части. О её формальном определении будет рассказано ниже.

Используя введенные понятия, можно дать формальное определение декомпозиции: разделить форму *O* на *q* частей, используя *m* разрезов, так чтобы суммарная стоимость *m* разрезов была минимальна и одновременно вогнутость каждой части не должна быть больше порогового значения *ε*.

По определению, множество называется выпуклым, если оно содержит отрезок, соединяющий любые две точки, принадлежащие множеству. Это позволяет определить взаимосвязь двух внутренних точек множества.

**Определение 4.** Точки *p1* и *p2* , принадлежащие форме *O*, называются ***мьютексными (взаимоисключающими)***, если *O* не содержит отрезок соединяющий *p1* и *p2*. Обозначается как *p1* ≈ *p2*.

**Рис. 2**

На рисунке 2 *p1* ≈ *p2* и *p1* ≈ *p4*. Очевидно, что после декомпозиции *p1* и *p2* не могут быть в одной части. Таким образом, мьютексные точки являются ограничением выпуклой декомпозиции. Обычно результатом выпуклой декомпозиции является огромное число частей, с которыми тяжело работать, поэтому будем искать декомпозицию на примерно выпуклые части. Очевидно, что для приблизительно выпуклой декомпозиции *p1* ≈ *p2* и *p1* ≈ *p4* – различны. Определим меру вогнутости для мьютексной пары точек *a* и *b*: Concavity(*a,b*). При этом если форма *O* содержит отрезок, соединяющий *a* и *b*, то Concavity(*a,b*) = 0. Из это следует, что на рисунке 2: Concavity(*p1*, *p2*) > Concavity(*p1*, *p4*) > Concavity(*p1*, *p3*) = 0.

**Определение 5**. Для формы *O* ***ε-мьютексным множеством, Mε(O),*** называется множество всех мьютексных пар, чей вес не меньше *ε*.

Введенное определение позволяет исключить мьютексные пары с малым весом.

Множество всех возможных разрезов формы *O* назовем ***кандидатным***, *C*(*O*). Каждый разрез из *C*(*O*) может удовлетворять мьютексной паре из Mε(O). То есть, чтобы выполнить приблизительно выпуклую декомпозицию нам надо выбрать такие разрезы из кандидатного множества *C*(*O*), которые удовлетворяют всем мьютексным парам из Mε(O).

Пусть кандидатное множество *C*(*O*) = {cut1 … cutn} включает *n* разрезов, Mε(O) = {mp1 … mpm} – *m* ε-мьютексных пар. Результат декомпозиции – множество индексов *I*, означающее, что если , то cuti – решающий разрез. То есть каждому разрезу cuti ставится в соответствие бинарная переменная *xi* : . Каждому соответствует подмножество мьютексных пар, которым удовлетворяет этот разрез. То есть в Mε(O) можео выделить *n* подмножеств мьютексных пар для каждого разреза. Для мьютексной пары среди всех разрезов, который ей удовлетворяют, должен быть хотя бы один, индекс которого принадлежит результирующему множеству *I*. То есть:

|  |  |
| --- | --- |
|  | (1) |

Пусть **x** = (*x1*, *x2*, …*xn*)*T*, **c** = (cost(cut1), cost(cut2), …, cost(cutn))*T*, **A** = {*aij | I =* 1 *... m, j =* 1 *... n*}, **1** = (1, 1, …, 1)*T*, тогда задача (2) перепишется в задачу целочисленного линейного программирования:

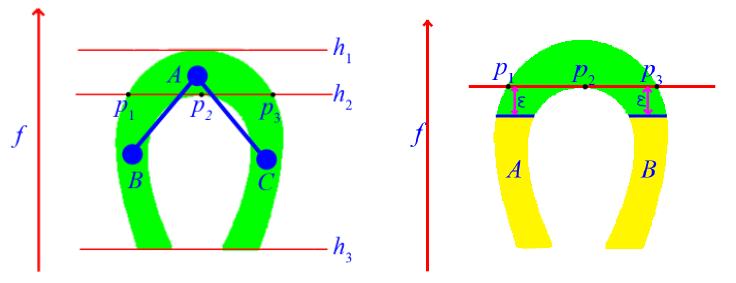
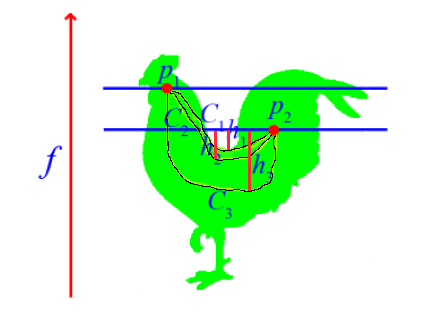
|  |  |
| --- | --- |
| min **c***T***x Ax** ≥ **1**, *xi*  {0, 1} | (2) |

Это NP-полная задача, однако мы можем ослабить условие на *xi*:

|  |  |
| --- | --- |
| min **c***T***x Ax** ≥ **1**, 0 ≤ *xi* ≤ 1 | (3) |

(3) – задача линейного программирования. Решив её, мы получим приближенное оптимальное решение задачи (2). *xi* может рассматриваться как вероятность выбора cut*i*. После решения (3) мы можем последовательно выбирать разрезы, имеющие наибольшую вероятность, пока не удовлетворим все мьютексные пары из *M*ε(*O*).

Для формы *O*, *M*ε(*O*) и *C*(*O*) – настолько большие, что с ними не имеет смысла работать напрямую. Но многие мьютексные пары избыточны, то есть когда некоторые мьютексные пары удовлетворены, другие также автоматически удовлетворены. Далее будут показано, как избавиться от избыточных мьютексных пар и получить достаточно малое *M*ε(*O*), а как уменьшить множество кандидатных разрезов *C*(*O*).

Определение *M*ε(*O*) и *C*(*O*) основано на теории Морса. Она имеет 2 основные операции. Первая – построение функция Морса *f* : *M* → *R*, которая может рассматриваться как проекция на выбранное направление. Вторая – построение графа Риба.

**Рис.3 Рис.4**

Граф Риба подковы Измерение вогнутости двух точек

Таких направлений существует бесконечно много. Чем больше мы рассмотрим, тем больше будет качество декомпозиции и тем больше времени потребуется на работу алгоритма. Баланс достигается экспериментальным способом.

Осталось разобраться с понятием вогнутости и дать формальное определение «приблизительности» выпуклой декомпозиции.

**Определение 6.** Для двух точек *p*1 и *p*2 формы *O* кривую их соединяющую и лежащую внутри *O* назовем **путем** между *p*1 и *p*2. Все пути соединяющие *p*1 и *p*2 обозначим *C*(*p*1, *p*2).

Очевидно, что количество путей в *C*(*p*1, *p*2) может быть бесконечным. Для каждого пути вводится понятие вогнутости. Пусть точка *p* принадлежит пути *С*, соединяющем *p*1 и *p*2 и пусть *f*(*p*1) ≥ *f*(*p*2), тогда:

|  |  |
| --- | --- |
|  | (4) |
|  | (5) |

На рисунке 4 *С*1, *С*2 и *С*3 – пути между точками *p*1 и *p*2, при этом .

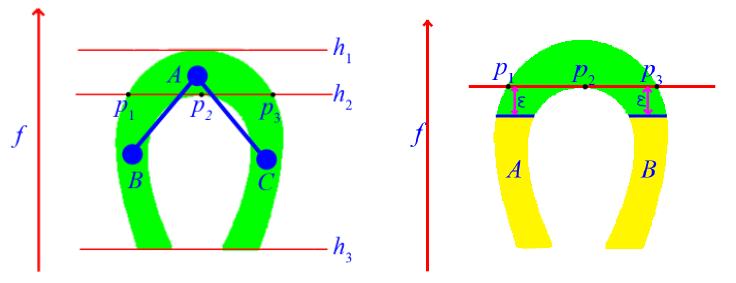
Вогнутость для 2 точек *p*1 и *p*2 – минимум меры вогнутости среди всех путей соединяющих *p*1 и *p*2:

Вогнутость же части *P* – максимум вогнутости случайных двух точек из *P*:

*M*ε(*O*) – слишком большое, поэтому чтоб уменьшить сложность, будем рассматривать *M*ε(*O*) ни как множество пар мьютексных множеств точек.

**Определение 7.** Если для двух непересекающихся множеств *A* и *B* существует мьютексная пара точек, состоящая из точки из *A* и точки из *B*, то множества *A* и *B* называются мьютексным: *A* ≈ *B*.

Теперь можно дать определение максимальной и минимальной вогнутости для двух мьютексных множеств.

Если , то мьютексная пара множеств может быть проигнорирована. Если же , то каждая пара точек из *A* и *B* формируют мьютексную пару, вес которой не меньше . Теперь нашей задачей является нахождение множеств точек *A* и *B*: , и также найти разрезы которые их разделят. Зная граф Риба это очень легко сделать. Так как он показывает изменения количества связных компонент, то два соседних множества и являются мьютексной парой. Разрезы же, которые их разделяют, и есть разрезы-кандидаты.

**Рис. 5.**

На рисунке 5 два множества *A* и *B* – мьютексная пара. В то же время два кандидатныых разреза – оба разделяют *A* и *B*. Очевидно, что .

