Распознавание жестов мыши

Ивашева В. М., студент математико-механического факультета
СПбГУ, valeria.ivasheva@outlook.com

Аннотация

В последнее время все чаще используются нестандартные методы ввода. Данная статья рассматривает один из них, а именно мышиные жесты. Жесты мышью — механизм, способный сделать взаимодействие пользователя и системы более удобным, но требуется обеспечить высокую точность распознавания жеста. В данной работе приводится реализация и численное сравнение нескольких существующих алгоритмов. Точность классификации лучшего из рассмотренных (то есть отношение правильно классифицированных жестов ко всем жестам) составила 87% на наборе из 9 нетривиальных жестов.

Введение

Для расширения способов взаимодействия человека и компьютера вводятся все новые и новые устройства. Но помимо этого можно модернизировать и стандартные методы управления. Одним из перспективных вариантов являются мышиные жесты.

Жест мыши — это метод управления программами в компьютере при помощи движений мыши и ассоциированных с ними команд. В некоторых случаях "рисовать" оказывается легче и быстрее, чем вызывать их обычным способом или с помощью горячих клавиш. Например, для добавления элемента на диаграмму вместо того, чтобы использовать технологию drag-and-drop, можно изобразить необходимый жест.

На кафедре системного программирования математико-механического факультета СПбГУ разрабатывается REAL.NET [1] — инструмент для создания визуальных предметно-ориентированных языков. В данном проекте добавление элементов на сцену происходит с использованием технологии drag-and-drop. Возможность добавлять элементы с помощью жестов мыши может сделать REAL.NET более удобным для пользователей. Поскольку готовых библиотек для .NET, реализующих эту возможность, не существует, было решено создать свой инструмент для распознавания жестов мыши, обладающий следующими свойствами:

1. уникальность — возможность выбора жестов так, чтобы у каждого была единственная ассоциированная команда;

2. расширяемость — возможность легко добавлять новые жесты;

3. многообразие — возможность распознавания более сложных жестов, состоящих не только из элементарных направлений.

Таким образом, была поставлена задача разработать на .NET компоненту, позволяющую пользователю задать довольно много (порядка одного-двух десятков) классов жестов и классифицировать рисуемые мышью жесты с достаточной точностью (не менее 85%).

Существующие решения

1. Браузеры: Opera, Яндекс.Браузер.

   Браузеры Opera и Яндекс.Браузер поддерживают управление с помощью жестов мыши. Например в Яндекс.Браузере, для того, чтобы вернуться на предыдущую страницу, необходимо удерживать правую кнопку мыши и сдвинуть влево, для перехода на следующую — вправо. Команд для браузера, нуждающихся в поддержке жестов, немного, и все они реализованы с помощью простейших ломаных траекторий. Распознать такой жест не сложно, каждому отрезку ломаной присвоить направление, к которому принадлежит конец отрезка. Выделить 1-3 главных направления (вектора максимальной длины), с помощью которых можно распознать жест.

2. Различные утилиты.

   Для различных операционных систем существуют специальные утилиты, благодаря которым мышиные жесты можно добавить в любую программу. Например, для Windows это StrokeIt, StrokesPlus, Mouse gestures recognition, Justgestures и другие. Главным недостатком этих утилит является ограниченное число заранее определенных команд, таких как вставка, удаление, копирование и прочие. Есть способ справиться с ограничением, например, каждому жесту сопоставлять не одну команду, а целый список, из которого пользователь выберет нужную. Но в таком случае возникает вопрос, не легче ли просто вызвать ее привычным способом.

3. Visual Paradigm.

   В CASE-системе Visual Paradigm мышиные жесты являются полезным инструментом визуального моделирования. Они делятся на три группы: жесты рисования, создающие объекты, жесты команды и жесты соединения. Жестов здесь больше, чем во всех вышеописанных решениях, однако они жестко "зашиты" в код.

В большинстве открытых существующих решений используют машинное обучение. Но для этого нужна обучающая выборка, состоящая из объектов, распределенных по классам. При данном подходе трудно обеспечить возможность легкого добавления новых жестов, поскольку пользователям придется самим пополнять эту выборку, и это довольно трудоемкий процесс. Конечная цель работы — внедрение механизма распознавания в REAL.NET, где мышиные жесты задаются для каждого элемента визуального языка при его описании. Весь смысл REAL.NET и подобных технологий в том, что создать визуальный язык с его помощью очень просто и быстро, а если на обучение классификатора уходит больше времени, чем на создание самого языка, то толку от такого механизма для этого случая использования мало.

