САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра системного программирования

РАСПОЗНАВАНИЕ НАРИСОВАННЫХ ДИАГРАММ В ПРОЕКТЕ QREAL

Дипломная работа студента 545 группы

Осечкиной Марии Сергеевны

Научный руководитель	…… / подпись /	ст. преп. Литвинов Ю. В.
Рецензент	…… / подпись /	ст. преп. Кириленко Я. А.
“Допустить к защите” заведующий кафедрой,	…… / подпись /	д.ф.-м.н., проф. Терехов А.Н.

Санкт-Петербург
2012

SAINT PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Software Engineering Chair

HANDWRITTEN DIAGRAM RECOGNITION IN QREAL

by

Maria, Osechkina

Graduate paper

Supervisor	……	Senior lecturer Y. V. Litvinov
Reviewer	……	Senior lecturer I. A. Kirilenko
“Approved by” Head of Department	……	Professor A. N. Terekhov

Saint Petersburg
2012

Оглавление

Введение....4
Постановка задачи....6
Распознавание нарисованных от руки диаграмм....7
Существующие решения....9
Формат входных данных....12
Множество распознаваемых объектов....14
Этапы алгоритма распознавания диаграмм....15
Разбиение диаграммы на объекты....16
Разделение на элементы и связи между ними....19
Распознавание элементов....20
Вывод результатов распознавания....21
Тестирование....22
Эксперименты....23
Заключение....25
Список литературы....26

Введение

Диаграммы и схемы являются незаменимыми инструментами в бизнесе и индустрии. Они широко используются при разработке бизнес-отчетов, научных статей, технической документации и других типов документов. Кроме того, создание диаграмм является неотъемлемой составляющей любого UML-средства и частью DSM-подхода. Большинство людей, которым требуется создать диаграмму, используют специализированные графические редакторы, к примеру, Microsoft Visio. Как правило используемые графические редакторы довольно эффективны, но, тем не менее, как показывают эксперименты, рисование диаграммы от руки занимает в 10 раз меньше времени, чем ее создание в специализированном редакторе [2]. Можно предположить, что автоматизированный инструмент для распознавания отсканированных, сфотографированных или нарисованных с помощью графического планшета диаграмм значительно сэкономит время разработчика.

Существует множество решений, предназначенных для графических планшетов, которые позволяют распознавать нарисованные пользователем диаграммы динамически, - в процессе создания диаграммы с учетом последовательности рисования штрихов. Другой тип распознавания — статический, при котором не предпринимается никаких попыток распознавания до того, как пользователь даст соответствующую команду. В то время как было опубликовано несколько работ на тему динамического распознавания, развитие в области статического распознавания минимально. Одной из причин является заметное усложнение задачи при отказе от использования информации о последовательности рисования штрихов: зная точки начала и конца штриха, легче понять когда пользователь начал рисовать элемент или связь. Например, связь, как правило, рисуется одним росчерком, но при этом может пересекать несколько элементов и разбиваться на несколько ломаных. Гораздо проще выделить возможные связи, обладая информацией о штрихах, чем, зная лишь статическую картинку, на которой придется искать точки, подозрительные на начало и конец связей, а также возможные точки пересечения с объектами.

В некоторых случаях у пользователя может не оказаться под рукой планшета или другого необходимого программного и аппаратного обеспечения для рисования диаграммы и сопутствующего динамического распознавания. Иногда удобнее рисовать диаграмму от руки на листе бумаги или, например, при обсуждении какой-то задачи - на доске перед аудиторией. Впоследствии нарисованную диаграмму как правило надо где-то хранить, возможно, осуществлять на ней поиск, редактировать. В любом случае нарисованная от руки диаграмма не имеет под собой логической модели, которая строится автоматически при формировании такой же диаграммы с помощью CASE-средства. А ведь все преобразования диаграмм, их оптимизация и генерация кода осуществляются на основе именно логической модели. В данном случае было бы полезно иметь инструмент для статического распознавания диаграмм и построения по ним соответствующей логической модели.

На кафедре системного программирования математико-механического факультета СПбГУ разрабатывается metaCASE-система QReal [8]. В QReal для быстрой разработки новых редакторов диаграмм имеется метаредактор с графическим редактором формы элементов. Он позволяет создавать метамодели новых языков, описывая имеющиеся на диаграммах элементы и связи между ними, задавать визуальное представление элементов на диаграммах. Созданную метамодель можно скомпилировать в подключаемый модуль и подключить его к QReal прямо в процессе работы. Для создания диаграммы раньше приходилось перетаскивать элементы с палитры на рабочую сцену, что было не очень удобно. Для упрощения человеко-машинного взаимодействия в проекте была реализована поддержка

жестов мышью: с каждым элементом ассоциирован определённый жест мышью, выполненный с каким-либо модификатором (с зажатой правой кнопкой мыши), в процессе распознавания жеста пользователя в заданном месте создается соответствующий объект. В силу расширяемости metaCASE-систем и необходимости расширения набора жестов была реализована возможность динамического расширения набора жестов.

Как было упомянуто ранее, в некоторых случаях диаграмму удобнее рисовать от руки на листе бумаги или на доске. Хотелось бы иметь возможность подать сфотографированную или отсканированную диаграмму на вход инструменту распознавания и получить диаграмму QReal, по которой можно генерировать код. При этом требуется, чтобы набор объектов, которые могут быть выделены на диаграмме, можно было динамически расширить, как и в случае с жестами мышью: хотелось бы, чтобы при создании нового редактора можно было просто загрузить в инструмент распознавания описание новых элементов, и после этого множество распознаваемых диаграмм было автоматически расширено.

Не следует забывать, что распознавание нарисованной от руки диаграммы является частью технологического процесса проектирования систем с помощью CASE-средств, то есть по определению диалоговым режимом. Если система что-то распознает не так, пользователь сам исправит ошибочный кусок традиционными средствами.

Статическое распознавание довольно сложная задача, так как включает в себя процесс разделения диаграммы на объекты без какой-либо информации о том, как и в каком порядке пользователь рисовал объекты. В случае с динамическим распознаванием имеется упорядоченная последовательность рисования штрихов, и требуется понять в какой момент пользователь закончил рисовать одну фигуру и начал рисовать другую, зная когда пользователь прерывал рисование. В случае со статическим распознаванием, необходимо кроме всего прочего выделить простейшие формы и объединить их так, как подразумевал пользователь. Некоторая неоднозначность при распознавании диаграмм сильно усугубляется в случае, когда мы отказывается от информации о последовательности штрихов, которые в свою очередь являются последовательностью точек, и имеем лишь растровое изображение.

Целью данной дипломной работы является создание прототипа инструмента статического распознавания диаграмм в рамках проекта QReal для распознавания простейших объектов и связей между ними.

Постановка задачи

Для создания прототипа инструмента распознавания диаграмм были поставлены следующие задачи:

1) Сформулировать требования к формату входных данных и к множеству распознаваемых объектов на диаграммах.
2) Выбрать и реализовать алгоритм разбиения диаграммы на объекты.
3) Отделить элементы от связей между ними.
4) Применить алгоритм распознавания жестов мышью, реализованный в рамках курсовой работы автора за 4 курс, к выделенным объектам.
5) Вывести результат.
6) Разработать и реализовать методику тестирования прототипа распознавания диаграмм.

