Деформация поверхностей на основе функций радиального базиса с компактным носителем

М. А. Сенин(1), Н. Е. Кожекин (2), В. В. Савченко (3)

(1) Московский Физико-Технический Институт[1], Россия,
(2) Tokyo Institute of Technology[2], Japan,
(3) Hosei University[3], Japan.

Аннотация

В данной работе представлен подход к решению задачи анимации деформируемых объектов. Алгоритм, использующий функции радиального базиса с компактным носителем (compactly supported radial basis functions — CSRBF), позволяет вычислять деформации в реальном времени. В основе алгоритма лежит техника пространственного отображения. Гладкие локальные деформации могут быть заданы небольшим количеством контрольных векторов. Локальность деформации определяется радиусом поддержки.

1 Введение

В данной работе рассматривается проблема анимации деформируемых объектов. Анимация используется при решении многих прикладных задач. Например, в компьютерных играх, при моделировании процессов с изменяющейся геометрией, в кино, в графических пользовательских интерфейсах и т.п. Поверхности объектов состоят, как правило, из большого количества геометрических примитивов, поэтому было бы крайне неэффективно задавать деформацию, явно указывая перемещение каждого примитива. Необходимо задавать деформацию на некотором макроскопическом уровне, например, определяя перемещения относительно небольшого количества контрольных точек и интерполируя положение остальных точек. При этом важно, чтобы результат интерполяции совпадал с интуитивно ожидаемым результатом. Здесь следует отметить, что точное физическое моделирование требуется далеко не всегда. Существует множество областей (например, компьютерные игры) в которых важно, чтобы деформация лишь выглядела правдоподобно. Это позволяет упростить вычисления и ускорить работу алгоритма. Деформация должна быть гладкой и, что не менее важно, локальной, т.е. должна затрагивать небольшую область вокруг контрольной точки. Так же предпочтительно производить вычисления в режиме реального времени. В течение последних лет было предложено много подходов к решению данной проблемы, однако, несмотря на большое практическое значение, она не была полностью решена.

Целью работы было создание алгоритма быстрого вычисления гладких, локальных и правдоподобных деформаций, заданных небольшим количеством контрольных векторов.

Алгоритм основан на использовании CSRBF предложенных в [1] для интерполяции точек поверхности деформируемого объекта. Эта работа является продолжением работы [2] в которой CSRBF были использованы для восстановления поверхностей трехмерных геометрических моделей. Оптимизация алгоритма позволяет производить вычисления деформации поверхности в реальном времени.

2 Обзор

Деформацию объекта можно рассматривать как непрерывную трансформацию одного тела (исходная форма объекта) в другое (форма объекта после деформации). Существующие методы трансформации можно разделить на три группы:

[1]Россия, Москва 113303, ул. Керченская, д. 1«А», корп. 1, МФТИ

[2]Tokyo Institute of Technology, Faculty of Engineering, Department of Mechanical Sciences and Engineering, Hagiwara's laboratory, 2-12-1, O-okayama, Meguro-ku, Tokyo, 152-8552

[3]Hosei University, Faculty of Computer and Information Sciences, 3-7-2 Kajino-cho Koganei-shi, Tokyo 184-8584

- отображение пространства на себя;
- метаморфоза;
- модификация определяющих функций.

Подробное описание всех этих методов выходит за рамки данной статьи, поэтому ограничимся лишь кратким описанием некоторых из них. Более полный обзор можно найти в работе [3].

Отображение может задаваться некоторыми предопределенными функциями и контролироваться параметрами этих функций. Другим подходом является задание параметров отображения при помощи дифференциальных уравнений или при помощи набора контрольных векторов.

Хорошо известным примером пространственных отображений являются Свободные деформации (Free-Form deformations), где трансформация определяется набором точек, задаваемым пользователем. Первой работой на эту тему стала работа [4], за которой последовали работы [5, 6, 7]. Обобщение методов трансформации основанных на пространственном отображении было сделано в работе [8]. Были описаны прямое и обратное отображение для поверхностей заданных неявными функциями.

