Реализация инструмента рисования для графических редакторов «Динамическая кисть»

Домнин Е.О., Павельев А.А., МГТУ им. Н.Э. Баумана,
domnin12@gmail.com, pavelyev@bmstu.ru

Аннотация

В данной статье будут рассмотрены особенности реализации инструмента рисования для графических редакторов «Динамическая кисть» для использования с графическим планшетом в качестве устройства ввода.

1 Введение

Современные дизайнеры и художники, как профессионалы, так и любители используют для рисования графические планшеты и соответствующее программное обеспечение, предоставляющие больше возможностей для создания изображений на персональном компьютере, чем классические мышь с клавиатурой.

Одной из ключевых особенностей графического планшета является датчик давления. Как правило этот датчик используется для динамического изменения параметров кисти, таких как прозрачность или размер мазка. Инструмент рисования позволяющий динамическое изменение параметров называется “Динамической кистью”. Также важно учесть частоту поступления от него данных, как правило они поступают с частотой около 125 Гц, то есть точка каждые 8 миллисекунд.

2 Анализ существующих алгоритмов

В качестве входных данных нам поступает некоторый набор точек, по которым мы рисуем кривую. Точка состоит из координат по осям X и У, а также уровня давления в ней. Так что у нас есть две подзадачи: обработка набора точек и его отрисовка. Будем называть мазком результат отрисовки нашей кисти в одной точке.

2.1 Базовый алгоритм

Самым простым способом отрисовки будет рассмотрение каждых двух точек в наборе как начало и конец отрезка, которые мы соединяем, проводя линию мазком, который должен быть в начальной точке.

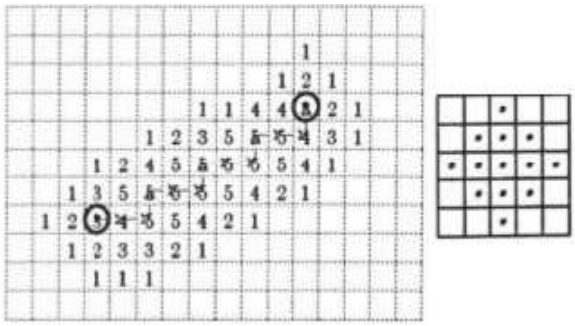

Рис. 1. Визиты пикселей в базовой реализации

Обозначим длину линии соединяющей точки как d, а радиус круга, описывающего мазок, как r, тогда получаем $O(dr^2)$ визитов пикселей, $O(d)$ вычислений, деградация изображения будет заметна при достаточно большом расстоянии между точками.

2.2 Наивный алгоритм отрисовки мазками

В случае наивного алгоритма каждые две точки в наборе мы рассматриваем как начало и конец линии, с помощью алгоритма, аналогичного Брезенхему, мы выбираем из линии пиксели и каждый из них считаем центром отдельного мазка, это позволяет плавно изменять параметры между мазками. По сложности алгоритм отличается от базового незначительно: $O(d((r_1 + r_2)/2)^2)$ визитов пикселей, где r_1 это радиус мазка в начальной точке, а r_2 радиус мазка в конечной точке, вычислений также $O(d)$ [1].

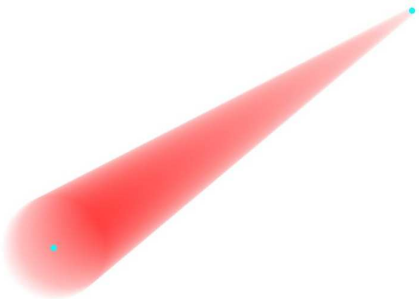

Рис. 2. Динамическое изменение толщины и прозрачности между точками

2.3 Алгоритм отрисовки мазками с модификацией отступов

В наивном алгоритме мы рисовали новый мазок на каждом следующем пикселе на пути линии, попробуем уменьшить частоту отрисовки мазков в зависимости от их размера. Сверху показан результат рисования кистью в виде незаполненного круга, снизу - заполненного.

При отступе в 25 % радиуса мазка получаем хорошо заметные отдельные мазки в случае прозрачной кисти, а не линию, как видно на рисунке.

Рис. 3. Отступ 25% радиуса мазка

При отступе в 1% мазки практически незаметно. Такой отступ сэкономит ресурсы относительно наивной реализации для кистей с радиусом более 100 пикселей.