Распознавание нарисованных от руки диаграмм

Конечной целью данного исследования является распознавание отсканированных и сфотографированных диаграмм. Для распознавания такого рода диаграмм прежде всего потребуется отделить фон от контура. Есть различные алгоритмы и библиотеки, предоставляющие такую возможность.

Диаграммы, нарисованные пользователем, могут содержать в себе названия объектов, комментарии, информацию о кратности связи и другую текстовую информацию. Для распознавания этой информации прежде всего потребуется обнаружить текст на диаграмме, а затем распознать его. Может случиться так, что распознанный текст даст дополнительную информацию о свойствах объекта: полезно уметь различать к какому объекту и свойству объекта относится текст. К примеру, надпись рядом с объектом вероятно может обозначать имя этого объекта. Логично, если при распознавании такой ситуации инструмент распознавания автоматически присваивал имени соответствующего объекта распознанное значение текста. В случае если текст находится рядом с концом связи и удовлетворяет шаблону кратности связи, следовало бы автоматически изменять кратность связи у распознанной и сгенерированной связи на конечной диаграмме.

Объекты в QReal, как и в любом другом UML-средстве, можно условно разделить на элементы и связи между ними.

В большинстве случаев графическое представление элементов в QReal является набором из простых форм — отрезков, окружностей и дуг. Оно может быть несвязным, к примеру, представлять собой 2 концентрические окружности. Графическое представление объекта в QReal может включать не только набор ломаных, но и заштрихованные области. Кроме того в некоторых редакторах графическое представление объекта — иконка, которая может включать в себя разные цвета. Если пользователю захочется нарисовать диаграмму от руки так, что цветовая палитра элементов соответствует графическому представлению объектов в QReal, можно воспользоваться данным фактором при распознавании, но в то же время, если пользователь будет рисовать лишь одним цветом, распознавание должно оставаться корректным.

При распознавании связи необходимо определить элементы, которые она соединяет, ее тип и направление, в случае если связь направленная. Тип связи можно определить по виду ее концов, например у наследования на конце стрелка, у агрегирования — заштрихованный ромб. Иногда тип связи и ее направление легче определить с помощью семантики: в метаредакторе в QReal для каждого типа связи указывается, какие пары объектов и в каком направлении может соединять эта связь.

Может случиться так, что на диаграмме часть объектов была распознана верно, а часть — нет. Полезно иметь возможность выделить неверно распознанную часть диаграммы и запросить альтернативные варианты распознавания. Ошибка в распознавании может произойти как при неправильном разделении на объекты, если, например, объединили то, что не следовало, приписав связь к объекту, или не включили в объект то, что следовало включить, так и при неверном распознавании верно выделенных объектов. Во втором случае можно усовершенствовать этап распознавания отдельных элементов, включив в систему обучение.

В качестве множества распознаваемых объектов были выбраны некоторые элементы из редакторов QReal, представляющие собой простые многоугольники, и связи без типа. Основное требование к элементам — отсутствие точек, в которых сходится более двух

кривых, и замкнутость. На этапе создания прототипа мы поставили такое ограничение для простоты выявления связей.

В прототипе не предполагается поиск и распознавание текста на диаграмме, заштрихованных областей и цветных фигур, графическим представлением которых являются иконки.

Диаграмма должна представлять собой черно-белое изображение с контуром шириной в 1 пиксель в формате PNG. Диаграмму можно нарисовать и в самом приложении, но последовательность штрихов при распознавании учитываться не будет.

Существующие решения

В наше время широкое распространение получили системы оптического распознавания символов, предназначенные для автоматического перевода рукописного и печатного текста в электронный формат. Кроме того, есть немало систем для динамического распознавания введенного текста и нарисованных диаграмм, - пользователь пишет текст или рисует диаграмму с помощью мыши, пера и планшета или интерактивной доски, а приложение по мере рисования распознает введенную информацию. При этом может использоваться информация о последовательности рисования, перерывах в рисовании, отпускании кнопки мыши или отрывании пера от планшета.

Интерактивные доски

Все большую популярность получают интерактивные доски, которые предоставляют возможность взаимодействовать с приложением с помощью большого экрана и руки, стилуса или пера: изображение проецируется на экран, пользователю предоставляется возможность управлять приложением с помощью экрана и специального пера, которое выступает в качестве мыши. Интерактивные доски удобно использовать в учебном процессе, на презентациях и совещаниях. Существует программное обеспечение, позволяющее распознавать рукописный ввод на интерактивной доске. Так как интерактивная доска по сути интерфейс для удобного взаимодействия пользователя с приложением, с ее помощью, безусловно, можно использовать существующие технологии распознавания UML-диаграмм на статус-митингах и совещаниях. Но интерактивные доски дороги, поэтому возможности их использования ограничены

Динамическое распознавание диаграмм

Довольно удобно при создании диаграмм пользоваться не перетаскиванием с палитры на рабочую сцену, а рисовать желаемый элемент мышью или пером на планшете, чтобы впоследствии элемент был сгенерирован приложением в нужном месте. Это экономит время и позволяет не отвлекаться от продумывания структуры диаграммы на особенности инструментария, используемого для создания диаграммы. В некоторые UML-средства уже встроена возможность использования жестов мышью — пользователь изображает символичное представление объекта по заданному образцу, после чего генерируется требуемый объект, соответственно размеру и расположению фигуры, нарисованной пользователем. В качестве примера UML-редакторов, позволяющих создавать диаграммы при помощи жестов мышью, можно привести Visual Paradigm и Rational Rose.

Распознавание документов

Существует множество алгоритмов и систем оптического распознавания символов, используемого повсеместно: автоматический перевод рукописного и печатного текста в электронный формат широко используется для конвертации книг и документов, для автоматизации систем учета. Когда встала задача статического распознавания диаграмм, в первую очередь была предпринята попытка использовать какие-либо идеи и решения распознавания документов для прототипа распознавания диаграмм. Однако методы распознавания текста основаны на выделении объектов — абзацев, строк, слов и прочее [4] [6]. При этом используется принцип несвязности объектов. Для выделения объектов на диаграммах этот способ не подходит, так как объекты могут пересекаться. Более того в большинстве случаев диаграмма создается для того, чтобы отразить взаимосвязи между объектами, - сложно придумать диаграмму, несущую какую-либо смысловую нагрузку, все объекты которой отделены друг от друга. Диаграммы содержат множество связей, которые по

определению должны пересекать объекты в случае, если семантика диаграммы корректна. Поэтому принцип разделения документов на объекты на основании их несвязности не подходит для распознавания диаграмм и необходимой при этом сегментации.

Статическое распознавание диаграмм

Достижения в области статического распознавания диаграмм не такие впечатляющие, как в области динамического распознавания. Это связано с тем, что потеря информации о последовательности рисования штрихов влечет усложнение задачи: перед тем как начать распознавание, необходимо провести сегментацию изображения, то есть выделить отдельные объекты на диаграмме. Затруднительно выделять объекты, имея лишь контур и не зная, каким образом он был нарисован.

