Санкт-Петербургский Государственный Университет

Математико–механический факультет

Кафедра системного программирования

Векторизация контура при распознавании UML-диаграмм

Курсовая работа студентки 361 группы

Анускиной Ирины Михайловны

Научный руководитель: Осечкина М.С.

Санкт-Петербург

2013

Оглавление

Введение

Диаграммы широко используются при разработке технической документации, научных статей, бизнес-отчетов и других типов документов.

Создание диаграмм является неотъемлемой составляющей любого UML-средства.

В CASE-системах диаграммы имеют под собой логическую модель, что позволяет их редактировать, оптимизировать и генерировать по ним код.

На сегодняшний день существует много редакторов, которые позволяют рисовать UML-диаграммы. Однако, как показывает практика, рисовать их от руки гораздо удобней и быстрее. Было бы полезно впоследствии нарисованную диаграмму использовать, например, построить по ней логическую модель и загрузить в CASE-систему для последующей генерации кода.

В рамках студенческого проекта «Распознавание нарисованных UML-диаграмм», проходящего на кафедре системного программирования, разрабатывается инструмент, который распознает отсканированную или сфотографированную UML-диаграмму, с возможностью разбиения диаграммы на объекты и выделением элементов и связей между ними.

Одним из этапов распознавания является векторизация контура.

Векторизация – это процесс построения векторной модели растрового изображения. Такая модель является предпочтительней, когда над изображением предполагается проводить некоторый анализ. Статическое распознавание предполагает выделение объектов на растровом изображении без какой-либо информации о том, в каком порядке пользователь рисовал объекты, что усложняет задачу. Поэтому гораздо удобней выделять объекты, имея последовательность векторов, с помощью которых представлена картинка, нежели просто набор пикселей.

На вход инструмента распознавания подается фотография или скан. Далее предполагается выделение контура от фона. Таким образом, на этапе векторизации мы имеем двухцветное черно-белое растровое изображение.

Процесс векторизации усложняется наличием на изображении линий произвольной ширины, разветвлений, пересечений и шумов.

В данной работе описана реализация алгоритма векторизации контура произвольной ширины для инструмента распознавания нарисованных UML-диаграмм.

Постановка задачи

В рамках данной работы были поставлены следующие задачи:

1) Изучить существующие алгоритмы векторизации, их преимущества и недостатки и выбрать тот из них, который удовлетворяет следующим требованиям:
    - Возможность корректной обработки разветвлений на растре;
    - Возможность частичного разбиения диаграммы на объекты;
    - Отсутствие ложных ответвлений;
2) Реализовать алгоритм векторизации для контура произвольной ширины, что позволит распознавать снимки диаграммы, нарисованной как ручкой на листке бумаги, так и маркером на доске;
3) Проанализировать, удовлетворяет ли результат векторизации вышеперечисленным требованиям в зависимости от различных характеристик входного изображения – ширины линий, внешних шумов, размытия.

Обзор существующих алгоритмов

На сегодняшний день векторные модели распространены в различных информационных системах. Так отсканированные изображения карт используются в географических системах, сканы инженерных чертежей используются в системах автоматизированного проектирования. Поэтому на сегодняшний день разработано множество подходов к векторизации.

Алгоритмы утончения (thinning algorithms)

Данное семейство алгоритмов [1] предполагает итеративное удаление внешнего слоя пикселей на растре, до тех пор пока не получится однопиксельный скелет. Решение об удалении пикселя принимается при помощи так называемого шаблона (рис. 1).

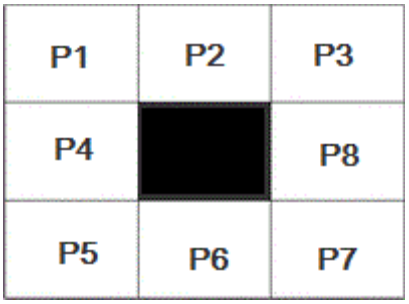

Рис. 1

В центре находится рассматриваемый пиксель. Каждый из восьми соседних пикселей либо принадлежит контуру, либо является фоновыми. В зависимости от комбинаций соседних пикселей и принимается решение об удалении текущего пикселя.

Рис. 2

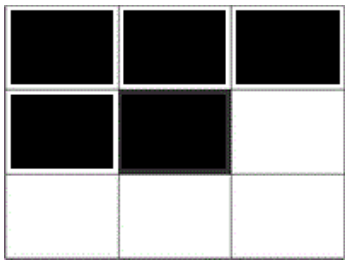

Рис. 3

В первом случае (рис. 2) удаление пикселя невозможно, поскольку это приведет к разрыву линии, во втором случае (рис. 3) пиксель может быть удален.