Предлагаемое решение

Мышиные жесты можно разделить на два вида: одноштриховые и многоштриховые (т.е. состоящие из нескольких штрихов). Для них могут применяться разные подходы к распознаванию. Далее в статье будут рассмотрены алгоритмы для распознавания одноштриховых жестов.

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 рассматривается как список точек траектории движения мыши.

2. Построение классификатора.

   Определяем классификатор — 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.

Рассмотрим алгоритм, классификатором которого является список характеристических точек. Характеристические точки — это точки, однозначно определяющие жесты или фигуры. Например, в прямоугольнике характеристические точки — это вершины. Метрика — сумма евклидового расстояния между точками из списков поэлементно. Для того, чтобы выполнялись все аксиомы метрики и сравнение было точным, необходимо иметь списки одинаковой длины. А также фигуры должны быть одинакового размера, поэтому каждый жест вписывается в квадрат со стороной максимальной длины идеального жеста с сохранением соотношения сторон.

Идеальные жесты описываются как списки характеристических точек, упорядоченные по направлению изображения. Дополняем каждый список до удвоенного количества максимальной длины для того, чтобы исключить возможность соответствия одинакового набора характеристических точек нескольким идеальным жестам. Например, ромб и окружность при 4-х элементном классификаторе будут при данном подходе представляться четырьмя характеристическими точками и выглядеть с точки зрения алгоритма одинаково, что приведет к неверному распознаванию окружности как ромба. Но если взять количество точек, участвующих в распознавании, как удвоенное, окружность будет сопоставляться с восьмиугольником, это гарантирует, что она не будет ошибочно отнесена к какому-либо классу. Таким образом, у каждого идеального жеста есть список характеристических точек, с которыми и будет происходить дальнейшее сравнение.

Алгоритм приведения списка до нужного количества точек

N — модуль разности длины списка и необходимого количества точек.

Если длина списка меньше нужного:

- Находим сегмент наибольшей длины и добавляем точку, являющуюся серединой этого отрезка. Повторяем данную операцию N раз.

Если длина списка больше:

- Находим сегмент наименьшей длины и удаляем точку, являющуюся началом данного отрезка. Если это первая точка списка, удаляем не ее, а вторую. Повторяем данную операцию N раз.

Алгоритм нахождения характеристических точек

Для нахождения характеристических точек рассмотрим два алгоритма. Начальные условия — это список точек пути мыши и заданная точность $\epsilon > 0$.

1. Алгоритм Дугласа-Пекера. [2]

   Первая и последняя точка списка отмечены к сохранению, они составляют отрезок AB. Находим самую удаленную точку от отрезка AB. Если расстояние между ними $<= \epsilon$, сохраняем данную точку, а все до нее отбрасываем. Иначе, алгоритм рекурсивно вызывает себя на наборе от начальной до данной и от данной до последней.

2. Алгоритм нахождения характеристических точек через углы. [3]

   Отмечаем первую точку как характеристическую. Далее в цикле от 2 до предпоследней: A — последняя отмеченная характеристическая точка, B — точка, рассматриваемая на данный момент, C — следующая за B. Если угол ABC(см. рис. 1) > 20 и расстояние AB $> \epsilon$, тогда отмечаем B как характеристическую. Последнюю точку также вносим в список характеристических.

Список, получившийся в результате алгоритмов Дугласа-Пекера или нахождения через углы, приводим до нужного количества точек с помощью алгоритма выше.

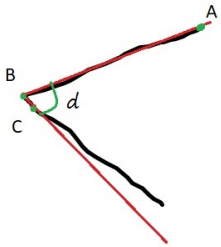

Рис. 1: Угол для проверки характеристической точки

Эксперименты

Было проведено тестирование на объектах, графическое представление которых представлено на рисунке 2. Для сравнения было протестирован также алгоритм, использующий построение сетки вокруг жеста [4]. Жест вписывается в прямоугольник, стороны которого разбиваются на 8 частей. Классификатор — ячейки прямоугольника, через которые проходит жест. Метрика — расстояние Левенштейна. Результаты тестирования представлены в таблице 1. Точность (precision) показывает насколько мы можем доверять классификатору (какая часть распознанных объектов была верно распознана). Полнота (recall) выражает как много истинных объектов выделяет алгоритм.

