САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

# РАСПОЗНАВАНИЕ НАРИСОВАННЫХ ДИАГРАММ В ПРОЕКТЕ QREAL

Курсовая работа
Дымниковой Натальи Александровны
244 гр.

Научный руководитель: ст.преп. Литвинов Ю.В.

Санкт-Петербург
2015

# Оглавление

# Введение

Есть всего несколько способов передавать информацию: с помощью слов, жестов и картинок. Картинки, в этом случае диаграммы, широко используются в бизнесе, в научных статьях, в технической документации и в других типах документов.

Существует большое количество специальных графических редакторов для создания диаграмм, к примеру, Microsoft Visio. Количество редакторов как оффлайн, так и онлайн достаточно велико, и среди них можно найти довольно эффективные и удобные, но, как говорит здравый смысл и показывают эксперименты, рисование диаграммы от руки практичнее и занимает в 10 раз меньше времени, чем в специализированном редакторе [2]. Поэтому создание программы, которая будет распознавать нарисованные диаграммы от руки, улучшит и облегчит жизнь многим людям.

Иногда во время какого-то обсуждения или доклада кто-то рисует на доске или бумаге диаграмму, которую необходимо сохранить и в будущем редактировать или извлечь из нее логическую модель.

На кафедре системного программирования математико-механического факультета СПбГУ ведется разработка metaCASE-системы QReal [4]. В этом проекте 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.

Как было сказано ранее, иногда диаграмму удобнее нарисовать от руки на листе бумаги или на доске. Хотелось бы иметь возможность получить из сфотографированного или отсканированного изображения диаграмму QReal, по которой можно генерировать код в дальнейшем.

Целью данной курсовой работы является создание прототипа инструмента статического распознавания диаграмм простейших объектов и связей в рамках проекта QReal.

## Постановка задачи

Для создания программы распознавания диаграмм были поставлены следующие задачи:

1) Определить требования к формату входных данных и к множеству распознаваемых объектов на диаграммах.
2) Выбрать и реализовать способ разбиения диаграммы на объекты.
3) Отделить элементы от связей между ними.
4) Определить тип элемента.
5) Вывести результат.

# Обзор существующих решений

Существует два типа распознавания текста или диаграмм — динамический и статический. Первый считается проще для распознавания, так как приложение имеет доступ к информации о последовательности рисования, направлении, перерывах, отпускании кнопки мыши или отрывании электронного пера. Однако задача данной курсовой работы — именно статическое распознавание диаграмм.

Был создан инструмент распознавания диаграмм [1] для изображений с однопиксельными линиями. Автор работы на первом этапе проводит сегментацию — разделение диаграммы на объекты, далее полученные объекты разделяет на фигуры и связи для дальнейшей обработки. После этого определяется тип фигуры при помощи сравнения с образцом. Разделение объектов происходит путем нахождения разветвления контура — так как все линии однопиксельные, то в точке пересечения линии с фигурой сходятся три линии.

Еще один пример — инструмент статического распознавания диаграмм [2], где распознаваемые объекты — это окружность, эллипс, прямоугольник, треугольник, ромб, а также связи между ними. Распознавание объектов проводится, учитывая отношения 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.

При этом авторы требуют от пользователя начинать и заканчивать связь рядом с контуром фигуры. Однако для удобства пользователя от этого ограничения пришлось отказаться.

## Входные данные

На вход программа принимает цветное изображение в формате PNG с нарисованной диаграммой с простым фоном, то есть с фоном, на котором может быть небольшой шум, который легко отсеивается. Главное отличие от существующего прототипа [1] состоит в том, что допускается толщина линии больше одного пикселя.

Так как в программе ищутся контуры линий, то необходимо обеспечить выполнение двух требований. Во-первых, нужна замкнутость контуров. Во-вторых, их связность. То есть в диаграмме все фигуры должны быть соединены, и линии — непрерывны.

Также необходимо ввести множество распознаваемых фигур. Фигуры будем определять инвариантные относительно переноса и растяжения, но не инвариантные относительно поворота. Среди фигур на изображении могут встречаться круги, прямоугольники, ромбы и флажки (прямоугольники, одна сторона которых выпуклый или вогнутый треугольник).

Связи на данном этапе могут быть просто прямыми линиями, нарисованными от руки, то есть прямыми с небольшой погрешностью.

Ещё одно требование, появившееся в ходе работы, касается циклов. Необходимо, чтобы их количество было существенно меньше, чем количество фигур, а площадь больше средней.

Также для упрощения работы на изображении не допускается наличие текста, комментариев или другой посторонней информации.