Большинство методов трансформации не приспособлены для создания локальных деформаций заданных набором произвольных контрольных точек. Хотя метод описанный в работе [9] был разработан именно для локальных пространственных отображений, он может приводить к неправдоподобным деформациям, если ограничивающие сферы нескольких контрольных точек пересекаются.

Общие математические основы методов пространственных деформаций можно найти в работе [10]. Большое количество литературы посвящено вопросу интерполяции поверхностей заданных набором точек. Именно выбор метода интерполяции во многом определяет свойства деформации. Один из подходов заключается в использовании методов интерполяции минимизирующих энергию упругой деформации [11, 12, 13]. Эти методы широко обсуждаются в литературе, в частности в работах [14, 15]. Насколько нам известно, первыми публикациями, описывающими использование контрольных точек, были работы [16, 17]. В работе [18] обсуждается метод, в котором точки выбираются так, чтобы полученная конфигурация соответствовала минимуму энергии упругой деформации.

Ещё одним подходом является использование при интерполяции радиальных функций. Преимущества использования радиальных функций обсуждались во многих работах. То, что величина деформации в некоторой точке зависит лишь от расстояния до контрольных точек, соответствует интуитивным ожиданиям и делает деформации правдоподобными. Радиальные функции применялись для двумерной и трехмерной анимации (см. [19]), в медицинских приложениях (см. [16, 20]), для восстановления поверхностей по набору точек, полученных в результате лазерного сканирования. Для увеличения скорости работы были предложены специальные методы для случая thin-plane сплайнов. Они обсуждались в работах [21, 22] а также недавних работах [23, 24].

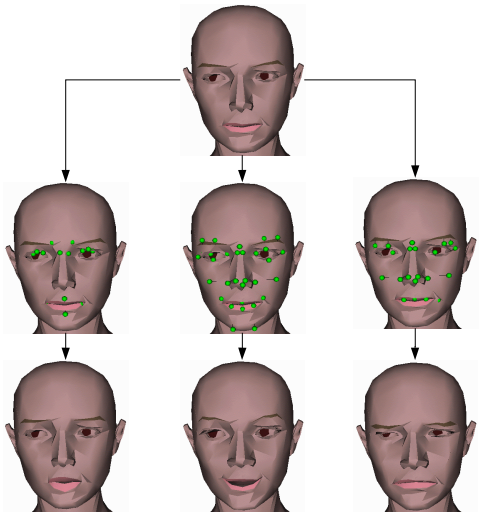


Рис. 1: Анимация мимики.

В целом, методы, основанные на использовании функций радиального базиса можно разделить на три группы. Первая группа состоит из «наивных» методов, которые применимы в случае относительно небольших моделей и работают достаточно хорошо в приложениях связанных с трансформациями (см. например [25, 26]). Эти методы требуют большого объема вычислений и поэтому практически неприменимы для анимации в реальном времени. Вторая группа состоит из быстрых методов, позволяющих обрабатывать большие объемы данных. Их описание можно найти в работах [24, 21]. Эти методы не являются локальными.

К третьей группе относятся методы, использующие CSRBF [1]. Они применялись, в том числе при реконструкции поверхностей (см. работы [2, 27]). Очень хороший обзор этой группы методов может быть найден в работе [28], которая также содержит описание задач решаемых этими методами и ограничений по применению. В ней также обсуждается вопрос интерполяции данных и приводится быстрый алгоритм конструирования C^2–непрерывных интерполяционных функций.

