# Алгоритм сеточной векторизация пиксельных изображений

Пиксели контуров будем называть черными, все остальные - белыми.

## Постановка задачи

Векторизации подвергается двумерное пиксельное монохромное изображение типа контурной карты. Изображение состоит из непересекающихся областей, ограниченных замкнутыми контурными линиями. Все незамкнутые линии в данной версии алгоритма будут проигнорированы.

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

Параметром алгоритма является точность аппроксимации линий исходного изображения ломаными линиями полигонов. Эта точность определяется шагом сетки, накладываемой на изображение, размер 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 линиями, окружающими 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 не должна пересекать одну сторону ячейки сетки более одного раза.

Другим условием является то, что линии изображения должны быть тонкими. Но, как будет видно дальше, это ограничение может быть ослаблено.

## 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. Ребра графа соединяют соседние узлы, но только те, между которыми не проходит контурная линия. Иными словами, если сторона ячейки сетки не пересекается с 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 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, наиболее трудоемкой операцией для каждого узла является объединение компонент связности, а оно не может быть сложнее, чем O(n).

## Определение пограничных точек

## Идеальный случай – линии тонкие

## Определение контуров, как компонент связности

## Реальный случай – линии толстые. Коррекция узлов сетки

## Сборка контуров

## Идеальный случай, линии плавные

## Реальный случай, линии изломанные

На первом этапе определяем компоненты связности графа. Одна компонента связности - это множество вершин, лежащих внутри замкнутого контура исходного изображения. Но поскольку наша цель - аппроксимировать этот контур ломаной, попутно с определением компонент связности находим точки, принадлежащие этой ломаной. Такими точками будут черные пиксели, обнаруженные нами в процессе определения ребер графа.

На каждом отрезке между двумя соседними узлами сетки таких точек будет не более одной (первый черный пиксель, найденный при прохождении отрезка). Помимо местоположения этих точек, важно знать их принадлежность к контуру (компоненте связности), точнее к паре смежных контуров.

Поскольку во время обнаружения точек результат определения компонент связности еще не готов, временно будем привязывать точки к паре узлов сетки (будем называть эту пару узлов охраняющими). По окончании определения к.с. графа по этой привязке можно будет определить, к каким контурам принадлежит конкретная точка.

## Алгоритм определения к.с.

Просматриваются все узлы слева направо, сверху вниз. Для каждого узла определяется его смежность с верхним и с левым соседом, если таковые имеются (узлы смежные, если на соединяющей их линии нет черных пикселей).

Если текущий узел не смежен ни с одним соседом, он один составляет новую временную компоненту связности.

Если текущий узел смежен только с одним соседом, он включается в компоненту соседа.

Если текущий узел смежен с обоими соседями, он включается в компоненту левого соседа,

и компонента верхнего соседа объединяется с компонентой левого.

Во время выполнения процедуры все компоненты связности считаются временными, но по окончании процедуры все временные компоненты делаются постоянными.

## Создание разделительных точек

Попутно с определением компонент связности создаем разделительные точки, из которых позднее будем строить контуры векторного изображения. В процессе выяснения смежности мы проверяем все пиксели на отрезке, соединяющем узлы. Когда на этом отрезке встречается черный пиксель, создаем разделительную точку с координатами этого пикселя и ссылками на два узла по концам отрезка.

## Распределение точек по контурам

На следующем этапе точки контуров, найденные ранее, выстраиваются в ломаные, аппроксимирующие линии исходного изображения.

Все множество точек разбиваем на подмножества таких точек, что относятся к одной компоненте связности. Для этого просматриваем все точки и для каждой определяем ее к.с. опосредованно через охраняющие точку узлы.

Точки на замкнутых линиях удвоятся и попадут в две разные компоненты, точки на незамкнутых линиях - только в одну.

## Сборка контуров

Неупорядоченное множество точек, относящихся к одному контуру, следует выстроить в линию. Для этого возьмем любую точку и сделаем ее первой точкой контура. Следующей точкой контура будет ближайшая к ней и т.д., пока все точки не будут выстроены.

Легко понять, что расстояние между соседними точками контура не превышает диагонали ячейки сетки. Если расстояние до ближайшей точки превышает диагональ ячейки, то что-то пошло не так.

Точки замкнутого контура обязательно выстроятся в линию, т.е. расстояние между первой и последней будет не больше диагонали ячейки.

Если контур не замкнутый, то таким алгоритмом можно выстроить лишь часть линии, от первой точки до одного из концов линии.

## 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.