# Алгоритм сеточной векторизация пиксельных изображений

## Постановка задачи

Векторизации подвергается двумерное пиксельное монохромное изображение типа контурной карты. Изображение состоит из непересекающихся областей, ограниченных замкнутыми линиями. Линии исходного изображения будем называть контурными линиями или просо контурами. Для определенности будем считать линии черными, а фон – белым.

Результатом векторизации будет множество полигонов, заданных своими точками. Координаты точек – целочисленные, начало координат – в верхнем левом углу изображения, единица измерения – пиксель.

Точность аппроксимации контуров полигонами определяется шагом сетки, накладываемой на изображение. Величина этого шага и является параметром алгоритма.

## Общая схема алгоритма

На изображение накладывается сетка с квадратными ячейками. Вначале определяется, какие узлы сетки попадают внутрь каждой области. Далее для каждой области находится множество точек пересечения линий сетки с контурами область. Эти точки станут узлами аппроксимирующих полигонов. На последнем этапе неупорядоченные множества точек выстраиваются в последовательности, составляющие полигоны.

## Условия применимости алгоритма

Алгоритм применим для изображений, линии которых являются плавными. Иными словами, линия изображения, как правило, не пересекает сторону ячейки сетки более одного раза. Это правило не должно нарушаться и при параллельном переносе сетки.

Другим условием является то, что линии изображения должны быть тонкими. Впрочем, оба эти требования не являются абсолютными, просто качество аппроксимации будет лучше, если исключений из правил будет меньше.

## Распределение узлов сетки по областям

Чтобы определить принадлежность узла сетки той или иной области, воспользуемся алгоритмом поиска компонент связности в неориентированном графе. В данной интерпретации узлы сетки – это узлы графа. Ребра графа соединяют соседние узлы, но только те, которые не разделены контурной линией. Иными словами, если сторона ячейки сетки не пересекается контурной линией, она является ребром графа.

Регулярность нашего графа позволяет воспользоваться упрощенной версией такого алгоритма.

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, т.е. узла графа, 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 - включаем очередной узел в компоненту, к которой принадлежат оба смежных узла. Если соседи данного узла принадлежат разным компонентам связности, объединяем эти компоненты в одну 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 компонент связности, а оно не может быть сложнее, чем O(n).

## Определение точек полигонов

Точки полигона будет располагаться на пересечении линий сетки с контурными линиями. Для определения точки пересечения последовательно просматриваем все пиксели отрезка сетки между двумя узлами. Черный пиксель на линии сетки и будем считать местом пересечения.

Возможна ситуация, когда на отрезке линии сетки таких пикселей более одного. В этом случае возьмем средний из них.

В итоге для каждой замкнутой области мы получим неупорядоченное множество окружающих ее точек.

## Сборка полигонов

Неупорядоченное множество точек, окружающих каждую область, следует выстроить в полигон. Для этого возьмем любую точку и сделаем ее первой точкой полигона. Следующей точкой полигона сделаем ближайшую к ней и т.д., пока все точки не будут выстроены. Если требования к применимости алгоритма выполняются, то процесс построения полигона успешно завершиться. В реальности возможны искажения, которые заключаются в образовании петель и потере отростков. Искажения поддаются коррекции, для каждого вида своя процедура.

## 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.

## Толщина контурных линий

Из-за того, что линии матричного изображения бывают толстыми, некоторые узлы сетки оказываются целиком окруженными черными пикселями. 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 ближайший белый пиксель. В частности из двух направлений переноса узла (вверх и влево от его законного положения) выбирается то, на котором первый белый пиксель встретится раньше.

Если узел сетки смещен, путь к нему из соседнего узла прокладывается буквой Г (как в Манхеттене). При этом четные пиксели распознаются только на первой ветке буквы.

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.

## Приложения алгоритма