Алгоритмы такого рода подразделяются на два множества – параллельные и последовательные. В последовательных алгоритмах пиксели проверяются на удаление в фиксированном порядке на каждой итерации, и удаление пикселя p на n-ой итерации зависит от результатов всех предыдущих n-1 итераций. В параллельных алгоритмах удаление пикселя на n-ой итерации зависит только от результатов текущей итерации, таким образом все пиксели могут проверяться параллельно на каждой итерации.

Основным недостатком подобного рода алгоритмов является временная сложность, поскольку на каждой итерации происходит обращение к каждому отдельному пикселю. При

больших изображениях время обработки значительно возрастает, его можно оценить как O(wN), где w – ширина линии, N – общее число пикселей на растре.

На выходе алгоритмов утончения получается однопиксельный скелет, по которому в дальнейшем необходимо построить ломанные.

При изучении был реализован один из параллельных алгоритмов утончения – Zang-Suen algorithm [2]. Обработка поданного на вход изображения (рис. 4) составила 6 секунд.

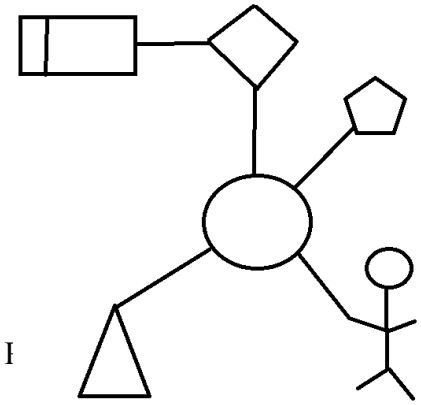

Рис. 5

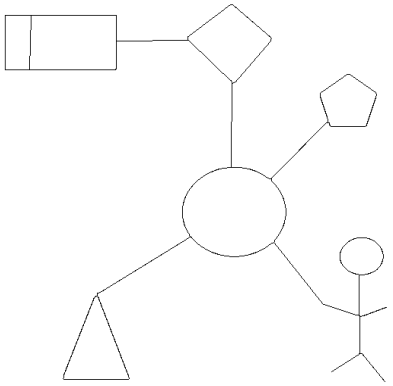

Рис. 4

Получившийся скелет изображен на рисунке 5.

Алгоритм отслеживания контура (contour following algorithms)

Данные алгоритмы [1] позволяют находить границы объектов на изображении. Результат работы такого рода алгоритмов – последовательность пикселей, которая описывает замкнутую часть пикселей на границах объектов (рис. 6). По этим последовательностям в дальнейшем могут быть вычислены ломанные, которые окружают объекты на растре.

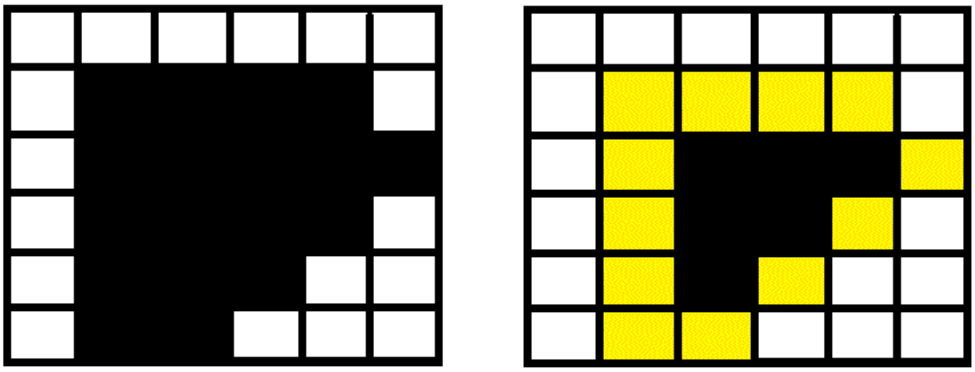

Рис. 6

Существенным недостатком таких алгоритмов являются ошибки при разветвлении. Так при обработке растра, изображенного ниже (рис. 7), в результате получим скелет, в котором будет отсутствовать линия, находящаяся внутри фигуры (рис. 8).