Был создан инструмент статического распознавания диаграмм [2], где распознаваемые объекты — это окружность, эллипс, прямоугольник, треугольник, ромб, а также связи между ними. Авторы предлагают выделять фигуры по аналогии с человеческой моделью выделения примитивов — на основании непрерывности и гладкости. После сегментации авторы предлагают исправить возможные ошибки — некоторые линии, относящиеся к одному объекту, могли быть разделены в процессе сегментации. Эти линии имеют близкие конечные точки. На основании этого соображения авторы предлагают объединять некоторые линии для исправления погрешности сегментации: если концы двух линий можно разделить на две пары близких концов так, что в каждой паре есть по одному концу из обеих линий, то эти линии должны быть объединены. После разделения диаграммы на объекты предлагается определить, является ли объект элементом (окружностью, эллипсом, треугольником, прямоугольником или ромбом) или связью между элементами. Для этого авторы предлагают проверить, есть ли в окрестности концов объекта контуры других объектов, а также проверить наличие «дыр» в объекте. При этом предполагается, что связь не может начинаться внутри объекта вдали от контура, - для корректного выделения связей в данном случае необходимо, чтобы концы связи были близки к контурам элементов. Распознавание объектов предлагается проводить на основании отношений между площадью выпуклой оболочки, ее периметром, площадью, высотой и шириной минимального по площади прямоугольника, описанного около выпуклой оболочки, максимума площадей прямоугольников, вписанных в выпуклую оболочку, площадью и периметром максимального по площади вписанного в выпуклую оболочку треугольника. Авторы утверждают, что некоторые отношения между этими величинами хорошо классифицируют распознаваемые объекты. Для улучшения распознавания предлагается использовать обучение посредством метода опорных векторов.

При создании описываемого прототипа была использована идея разделения на объекты при помощи гладкости и непрерывности. Но алгоритм исправления ошибок сегментации, описанный авторами, дает хороший результат лишь при условии, что линии пересекают объект не более чем в двух местах, иначе получится так, что есть три или больше линий, которые надо объединить в один объект, и предложенный авторами принцип рассмотрения пар концов не даст требуемого результата. Кроме того, авторы требуют от пользователя начинать и заканчивать связь рядом с контуром фигуры. Для удобства рисования нами было принято решение отказаться от этого ограничения, что потребовало изменения алгоритма сегментации и определения связей и элементов.

Семантика связей в распознавании диаграмм

Инструментарий InkKit [1] – еще один пример инструментария распознавания простейших диаграмм, таких как ненаправленные графы, направленные графы, организационные структуры. Для интерпретации той или иной диаграммы в InkKit

используется вид связи: авторы описывают, как можно сгенерировать структурированный текст, полностью отражающий содержание диаграммы, в зависимости от типов связей, используемых на диаграмме. Таким образом, по виду связи восстанавливается ее тип.

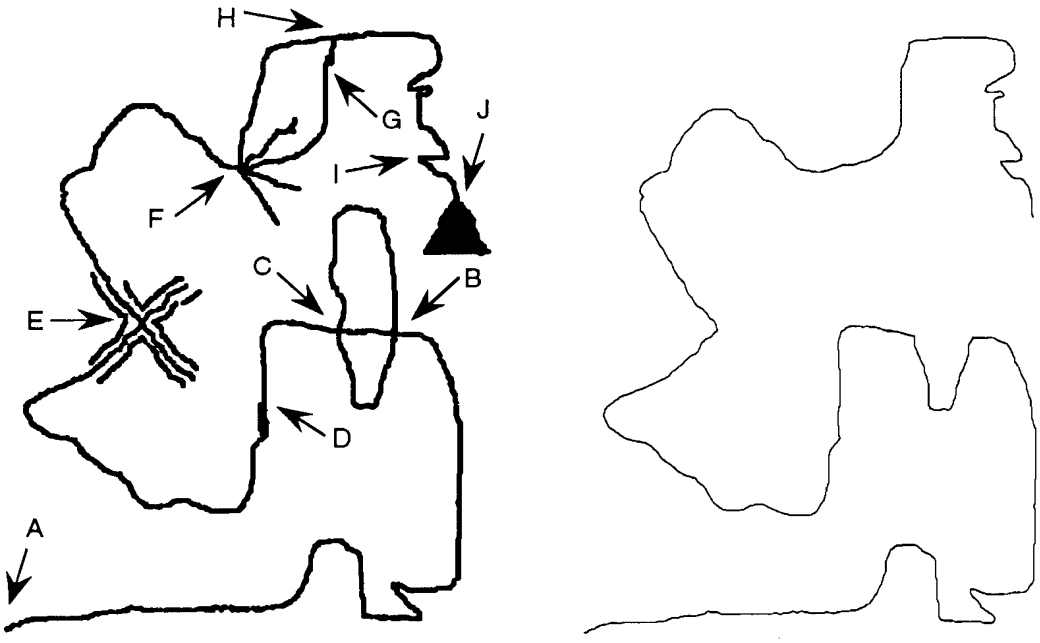
Заметим, что помимо изображения связи можно использовать дополнительную информацию об объектах, которые она соединяет. В QReal для каждой связи в meta-редакторе, содержащем ее описание, записано какие пары элементов она может соединять. Если это направленная связь, то также описывается, в элементах какого типа она может начинаться и заканчиваться. При распознавании типа связи можно использовать эту информацию — если элементы, которые соединяет связь, уже распознаны, то можно сократить область поиска типа связи.

Формат входных данных

Как уже было упомянуто выше, на вход прототипу распознавания диаграмм можно подать черно белое изображение диаграммы с шириной контура в 1 пиксель в формате PNG. Иногда, чтобы добиться выполнения условия для ширины контура достаточно понизить разрешение изображения. Кроме того, диаграмму можно рисовать непосредственно с помощью прототипа распознавания.

Очевидно, что при переходе к распознаванию отсканированных и сфотографированных диаграмм работать придется с растровым изображением, на котором придется отделить фон от контуров диаграммы. Предположим, что контур диаграммы одноцветный, и мы выбрали алгоритм отделения фона от контура, который всегда корректно работает. В итоге мы получим двухцветное растровое изображение. Для упрощения последующего расширения прототипа до полноценной системы распознавания диаграмм необходимо, чтобы растровое изображение, которое будет подаваться на вход инструменту распознавания, было легко свести к формату входных данных описываемого прототипа. Для этого либо алгоритмы сегментации диаграммы и распознавания объектов в прототипе должны быть рассчитаны на двухцветное растровое изображение, либо потребуется проводить векторизацию для изображений, которые впоследствии будут подаваться на вход системе распознавания отсканированных и сфотографированных диаграмм. Векторизация - это преобразование растрового изображения в векторное, то есть процесс, обратный растеризации. Разработано множество алгоритмов векторизации, но большинство из них либо вычислительно сложны, либо могут дать ложное разделение на объекты. Например, если есть связь, пересекающая фигуру, то некоторое алгоритмы векторизации объединяют отрезок связи, не лежащий внутри фигуры, с фигурой.

Был предложен алгоритм векторизации, который делает возможным пользователю самому корректировать направление линий в точках разветвлений [5]. Авторы этого алгоритма предлагают рассматривать точки пересечения окружности с центром в точке, из которой мы хотим продолжить векторизацию, и растровой линией, то есть границами пикселей, относящихся к контуру. В статье приводится пример векторизации изображения, время обхода составило 14 секунд (Рис. 1).