Таким образом, мы нашли *M*ε(*O*) и *C*(*O*).

Кратко алгоритм приблизительной декомпозиции можно записать следующим образом:

1. Выбираем множество направлений, считаем функцию Морса.
2. **for** каждой функции Морса **do**
   1. строим граф Риба
   2. считаем мьютексные пары и добавляем их в мьютексное множество
   3. считаем кандидатные разрезы и добавляем их в множество кандидатных разрезов

**end**

1. **for** каждого разреза в множестве кандидатных разрезов **do**

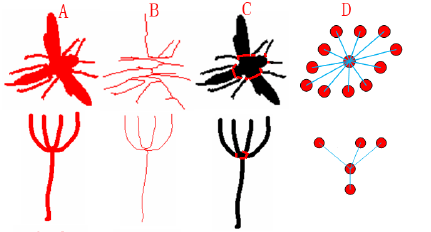
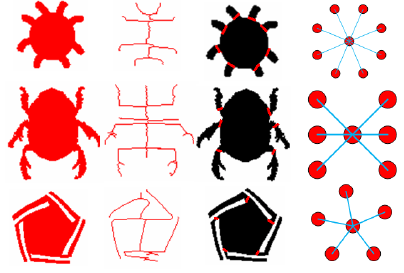
**for** каждой пары в мьютексном множестве **do**

Проверяем удовлетворяет ли разрез мьютексной паре

**end**

**end**

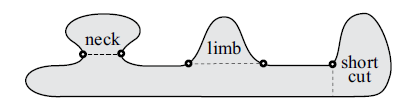
1. Решаем задачу линейного программирования (3)
2. Выбираем решающие разрезы, основываясь на их стоимости.

Примеры декомпозиции:

В столбце A показаны входные формы; в столбце B – графы Риба, построенные по высотной функции вдоль вертикали; С – результат декомпозиции; D – граф выпуклости формы.

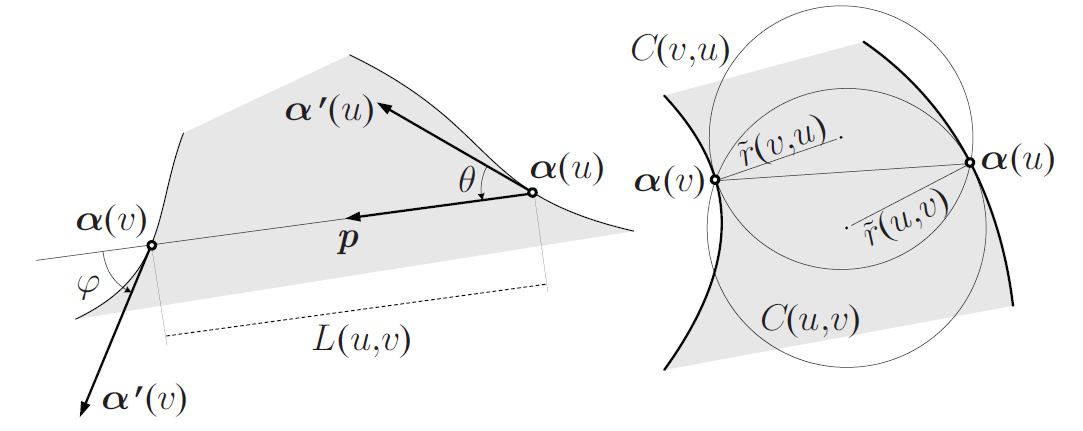
## Метод связности

Данный метод основан на анализе поведения границы формы, а точнее на особенностях её изменения при переходе между частями. Важным аспектом является то, что выделенные части последовательно удаляются и дальнейшая декомпозиция идет без их участия. Выделают 3 типа частей: «шея», «конечность» и «отрез».



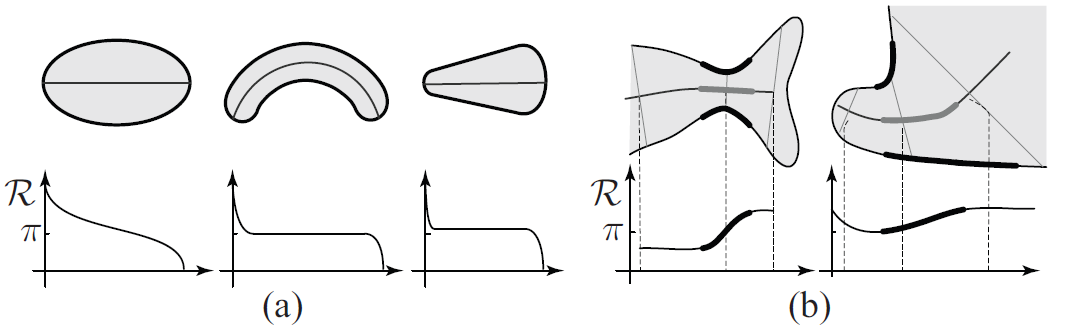
**Рис.6.**

Анализ проходит на основании скелета сглаженной локальной симметрии (Smoothed Local Symmetries - SLS). Представим головку машины, которая последовательно скользит по осям такого скелета. Таким образом, форма – это объединение ребер, соединяющих симметричные точки границы формы, с центром на оси.

Начнем с простой формы, чья граница представляется непрерывной в кривой , заданной параметрически.

**Рис.7**

Будем рассматривать отрезок с концами и , где и –параметрические положения кривой. Длина отрезка , а его направление . Касательные в этих точках соответственно обозначим и . Обозначим угол между и *p* как , а угол между *p* и как , где . Тогда полный угол поворота это .



**Рис.8**

Ниже будет показано, что величина связности равна . На рисунке 8 показано изменение связности при последовательном движении по оси симметрии формы. Также показано, что по характеру изменения *R* можно судить о виде части (Рис.6).

Частично касательной назовем окружность , которая касательна к границе формы в точке и проходит через точку . Её радиус (см. Рис.7). Теперь мы можем определить частные производные по и :

|  |  |
| --- | --- |
|  | (6) |
|  |  |
|  | (7) |

– кривизна кривой . Далее нам будут важны следующие величины:

|  |  |
| --- | --- |
|  | (8) |

Также необходимо формализировать понятие асимметрии . – разница между углами . А точнее: .

**Определение 8.** Две точки и асимметричны, если .

Из определения следует, что движение симметричной пары точек по скелету сглаженной локальной симметрии перпендикулярно градиенту асимметрии . Рассмотрим малый шаг вдоль оси. При этом смещение симметричной пары будет и соответственно. Поэтому . Для симметричной пары , : две частично касательные окружности совпадают друг с другом (бикасательная окружность) и мы можем связать и :

|  |  |
| --- | --- |
|  | (9) |

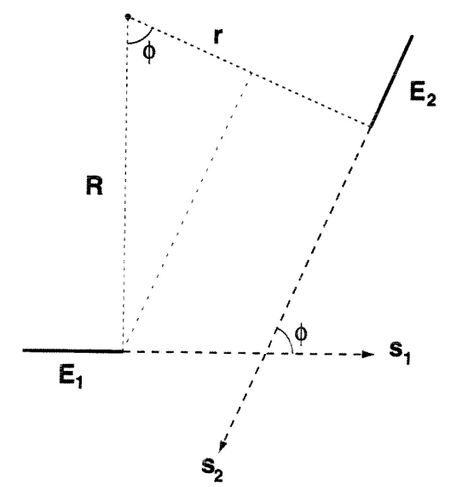
Точки симметричной пары движутся по границе в противоположных направлениях, поэтому и имеют различные знаки, в то время как и имеют одинаковые знаки, и бикасательная окружность вписана или описывает обе стороны.

Шаг можно разбить на две составляющие: перпендикулярную и параллельную :

|  |  |
| --- | --- |
|  | (10) |

Обычно не равен нулю, а и в большой степени компенсируют друг друга, поэтому . Следовательно, параллельной составляющей можно пренебречь. Поэтому с учетом (6): и:

|  |  |
| --- | --- |
|  | (10) |

Перейдем теперь к понятию связности границ поверхности:

**Определение 9.** Пусть и – две границы поверхности, и – их продолжения. Тогда две границы называются связными, если и пересекаются и их смежный угол пересечения острый.

В [3] вводится мера связности, как , где – это любая сглаживающая кривая, соединяющая и . В нашем варианте параметризации прямой .

Мы можем найти изменение связности при движении вдоль SLS на малый шаг . С учетом (8) и того что мы получаем и

|  |  |
| --- | --- |
|  | (11) |

где и .

Можно сделать вывод, что изменение связности определяется средней нормированной величиной кривизны в симметричных точках. Также заметим, что знак не зависит от направления . Далее видим, что вторая производная пропорциональна связности:

|  |  |
| --- | --- |
|  | (12) |

Резкое изменение является индикатором переходной области. Однако эта величина зависит от масштаба изображения, поэтому определим величину силы перехода:

**Определение 10. Локальной силой перехода** симметричной пары назовем величину .

Значительные изменения силы перехода является необходимым условием переходной области. На практике выставляется порог значения силы .