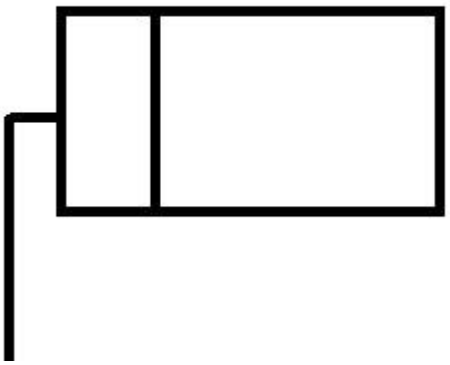

Рис. 8

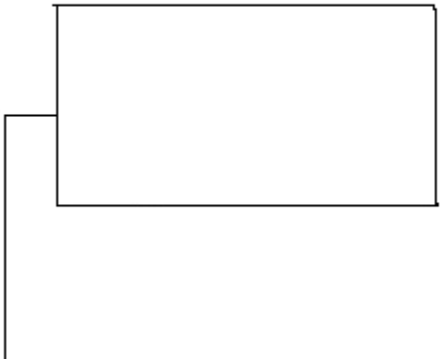

Рис. 7

Median line determination algorithms

Существует ряд алгоритмов [1], которые позволяют определять так называемые медианы линий на растре. Это можно делать с помощью вычисления для каждого пикселя дистанционного расстояния до первого фонового пикселя. После вычисления дистанционного расстояния для каждого пикселя на растре находятся локальные максимумы, которые образуют скелет изображения.

Помимо высокой временной сложности, которая пропорциональна количеству пикселей на растре, к недостаткам подобного рода алгоритмов относится низкое качество скелета, который может оказаться несвязным множеством, что усложнит построение векторной модели.

Описанные выше подходы обладают рядом преимуществ и недостатков. Все алгоритмы достаточно просты в реализации, однако данные получаемые на выходе требуют дальнейшей обработки. Основные особенности рассмотренных подходов представлены ниже (табл. 1).

	Thinning	Contour following	Median determination
Временная сложность	Высокая - O(wN) w – ширина линии N – число пикселей	Пропорциональна количеству пикселей на растре	Пропорциональна количеству пикселей на растре
Ложные ответвления	Порождение ложных ответвлений	Ошибки при разветвлении	Порождение ложных ответвлений
Качество скелета	Связанный скелет	Связанный скелет	Несвязный скелет

Табл. 1

Реализованный алгоритм

По итогам анализа преимуществ и недостатков существующих подходов, для реализации был выбран алгоритм векторизации [3], который совмещает в себе подходы описанные в предыдущем разделе. Алгоритм работает для контура произвольной ширины, что позволяет избавиться от предварительного утончения, требующего высоких временных затрат. Предполагается обход контура с обеих сторон и одновременное вычисление медианных значений линий на растре. Алгоритм позволяет корректно обрабатывать случаи разветвления и продолжать векторизацию по всем распознанным направлениям, игнорируя при этом ложные ответвления. На выходе алгоритма имеется набор уже построенных ломанных. Немаловажной причиной, по которой был выбран алгоритм, является то, что он уже на стадии векторизации позволяет осуществить сегментацию и представить диаграмму как набор линий.

Входные данные

На вход алгоритма подается двухцветное растровое изображение, стартовое значение, с которого требуется начать процесс векторизации, а также направление векторизации.

Отслеживание контура

Алгоритм предполагает обход контура с обеих сторон. Имея стартовое значение, вычисляется крайний левый – L_0 и крайний правый – R_0 пиксели, принадлежащие контуру. Они являются стартовыми значениями для процедуры обхода контура (рис. 9).

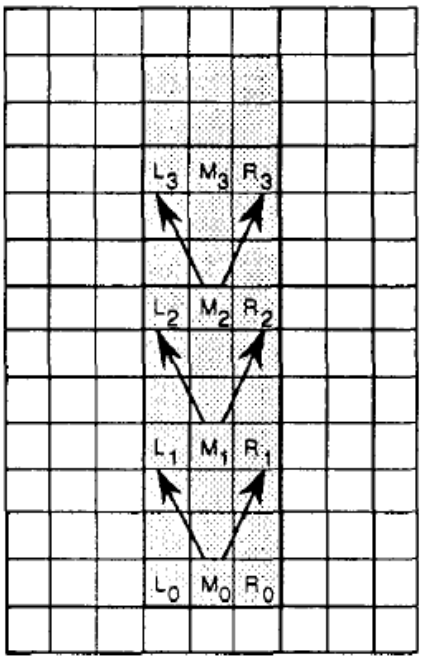

Рис. 9

Последующие значения L_1, L_2, и т.д., а также R_0, R_1, и т.д. вычисляются следующим образом:

- Для текущего контурного пикселя рассматриваются его соседи (рис. 10);

- В случае обхода правого контура соседние пиксели просматриваются против часовой стрелки, начиная с первого фонового пикселя, до первого не фонового – он и становится следующим контурным пикселем;
- В случае же левого обхода, соседние пиксели просматриваются по часовой стрелки, аналогично выбирается следующий контурный пиксель.

