Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Ивашева Валерия Михайловна

Распознавание многоштриховых жестов мыши

Курсовая работа

Научный руководитель:
к. т. н., доцент Литвинов Ю. В.

Санкт-Петербург
2019

Оглавление

Введение

Жест мыши — это способ управлять программами в компьютере при помощи движений мышью и ассоциированных с ними команд. Они могут быть одноштриховыми и многоштриховыми, т. е. состоять из нескольких штрихов. Иногда нарисовать жест оказывается быстрее и удобнее, чем вызвать команду привычным способом. Таким образом, поддержка мышиных жестов может облегчить жизнь пользователям. В рамках предыдущей работы автора был реализован инструмент распознавания одноштриховых жестов [1], обладающий следующими свойствами:

1. уникальность — возможность каждую команду ассоциировать с единственным штрихом;
2. разнообразие — возможность распознавать сложные жесты, состоящие не только из элементарных направлений (влево-вправо, и т.п.);
3. расширяемость — возможность легкого добавления новых жестов.

Так как на данный момент, нет возможности исправить нарисованный жест при неверном распознавании до корректного. Хочется уметь работать с несколькими штрихами, также это увеличит множество жестов для распознавания. Поэтому было решено добавить возможность распознавания многоштриховых жестов, не нарушая основные свойства инструмента. Таким образом, целью работы является разработка инструмента, который умеет распознавать и классифицировать многоштриховую траекторию мыши, к одному из установленных классов.

Для достижения цели надо было решить следующие задачи.

1. Изучение алгоритмов распознавания многоштриховых жестов.
2. Реализация алгоритмов.
3. Сбор базы многоштриховых жестов.
4. Тестирование алгоритмов и сравнение точности распознавания.

1. Общий случай алгоритма распознавания жеста

Есть список идеальных жестов, соответствующих некоторым объектам или командам. Среди них с помощью алгоритма находим фигуру, наиболее похожую на жест, введенный пользователем. Алгоритм распознавания жестов следующий.

1. Определение объекта распознавания.
   Каждый жест мыши — это список штрихов, штрих — список точек траектории мыши.

2. Построение классификатора.
   Определяем множество признаков, по которым происходит сравнение жестов. На множестве классификатора вводим метрику.

3. Выбор объекта.
   Вычисляем расстояние между пользовательским жестом и каждым жестом из списка идеальных. Жест, которому соответствует минимальное расстояние, и есть искомый. Выполняем ассоциированную с ним команду.

2. Обзор существующих решений

2.1. QReal

Ранее на кафедре системного программирования была разработана CASE-система QReal[4], позволяющая быстро создавать визуальные предметно-ориентированные языки, редакторы и генераторы исходных кодов для них. В нем реализована поддержка жестов мыши. В QReal инструмент распознавания мышиных жестов имеет схожий принцип работы, однако здесь приводятся другие алгоритмы, не взятые за основу в QReal.

Алгоритм распознавания в QReal

Жест мыши рассматривается как список ячеек. Жест пользователя вписывается в прямоугольник, стороны которого разбиваются на n равных частей, таким образом получим n^2 ячеек. Список ячеек — ячейки, через которые проходит жест. На рисунке 1 (слева) — это ячейки красного цвета. Расстояние между жестами — это расстояние между списками ячеек. Оно вводится как норма разности соответствующих матриц, полученных следующим образом. M — матрица h*w, где w — это ширина описанного около жеста прямоугольника, а h — это высота. Элемент матрицы $M_{i,j}$ равен расстоянию μ от ячейки с координатами i,j до ближайшей ячейки вспомогательного списка. Ячейки рассматриваются как точки в $\mathbb{R}^2$. Расстояние μ между ними это расстояние в $\mathbb{R}^2$. На рисунке 1 (справа) изображена матрица жеста, расстояние — сумма модулей разности координат.