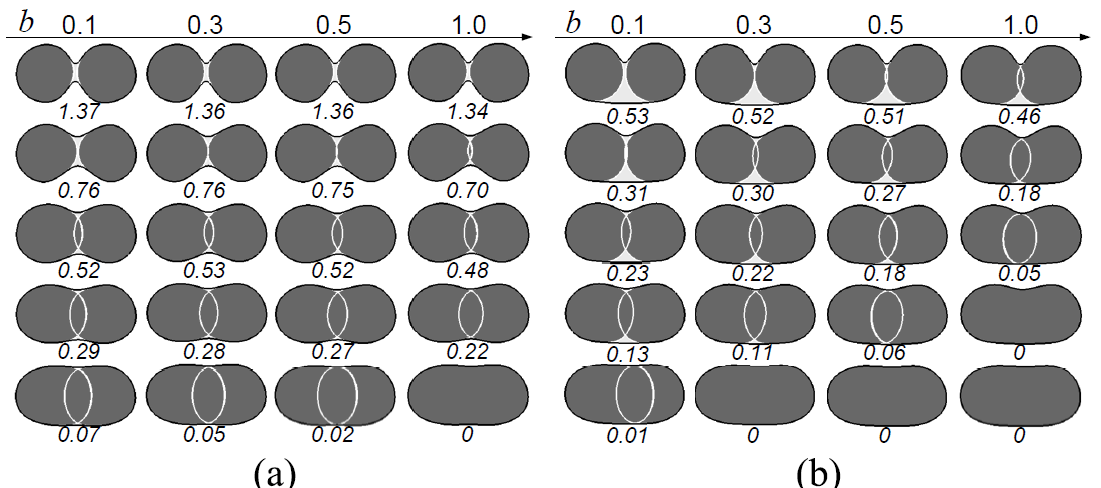
**Определение 11. Полной силой перехода** между частями называется величина изменения связности в переходной области:

,

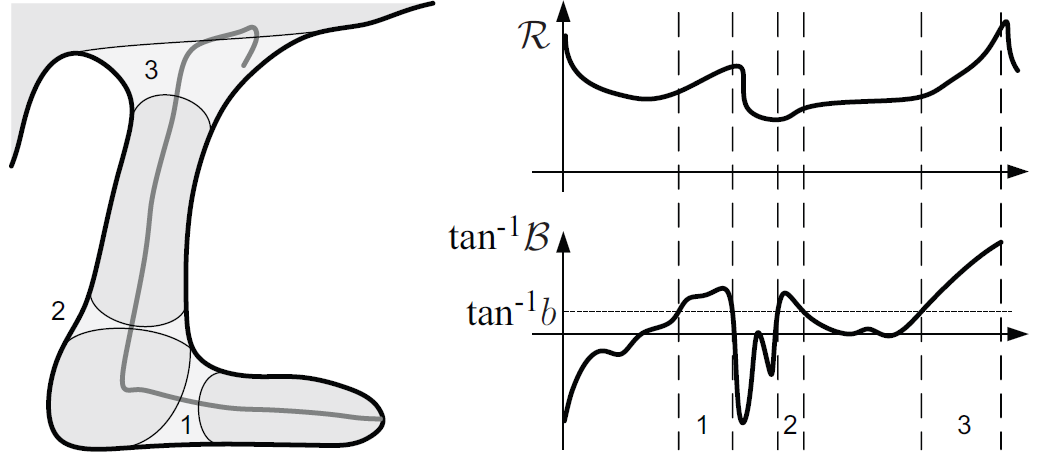
где – отрезок оси с .

Измерение включает в себя два шага. Во-первых, мы находим сильные изменения локальной силы перехода . Далее для каждой из такой областей вычисляется . Считается, что мы обнаружили переходную область, если больше некоторого заранее установленного порога. Чаще всего порог выбирают в диапазоне

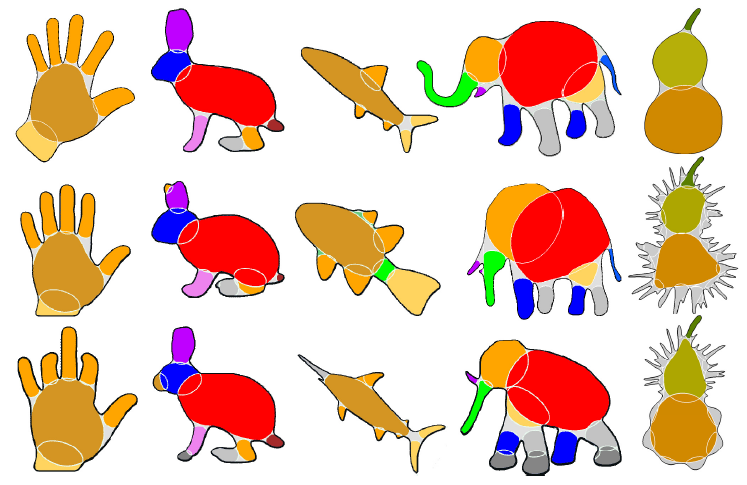
На рисунке ниже можно видеть эксперименты с варьированием пороговых величин и (под каждым изображением записана величина ).



На следующем рисунке показаны изменения связности и силы перехода для элемента формы.



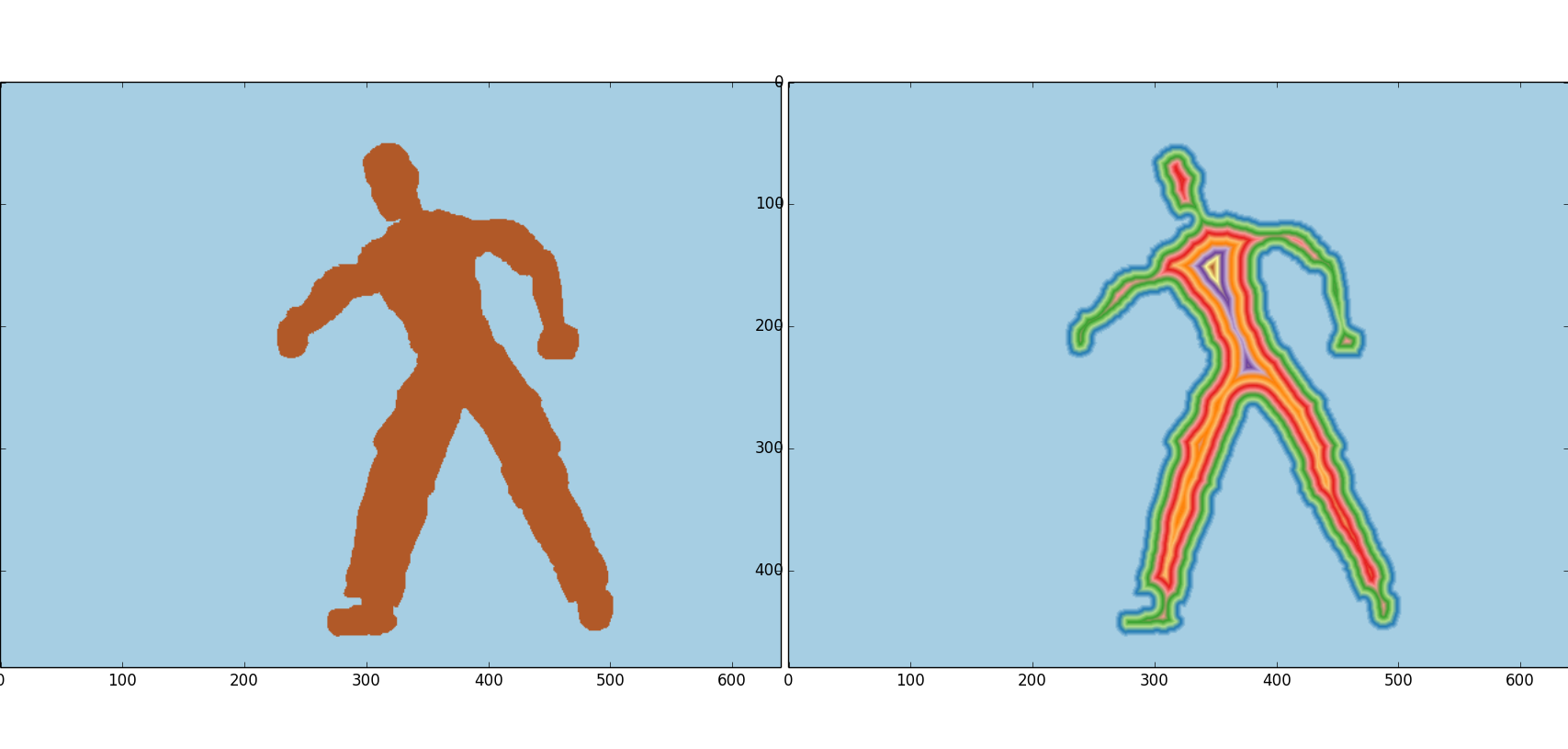
Примеры результатов декомпозиции:



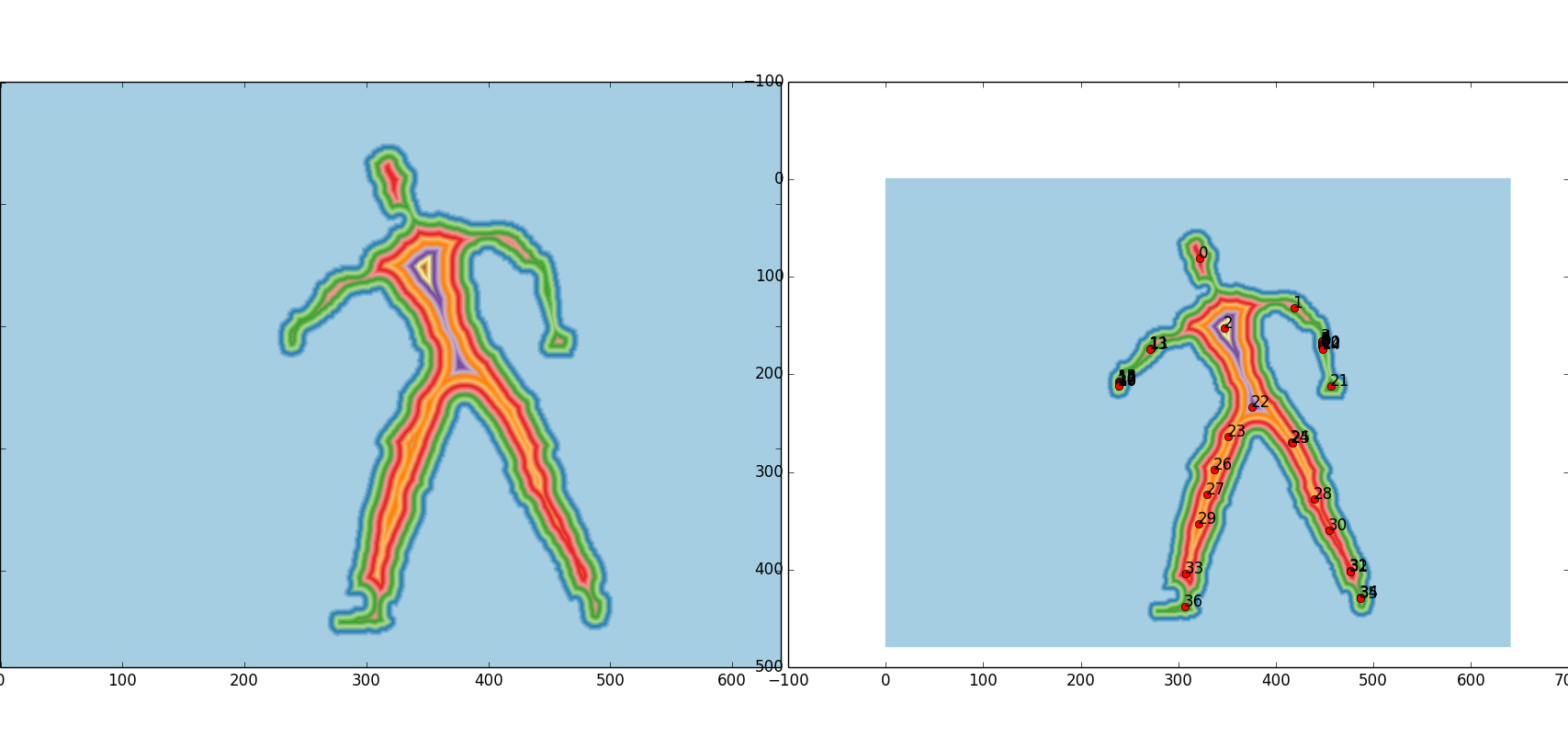
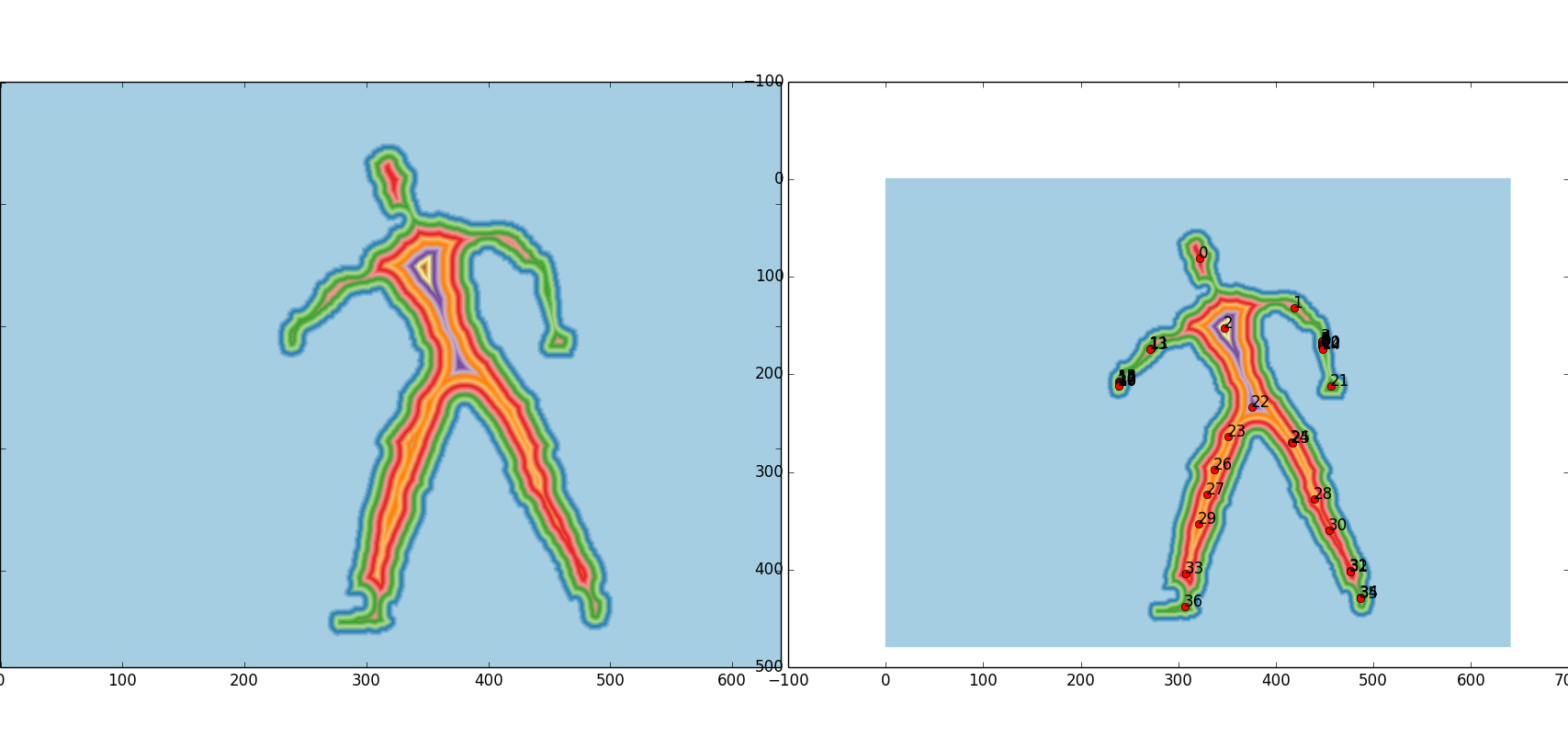
## Глубинный метод

В этом разделе рассмотрим предложенный мною метод декомпозиции.

Как известно, *скелетом* фигуры называется множество центров всех её максимальных пустых кругов. В [1] рассматривается ещё одно определение скелета, основанное на понятии дистанционной функции формы. Дистанционная функция для каждой внутренней точки формы задает «глубину» ее расположения относительно границы, равную расстоянию от этой точки до границы фигуры. Дистанционная функция строится последовательным обходом всех внутренних точек формы, начиная со «слоя» точек максимально близких к границе. Пройдя один слой полностью, мы переходим на следующий слой внутрь формы. Визуализация результата представлена на рисунке ниже:

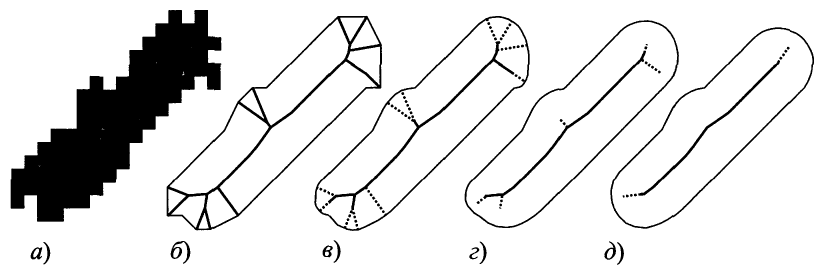


**Рис.9.** Начальная форма и её глубинная функция

Следующим шагом является поиск максимумов глубинной функции. Тем самым мы найдем точки, максимально удаленные от границы:

Далее эти максимумы будут рассматриваться как центры формирования частей, на которые мы и декомпозируем форму.

Можно заметить на реальном примере, что таких максимумов получилось очень много. Многие из них необоснованны и интуитивно (к чему мы и стремимся) непонятны. Главное причиной этого является наличие «шумовых эффектов» в форме. Скелет формы (глубинная функция) очень чувствительна к локальным свойствам её границы. В алгоритмах дальнейшей работы со скелетом формы для устранения таких эффектов применяется его регуляризация, например, методом стрижки:



**Рис.10.** Исходное растровое изображение (а), минимальный разделяющий многоугольник и его скелет (б), последовательная стрижка ребер (в)-(д).