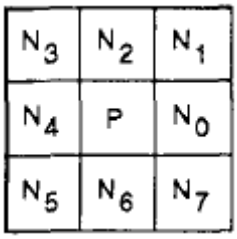

Рис. 10

Имея набор таких пар (L_k, R_k) можно определить так называемые медианные значения M_k, которые вычисляются как среднее между L_k и R_k. Эти значения будут использоваться в качестве выходных данных реализуемого алгоритма.

Обработка разветвлений

Согласно алгоритму, обход контура и вычисление соответствующих медианных значений осуществляется для одной линии. Поэтому когда есть разветвление необходимо определить все возможные направления и продолжить обход по всем из них, а полученные результаты объединить.

Критерием для определения разветвления является тот факт, что текущее медианное значение стало равным предыдущему медианному значению.

Для определения всех направлений разветвления используется метод построения концентрических окружностей. Строятся окружности с центром в точке разветвления и радиусом равным расстоянию до предыдущего медианного значения. Таким образом, число окружность есть число вычисленных медианных значений (рис. 11).

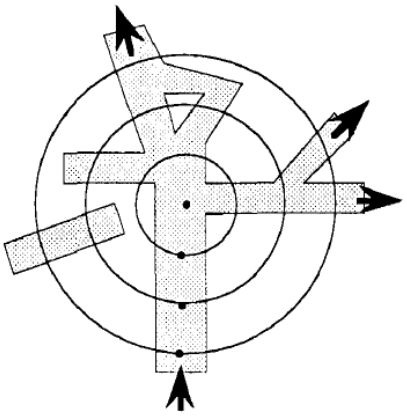

Рис. 11

После того как окружности построены, каждая из них обходится против часовой стрелки, начиная с медианного значения, расположенного на окружности. В ходе этого обхода сохраняются все нефоновые пиксели кроме первого и последнего. Сохраненные пиксели группируются в пары противоположных значений. Так, на рис. 12 парами противоположных значений для последней окружности будут – (O,P), (Q,R), (S,T), (U,V).

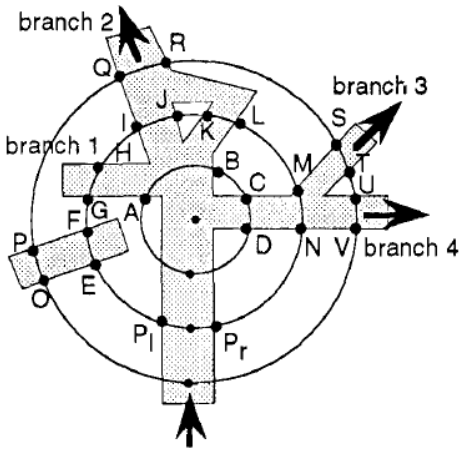

Рис. 12

После этого осуществляется процедура обхода контура по часовой стрелке, начиная с некоторого контурного пикселя уже обработанной линии. Если встречается пиксель соответствующий первому значению одной из вышеупомянутых пар противоположных значений, то количество распознанных веток увеличивается и сохраняется, пока не встретится пиксель соответствующий противоположному значению одной из пар. Тем самым определяются все возможные ветки и пары им принадлежащие, а также игнорируются отдельные несвязанные объекты на растре.

Дальнейший процесс векторизации проходит лишь по тем веткам, которые пересекли последнюю окружность. Это позволяет избежать обработки небольших ответвлений.

Стартовым значением для векторизации распознанных веток будет служить среднее между значениями пары, расположенной на последней окружности.

Критерий остановки

Критерием остановки алгоритма является тот факт, что при текущих L_k и R_k – левом и правом контурном пикселе при обходе контура выполняются равенства $L_k = R_{k-1}$ и $R_k = L_{k-1}$.

Еще одним критерием остановки служит тот факт, что текущее медианное значение стало равным стартовому значению, т.е. произошло замыкание линии.

Эксперименты

Ниже приведены результаты векторизации. На рис. 13 изображен снимок диаграммы, нарисованной на обычном листке бумаги, на рис. 14 представлен результат выделения контура соответствующего снимка, на рис. 15 - результат работы алгоритма векторизации.