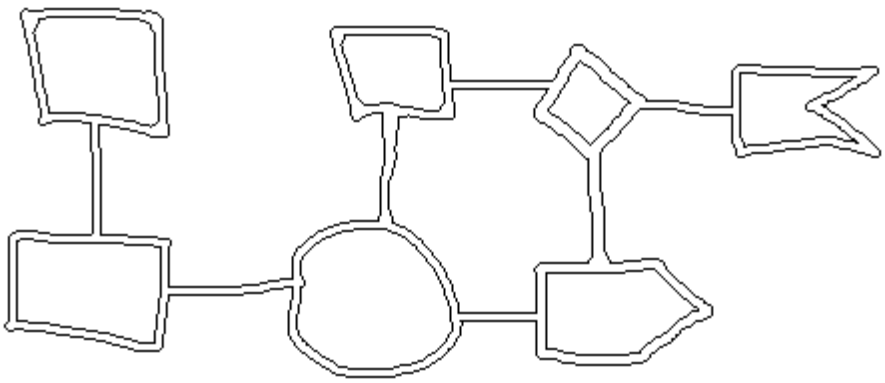
# Реализация

В написании данной курсовой работы использовалась библиотека OpenCV [3].

Программа разбита на несколько отдельных этапов:

1) Конвертация цветного изображения в изображение с двумя цветами — черный и белый — при помощи процедуры cvtConvert из библиотеки OpenCV. При этом отсеивается лишний шум, отделяется диаграмма от фона.

2) Нахождение контуров и их аппроксимация (также при помощи возможностей OpenCV). На этом этапе находятся внутренние и внешние границы. То есть в результате получается внешняя связная граница и множество внутренних.



На этом этапе также нужно исключить внутреннюю границу цикла, иначе алгоритм будет думать, что это одна из фигур. Для этого считается средняя площадь всех контуров, без учета внешнего (его площадь максимальна, так что его легко игнорировать). Площадь циклов будет существенно отличаться от средней площади, значит можно удалить этот контур.

Таким образом мы разобьем диаграмму на фигуры — это будут оставшиеся внутренние контуры.

3) Определение типа фигур.

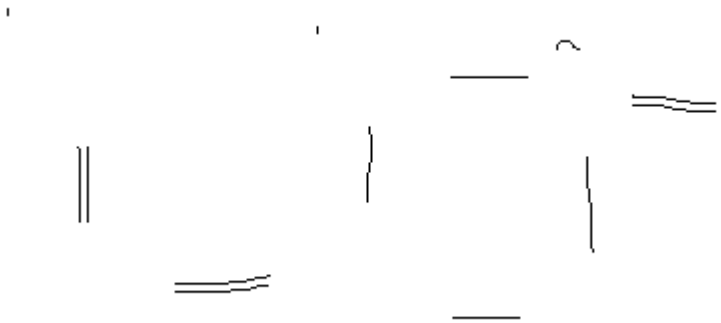
Самое простое, что можно распознать — окружность, потому что S / P^2 = 0.25 / Pi = 0,079577, то есть константа. Таким образом если это отношение для контура выполняется с какой-то погрешностью, то можно считать это окружностью. Для большей точности добавлена еще одна проверка — в библиотеке OpenCV есть функция cvMinEnclosingCircle, возвращающая минимальную окружность, в которую можно вписать фигуру. Площадь нашего контура и получившейся окружности также не должны сильно отличаться.

Далее можно определить прямоугольник — площадь минимального прямоугольника, в который можно вписать нашу фигуру, должна быть близка к площади фигуры. Этот прямоугольник можно получить с помощью функции cvBoundingRect.

Затем — ромб. Площадь прямоугольника, его описывающего, должна быть больше примерно в 2 раза, чем площадь фигуры.

Остаются флаги — если флаг вогнутый, то существует вертикальная линия, которая пересечет его в 4 точках, а если выпуклый, то только в 1 или 2. Так их можно различить между собой.

4) На данном этапе необходимо распознать связи и найти объекты, которые ими соединяются. Это можно сделать следующим образом: из внешнего контура вырежем все найденные нами слегка увеличенные контуры, тогда останутся линии и некоторые остатки фигур, которые слишком малы и они будут игнорироваться.



Некоторые связи представляются двумя линиями, потому что во внешнем контуре было именно две линии. А там, где осталась только одна, вторая была удалена, как часть внутреннего цикла на втором этапе.

Линии можно находить с помощью средств OpenCV — функции cvHoughLines2, которая возвращает последовательность линий, после чего оставлять только одну из двух линий, расстояние между началами и концами которых мало.