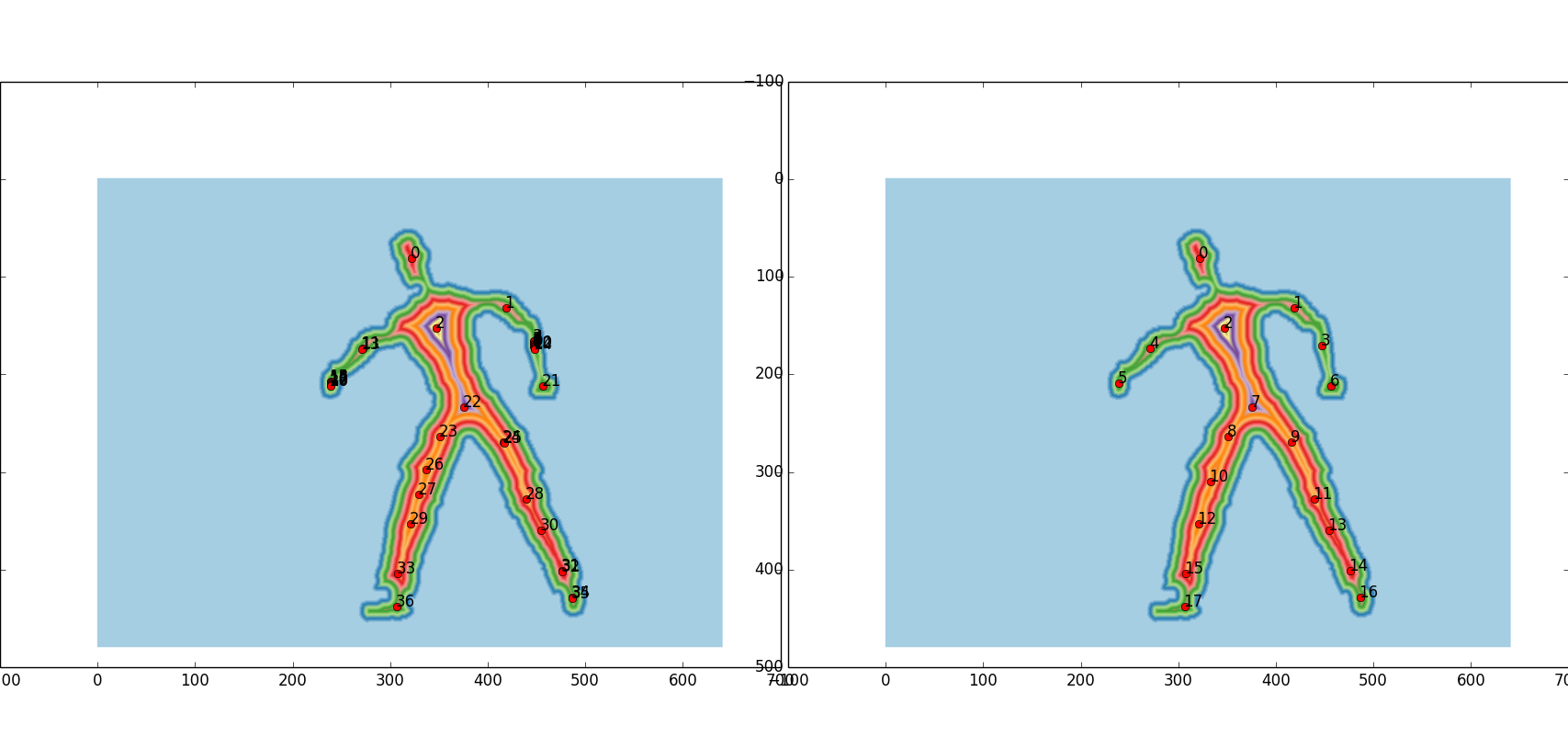
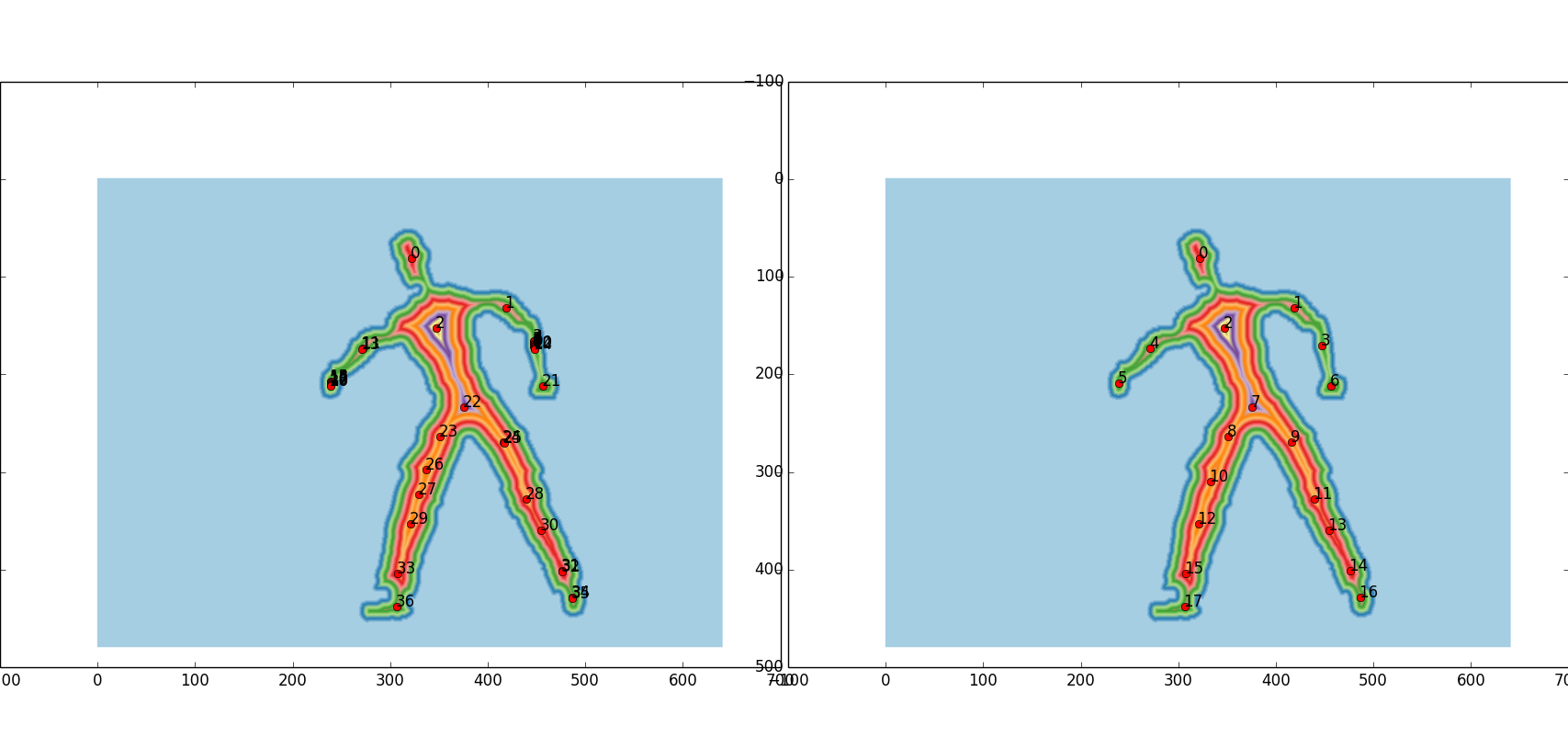
Однако регуляризованный скелет не дает нужных нам точек максимума, а вернее дает целую их последовательность, среди которой нужно выделить «центры» будущих частей. Поэтому предлагается сгруппировать найденные максимумы дистанционной функции следующим образом.

1. Каждой точке присваиваем метку группы. Если точка не входит ни в какую группу ставим метку (-1).
2. Инициализируем текущий номер группы: group = 1.
3. **for** каждой точки **do**

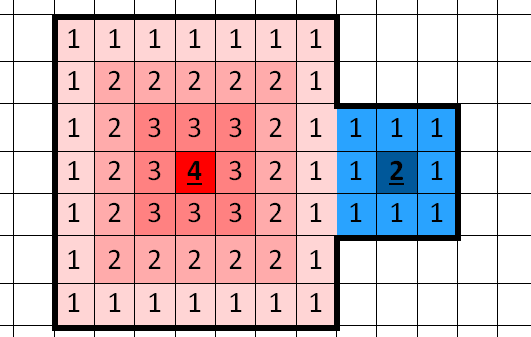
**if** точке не присвоена метка группы **then**

1. присваиваем точке текущую метку группы
2. ищем другие точки, которые находятся на расстоянии меньшем, чем глубина рассматриваемой точки
3. присваиваем найденным точкам текущую метку
4. переходим в начало шага 3, но цикл осуществляется по найденным точкам
5. Для каждой группы ищем их геометрические центры

Найденные центры и будут центрами формирования частей. На рисунке ниже показан результат работы описанных шагов.



