# Алгоритм сеточной векторизация пиксельных изображений

## Постановка задачи

На вход поступает двумерное пиксельное монохромное изображение типа контурной карты. Изображение состоит из областей, ограниченных замкнутыми линиями. Все незамкнутые линии в данной версии алгоритма будут проигнорированы. Линии предполагаются непрерывными и тонкими. В дальнейшем цвет линий будем называть черным, а цвет фона – белым.

Результатом векторизации должно быть множество полигонов, заданных своими точками. Координаты точек – целочисленные, измеряются в пикселях, начало координат – в верхнем левом углу изображения.

Алгоритм векторизации имеет параметр, от которого зависит точность аппроксимации замкнутых линий исходного изображения полигонами. Чем выше точность аппроксимации, тем больше точек должно содержать векторное изображение, и тем больше его объем. Параметр позволяет достичь желаемого компромисса между точностью и объемом векторного изображения.

## Общая схема алгоритма

На изображение накладывается сетка с квадратными ячейками (размер ячейки и является параметром алгоритма). (рис)

Вначале определяется, какие узлы сетки попадают внутрь каждой области исходного изображения. Далее для каждой области находится множество точек пересечения линий сетки с линиями, окружающими область, эти точки и станут узлами аппроксимирующих полигонов. На последнем этапе неупорядоченные множества точек выстраиваются в последовательности, составляющие полигоны.

## Условия применимости алгоритма

Начнем с того, что параметр алгоритма регулирует отношение точность-объем только до определенного предела своего значения. Начиная с этого предела, алгоритм перестает работать устойчиво, т.е. сопоставлять одному контуру исходного изображение один аппроксимирующий полигон.

Нет возможности определить это предельное значение численно, но можно сказать, что величина ячейки сетки не должна быть большой. Более формально можно выразить это как требование, чтобы одна контурная линия не пересекала границы одной ячейки более двух раз. В сущности, формулировка не имеет значения, т.к. неясно, каким образом проверить такое или подобное условие, не выполнив сам алгоритм. (рис)

## Распределение узлов сетки по областям

Для того, чтобы определить принадлежность узла сетки той или иной области, воспользуемся вариантом алгоритма поиска компонент связности в неориентированном графе. В данной интерпретации узлы сетки – это узлы графа. Если два соседних узла сетки не разделены контурной линией, значит, между ними есть ребро. Иными словами, если сторона ячейки сетки не пересекается с контурной линией, она является ребром графа. Установить факт пересечения можно, просмотрев все пиксели изображения, лежащие вдоль стороны ячейки.

Регулярность нашего графа позволяет воспользоваться упрощенной версией алгоритма поиска компонент связности.

Просматриваем поочередно все узлы сетки в направлении слева направо и сверху вниз, и каждый узел включаем в одну из компонент связности. Для этого у очередного узла сетки проверяем наличие двух ребер: ведущего к верхнему и ведущего к левому соседу.

Если ребер нет, создаем новую компоненту связности, состоящую из одного этого узла.

Если есть ровно одно ребро, включаем очередной узел в ту компоненту, к которой принадлежит смежный с ним узел.

Если существуют оба ребра и смежные узлы принадлежат одной компоненте связности, поступаем, как в случае с одним ребром.

Если соседи данного узла принадлежат разным компонентам связности, объединяем эти компоненты и включаем очередной узел в эту объединенную компоненту связности.

Верхняя оценка временной сложности алгоритма – O(n2), где n – количество узлов сетки. В самом деле, всего узлов n, наиболее трудоемкой операцией для каждого узла является объединение двух компонент связности в одну (соединение двух коллекций), а оно не может быть сложнее, чем O(n).

## Определение точек полигонов

Сначала рассмотрим идеальный случай – все линии исходного изображения тонкие. Тогда точки полигонов будут располагаться на пересечении линий сетки с линиями изображения. Для определения точки пересечения последовательно просматриваем все пиксели отрезка сетки между двумя узлами. Там, где находится пиксель цвета линии и есть точка пересечения.

Каждая точка пересечения ребра графа лежит на границе двух областей, и значит, принадлежит двум полигонам, аппроксимирующим эти области. Таким образом, нетрудно найти все точки каждого полигона, и остается определить порядок следования точек в полигонах.

## Определение порядка следования точек

Берем произвольную точку полигона и помещаем ее в очередь. Находим точку, ближайшую к последней точке очереди, и помещаем ее в очередь. Повторяем это действие, пока есть еще точки, не поставленные в очередь. Если условие применимости алгоритма выполнено, т.е. размер ячеек не слишком велик, все точки, лежащие на границе области, окажутся в очереди и составят полигон (первая точка смыкается с последней точкой очереди).

Если условие применимости не выполнено, возможны ошибки, которые можно разделить на петли и потерянные участки. (рис.) Ошибки можно исправить при помощи корректирующих процедур.

## Толщина контурных линий

Из-за того, что линии матричного изображения бывают толстыми, некоторые узлы сетки оказываются целиком окруженными черными пикселями. (рис) Это приводит к возникновению ложных компонент связности и, как следствие, к искажению контуров.

Выход в выносе узлов сетки за пределы контурных линий, иными словами, в переносе узла на какой-то белый пиксель, имеющийся в его окрестности. В частности из двух направлений переноса узла (вверх и влево от его законного положения) выбирается то, на котором первый белый пиксель встретится раньше.

Линии сетки, соединяющие смещенные узлы уже не параллельны осям координат и исследование пикселей, через которые они проходят, выполняется алгоритмом Брезенхайма (хотя на фоне общих затрат, получаемая от этого экономия не велика).

Другое следствие ненулевой толщины контурных линий то, что определять положение разделительной точки лучше не по первому встреченному черному пикселю, а по среднему между первым и последним черными пикселями.

## Незамкнутые контуры

Описанная процедура находит точки, принадлежащие и незамкнутым контурам тоже.

Одна компонента связности может содержать несколько незамкнутых линий. Сборка точек в незамкнутые линии отличается тем, что процедуру поиска ближайших точек нужно повторять дважды, для первой и второй части линии. Нужно иметь в виду, что линий в одной компоненте связности может быть более одной.

Для построения второй части линии нужно повторить процедуру поиска ближайших точек, еще раз начав с первой выбранной точки.