1

Рис 1

Заметим, что при использовании векторизации будет частично реализовано разделение диаграммы на объекты: диаграмма представляется как набор линий. Несмотря на то, что описываемый прототип предполагает распознавание оффлайн, желательно минимизировать время сегментации. В контексте данной задачи можно воспользоваться более простым методом для разбиения диаграммы на формы, нежели поиск пересечения концентрических окружностей с границами пикселей.

Было принято решение интерпретировать нарисованную диаграмму как растровое изображение, а процесс векторизации совместить с процессом сегментации.

Множество распознаваемых объектов

Перед тем как приступать к распознаванию диаграммы в целом необходимо определиться с множеством фигур, которые пользователь может изобразить на диаграмме. Необходимо предоставить пользователю образцы объектов, ориентируясь на которые он будет рисовать фигуры на диаграмме. При распознавании диаграмм разделение на объекты и распознавание выделенных объектов проводится в соответствии с набором образцовых объектов — как раз они ищутся на диаграмме при сегментации, с ними впоследствии сравниваются выделенные фигуры.

Введем ограничение на набор распознаваемых фигур. Фигуры будем разделять на элементы, инвариантные относительно переноса и растяжения, но не инвариантные относительно поворота, и связи между элементами, расположение которых может быть произвольным, но начало и конец должны совпадать с элементами. На данном этапе будем распознавать связи без типа, представляющие собой обычную ломаную без направления, и простые объекты. Мы ограничились связями без типа, так как распознавание вида конца связи несколько сложнее, чем распознавание объекта, в первую очередь из-за разницы в размерах. Кроме того, объекты в QReal нельзя вращать, то есть на алгоритм распознавания жестов не накладывалось требование инвариантности объектов относительно поворота. А расположение стрелок или, как в случае с агрегированием, ромба на конце связи, соответствует направлению связи, таким образом стрелки и прочие фигуры, обозначающие тип связи, могут быть повернуты произвольным образом. Для упрощения задачи в рамках прототипа мы ограничились связями без типа, графическое представление которых — линия, соединяющая два объекта. Следует заметить, что концы линии не обязательно должны быть близки к контуру фигуры: связь может начинаться и заканчиваться внутри фигуры, вдали от контура.

Главным критерием для распознаваемых элементов является отсутствие точек разветвления, то есть точек, которые служат началом более, чем для трех ломаных. Это важно, так как искать точки пересечения связи с объектом удобно именно как точки разветвления. Если среди распознаваемых элементов есть содержащие точки разветвления, то распознавание усложняется, и процесс выделения объектов, вероятно, придется объединить с распознаванием отдельных объектов. Впоследствии планируется усложнить схему алгоритма, но для прототипа мы решили ограничиться простейшим набором фигур, - равнобедренным треугольником, прямоугольником, одна из вертикальных сторон которого может быть выпуклым или вогнутым углом (символом приема и посылки сигналов в SDL), и ромбом. Распознаваемые фигуры должны быть многоугольниками или окружностью. Сфотографированная диаграмма может содержать текст, комментарии, информацию о кратности связей, но в рамках данной работы мы ограничимся диаграммами, не содержащими текст.

Этапы алгоритма распознавания диаграмм

В общем случае алгоритм распознавания нарисованных диаграмм можно разбить на несколько этапов.

1) Сегментация — разделение диаграммы на объекты. На этом этапе проводится выделение из контура диаграммы отдельных объектов, - элементов или связей, которые впоследствии необходимо распознать.

2) Разделение на элементы и связи между ними. Принцип распознавания элементов в корне отличается от распознавания связей, поэтому перед тем как приступать к распознаванию отдельных объектов, необходимо определить, что это за объект — элемент или связь между элементами.

3) Распознавание объектов. Распознавание элемента подразумевает определение типа элемента. Распознавание связи сводится к определению, какие элементы соединяет эта связь. Для распознавания элементов необходимо сравнить элементы, выделенные на диаграмме, с образцами элементов из множества распознаваемых объектов. Для определения того, на какой образец наиболее похож нарисованный пользователем элемент, вводится функция на декартовом произведении множеств элементов, которая показывает похожесть двух элементов. С помощью этой численной характеристики выбирается образец, на который наиболее похож нарисованный пользователем элемент.

4) Вывод распознанной диаграммы. На этом этапе производится прорисовка графического представления распознанных элементов и связей межу ними. Необходимо, чтобы пользователь мог усмотреть соответствие между объектами на нарисованной им диаграмме и распознанным вариантом диаграммы. Поэтому при прорисовке образцов элементов необходимо перенести и масштабировать их соответственно нарисованным пользователем элементам. На этом же этапе строится соответствующая логическая модель.

Разбиение диаграммы на объекты

В прототипе выделение объектов происходит в два этапа. Первый этап — разделение на компоненты, которые являются непрерывными кривыми и представляют собой упорядоченную цепь из пикселей. Второй этап — слияние некоторых компонент.

Разделение диаграммы на объекты основано на поиске точек пересечения линий, изображенных пользователем. Так как по введенным ограничениям распознаваемые элементы могут быть только многоугольниками и не могут содержать разветвлений, то точки пересечения линий на нарисованной диаграмме — это либо пересечение связи с элементом, либо самопересечение элемента вблизи точек, в которых пользователь начал и закончил рисовать элемент.

Чтобы локализовать развилки, требуется рассмотреть пересечение контура диаграммы с окрестностью каждой точки контура диаграммы. В качестве окрестности выступает квадрат $3X3$ с центром в точке из контура. Диаграмму предлагается разделить на компоненты, концами которых являются либо точки разветвления, либо точки в которых пользователь начинал или заканчивал штрихи при рисовании. Существует $2^8=256$ различных вариантов окрестностей $3X3$ точки на контуре, так как центр квадрата имеет 8 соседей, каждый из которых может принадлежать или не принадлежать контуру диаграммы. Хотелось бы избежать полного перебора при определении содержит ли окрестность точку разветвления. Мы предлагаем алгоритм, который проверяет, можно ли представить пересечение контура диаграммы с окрестностью точки на контуре как цепь. В случае, если можно, все точки из пересечения добавляются к той же компоненте, что и центральная точка.

Алгоритм определения точек разветвления.

1) Если в окрестности точки есть три ячейки, принадлежащие контуру диаграммы, и две из них являются соседними углами в окрестности, то окрестность содержит разветвление (Рис 2). Для каждой точки из пересечения контура с окрестностью, не отнесенной ни к одной компоненте, начинается построение новой компоненты.

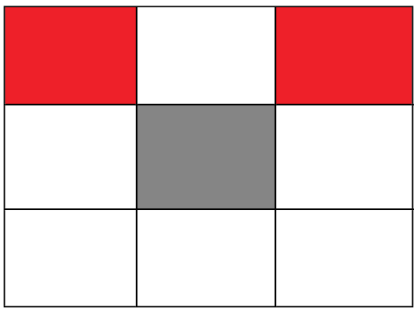


Рис 2

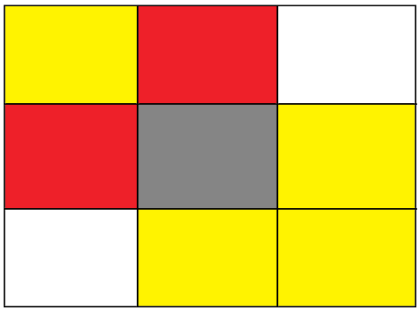


Рис 3

2) Рассмотрим случай, когда центральная ячейка имеет соседнюю по вертикали и по горизонтали. На рисунке 3 серым и красным выделены ячейки, принадлежащие контуру. Если хотя бы одна из желтых ячеек тоже принадлежит контуру, окрестность содержит разветвление контура, и необходимо начать построение новых компонент для ячеек из пересечения контура с окрестностью, которые еще не отнесены ни к одной компоненте.