2.2. $P

Это распознаватель жестов, предназначенный для быстрого добавления жестов в пользовательский интерфейс. Представляет жесты в виде облаков точек, $P[2] может обрабатывать как одноштриховые, так и многоштриховые одинаково, причем порядок и направление штрихов игнорируются. На рисунке 2 изображено облако точек. При сравнении двух облаков точек $P находит решение задачи назначения между

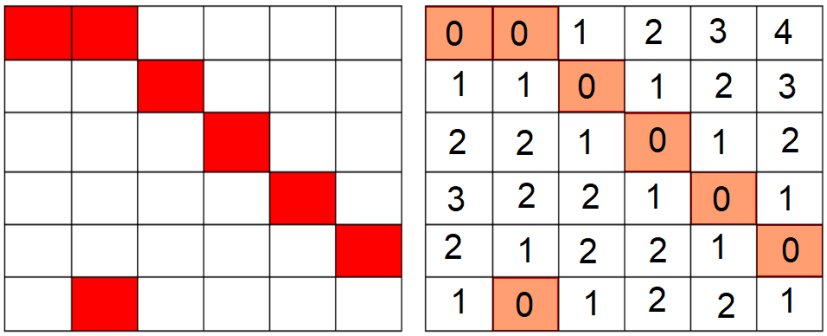

Рис. 1: Список ячеек и матрица расстояний жеста, QReal

двудольными графами с использованием приближенного венгерского алгоритма.

Алгоритм расстояния между облаками в $P.

C_1 — точки первого облака (на рис. 2 светло-зеленые), C_2 — точки второго облака (на рис. 2 темно-зеленые), D = $\varnothing$ — распределенные точки.

1. Для каждой точки p_1 из первого облака находим ближайшую точку p_2 из второго, которую еще не соединили.

2. p_2 добавляем в D.

3. Повторяем до тех пор, пока $C_2 \backslash \mathrm{D} \neq \varnothing$.

4. Находим временное расстояние между C_1 и C_2 = сумма евклидовых расстояний между точками C_1 и D.

5. Повторяем пункты 1-3, теперь C_1 — точки второго облака, C_2 — точки первого облака, D = $\varnothing$ — распределенные точки.

6. dist2 = dist1, из пункта 4.

7. Расстояние между облаками — min(dist1, dist2).

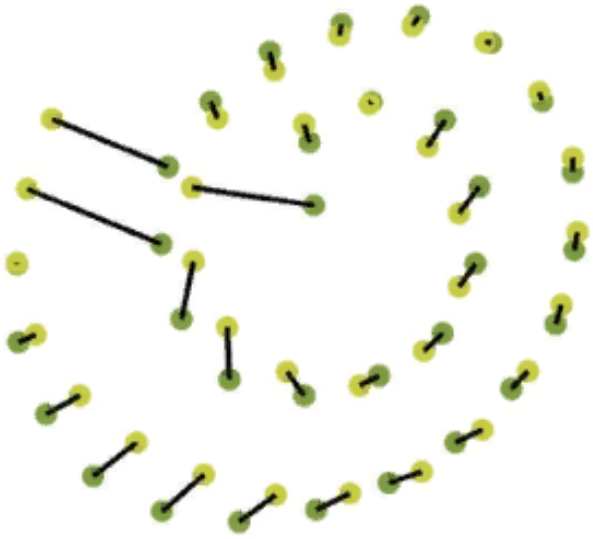

Рис. 2: Жесты, как облака точек, $P

2.3. Алгоритмы машинного обучения

Обучающая выборка — это множество объектов (в нашем случае жестов), распределенных ранее по классам.

Некоторые алгоритмы, применимые к распознаванию жестов

1. Метод k ближайших соседей.

   Объект относится к классу, которому принадлежит большинство соседей — ближайших объектов из обучающей выборки. Может потребовать много памяти для хранения данных. Данный метод используется в комбинации с некоторым классификатором, и если классификатор будет работать достаточно долго, тогда общее время работы увеличится пропорционально количеству элементов обучающей выборки. При таком подходе пользователю будет быстрее вызвать команду привычным способом, чем с помощью жеста.