Затем, чтобы найти требуемые начало и конец, нужно найти ближайшие фигуры к концам линии.

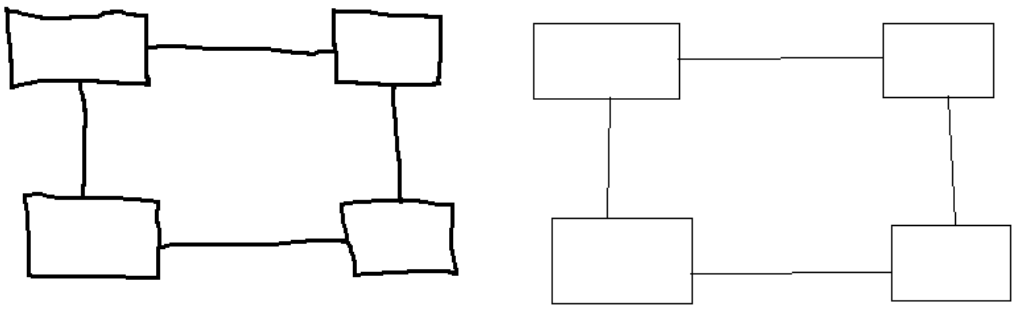
Таким образом, находятся сначала все фигуры, затем все связи между ними, и полученными результаты выводятся на новом изображении, при этом расположение объектов не меняется.

## Апробация

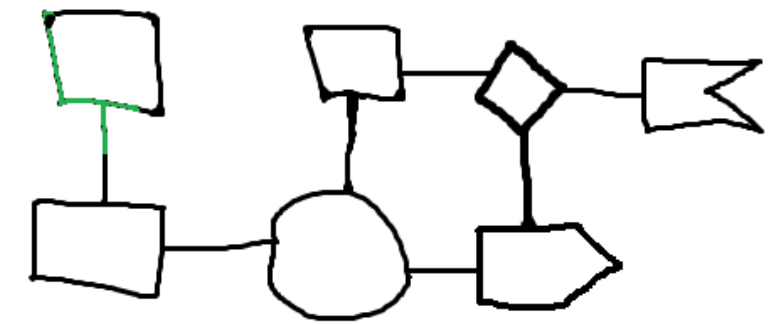
При тестировании было распознано 6 диаграмм, 28 фигур, из которых 5 неверно классифицировано. Таким образом корректно было определено 83% фигур. Большинство ошибок программа допускала при распознавании окружностей, принимая их за прямоугольники. Также иногда выпуклые флажки и совсем редко ромбы программа также принимала за прямоугольники.

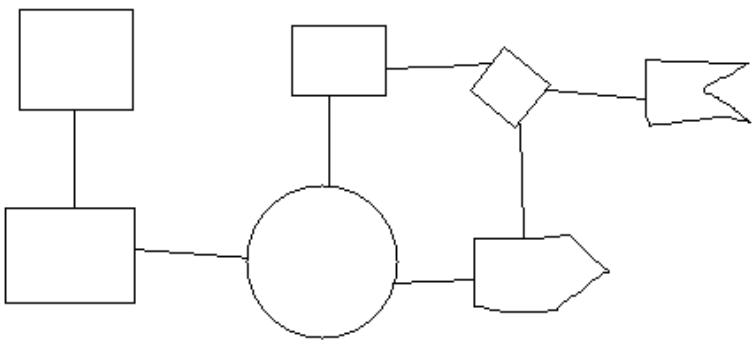
Несколько экспериментов:

1)



2)





# Заключение

В рамках данной курсовой работы были получены следующие результаты:

1) Сформулированы требования к формату входных данных и множеству распознаваемых объектов.
2) Разработан и реализован алгоритм разбиения диаграммы на фигуры и связи, основанный на непрерывности и гладкости линий.
3) Придуман способ определения типа объекта, основная идея которого — сравнение площадей и периметров фигуры и описанного вокруг нее прямоугольника.
4) Найдены связи между объектами.
5) Полученный результат сохраняет положение объектов и связей между ними.

## Список используемой литературы

[1] М.С. Осечкина, “Распознавание нарисованных диаграмм в проекте QReal“ (дипломная работа),
URL: http://se.math.spbu.ru/SE/diploma/2012/s/Osechkina_diploma.pdf (дата обращения 13.05.15)

[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 обращения 13.05.15)

[3] Домашняя страница открытой библиотеки компьютерного зрения — OpenCV,
URL http://opencv.org (дата обращения 13.05.15)

[4] Домашняя страница проекта QReal на GitHub,
URL https://github.com/qreal/qreal (дата обращения: 13.05.15)