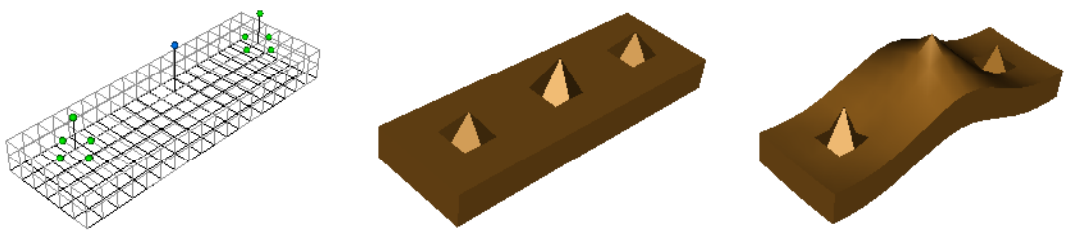
Одним из основных преимуществ использование функций с компактным носителем (ограниченным радиусом поддержки) является возможность делать деформации локальными без потери гладкости. Гладкие локальные деформации могут быть использованы при анимации изменяющихся объектов, особенно хорошо они подходят для симуляции лица. В последние годы появился значительный интерес к моделированию виртуальных актеров, большое внимание было уделено моделированию выражений лица говорящего человека. Задачи, поставленные в этой области исследований, до сих пор являются наиболее сложными, поскольку скорость, гладкость и правдоподобность деформаций становятся очень важны. Попытки решения этих задач были предприняты в работах [29, 30], обзор может быть найден в [31]. При моделировании лица говорящего человека преобладали два подхода: двухмерный и трехмерный. В первом случае создается трехмерная модель лица или всей головы (см. рис. 1). Во втором случае используются изображения. Двухмерный подход имеет ряд ограничений, однако требует меньше системных ресурсов, что позволяет использовать его даже в таких устройствах, как мобильные телефоны.

Алгоритм представленный в данной работе позволяет получить хорошие результаты даже в таких сложных случаях, как симуляция лица.

3 Деформация поверхности с использованием CSRBF

Объект анимации обычно моделируется скелетом и поверхностным слоем (см. например [32, 33]). Скелет определяет характерные макроскопические движения объекта; изменения поверхностного слоя задаются относительно скелета. Таким образом, общее изменение формы объекта может быть представлено в виде суммы глобальных перемещений скелета и локальных перемещений поверхностного слоя. Мы будем рассматривать именно локальные перемещения поверхностного слоя.

Как было сказано выше удобно задавать перемещение набором контрольных векторов (т. е. набором точек образы которых предопределены) и интерполировать по ним деформируемую поверхность. Как правило (см. [18]) предполагается, что перемещение должно выглядеть как эластичная деформация — в подавляющем большинстве случаев пользователь интуитивно ожидает именно этого. Как показывает пример на рисунке 2 интерполяция с использованием CSRBF позволяет получить деформацию с хорошими визуальными свойствами соответствующую ожиданиям пользователя.



(a) (b) (c)

Рис. 2: Пример пластичной деформации модели кирпича. (a) — Исходная форма кирпича. Вектора (точки) показывают направление и величину деформации. (b), (c) — различные деформации в зависимости от радиуса поддержки.

Для конструирования отображения $\mathbb{R}^3 \to \mathbb{R}^3$ мы будем использовать интерполяцию, основанную на CSRBF. Мы можем построить отображение $T : \mathbb{R}^3 \to \mathbb{R}$ для каждой координаты

и получить требуемое отображение $(x, y, z) \to (\tilde{x}, \tilde{y}, \tilde{z})$, где — $\tilde{x} = T_x(x, y, z)$, $\tilde{y} = T_y(x, y, z)$ и $\tilde{z} = T_z(x, y, z)$).