2. Нейронные сети.

   На вход подаётся вектор значений, характеризующих конкретный

пример. Специально подбирается матрица параметров (веса нейронов), с помощью которой (возможно, в несколько этапов) производятся вычисления, приводящие нас к ответу. Обучение нейронной сети иногда занимает довольно много времени, а также требует обучающей выборки.

3. AdaBoost.

   AdaBoost — алгоритм машинного обучения, основная цель которого состоит в том, чтобы создать сильный классификатор на основе слабых. Каждый следующий классификатор пытается исправить ошибки предыдущего классификатора. После каждого вызова обновляются веса, которые отвечают важности каждого из объектов обучающего множества для классификации.

Во многих алгоритмах для обучения нужна обучающая выборка. К сожалению, данный способ противоречит одному из свойств нашего приложения, а именно расширяемости. Ведь пользователю придется самому пополнять обучающую выборку, а это довольно трудоемкий процесс.

3. Алгоритмы построения классификатора

На данный момент в реализованном инструменте используется классификатор, в котором множеством признаков для распознавания одноштриховых жестов является множество характеристических точек. Характеристические точки — это точки, с помощью которых можно однозначно определить фигуру или жест. Метрика — сумма евклидового расстояния между точками списка поэлементно. Точки упорядочены по мере их появления.

Алгоритм нахождения характеристических точек

1. Первая и последняя точки — характеристические.

2. Если угол ABC > 20 и расстояние AB $> \epsilon$, это характеристическая точка.
   A — последняя добавленная характеристическая точка,
   B — точка, рассматриваемая на данный момент,
   C — следующая за B,
   ϵ — заданная точность.

Данный алгоритм можно использовать и на многоштриховых с одним дополнительным ограничением, жесты должны учитывать направление. Для этого нужно “склеить” штрихи жеста в один. В ходе экспериментов оказалось, что данный алгоритм не может быть использован на практике для многоштриховых жестов.

Также с данным алгоритмом пользователю придется запоминать еще и направление. Хочется позволить пользователю, запоминать саму фигуру, а не способ рисования. Поэтому было решено рассмотреть еще один классификатор.

3.1. Классификатор, основанный на венгерском алгоритме

Множество признаков — это множество точек, заранее ограниченного количества. Данное количество должно быть минимум в 2 раза

больше, чем максимум из количества точек, описывающих идеальные жесты. Если количество точек = 4, окружность и ромб задаются одинаковыми точками (вершинами ромба), тогда алгоритмом будет игнорироваться одна из фигур. Для того, чтобы не возникало таких ситуаций увеличивается количество точек.

Метрика — сумма евклидового расстояния между списками точек.

Для того, чтобы наша метрика не противоречила основным аксиомам, необходимо иметь фигуры одинакового размера, а также чтобы их задавало одинаковое количество точек. Поэтому нам потребуется еще один алгоритм, задачей которого будет нормализовать фигуры и дополнить либо сократить множество точек до заданного числа.

Приведение жеста к нужному формату
(n — итоговое количество точек в списке)

1. Переводим жест в первую координатную четверть.

2. Вписываем фигуру в квадрат со стороной 100.

3. Считаем "периметр" (P) нашей фигуры, т. е. расстояние, которое проходит мышь при нажатой кнопке, но уже в приведенных координатах.

4. Находим расстояние $\epsilon = P/n$.

5. В цикле ищем точки, находящиеся на расстоянии $>= \epsilon$ друг от друга.
   Массив точек, получившийся в результате и есть искомый.

Так как в общем случае жест не зависит от направления, данные списки нужно упорядочить так, чтобы минимизировать расстояние между ними. Для этого можно использовать венгерский алгоритм [3]. Венгерский алгоритм решает задачу о назначениях. Жест пользователя и идеальный жест, с которым происходит сравнение являются множествами двудольного графа. Веса между вершинами — это евклидово расстояние между данными точками.