Формулы для вычисления точности и полноты.

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

α — класс, для которого мы считаем формулу, TP — верно распознанный жест, FP — жест распознанный как α, но не являющийся им на самом деле, FN — жест, являющийся α, но не распознанный алгоритмом. С precision и recall, вычисленными для каждого объекта, можно ознакомиться в таблице 2. Precision и recall для каждого алгоритма считаем как усредненные значения по всем классам.

Заключение

В результате работы был создан инструмент для распознавания мышиных жестов [5]. Было протестировано 3 алгоритма. Наилучший результат показал алгоритм, находящий характеристические точки через

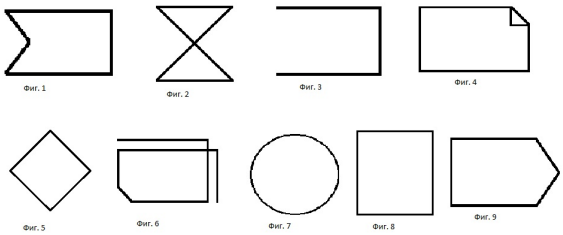

Рис. 2: Фигуры, на которых происходило тестирования

	Фиг. 1	Фиг. 2	Фиг. 3	Фиг. 4	Фиг. 5	Фиг. 6	Фиг. 7	Фиг. 8	Фиг. 9	Всего
Идеальное распознавание	115	142	133	128	147	122	147	122	121	1177
Разбиение на ячейки + расстояние Левенштейна	92	135	133	97	137	2	117	107	0	820
Дуглас-Пекер	41	121	32	104	141	83	94	87	96	799
Характеристические точки	103	140	83	107	145	121	90	122	118	1029

Таблица 1: Результаты тестирования алгоритмов

	Фиг. 1	Фиг. 2	Фиг. 3	Фиг. 4	Фиг. 5	Фиг. 6	Фиг. 7	Фиг. 8	Фиг. 9
Precision (разбиение на ячейки)	0.53	1	0.49	1	0.9	0.03	0.64	0.96	0
Recall (разбиение на ячейки)	0.8	0.95	1	0.75	0.93	0.01	0.79	0.87	0
Precision (Дуглас-Пекер)	0.57	0.67	0.51	0.72	0.68	0.49	0.8	0.98	0.67
Recall (Дуглас-Пекер)	0.35	0.85	0.24	0.81	0.95	0.68	0.63	0.71	0.79
Precision (Характеристические точки)	0.66	0.76	0.83	0.99	0.95	0.99	0.81	0.99	0.93
Recall (Характеристические точки)	0.89	0.98	0.62	0.83	0.98	0.99	0.61	1	0.97

Таблица 2: Результаты тестирования алгоритмов, precision и recall

углы, его precision = 0.88, recall = 0.87. Данная точность и полнота недостаточны для использования в визуальных системах, так как приносит неудобство пользователю. Направлением дальнейшего развития является их улучшение с целью дальнейшего внедрения в среду визуального моделирования, а также реализация поддержки многоштриховых жестов.

Литература

[1] Литвинов, Ю. В., Кузьмина, Е. В., Небогатиков, И. Ю., Алымова, Д. А., Среда предметно-ориентированного визуального моделирования REAL.NET. // СПИСОК - 2017. 7-ая всероссийская научная конференция по проблемам информатики (Санкт-Петербург, 25-27 апр. 2017 г.). – Санкт-Петербург, 2017. – С. 80-89.

[2] David Douglas Thomas Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature// The Canadian Cartographer, Vol 10, No. 2, pp. 112-122

[3] Pawel Hofman, Maciej Piasecki, Efficient Recognition of Mouse-based Gestures//. URL `https://www.ii.pwr.edu.pl/~piasecki/publications/hofman-piasecki-v1-1.pdf` (дата обращения: 14.04.2019)

[4] Осечкина, М. С., Курсовая, Визуальное программирование при помощи мыши// сайт кафедры системного программирования СПбГУ. URL `http://se.math.spbu.ru/SE/YearlyProjects/2010/YearlyProjects/2010/345/Osechkina_report.pdf` (дата обращения: 14.04.2019)

[5] Github-репозиторий проекта. URL `https://github.com/valeria-ivasheva/recognizer` (дата обращения: 14.04.2019)