Рис. 4. Отступ 1% радиуса мазка

В отличие от простого алгоритма мы получаем некоторый выигрыш в скорости, но при этом происходит ухудшение качества изображения. Оптимальным вариантом является использование отступа в 25% для непрозрачной кисти [2]. По сложности алгоритм отличается от наивной отрисовки мазками в $min(r_1, r_2)/4$ раз по визитам пикселей, что не меняет порядок сложности в $O(d((r_1 + r_2)/2)^2)$ визитов пикселей, где r_1 это радиус мазка в начальной точке, а r_2 радиус мазка в конечной точке, вычислений также $O(d)$.

2.4 Алгоритм Бидермана

Данный алгоритм требует, чтобы кисть была выпуклым многоугольником, не поддерживает прозрачность и изменение параметров между точками. Идея заключается в том, чтобы в случае симметричной кисти представить её как кисть высотой $2r$ и шириной в один пиксель - центральный столбец кисти. Тогда в начале и конце отрезка можно нарисовать полные мазки, а между ними закрасить всё этой упрощённой кистью. Для асимметричной кисти требуется взять две кисти - горизонтальный и вертикальный разрезы. Соответственно, алгоритм прохода по линии, аналогичный Брезенхему, решает, какой из мазков использовать: при движении по горизонтали - вертикальный, при движении по вертикали - горизонтальный и при движении по диагонали - оба [1].

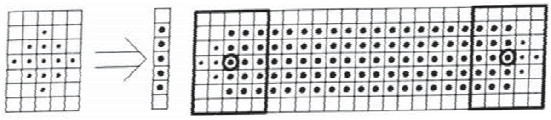

Рис. 5. Симметричный случай Бидермана

В этом алгоритме мы получаем $O(dr)$ визитов пикселей, $O(d)$ вычислений, ухудшение качества изображения и отсутствие поддержки прозрачности.

2.5 Алгоритм с использованием сплайнов

В статье [3] рассматривается два вида сплайнов: Б-сплайны и интерполяционные сплайны. Разница заключается в том, что интерполяционный обязательно проходит через центры точек, а Б-сплайн аппроксимирует и не обязан проходить через них, однако общая идея одинаковая. Так как нам важно наиболее близкое соответствие рисунка мазку, мы будем использовать одну из разновидностей интерполяционного сплайна. На некотором расстоянии друг от друга выбираются контрольные точки, которые мы назовём узлами и в которых мы будем хранить значения различных параметров. С помощью сплайнов мы строим кривую, соединяющую точки мазков в узлах. С помощью построенной кривой получаем набор промежуточных междуузлами точек, которые соединяются одним из алгоритмов соединения двух точек, линейно изменяя параметры от узла к узлу. Чем большая степеньиспользуется для полиномов в сплайне, тем больше требуется вычислений и тем сильнее он колеблется между узлами, потому оптимально использовать кубический сплайн [4]. Основные проблема алгоритма - сложность предварительных вычислений, которые зависят только от количества точек и возможность “выброса”- некрасиво построенного сплайна между точками. Таким образом получаем $O(NX_1)$ визитов пикселей, $O(NX_2)$ вычислений, где N - количество интерполированных точек, X_1 - количество ви-

зитов пикселей a, X_2 - сложность вычислений в соединяющем их алгоритме и возможную деградацию изображения из-за “выбросов”относительно наивного метода. [3]

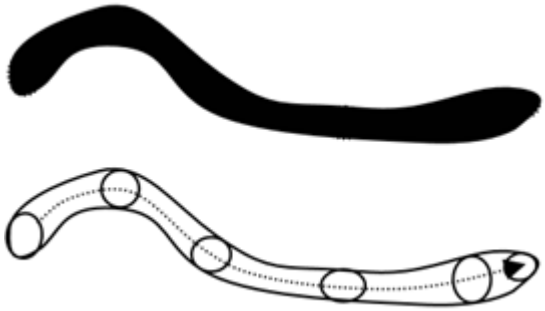

Рис. 3. Отрисовка сплайном

2.6 Итоговое сравнение алгоритмов

Табл.1 Сравнение алгоритмов

Алгоритм	Базовый	Наивный	Мод. отступов	Бидерман	Сплайны
Операций с пикселями	$O(dr^2)$	$O(dr^2)$	$O(dr^2)$	$O(dr)$	$O(dr^2)$
Предварит. вычислений	$O(d)$	$O(d)$	$O(d)$	$O(d)$	$O(N)$
Поддержка всех параметров	Да	Да	Да	Нет	Да
Деградация изображения	Да	Нет	Да	Да	Да