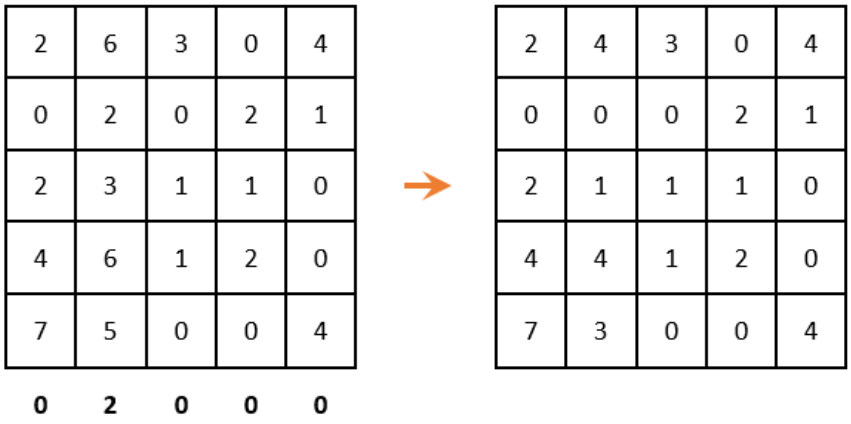

Рис. 3: Венгерский алгоритм, пункт 1

Алгоритм поиска расстояния, использующий венгерский алгоритм

L_1, L_2 — списки точек жеста, полученные в процессе алгоритма приведения к нужному формату.

1. Имеем матрицу n*n, $M_{i,j}$ — расстояние между точками L_1[i] и L_2[j].
2. Ищем минимальный элемент в каждой строке и вычитаем его из каждого элемента данной строки. Аналогично со столбцами. (Рис 3)
3. Вычеркиваем строки и столбцы, которые содержат 0. (Рис 4)
4. Среди оставшихся(на рис. 4 зеленые) находим минимальный — min, вычитаем min из всех оставшихся.
5. Прибавляем min ко всем элементам, находящихся на пересечении вычеркнутых строк и столбцов (на рис 4 золотые ячейки).
6. Повторяем пункты 1-4 до тех, пока в каждой строке и столбце не удастся выбрать один ноль (как на рис. 5)
7. Находим сумму расстояний между точками L_1[i], L_2[j], где i — номер строки нуля, j — номер столбца.