3) В случае если ни одно из условий 1 и 2 не выполняется пересечение можно представить как растровый отрезок. Каждую ячейку из пересечения рассматриваем

как центр окрестности, продолжая построение компоненты, пока не наткнемся на точку разветвления.

На рисунках 3-9 показан порядок добавления к компоненте ячеек из пересечения окрестности с контуром диаграммы, если не выполняются условия 1 и 2: остальные случаи получаются поворотом на углы, кратные 90, или симметрией. При добавлении ячеек компонента должна оставаться непрерывной упорядоченной цепью. Для выполнения этого условия сначала проверяется есть ли в компоненте две последующие ячейки, которые граничат по стороне с ячейкой, которую мы хотим добавить. Если есть, вставляем ячейку между ними. В противном случае добавляем ячейку в начало или в конец компоненты в зависимости от того, с какой ячейкой компоненты новая ячейка граничит по стороне или по вершине. Так как построение компоненты начинается от произвольной ячейки и продолжается посредством добавления соседних с уже содержащимися в компоненте ячеек, на любом шаге компонента будет оставаться непрерывной упорядоченной цепью из ячеек.

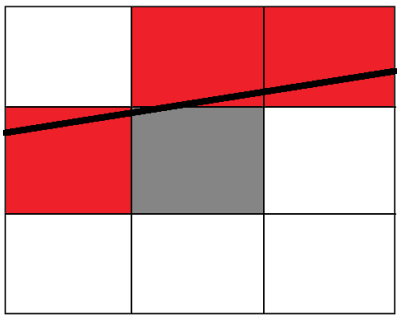


Рис 4

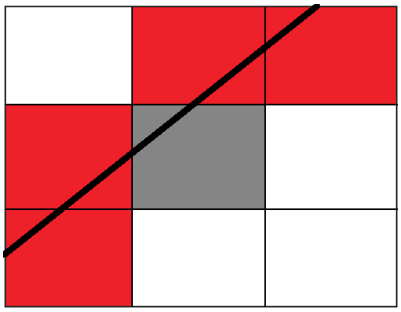


Рис 5

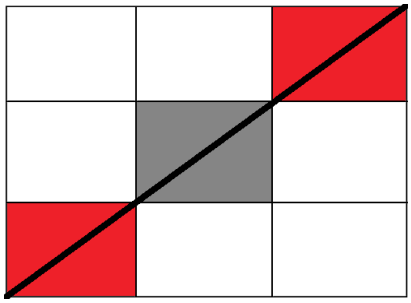


Рис 6

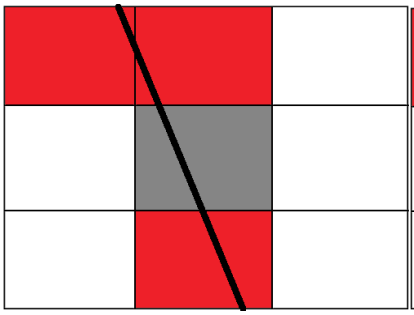


Рис 7

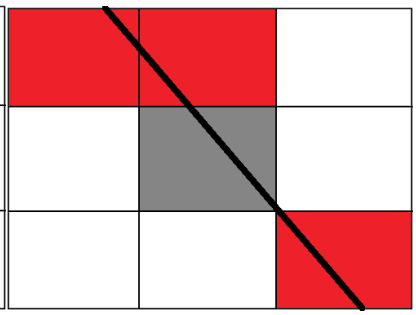


Рис 8

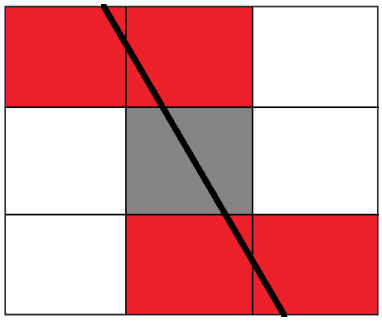


Рис 9

После того как диаграмму разбили на компоненты без развилок, следует объединить некоторые компоненты. Прежде всего проводится фильтрация по размеру: удаляются мелкие компоненты. Фильтрация необходима, так как иногда угол многоугольника локально представим в виде развилки из-за погрешности в рисовании. После того как удалены мелкие компоненты, необходимо объединить компоненты, которые образуют углы объекта: если в небольшой окрестности конца компоненты нет другого ее конца и есть единственный конец одной из других компонент, то эти две компоненты сливаются. После объединения углов требуется объединить части связей и элементов, разделенные точками пересечений. Слияние в точках пересечений проводится на основании гладкости. Рассмотрим аналог производной по времени. Усредним приращение нескольких крайних точек компоненты по отношению к ее концу.

$(\Delta x_{0,} \Delta y_0) = \sum_{i=1}^{n} \left(\frac{x_i - x_0}{n}, \frac{y_i - y_0}{n} \right)$ - вектор приращения для начала компоненты.

Аналогично рассчитаем вектор приращения для конца компоненты

$(\Delta x_{count}, \Delta y_{count}) = \sum_{i=count-n}^{count-1} (\frac{x_i - x_{count}}{n}, \frac{y_i - y_{count}}{n})$, *count* — размер компоненты.

Нам нужно слить те компоненты, которые являются продолжением друг друга. Заметим, что если два луча дополняют друг друга, то сумма введенных приращений для них 0. Исходя из этого факта, при объединении компонент будем рассматривать сумму приращений близких концов: если концы двух компонент достаточно близки и норма суммы приращений для этих концов не превосходит заданный порог, то две компоненты объединяются так, чтобы эти концы оказались соседними в объединенной компоненте. В случае если для конца какой-то компоненты есть несколько кандидатов на слияние, выберем того, сумма приращений с которым минимальна.

Заметим, что объединения компонент по гладкости недостаточно для выявления объектов. К примеру, на диаграмме может быть изображен ромб, в двух углах которого начинаются связи. Объединение компонент для углов в данном случае не сработает, так как для каждого угла будет третья компонента, проходящая через окрестность углов. Объединение по гладкости тоже не даст результатов. Для разрешения подобных случаев был применен поиск в глубину [7]: если при обходе графа из несамопересекающихся компонент мы находим цикл, то компоненты, образующие цикл, должны быть объединены.

После завершения поиска в глубину мы получим список компонент, которые можно подавать на вход алгоритму распознавания, предварительно определившись является ли компонента элементом или связью.

Разделение на элементы и связи между ними

Принцип распознавания связей в корне отличается от принципа распознавания элементов: если элементы сравниваются с заранее определенным множеством образцов элементов, то сравнение связи с другими связями не информативно. Элементы должны быть инвариантны относительно растяжения и переноса, но не инвариантны относительно поворота, - квадрат и ромб должны распознаваться как две разные фигуры. Связи в свою очередь могут быть как угодно повернуты и растянуты, более того, они могут иметь излом в произвольном месте. Перед тем как начать распознавание объектов, требуется отделить элементы от связей между ними.