2.7 Выводы

В большинстве современных редакторов используется алгоритм отрисовки мазками с отступами из-за его относительно простой реализации, почти незаметного ухудшения качества изображения по сравнению с наивной реализацией при использовании небольших отступов и поддержки всех возможных параметров. Однако, он плохо подходит для рисования прозрачной кистью, поэтому представляет интерес гибрид двух алгоритмов: рисующий без отступов в случае прозрачной кисти для лучшего качества картинки и рисующий с отступами при непрозрачной кисти для лучшего быстродействия.

Также интересным выглядит алгоритм, использующий сплайны, поскольку из-за недостаточно высокой частоты опроса у графического планшета при быстром движении просто соединяя точки линиями мы можем увидеть, что кривая состоит из нескольких отрезков, что некрасиво, а сплайн сгладит этот артефакт, однако у него велика сложность предварительных вычислений, что ставит под вопрос возможность комфортной интерактивной отрисовки.

Так как алгоритм Бидермана не позволяет динамически изменять параметры между точками, результат его работы будет слишком некрасив, а, следовательно, он не интересен, несмотря на скорость его работы.

Нужно определить возможно ли использование такого сложного алгоритма как отрисовка сплайнами при интерактивном вводе, требуется ли оно при рисовании с устройства ввода с относительно небольшой частотой опроса и насколько заметно отличие полученного изображения от идеала, для этого потребуется реализация алгоритмов и сравнительные эксперименты.

3 Алгоритм отрисовки сплайнами

Как уже было сказано ранее, оптимальным вариантом сплайна является кубический сплайн, однако у него есть несколько модификаций, для начала рассмотрим общий вид кусочно кубической интерполяции [5]. Для сплайна нам нужно чтобы были заданы точки $a = x_1 < x_2 <\ldots< x_n = b$ и соответствующие им значения $f(x_1), f(x_2)\ldots, f(x_n)$, по ним строится интерполирующая функция Pf таким образом что на каждом отрезке $[x_i, x_i + 1], y = 1,\ldots, n - 1$ она является многочленом P_i степени 3, таким, что

$$\begin{cases} P_i(x_i) = f(x_i), & P_i(x_i + 1) = f(x_i + 1) \\ P_i`(x_i) = d_i, & _i`(x_i + 1) = d_i + 1 \end{cases} \quad (1)$$

для $i = 1,\ldots, n - 1$ и где $d_i, i = 1,\ldots, n$ - свободные параметры, тот или иной способ выбора которых определяет метод кусочной интерполяции кубическими многочленами. Полученная функция Pf совпадает с f в точках $x_i, i = 1,.., n$ и для любого набора параметров $d_i Pf \subset C^{(1)}([a, b])$.

Коэффициенты многочлена P_i, записанного в форме

$$P_i(x) = a_{1,i} + a_{2,i}(x - x_i) + a_{3,i}(x - x_i)^2 + a_{4,i}(x - x_i)^2(x - x_{i+1}) \quad (2)$$

могут быть вычислены по интерполяционной формуле Ньютона с кратными узлами из которой получаем

$$a_{1,i} = f(x_i)$$
$$a_{2,i} = d_i$$
$$a_{3,i} = \frac{f(x_i;x_{i+1})-d_i}{x_{i+1}-x_i} \quad (3)$$
$$a_{4,i} = \frac{d_i+d_{i+1}-2f(x_i;x_{i+1})}{(x_{i+1}-x_i)^2}$$

, где $f(x_i;x_j) = \frac{f(x_j)-f(x_i)}{x_j-x_i}$ - разделённая разница.

Перейдём к другому виду

$$P_i(x) = c_{1,i} + c_{2,i}(x-x_i) + c_{3,i}(x-x_i)^2 + c_{4,i}(x-x_i)^3 \quad (4)$$

, где коэфициенты с берутся согласно (5).