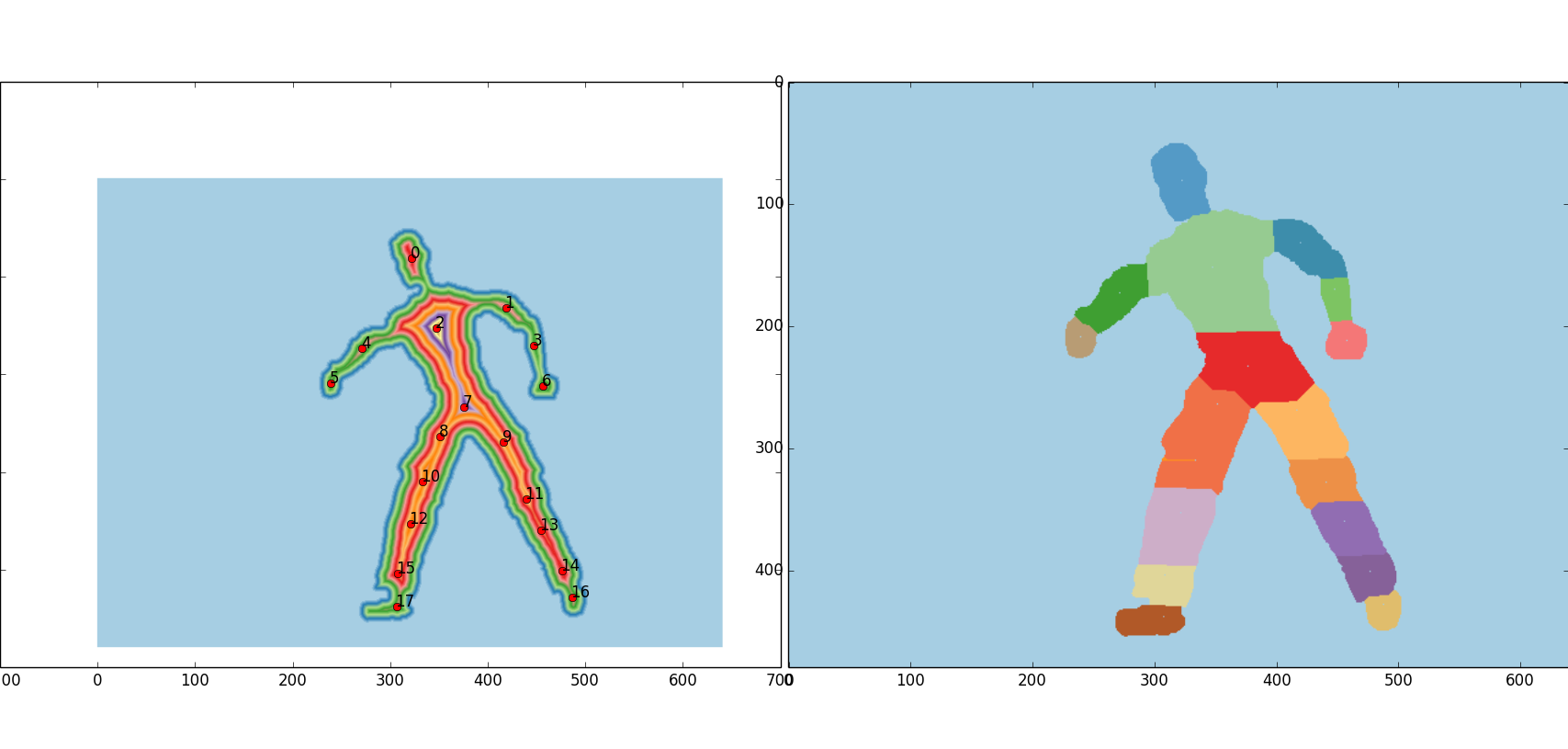
Следующим шагом является применение алгоритма водораздела с пиками в найденных нами точках.



Приоритет (высота) точек – это их глубина в нашей форме. Начиная с самых приоритетных точек, мы последовательно ставим самым близким к ним пикселям метку группы этой точки. Продолжаем это делать, удаляясь от самой приоритетной точки, и запоминаем, на какой глубине мы сейчас находимся. После проставления меток на очередном круге, мы пробегаем по всем остальным центрам, и если их глубина равна текущей, запускаем такую же операцию (водопоток) с этого центра. Для наглядности представлен рисунок с порядком выставления меток. Цветом обозначен индикатор группы.

В результате, всем пикселям формы присвоено какое-то цветовое значение, которое и является индикатором принадлежности к какой-либо части.

Примеры результатов декомпозиции:



# Алгоритм хеширования

В данном разделе предлагается алгоритм хеширования, который в результате помог осуществлять быстрый поиск похожих декомпозированных частей. В следующем же разделе, опираясь на описанные особенности хеширования, мы выберем алгоритм декомопозиции.

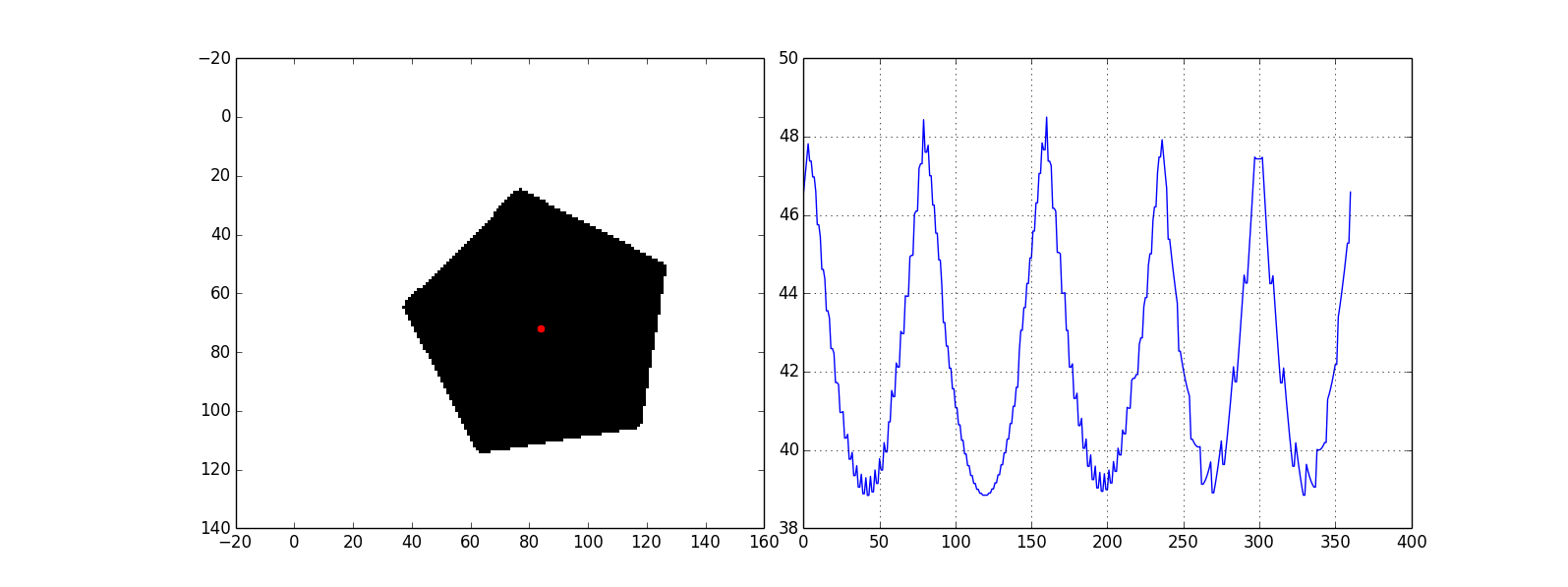
Алгоритм хеширования декомпозированных частей должен удовлетворять следующим свойствам:

1. Хеш должен быть сравниваемым. То есть, зная хеши 2 форм, мы можем сказать, не только схожи ли эти формы, но и степень их близости.
2. Хеш не должен зависеть от масштаба формы
3. Хеш не должен зависеть от поворота формы.