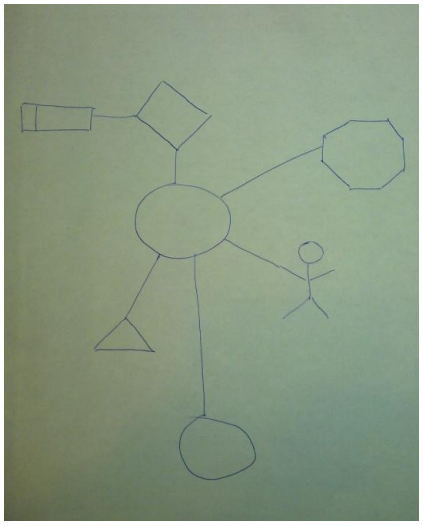

Рис. 13

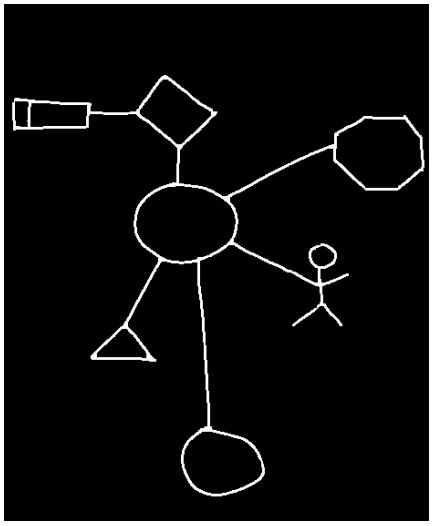

Рис. 14

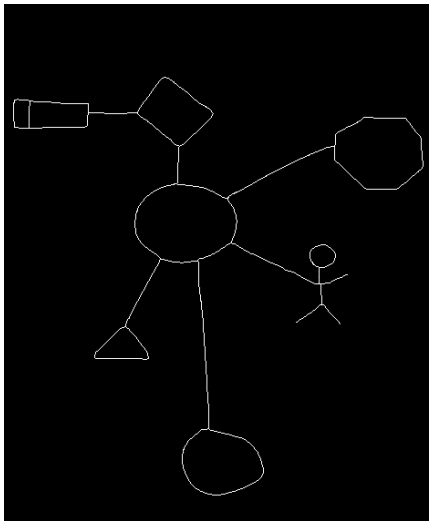

Рис. 15

На рис. 16 изображен снимок диаграммы, нарисованной мелом на обычной доске, далее аналогично представлены результаты выделения контура и векторизации (рис. 17 и рис. 18).