Опишем теперь построение отображения T. Пусть задано множество попарно различных контрольных точек $X = \{\vec{x}_1, \ldots, \vec{x}_N\} \subseteq \mathbb{R}^3$. Предположим далее, что нам известны значения $g_1, \ldots, g_N$ в контрольных точках. Это будут соответственно $\tilde{x}_1, \ldots, \tilde{x}_N$, $\tilde{y}_1, \ldots, \tilde{y}_N$ или $\tilde{z}_1, \ldots, \tilde{z}_N$. Будем искать непрерывную функцию, которая интерполирует эти значения в контрольных точках. Интерполяция с использованием радиальных функций имеет вид:

$$s_{g,X}(\vec{x}) = \sum_{i=1}^{N} \alpha_i \varphi(\|\vec{x} - \vec{x}_i\|), \tag{1}$$

где $\|\cdot\|$ обозначает стандартную Евклидову норму в $\mathbb{R}^3$, а φ — радиальная функция.

Коэффициенты α_i определяются из условия интерполяции

$$s_{g,X}(\vec{x}_j) = g_j, \quad 1 \leqslant j \leqslant N. \tag{2}$$

В нашей работе мы опирались на работу [1], в которой для $\mathbb{R}^3$ был предложен новый класс положительно определенных функций радиального базиса с компактным носителем:

$$\varphi(r) = \begin{cases} \psi(r), & 0 \leqslant r \leqslant 1 \\ 0, & r > 1, \end{cases} \tag{3}$$

где $\psi(r)$ многочлен одной переменной.

В наших приложениях мы использовали функцию $\psi_{2,0}(r) = (1-r)_+^2$, принадлежащую классу C^0. Эта функция требует наименьших вычислительных затрат и, как следует из результатов наших экспериментов, приводит к правдоподобным деформациям. Обсуждение этого вопроса может быть найдено в статье [34].

Интерполяция уникальна, потому что в нашем случае φ является положительно определенной функцией.

Условие (2) представляют собой систему линейных алгебраических уравнений (СЛАУ) на коэффициенты α_i.

Поскольку φ имеет ограниченный радиус поддержки, матрица полученной системы уравнений будет разреженной. После специальной сортировки (описанной в пункте 4.2), полученная матрица $A = \{\alpha_{ij}\} = \{\varphi(\|\vec{x}_i - \vec{x}_j\|)\}$ размера $N \times N$ будет не только разреженной, но и полосно-диагональной. Это позволяет хранить её в очень компактном виде. Кроме того, существует множество быстрых алгоритмов решения уравнений заданных такими матрицами. Это позволяет значительно уменьшить время вычислений и делает возможными выполнять их в режиме реального времени.

4 Описание алгоритма

Алгоритм состоит из следующих частей:

- сортировка входных данных;
- конструирование СЛАУ;
- решение СЛАУ;
- выполнение преобразования.

Хотя решение системы является наиболее критичным шагом, конструирование СЛАУ и выполнение преобразования также могут быть дороги с вычислительной точки зрения.

4.1 Сортировка входных данных

Для описания желаемой деформации пользователь определяет набор контрольных векторов. Этот набор может быть представлен в виде списка точек соответствующих начальному положению поверхности и списка точек с конечными положениями поверхности. Начальные точки могут не лежать на поверхности, т.к. реально задается преобразование пространства.

Начальные положения точек используются для конструирования левой части СЛАУ; конечные положения — для конструирования правой части.

Как было сказано выше, входные точки могут быть отсортированы так, что матрица СЛАУ будет полосно-диагональной. Для этого будет нужно для каждой начальной точки найти все другие начальные точки, удаленные на расстояние меньшее радиуса поддержки. Выполнение этой операции потребуется также на этапе выполнения преобразования. Для ускорения поиска мы строили восьмеричное дерево (см. [35]). Алгоритм построения восьмеричного дерева по заданным начальным точкам можно найти в работе [2]. Это позволило существенно ускорить поиск соседей точки.

Помимо этого на этапе сортировки входных данных может быть выполнено их нормирование и т.п. действия.