По введенному ограничению на распознаваемые объекты корректная диаграмма, подаваемая на вход инструменту распознавания, должна состоять из многоугольников и связей между ними. Можно утверждать, что объект является элементом тогда и только тогда, когда он замкнут. Но иногда может случиться так, что пользователь недорисовал фигуру и между началом и концом линии-контура многоугольника есть зазор (рис 10), или, наоборот, пользователь закончил рисовать позже чем следовало бы и получилась линия с самопересечением, концы которой недалеки от точки самопересечения (рис 11).

Рис 10

Рис 11

Чтобы определить, является ли компонента связью или многоугольником, надо определить, содержит ли она точки самопересечения и верно ли, что расстояние между концами компоненты пренебрежимо мало. Если хотя бы одно из этих условий выполняется, значит, компонента является элементом и в процессе распознавания рассматривается как многоугольник, в противном случае компонента является связью и ее распознавание сводится к тому, чтобы определить какие элементы пересекают концы этой связи.

Распознавание элементов

После разделения на элементы и связи следует этап распознавания.

Существует множество алгоритмов распознавания мышиных жестов: есть набор команд, каждая из которых ассоциирована с определенным движением мыши - жестом, по определенному сигналу (нажатию комбинации клавиш, отпусканию кнопки мыши, таймауту) приложение распознает нарисованный жест, выбирая из списка жестов, ассоциированных с командами, наиболее похожий на жест пользователя, после чего выполняется соответствующая команда. Данный подход нашел применение в различным браузерах для навигации между страницами, некоторых играх для управления героями, при рукописном вводе текста, в некоторых CASE-системах для создания элементов, и так далее. В контексте данной работы нас будут интересовать алгоритмы распознавания жестов мышью, позволяющие рисовать жест несколькими штрихами, в любом направлении и в любой последовательности, - алгоритмы распознавания многоштриховых жестов. Если для выделенных на диаграмме элементов задать последовательность штрихов, то для распознавания элементов можно использовать алгоритмы распознавания многоштриховых жестов. Выделить штрихи для растрового многоугольника просто — достаточно рассмотреть контур многоугольника, применив алгоритм векторизации. Но для некоторых алгоритмов распознавания жестов хватит растровой картинки, не обязательно выделять штрихи для изображения. Один из таких алгоритмов был разработан для распознавания многоштриховых жестов в проекте QReal [3] и применен для распознавания элементов в описываемом прототипе. Для распознавания жестов мышью необходимо иметь набор идеальных жестов, ориентируясь на которые пользователь будет рисовать, и с которыми будут сравниваться жесты пользователя. Идеальные жесты для объектов QReal строятся в соответствии с их графическим представлением: каждый объект состоит из отрезков, окружностей или дуг, которые заменяются на штрихи в случае с жестами мышью. Для того чтобы определить на какой идеальный жест больше всего похож жест пользователя было предложено использовать расстояние Хаусдорфа. Жест вписывается в прямоугольник со сторонами параллельными осям координат, который делится на ячейки. Количество ячеек по ширине и высоте задано заранее. После этого для жеста строится матрица, размеры которой равны количеству ячеек по высоте и ширине прямоугольника. Элементу матрицы на позиции *(i, j)* присваивается расстояние от ячейки прямоугольника с координатами *(i, j)* до ближайшей ячейки прямоугольника, через которую проходит жест пользователя. В качестве расстояния между ячейками берется максимум модуля разности координат. Расстояние между жестами определяется как норма разности соответствующих матриц. В случае, когда в качестве нормы рассматривается максимум модулей элементов матрицы, норма разности оказывается равна расстоянию Хаусдорфа [9] между соответствующими множествами ячеек прямоугольников, через которые проходит жест. Жест пользователя сравнивается с каждым из идеальных жестов посредством вычисления описанного расстояния между ними. Выбирается идеальный жест расстояние до которого минимально, если это расстояние не превосходит заданный порог, то генерируется объект, соответствующий выбранному жесту.

Как было упомянуто выше, распознавание связи в контексте данной работы сводится к определению объектов, которые пересекают концы связи, - в прототипе отсутствуют такие понятия, как направление и тип связи. Для этого достаточно определить, с какими компонентами граничила связь на этапе разделения растрового изображения на компоненты и к каким элементам в итоге относятся данные компоненты.

Вывод результатов распознавания

Чтобы пользователь мог соотнести нарисованную им диаграмму и диаграмму, сгенерированную приложением в результате распознавания, необходимо, чтобы положение и масштаб объектов на итоговой диаграмме соответствовали расположению объектов на изначальной диаграмме пользователя.

Размер и положение элемента можно ассоциировать с описанным прямоугольником, стороны которого параллельны осям координат. После распознавания фигуры, соответствующей элементу, на диаграмме достаточно найти центр и размер описанного около нее многоугольника, а при генерации итоговой диаграммы следует расположить элемент так, что описанный около него прямоугольник совпадает с прямоугольником, описанным около соответствующей фигуры, выделенной на диаграмме пользователя.

Связи на итоговой диаграмме представляют собой отрезки с концами связей, выделенных на диаграмме пользователя.

Тестирование

Автоматизация тестирования диаграмм, нарисованных пользователем, — отдельная трудоемкая задача. Прежде всего нужно определиться как дать ответ на проблему разрешимости «Распознал ли прототип нарисованную пользователем диаграмму корректно?» Когда пользователь сидит перед монитором и видит результат распознавания, он может сам для себя ответить на этот вопрос, но при автоматизации требуется знать заранее какой должен быть результат распознавания, чтобы сравнить его с диаграммой, сгенерированной прототипом. Первое, что приходит на ум для создания тестовой базы - задать несколько примеров диаграмм посредством описания того, какие элементы должна содержать диаграмма, и какие из этих элементов должны быть соединены связями. Например, диаграмма должна содержать три квадрата, один из которых должен быть соединен связями с двумя другими. При таком описании проверка корректности распознавания сводится к тому, чтобы определить соответствует ли набор распознанных объектов заранее заданному. Но может случиться так, что в результате неправильного разделения диаграммы и неправильного распознавания элементов, конечный набор элементов будет верный, но расположение и разбиение элементов будет не таким, как подразумевал пользователь: например, если при пересечении связей объединение по гладкости будет неверное. На рисунке 12 показано разбиение связей, которое подразумевалось пользователем, на рисунке 13 показано ошибочное разбиение связей. При этом и тот и другой случай можно описать как две пары квадратов, соединенных связями. Необходимо более точное описание тестового примера, нежели просто перечисление элементов и связей между ними.