3.1 Метод Акимы

Этот метод приближения используется для борьбы с выбросами приближающей функции, которые появляются, если значения функции в точках заданы с некоторой погрешностью. Поскольку выбросы нежелательны, а разница по сложности вычислений с простым кубическим сплайном незначительна, мы будем использовать этот метод. [4][6]

Разделённая разность $f(x_{i-1,x_i})$ является приближением к $f`(x_i)$ слева, а $f(x_i,x_{i+1})$ является приближением к $f`(x_i)$ справа. В методе Акимы эти приближения усредняются с весам, которые тем больше, чем меньше гладкость функции на соседнем отрезке. Окончательная формула для определения параметра d_i имеет вид (6), где $i = 3,4,\dots,n-2$ и $w_j = |f(x_j;x_{j+1}) - f(x_{j-1};x_j)|$

Для получения недостающих значений d_1, d_2, d_{n-1}, d_n точки экстраполируются по формулам (7).

$$c_{1,i} = a_{1,i} = f(x_i)$$
$$c_{2,i} = a_{2,i} = d_i$$
$$c_{3,i} = a_{3,i} - a_{4,i}(x_{i+1}-_i) = \frac{3f(x_i;x_{i+1}) - 2d_i - d_{i+1}}{x_{i+1}-x_i} \quad (5)$$
$$c_{4,i} = a_{4,i} = \frac{d_i + d_{i+1} - 2f(x_i;x_{i+1})}{(x_{i+1}-x_i)^2}$$

$$d_i = \begin{cases} \frac{w_{i+1}f(x_{i-1};x_i) + w_{i-1}f(x_i;x_{i+1})}{w_{i+1}+w_{i-1}}, \text{ если } w_{i+1}^2 + w_{i-1}^2 \neq 0 \\ \frac{(x_{i+1}-x_i)f(x_{i-1};x_i) + (x_i-x_{i-1})f(x_i;x_{i+1})}{x_{i+1}-x_{i-1}}, \text{ если } w_{i+1} = w_{i-1} = 0 \end{cases} \quad (6)$$

$$x_0 = 2x_2 - x_4$$
$$y_0 = (x_1 - x_0)(f(x_2;x_2) - 2f(x_2;x_1)) + y_1$$
$$x_1 = x_2 + x_3 - x_4$$
$$y_1 = (x_4 - x_3)\big(f(x_4;x_3) - 2f(x_3;x_2)\big) + y_2 \quad (7)$$
$$x_{n-2} = x_{n-3} + x_{n-4} - x_{n-5}$$
$$y_{n-2} = (x_{n-2} - x_{n-3})(2f(x_{n-3};x_{n-4}) - f(x_{n-4};x_{n-3})) + y_{n-3}$$
$$x_{n-2} = 2x_{n-3} - x_{n-5}$$
$$y_{n-2} = (x_{n-1} - x_{n-2})(2f(x_{n-2};x_{n-3}) - f(x_{n-3};x_{n-2})) + y_{n-2}$$

3.2 Проблема использования сплайнов при рисовании

Основное применение сплайнов в компьютерной графике - построение графиков и с этим они хорошо справляются, однако при использовании сплайнов в рисовании у нас могут не выполнятся условия упорядоченности точек по оси Ox и как следствие сплайн не может быть построен. Кроме того, в условиях сильно различающихся изменений по осям Ox и Oy сплайн может рисовать кривую с слишком большим изгибом. Следовательно, требуется модифицировать алгоритм рисования сплайна.

3.3 Предлагаемая модификация

Модификация должна решать две проблемы:

Первая - не по каждому набору точке можно построить сплайн, необходимо чтобы по одной из координат они не совпадали и были упорядочены по возрастанию, если представить точки в таком виде возможно, то строит-

ся сплайн, иначе точки соединяются прямыми линиями.

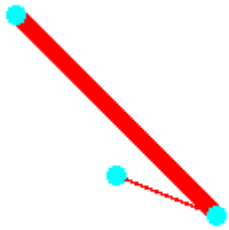

Рис. 4. Пример точек, по которым невозможно построить сплайн

Вторая - в результате построения сплайна мы можем получить кривую, которая будет иметь сильный изгиб, так называемый выброс, такой результат отрисовывать не стоит, поэтому проверяется выходит ли кривая за границы наименьшего прямоугольника, описывающего входные три точки и рисуется только если не выходит.