4.2 Конструирование СЛАУ

Для получения матрицы A в полосно-диагональном виде используется специальный алгоритм перестановки контрольных точек. Максимальная ширина полосы равна максимальному количеству соседей контрольной точки внутри сферы с радиусом равным радиусу поддержки. Псевдокод алгоритма перестановки контрольных точек передставлен на рисунке 3. Типичный пример матрицы представлен на рисунке 4.2.

```
1: // inList — список контрольных точек
2: // outList — отсортированный список
3: i := 0
4: while inList.size() ⩾ 0 do
5:    outList.add(inList[0])
6:    inList.removeFirst()
7:    while i < outputList.size() do
8:       // Находим индексы элеметнов inList,
9:       // соседних с i-ым элементом outList
10:      ids := getNeighborIds(outList[i], inList);
11:      for all 0 ⩽ j < ids.size() do
12:         outList.add(inList[ids[j]])
13:         inList.removeById(ids[j])
14:      end for
15:   end while
16:   i := i + 1
17: end while
```

Рис. 3: Перестановка контрольных точек.

Получение матрицы A в полосно-диагональном виде имеет два преимущества. Первое состоит в возможности хранения матрицы в компактном виде. Второе состоит в возможности применения быстрого алгоритма решения.

В данной работе для хранения полосно-диагональной матрицы была использована так называемая профильная форма, или несколько модифицированная схема конверта (см. работу [36]).

При хранении можно использовать массив для диагональных элементов; недиагональные элементы и соответствующие индексы первых ненулевых элементов строк матрицы помещаются в два дополнительных массива. Размер полосы зависит от выбранного радиуса поддержки r. Для примера приведенного на рисунке 1 хранение всех элементов матрицы потребует 4096 байт, в то время как при радиусе поддержки $r = 0.1$ нужно только 276 байт; при радиусе поддержки $r = 0.2$ (результат представлен на рисунке) — 332 байта, а при $r = 0.5$ — 1148 байт. Для примера показанного на рисунке 7 требуется хранить только 6320 байт, а не 33856 байт как в случае хранения всех элементов матрицы.

Рис.4: Пример матрицы

Этот результат становится более важным в случае больших моделей и более сложных деформаций. Хочется также обратить внимание на то, что радиус поддержки влияет не только на локальность деформаций, но и на скорость их выполнения и количество требуемой памяти.

4.3 Решение СЛАУ

При решении СЛАУ использовались явные методы решения для матриц в профильной форме. Это связано с желанием получить решение за известное число шагов и хранить матрицу в компактном виде. При отсутствии ограничений на занимаемую память может быть использован любой другой метод решения, например, метод сопряжённых градиентов.

Преимущества LU-декомпозиции обсуждались во многих работах, было создано большое количество программных реализаций (см. работу [37]). Для симметричной положительно определенной матрицы время работы может быть уменьшено вдвое (по сравнению с LU-декомпозицией) при использовании факторизации по Холецкому. Мы использовали именно этот метод.

После решения СЛАУ мы получаем всю необходимую для выполнения преобразования информацию.

4.4 Выполнение преобразования

При выполнении преобразования для каждой вершины определяется её новое положение. На рисунке 4 представлен алгоритм вычисления нового положения вершин. Полное смещение вычисляется один раз и запоминается. Для определения промежуточных положений используется фазовый коэффициент, значения которого могут находиться в диапазоне $[0, 1]$.

5 Программное обеспечение

Описанный выше алгоритм был разработан в рамках системы ПО, состоящей из набора библиотек, написанных на C++, обеспечивающих поддержку 3D моделирования и анимации; система основана на библиотеке с открытым исходным кодом *«The Visualization Toolkit»* (VTK) [38]. Алгоритм работает с выбранной пользователем областью. Возможность увеличения или уменьшения (путем изменения радиуса поддержки) трансформируемой области позволяет упростить довольно утомительный процесс определения контрольных векторов. Созданный в рамках данной работы C++ класс может быть использован в процессе конвейерной обработки VTK, т.е. может быть использован совместно с другими, имеющимися в VTK фильтрами преобразования. Это делает систему гибкой и удобной в использовании.