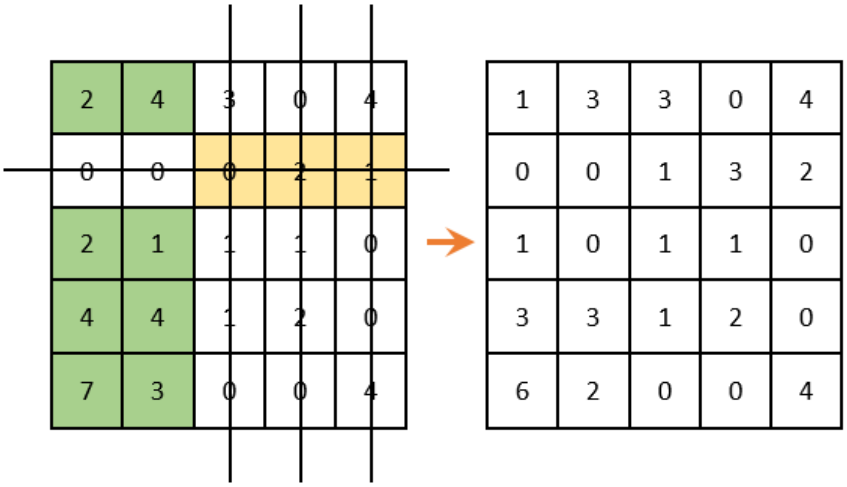

Рис. 4: Венгерский алгоритм, пункт 2-5

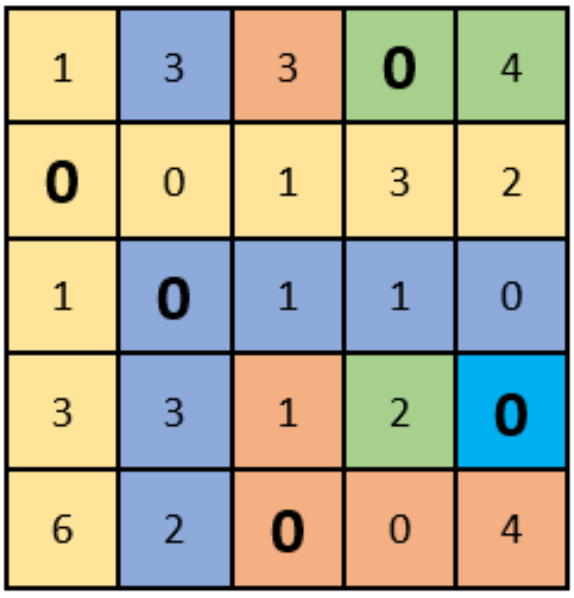

Рис. 5: Венгерский алгоритм, результат

	Идеальное распознавание	Алгоритм из QReal	Алгоритм из $P	Венгерский алгоритм
Фиг.1	122	117	121	121
Фиг.2	123	97	81	80
Фиг.3	119	56	90	97
Фиг.4	116	0	0	0
Фиг.5	130	0	0	0
Фиг.6	129	0	12	5
Фиг.7	128	0	0	0
Фиг.8	151	74	108	101
Фиг.9	160	0	0	0
Фиг.10	132	120	109	114
Фиг.11	152	143	90	103
Фиг.12	223	162	202	204
Фиг.13	158	151	81	84
Фиг.14	118	56	51	47
Фиг.15	135	102	45	48
Фиг.16	224	205	155	167
Всего	2320	1283	1145	1171

Таблица 1: Результаты тестирования алгоритмов

4. Эксперименты

Было проведено тестирование на объектах, графическое представление которых представлено на рис. 6 и 7. В составлении базы жестов с направлением (далее база 1) участвовало 18 человек, а базы многоштриховых (база 2) — 9. Таким образом, мы протестировали алгоритм на 1085 жестах с направлением и 2320 многоштриховых без направления. С результатами можно ознакомиться в таблицах 1 и 2. Для оценки качества работы алгоритмов на каждом классе по отдельности введем метрики precision (точность) и recall (полнота). Precision и recall для алгоритма в целом считаем как среднее значение по всем классам.

$\text{Precision}(\alpha) = \frac{TP}{TP+FP}$ $\text{Recall}(\alpha) = \frac{TP}{TP+FN}$

α — класс, для которого считается формула, TP — верно распознанный жест, FP — жест распознанный как α, но не являющийся им на самом деле, FN — жест, являющийся α, но не распознанный алгоритмом.

Precision и recall представлены в таблице 3. Для алгоритма характеристических точек с направлением посчитано на базе 1, остальные на базе 2.

	Фиг. 1	Фиг. 2	Фиг. 3	Фиг. 4	Фиг. 5	Фиг. 6	Фиг. 7	Фиг. 8	Фиг. 9	Всего
Идеальное распознавание	120	119	119	132	130	95	122	116	132	1085
Характеристические точки с направлением	16	3	0	123	3	87	0	0	87	319
Алгоритм из QReal	105	119	115	0	122	86	101	42	101	791
Венгерский алгоритм	93	96	18	54	116	5	109	31	117	639

Таблица 2: Результаты тестирования алгоритмов на базе многоштриховых с направлением

	Precision	Recall
Алгоритм из QReal	0, 56	0,52
Алгоритм из $P	0,48	0,47
Венгерский алгоритм	0,48	0,48
Характ. точки + направление	0,30	0,31