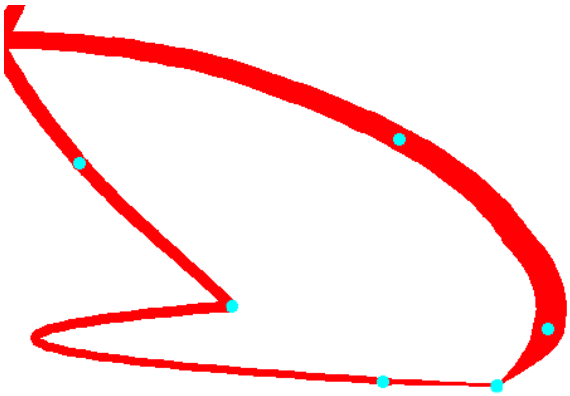

Рис. 5. Выброс

4 Алгоритм отрисовки мазками

Для отрисовки непрозрачной кистью будем использовать алгоритм с модификацией отступа 25%, а для прозрачной кисти наивный алгоритм для баланса между красотой картинки и скоростью работы.

5 Сравнительное тестирование алгоритмов

Поскольку однозначно лучшего алгоритма выбрано не было, было решено реализовать алгоритм отрисовки мазками, сплайнами и базовый, после чего провести сравнительное тестирование.

Для реализации использовался язык C++, библиотеки Qt среда разработки Qt Creator, тестирование проводилось под ОС Linux, а точнее Manjaro 16.10.

В качестве устройства ввода использовался графический планшет Wacom Bamboo Fun CTE-650, он передаёт до 133 точек в секунду.

На представленном ниже рисунке видны артефакты, связанные с частотой ввода:

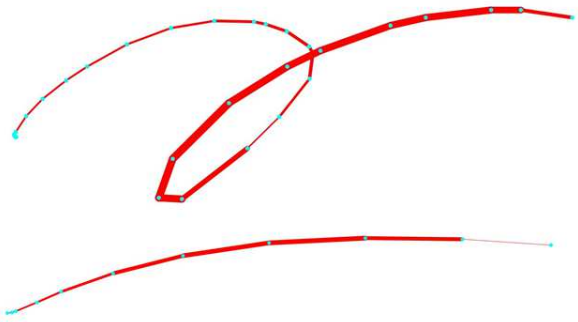

Рис. 6. Артефакты при быстрых штрихах

5.1 Тестовые данные

Важным является вопрос эталонного результата, ведь провести одну и ту же кривую, но с разными скоростями практически невозможно, ведь человек не может идеально контролировать силу нажатия и траекторию движения пера. Однако мы можем взять математическую функцию и различными способами выбрать точки, которые в ней стоит соединять. Так как скорость проведения пером по планшету может меняться возьмём два варианта выбора точек: с фиксированным шагом аргумента, симулирующую проведение кривой без ускорения и с увеличивающимся в геометрической прогрессии с знаменателем прогрессии 1.2 аргументом, для симуляции ускорения, более быстро возрастающие прогрессии дают бессмысленно плохой результат. Наиболее реалистичным вариантом, показывающим проблему, является диапазон от 25 до 100 точек. Так что будем выполнять тест для 30 и 60 точек с равным расстоянием между точками и 18 и 22 точек для случая с увеличивающимся расстоянием соответственно.

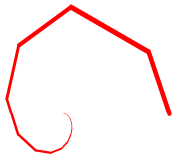

Рис. 7. Спираль Архимеда, 18 точек с ускорением, базовый алгоритм

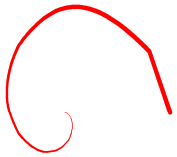

Рис. 8. Спираль Архимеда, 18 точек с ускорением, алгоритм

5.2 Эталонная версия

Эталонной версией будем считать версию с увеличенным на порядок количеством точек (1000), отрисованных наивным алгоритмом. С таким количеством точек расстояние между несколькими соседними получается не более нескольких пикселей и потому то, что кривая на самом деле состоит из отрезков не заметно.