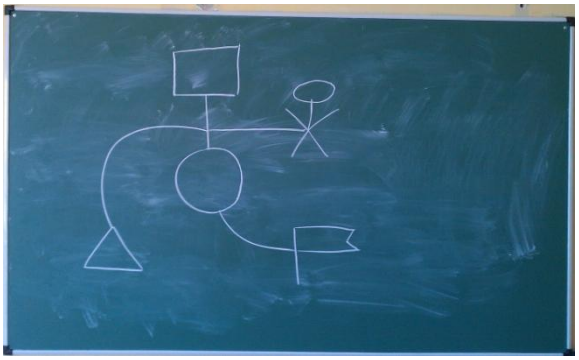

Рис. 17

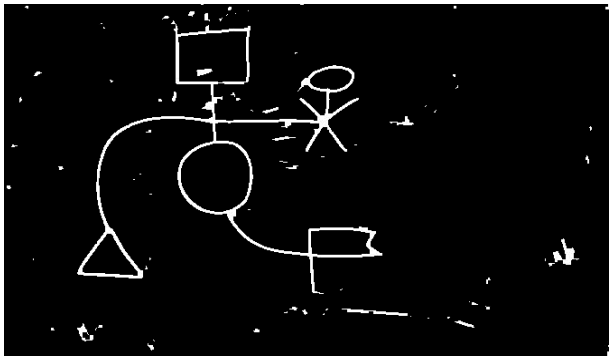

Рис. 16

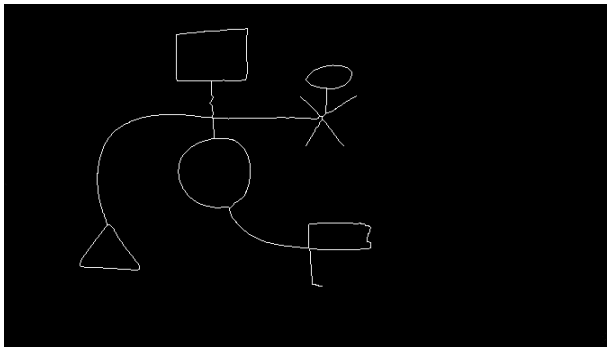

Рис. 18

В ходе экспериментов была выявлена проблема, связанная с векторизацией замкнутых объектов (круги, прямоугольники). В случае векторизации в одном направлении такого рода фигуры будут представлены в виде двух ломанных, что неудобно для последующей сегментации диаграммы. Можно также осуществить векторизацию в двух направлениях, однако в этом случае придется отслеживать точки, в которых два направления векторизации пересекутся во избежание повторной векторизации уже пройденного участка. Однако было принято решение проводить векторизацию в одном направлении и склеивать ломанные представляющие замкнутые объекты.

Еще одна проблема возникла при отслеживании разветвлений. При построении концентрических окружностей, на рассмотрение берутся только те ветки, которые пересекают последнюю окружность. Однако это неприемлемо, в том случае, когда последняя обработанная линия слишком большая, как правило это связь. В этом случае радиус последней концентрической окружности окажется большим, и окружность будет огибать объекты на диаграмме, не пересекая их. В этом случае необходимо рассматривать последнюю окружность, которая пересекла объекты на растре и соответствующие ей ветки.

Еще одной проблемой оказался критерий остановки алгоритма. Выяснилось, что критерий $L_k = R_{k-1}$ и $R_k = L_{k-1}$, описанный в предыдущем разделе, недостаточный, если соответствующие L_k и L_{k-1} находятся не друг за другом, а на некотором заданном расстоянии, и было бы удобно это расстояние регулировать. Поэтому критерий был несколько изменен.

Необходимо учитывать текущее направление векторизации. Таких направлений восемь (рис. 19).

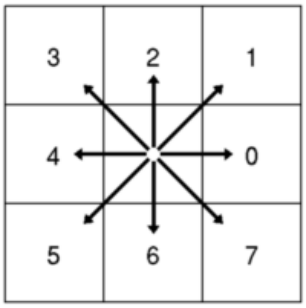

Рис. 19

В зависимости от направления критерии остановки алгоритма следующие:

- При направлении 0: $R_k.y < L_k.y$;
- При направлении 1: $R_k.x < L_k.x$ и $R_k.y < L_k.y$;
- При направлении 2: $R_k.x < L_k.x$;
- При направлении 3: $R_k.x < L_k.x$ и $R_k.y > L_k.y$;
- При направлении 4: $R_k.y > L_k.y$;
- При направлении 5: $R_k.x > L_k.x$ и $R_k.y > L_k.y$;
- При направлении 6: $R_k.x > L_k.x$;
- При направлении 7: $R_k.x > L_k.x$ и $R_k.y < L_k.y$.

Где $R_k.y$ и $R_k.x$ – x и y координаты соответствующего пикселя.

Заключение

В рамках данной работы были решены следующие задачи:

1) Изучены существующие алгоритмы векторизации, их преимущества и недостатки, выбран алгоритм, удовлетворяющий заявленным требованиям:
    - Возможность корректной обработки разветвлений на растре;
    - Возможность частичного разбиения диаграммы на объекты;
    - Отсутствие ложных ответвлений;
2) Реализован алгоритм векторизации для контура произвольной ширины;
3) Проанализированы результаты векторизации в зависимости от различных характеристик входного изображения. Были проведены эксперименты над снимками диаграмм с различной шириной линий – диаграмма, нарисованная ручкой на листке бумаги; с наличием внешних шумов и размытия - диаграмма, нарисованная мелом на обычной доске.

В дальнейшем планируется внедрить реализованный алгоритм в проект «Распознавание нарисованных UML-диаграмм». Планируется реализация процедур улучшение качества векторизации, сглаживание полученных ломанных, удаление внешних шумов на растре.

Список литературы

[1] Dov Dori, Liu Wenyin, «From Raster to Vectors: Extracting Visual Information from Line Drawings», Microsoft Research;

[2] Louisa Lam, Seong-Whan Lee, Ching Y. Suen, «Thinning Methodologies-A Comprehensive Survey», Journal IEEE Transactions on Pattern Analysis and Machine Intelligence;

[3] Peter R. van Nieuwenhuizen, Olaf Kiewiet, Willem F. Bonsvoort, «An integrated line tracking and vectorization algorithm», Eurographics European association for computer graphics.