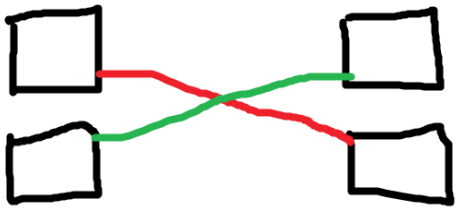


Рис 12

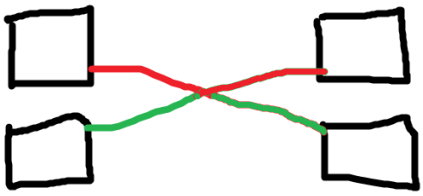
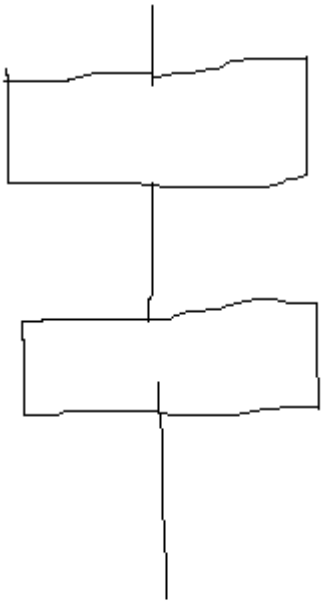


Рис 13

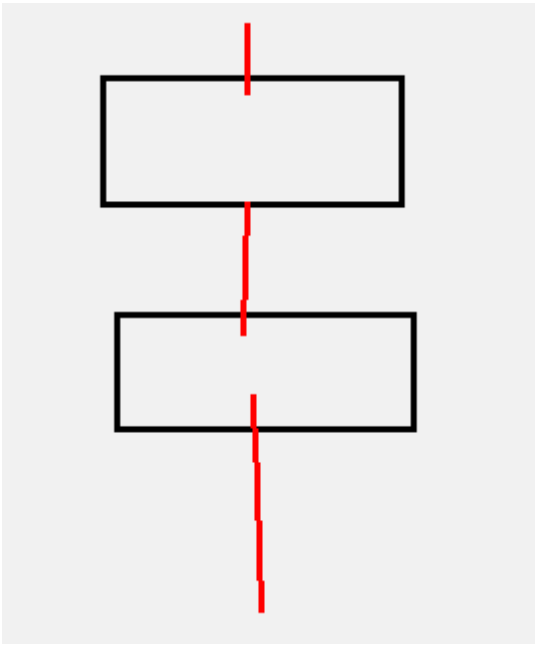
Мы предлагаем для каждого элемента указывать его местоположение. Но невозможно требовать от пользователя, чтобы он рисовал элементы в строго заданном месте, так чтобы их центр совпадал с заданной точкой, так как погрешность при дрожании руки все равно будет. Мы предлагаем в процессе создания отдельной диаграммы для тестовой выборки запрашивать рисование отдельных элементов. Когда пользователь закончил рисовать элемент, приложение получает сигнал (отпускание кнопки мыши, таймаут, нажатие комбинации клавиш), и элемент должен быть перенесен, так чтобы его центр находился в заданной точке. При этом каждому элементу на диаграмме присваивается порядковый номер, расположенный рядом с элементом. После того как пользователь закончил рисовать элементы, ему предлагается список связей, которые должны соединять элементы с заданными порядковыми номерами. Корректность распознавания при таких условиях создания тестовой выборки проверяется следующим образом: ищутся элементы, центры которых близки к заданным точкам. В соответствии с расположением этих объектов им присваиваются порядковые номера, а затем проверяется какие пары объектов соединены связями. При таком подходе к тестированию можно локализовать ошибки в распознавании отдельных элементов, разделении диаграммы на объекты и отделении элементов от связей.

Эксперименты

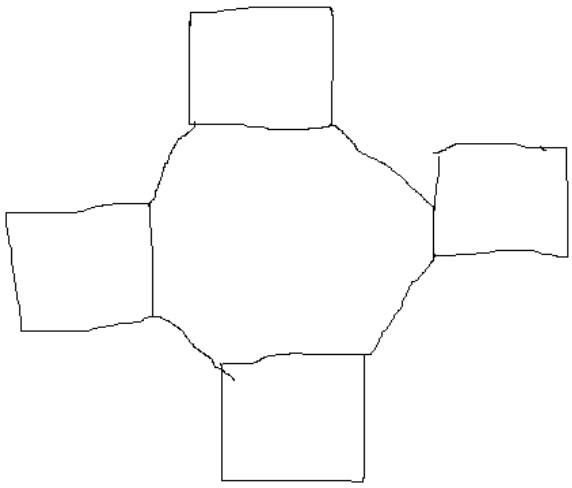
Ниже приведены примеры распознавания некоторых простейших диаграмм. Черным цветом изображены выделенные объекты, красным — связи между ними. Диаграммы для эксперимента были нарисованы в графическом редакторе: было необходимо получить двухцветное изображение с контуром шириной в 1 пиксель.