Табл 2. Описание экспериментов

Эксперимент	Функция	Расстояние между точками
1	Спираль архимеда	Геометрическая прогрессия
2	Спираль архимеда	Фиксированное
3	Кардиоида	Геометрическая прогрессия
4	Кардиоида	Фиксированное
5	Дельтоид	Геометрическая прогрессия
6	Дельтоид	Фиксированное
7	Ранункулоид	Геометрическая прогрессия
8	Ранункулоид	Фиксированное
9	Синус	Геометрическая прогрессия
10	Синус	Фиксированное
11	Квадратный корень	Геометрическая прогрессия
12	Квадратный корень	Фиксированное
13	Трифолиум	Геометрическая прогрессия
14	Трифолиум	Фиксированное

5.3 Алгоритм сравнения

Для сравнения по скорости выполнения выполним каждый из вариантов комбинации алгоритм/набор точек 100 раз, для усреднения результата и замерим суммарное время отрисовки в миллисекундах. Для сравнения совпадения с эталоном мы будем попиксельно сравнивать изображения, подсчитывая число пикселей закрашенных цветом, отличающимся от цвета фона, в переменной $pixels$, суммировать разницу цвета в переменной $diff$, тогда процент совпадения с эталоном может быть посчитана по формуле $\frac{diff}{pixels} * 100\%$.

6 Результаты сравнения

Совпадения. Некоторые эксперименты показывают примерно одинаковые результаты, особенно 10, 11 и 12, в целом мазок немного лучше базового, а сплайн то примерно равен, то заметно лучше. Лучшие результаты сплайн показывает на тестах 1, 8 и 5. В первом тесте сплайн показывает более чем в два раза больше совпадений, чем другие алгоритмы. Также многие тесты показывают высокий процент совпадений на 60\22 точках, 8 из 12 показали больше 75%.

Производительность. Базовый алгоритм на порядок быстрее алгоритма отрисовки мазка, а тот в свою очередь до 5 раз быстрее алгоритма отрисовки сплайнами, однако в среднем быстрее всего на 50%, а в 11м эксперименте при 30 точках даже медленнее на 32%. Однако даже в худшем случае, 4й эксперимент при отрисовке 60 точек 10 раз требует 3400 миллисекунд, что даёт менее 6 миллисекунд на отрисовку, тогда как точки поступают от планшета с частотой 8 миллисекунд что означает достаточную производительность.

6.1 Выводы

Как мы видим, в значительной части случаев все три алгоритма показывают схожие результаты совпадения с эталонной версией, однако достаточно часто сплайн показывает заметно больший процент совпадения с эталонной версией(до 30% разницы с базовым алгоритмом), но при этом по скорости он на порядок проигрывает базовому, а алгоритму мазками в зависимости от эксперимента проигрывает в производительности от нескольких процентов до 5 раз, однако даже в худшем случае мы получаем менее миллисекунды на построение связи между двумя точками, тогда как мы получаем точку раз в 8 миллисекунд, а значит его производительность достаточна для комфортной интерактивной работы.

7 Заключение

В рамках данной работы нами был проведён анализ существующих алгоритмов, используемых при соединении точек, два из них были успешно модифицированы, три реали-

зованы, после чего был проведено исследование скорости их работы и совпадения с эталонной версией отрисовки. Также в рамках исследования были подтверждены теоретические преимущества и недостатки алгоритмов. По итогам данного исследования можно утверждать, что алгоритм отрисовки со сплайнами может использоваться в современных графических редакторах и имеет преимущества над более примитивными алгоритмами.

Список литературы

Fishkin, Kenneth P. *Algorithms for brush movement* / Kenneth P. Fishkin, Brian A. Barsky // The Visual Computer. — 1985. — Vol. 1, no. 4. — Pp. 221–230. http://dx.doi.org/10.1007/BF02021811.

Jansson, Erika. *Brush Painting Algorithms* / Erika Jansson. — 2004.

Simulating Artistic Brushstrokes Using Interval Splines / Sara L. Su, Ying-Qing Xu, Heung-Yeung Shum, Falai Chen//Proceedings of the 5th IAST ED International Conference on Computer Graphics and Imaging (CGIM ’02, Kauai, Hawaii, August 2002). — 2002. — Pp. 85–90. http://people.csail.mit.edu/sarasu/pub/cgim02/.

Akima, Hiroshi. *A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures*. — 1970.

К.Ю, Богачев. *Практикум на ЭВМ. Методы приближения функций* / Богачев К.Ю. — 1998.

Özdemir, Hüseyin. *Comparison of linear , cubic spline and akima interpolation methods.* / Hüseyin Özdemir. — 2007.