В этом разделе под формой мы будем понимать простую форму, как одну из частей сложной декомпозированной формы. Первым шагом алгоритма является поиск центра масс формы. К примеру, для изображения размера N × N его можно найти по следующей формуле:

где

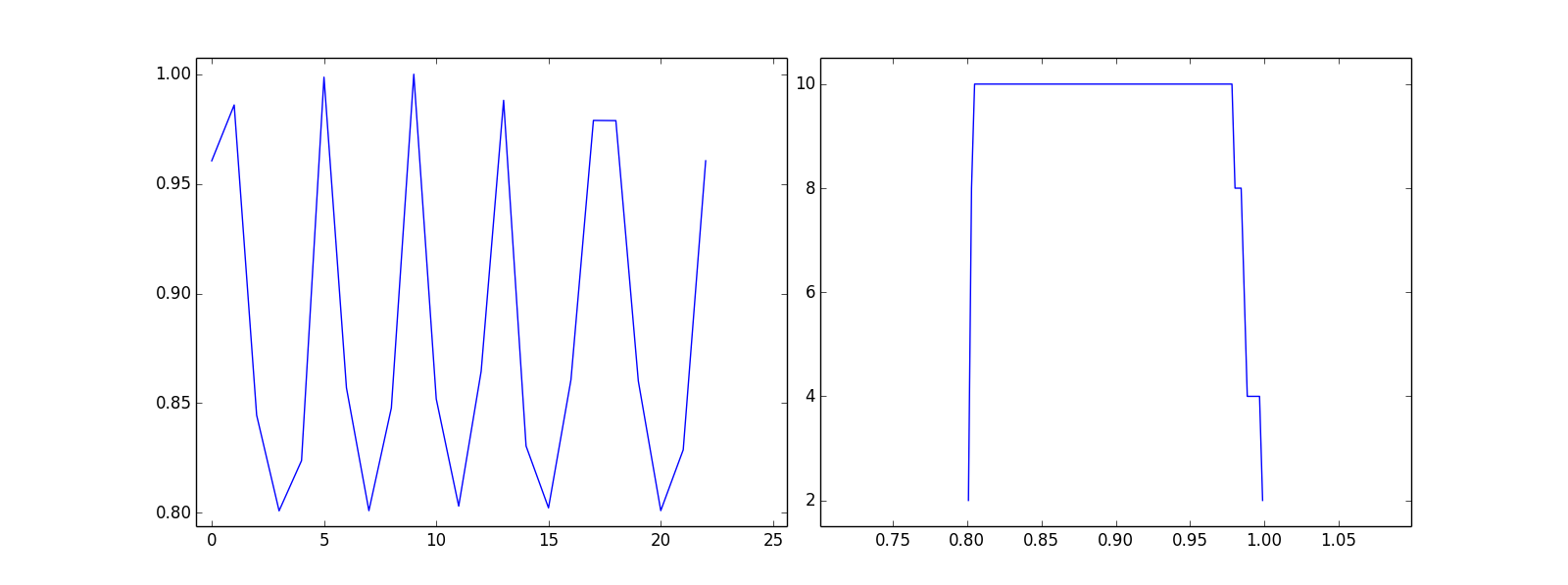
В [5] рассматриваются этот и другие методы поиска центроида в двухбитном изображении.

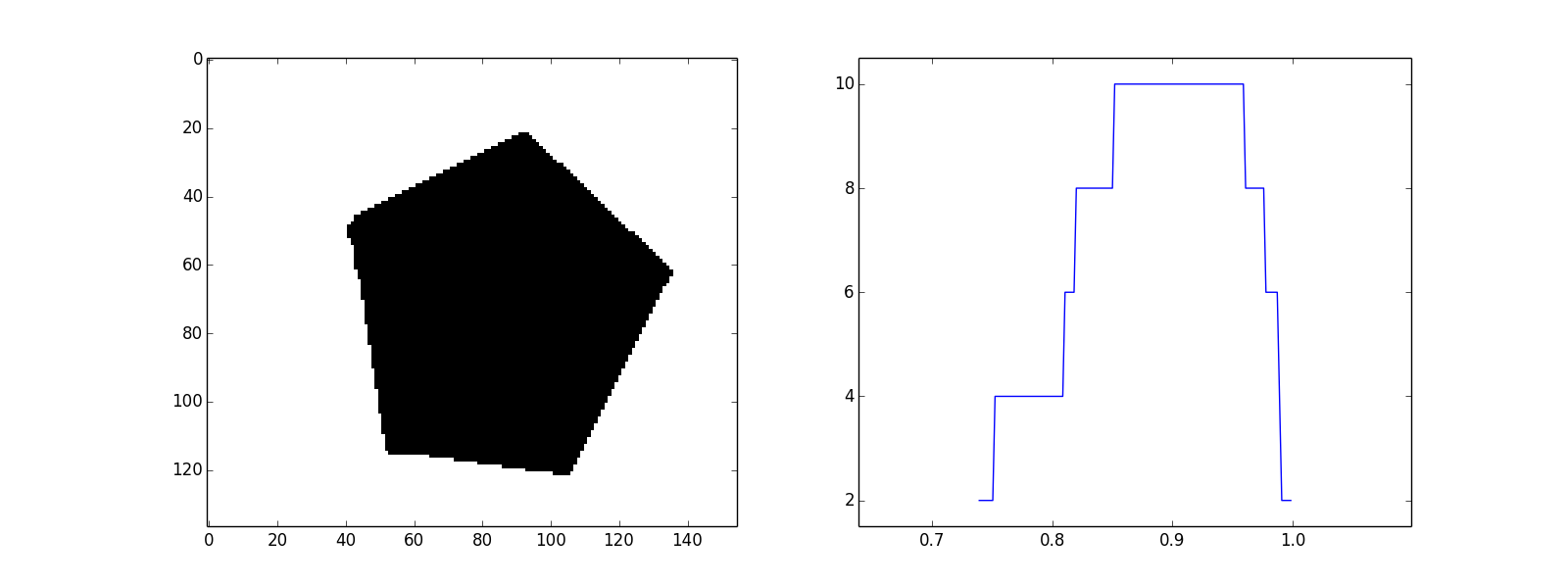
 Далее для каждой точки границы формы считается расстояние до центра масс:

На рисунке изображен график расстояний до границы пятиугольника. Можно видеть на нем пять максимумов и пять минимумов, которые соответствуют вершинам и серединам сторон пятиугольника соответственно. Из этого замечания и следует основная идея алгоритма: следующим шагом считаем, сколько раз встречаются одни и те же расстояния от центра масс до границы. Иначе можно сформулировать вопрос, опираясь на график: сколько раз прямая, параллельная оси X пересекает график.

Однако для того, чтобы ответить на него, необходима предварительная обработка графика. Причиной снова являются «шумы» изображения и неровность границы, в том числе и из-за «растровости» изображения. Для устранения таких эффектов предлагается применить Рамера — Дугласа — Пекера к графику расстояний. Алгоритм рекурсивно делит линию. Входом алгоритма служат координаты всех точек между первой и последней. Первая и последняя точка сохраняются неизменными. После чего алгоритм находит точку, наиболее удалённую от отрезка, соединяющего первую и последнюю. Если точка находится на расстоянии, меньшем чем ε, то все точки, которые ещё не были отмечены к сохранению, могут быть выброшены из набора, и получившаяся прямая сглаживает кривую с точностью не ниже ε. Если же расстояние больше ε, то алгоритм рекурсивно вызывает себя на наборе от начальной до данной и от данной до конечной точках (что означает, что данная точка будет отмечена к сохранению). По окончанию всех рекурсивных вызовов выходная ломаная строится только из тех точек, что были отмечены к сохранению.

Чтобы результат хеширования не зависел от масштаба, отнормируем все расстояния, разделив на максимальное.



На рисунке изображена кривая расстояний после применения алгоритма Рамера — Дугласа — Пекера, а также график частоты пересечений прямой параллельной оси X. Для сравнения, ниже представлен график другого пятиугольника – большего по размеру, повернутого и немного видоизмененного.И сравним их с графиками других простых форм:

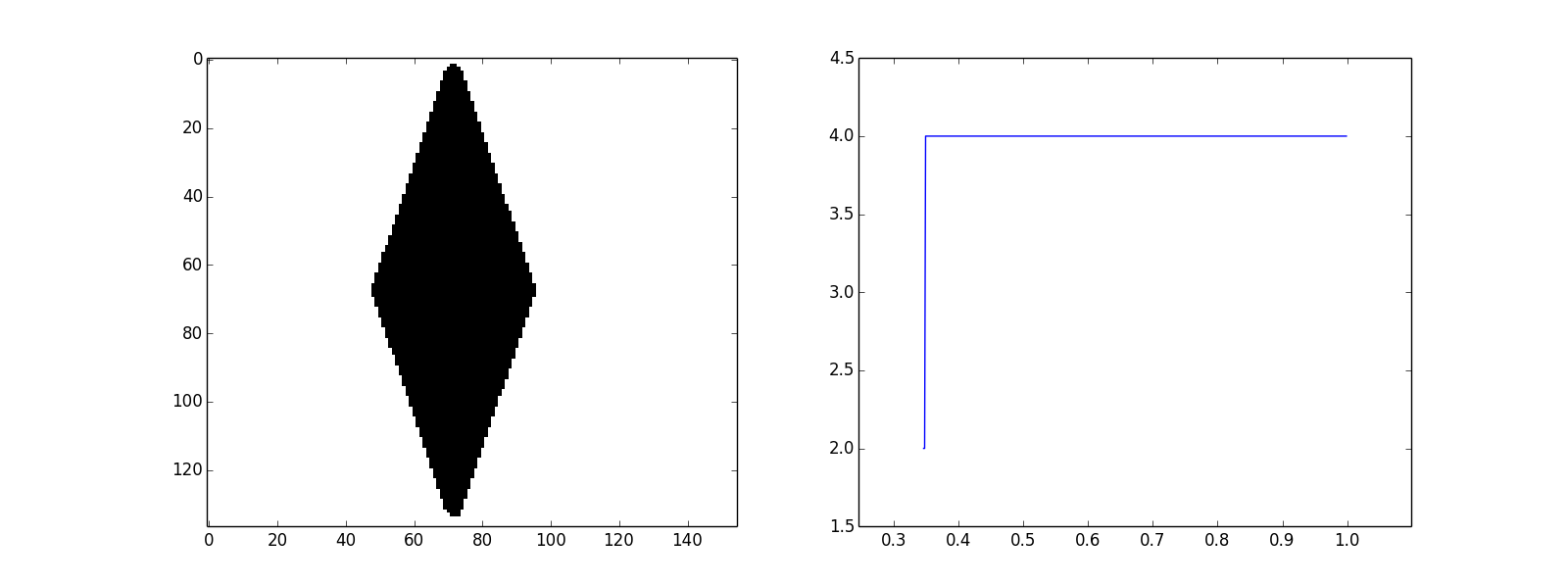
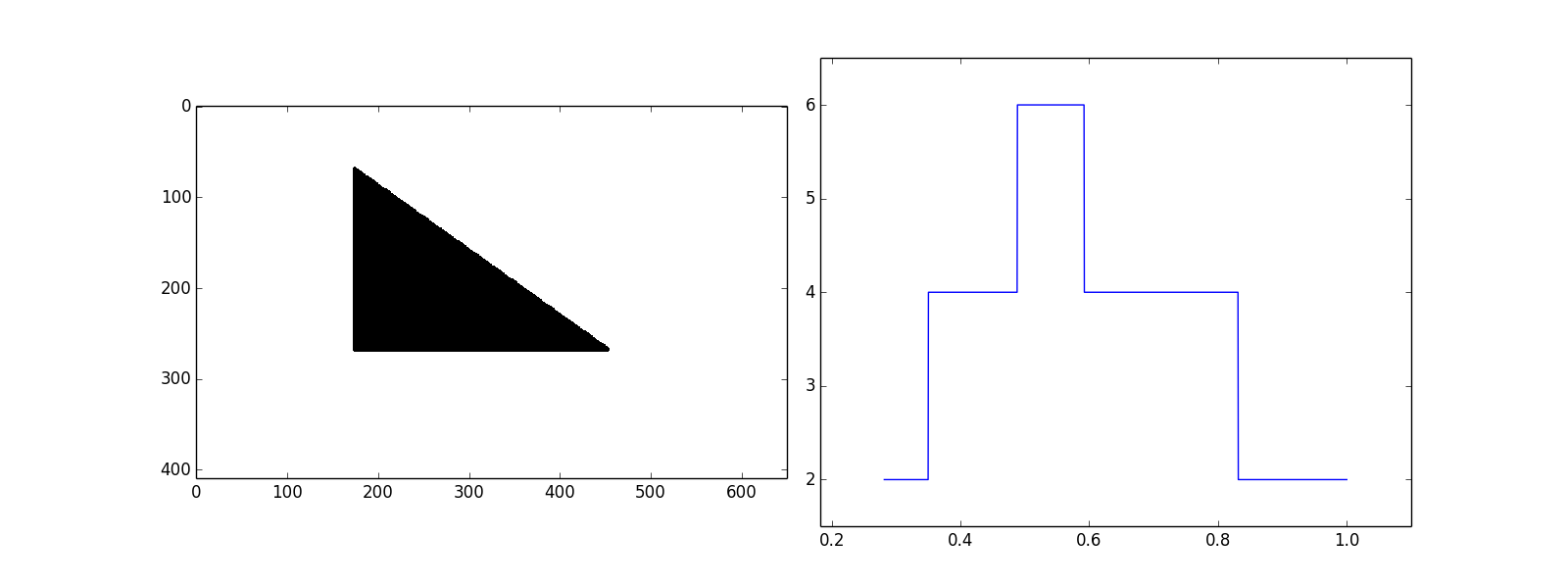


График зависимости частоты встречаемости расстояния от самого расстояния и предлагается взять за хеш. Свойства 2 и 3 мы выполнили при построении алгоритма. Осталось определить правила сравнения хешей. Предлагается мерой близости считать разницу площадей под графиками. Чем она меньше, тем более схожи фигуры.

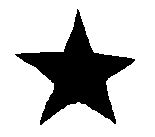
# Выбор алгоритма декомпозиции

К алгоритмам декомпозиции предъявлялись следующие требования в порядке убывания приоритета:

1. Декомпозиция должна совпадать с интуитивным восприятием формы человеком.
2. Части – результаты декомпозиции – должны легко хешироваться предложенным алгоритмом.
3. Скорость работы алгоритма должна быть высокой.
4. Простота реализации.

Глубинный метод.

Данный метод с уверенностью удовлетворяет 2-4 требованиям. При этом центр масс можно заменить уже найденными локальными максимумами удаленности от границы, а во многих случаях эти точки совпадут. Однако возникают проблемы с первым требованиям. Достаточно несложно придумать пример формы, при котором алгоритм не выделит части, которые интуитивно предполагаются. Для этого надо придумать форму, у которой будет один локальный максимум удаленности от границы, при этом были ответвления без подобных локальных максимумов. Примером может служить следующая форма звезды:



У неё один локальной максимум удаленности и следовательно один центр декомпозиции. При этом человек, скорее всего, интуитивно выделит части у этой формы – центр и конечности звезды.

Метод связности.

Этот алгоритм декомпозиции лучше других удовлетворяет первому требованию. При этом благодаря «эллиптичности» декомпозированных частей в них легко выделить центр масс и посчитать хеш. Однако алгоритм оказался очень сложен в реализации. Одна из основных проблем заключается в том, что форма представляется непрерывной в кривой, что сложно гарантировать.

Приблизительно выпуклая декомпозиция

Эксперименты показали, что данный алгоритм чаще всего декомпозирует форму на интуитивно понятные части. Второе требованием выполняется автоматически за счет выпуклости частей (или их приблизительной выпуклости, что легко настраивается варьированием параметра ε метода). То есть из любой точки (в том числе из центра масс формы) мы можем провести отрезок до другой точки формы (в нашем случае до её границы), и он будет лежать внутри формы. Также проводились эксперименты с производительностью данного алгоритма, при которых он хорошо себя показал, за счет возможности варьировать параметра декомпозиции, такие как ε «приблизтельность» декомпозиции.

Таким образом, выбор был сделан в пользу алгоритма приблизительно выпуклой декомпозиции.

# Особенности реализации

Описанные алгоритмы были реализованы на языках Python и C++. Для демонстрации алгоритма глубинной декомпозиции и алгоритма хеширования был создан веб-сайт. На нем пользователь может загрузить черно-белое изображение и увидеть результат декомпозиции и распознавание декомпозированных частей с помощью сравнения их хешей.

Веб-сайт был написан с использованием фреймворка Django. Он позволяет удобно выделить все алгоритмы анализа изображений в отдельное приложение, что позволяет на ранних этапах работать с ними напрямую и быстро, и одновременно в нужный момент легко встроить их в MVC структуру фреймворка.

В качестве СУБД используется MySQL. Управлением запросов занимается Apache HTTP-сервер. В качестве посредника между сервером и Django выступает модуль mod\_wsgi, который предоставляет WSGI-совместимый интерфейс для работы с web-приложениями, написанными на языке программирования Python. В роли серверной машины выступает облачный виртуальный сервер от провайдера DigitalOcean.

Для обработки изображений использовался пакет scikit-image. Реализация алгоритма приблизительно выпуклой декомпозиции была подготовлена при взаимодействии с Khaled Mamou, чья реализация алгоритма декомпозиции используется в таких крупных проектах 3D визуализации, как Unreal Engine 4.

# Заключение

В данной работе были рассмотрены алгоритмы декомпозиции форм. Также был предложен новый метод декомпозиции. Проведено сравнение этих методов. Предложен способ хеширования простых 2-битных изображений для их быстрого сравнения. Реализованы все описанные алгоритмы, и представлен веб-интерфейс для их тестирования.

Следующим шагом является исследование способа хранения и быстрого поиска сложной формы целиком на основе хешей декомпозированных частей. Здесь наиболее сложной является задача сравнения графов декомпозиции с вершинами, имеющими характеристики в виде хешей.

Также планируется усовершенствовать веб-сайт. Будут добавлены новые методы декомпозиции форм и анализа изображений вообще.

# Список литературы

[1] Местецкий Л.М. Непрерывная морфология бинарных изображений: фигуры, скелеты, циркуляры. – М.:ФИЗМАТЛИТ, 2009. – 288 с. – ISBN 978-5-9221-1050-1.

[2] Hairong Liu, Wenyu Liu, Longin Jan Latecki. Convex Shape Decomposition. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2010.

[3] Manish Singh, Donald D. Hoffman. Completing visual contours: The relationship between relatability and minimizing inflections. Perception & Psychophysics 1999, 61 (5), 943-951

[4] Xiaofeng Mi, Doug DeCarlo. Separating Parts from 2D Shapes using Relatability. ICCV 2007: 1-8

[5] Zhang Hai-yan, Wang Dong-mu, Song Ke—ou. An improved fast algorithm for searching centroid of object in binary image. Proc. SPIE 5286, Third International Symposium on Multispectral Image Processing and Pattern Recognition, 859 (September 29, 2003); doi:10.1117/12.538711; http://dx.doi.org/10.1117/12.538711