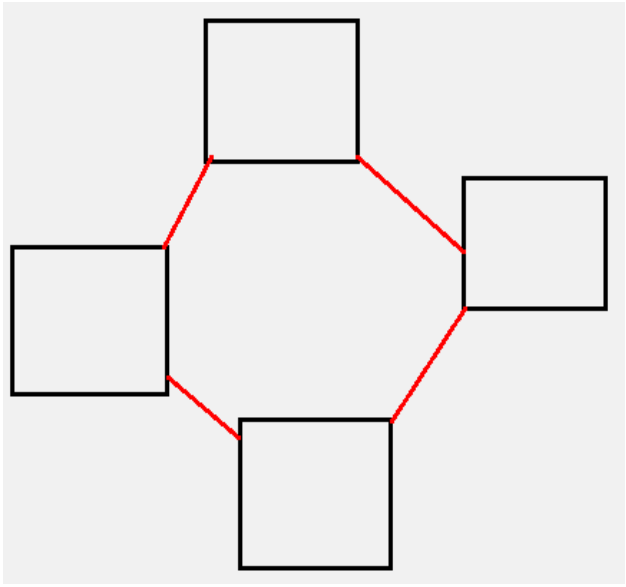
Нарисованная диаграмма 1



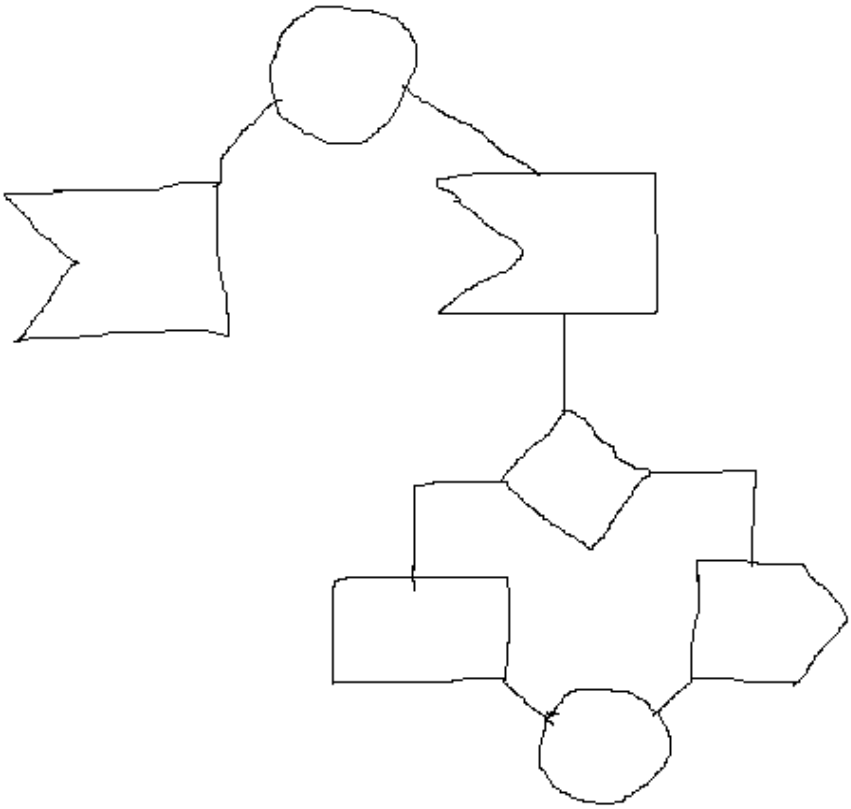
Результат распознавания 1



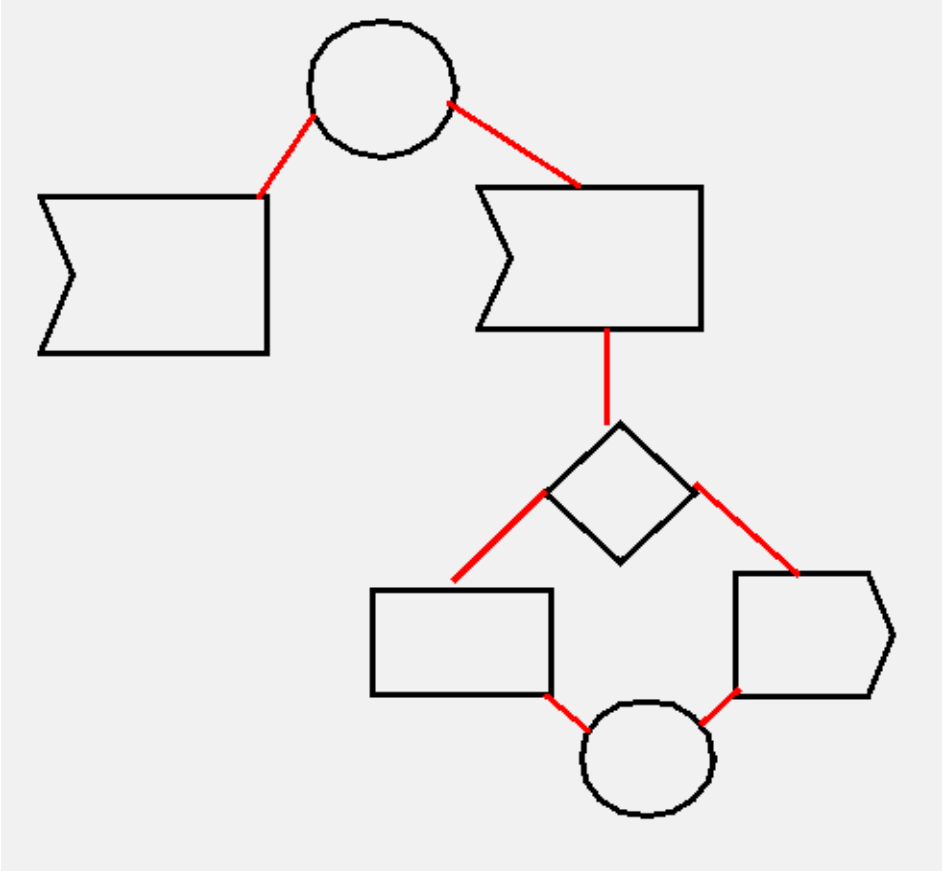
Нарисованная диаграмма 2



Результат распознавания 2



Нарисованная диаграмма 3



Результат распознавания 3

Заключение

В рамках данной дипломной работы были получены следующие результаты:

1) Сформулированы требования к формату входных данных и множеству распознаваемых объектов. Предполагается, что на вход инструменту распознавания подается растеризованная картинка, что удобно для планируемой реализации распознавания сфотографированных и отсканированных диаграмм. Распознаваемые объекты должны представлять собой многоугольники.
2) Разработан и реализован алгоритм разбиения диаграммы на объекты, основанный на непрерывности и гладкости связей и объектов, для корректного распознавания некоторых случаев было использовано выделение циклов с помощью поиска в глубину.
3) Выделены элементы и связи между ними на основании замкнутости элементов.
4) Для распознавания выделенных объектов был применен алгоритм распознавания жестов мышью, реализованный в рамках курсовой работы автора за 4 курс.
5) Реализован механизм вывода результатов распознавания, в виде, соответствующем нарисованной диаграмме

В дальнейшем планируется расширить множество распознаваемых объектов, добавив несвязные объекты и объекты, в которых одна точка может служить началом более двух кривых. Планируется реализация распознавания сфотографированных и отсканированных диаграмм. Для этой цели требуется внедрить алгоритм отделения контура от фона для изображений в созданный прототип распознавания диаграмм.

Список литературы

[1] Isaac J. Freeman, Beryl Plimmer, «Connector Semantics for Sketched Diagram Recognition», Proceeding AUIC '07 Proceedings of the eight Australasian conference on User interface - Volume 64, pp 71-78, 2007

[2] Khaled S. Refaat, Wael N. Helmy, Abdel Rahman H. Ali, Mohamed S. AbdelGhany, Amir F. Atiya, «A New Approach for Context-Independent Handwritten Offline Diagram Recognition using Support Vector Machines», UCLA Engineering Computer Science, URL http://www.cs.ucla.edu/~krefaat/Refaatetal08.pdf дата последнего обращения 20.05.2012

[3] Maria Osechkina, Yuri Litvinov and Timofey Bryksin, «Multistroke Mouse Gestures Recognition in QReal metaCASE Technology», unpublished

[4] Md. Abul Hasnat, S. M. Murtoza Habib, Mumit Khan, «Segmentation Free Bangla OCR using HMM: Training and Recognition», PAN Localization URL http://www.panl10n.net/english/outputs/Working%20Papers/Bangladesh/Microsoft%20Word%20-%2029_N_176.pdf дата последнего обращения 20.05.2012

[5] Peter R. van Nieuwenhuizen, Olaf Kiewiet, Willem F. Bronsvoort, «An Intrgrated Line Tracking and Vectorization Algorithm», Eurographics European association for computer graphics, URL http://diglib.eg.org/EG/CGF/Volume13/Issue3/v13i3pp349-359.pdf дата последнего обращения 20.05.2012

[6] Yuan Y. Tang, Hong Ma , Dihua Xi, Xiaogang Mao, Ching Y. Suen, «Modified Fractal Signatures (MFS): A New Approach to Document Analysis for Automatic Knowledge Acquisition», IEEE Transactions on Knowledge and Data Engineering Vol 9 Issue 5, pp 747 - 762, September 1997

[7] Кормен Т., Лейзерсон Ч., Ривест Р. Глава 22. Элементарные алгоритмы для работы с графами // Алгоритмы: построение и анализ (второе издание), 2005.— стр. 622-632

[8] Кузенкова А.С., Дерипаска А.О., Таран К.С., Подкопаев А.В., Литвинов Ю.В., Брыксин Т.А., «Средства быстрой разработки предметно-ориентированных решений в metaCASE-средстве QReal» // Научно-технические ведомости СПбГПУ, Информатика, телекоммуникации, управление. Вып. 4 (128). СПб.: Изд-во Политехнического Университета. 2011, С. 142-145

[9] Скворцов В. А. «Примеры метрических пространств» // Издательство Московского центра непрерывного математического образования, 2002, с. 7-12