Таблица 3: Precision и recall алгоритмов

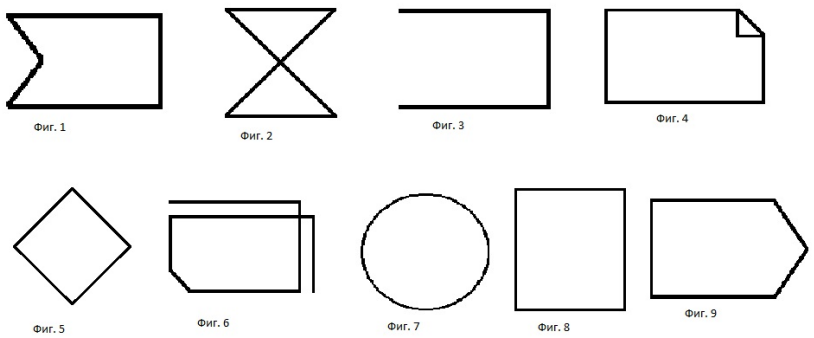

Рис. 6: Фигуры, на которых происходило тестирование алгоритмов с направлением

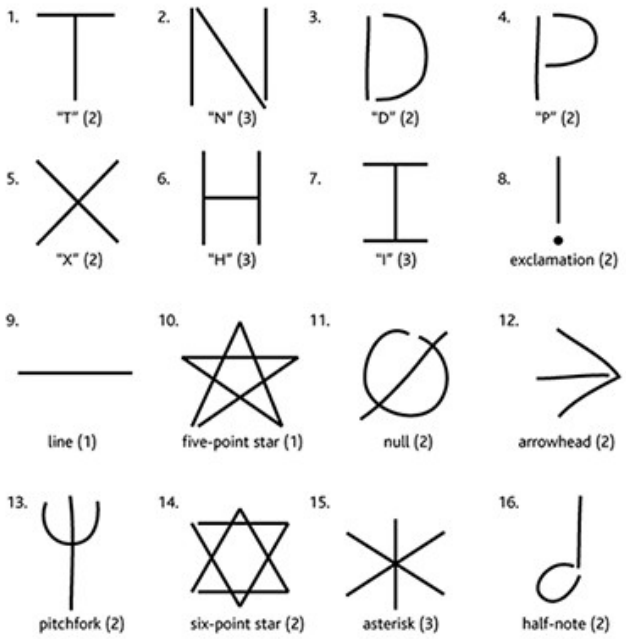

Рис. 7: Фигуры, на которых происходило тестирование алгоритмов

Заключение

Были реализованы и протестированы различные алгоритмы распознавания многоштриховых жестов. В ходе работы стало ясно, что инструмент, работающий на данных алгоритмах, не подходит для внедрения в визуальные системы , так как имеет недостаточную точность распознавания, что не очень понравится пользователям системы. Ознакомиться с результатами работы можно в GitHub-репозитории проекта [1]

Список литературы

[1] Github-репозиторий проекта. — URL: https://github.com/valeria-ivasheva/recognizer (online; accessed: 29.05.2019).

[2] Radu-Daniel Vatavu Lisa Anthony Jacob O. Wobbrock. Gestures as Point Clouds: A $P Recognizer for User Interface Prototypes // University of Washington. — URL: http://faculty.washington.edu/wobbrock/pubs/icmi-12.pdf (online; accessed: 29.05.2019).

[3] Wikipedia. Венгерский алгоритм // Википедия, свободная энциклопеди. — URL: https://ru.wikipedia.org/wiki/Венгерский_алгоритм (online; accessed: 29.05.2019).

[4] С. Осечкина М. Многоштриховые жесты мышью в проекте QReal // сайт кафедры системного программирования СПбГУ. — URL: http://se.math.spbu.ru/SE/YearlyProjects/2011/YearlyProjects/2011/445/445_Osechkina_report.pdf (online; accessed: 29.05.2019).