Полная сцена представляет собой набор анимационных объектов. Дополнительно организована поддержка движения объекта, позволяющая задавать перемещение локальной системы координат объекта вдоль некоторой траектории и изменение ориентации локальной системы координат. Поддерживается возможность задания цветов, текстур и деформаций для различных частей анимируемого объекта.

ПО включает в себя два приложения:

```
1: // inList — вершины до преобразования
2: // outList — вершины после преобразования
3: // alpha — коэффициенты интерполяции
4: i := 0
5: while i < inList.size() do
6:    // Начальное смещение
7:    dx := 0, y := 0, dz := 0
8:    // Для i-ой вершины находим соседние контрольные
9:    // точки и записываем их индексы в ids
10:   ids := getNeighborIds(inList[i])
11:   for all 0 ⩽ j < ids.size() do
12:      // Корректируем смещение
13:      φ := Phi(inList[i], ids[j])
14:      dx := dx + φ·alpha[ids[j]].x
15:      dy := dy + φ·alpha[ids[j]].y
16:      dz := dz + φ·alpha[ids[j]].z
17:   end for
18:   outList[i].x := inList[i].x + dx
19:   outList[i].y := inList[i].y + dy
20:   outList[i].z := inList[i].z + dz
21:   i := i + 1
22: end while
```

Рис. 4: Выполнение преобразования.

- редактор деформаций *«Picker»*, позволяющий пользователю интерактивно задавать контрольные вектора и выполнять соответствующие преобразования. Интерфейс редактора представлен на рисунке 5.
- компоновщик объектов анимации *«Animation composer»*, предоставляющий возможность интерактивно задавать траекторию движения анимационного объекта, а также ориентацию и временные метки, т.е. расписание движения объекта вдоль выбранной траектории. Использованы сплайны Кочанека-Бартельса (Kochanek-Bartels) для задания траектории и кватернионы [39] для интерполяции ориентации. Интерфейс редактора представлен на рисунке 6.

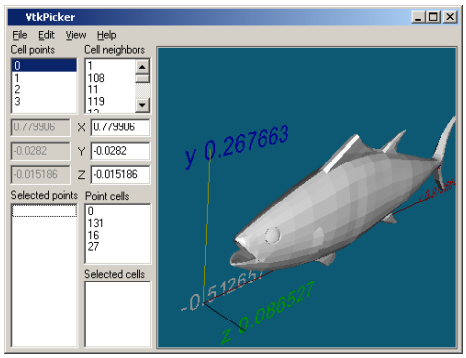


Рис. 5: «Picker».

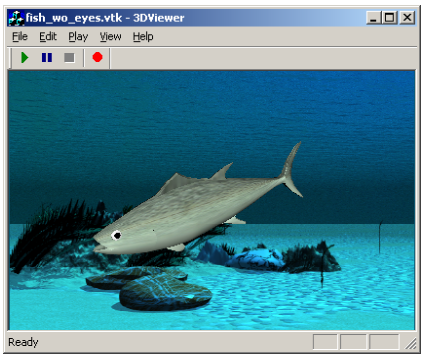


Рис. 6: «Animation Composer».

Кроме того, приложение *«Animation composer»* поддерживает режим *«Animation»* в котором программа отображает анимируемые объекты в соответствии с заданными траекториями,

поворотами и деформациями. Пользователь имеет возможность сохранить результат анимации в виде ролика. Приложения разрабатывались для использования на персональных компьютерах работающих под управлением ОС Windows и были успешно протестированы.

6 Результаты

При тестировании алгоритма использовалась состоящая из многоугольников модель рыбы (см. рис. 7). Модель содержала 974 вершин и 1119 многоугольников. Для задания деформации рта, хвоста и плавников использовалось 92 контрольных вектора. Средняя скорость вычислений и визуализации деформаций составила 632 кадра/сек.

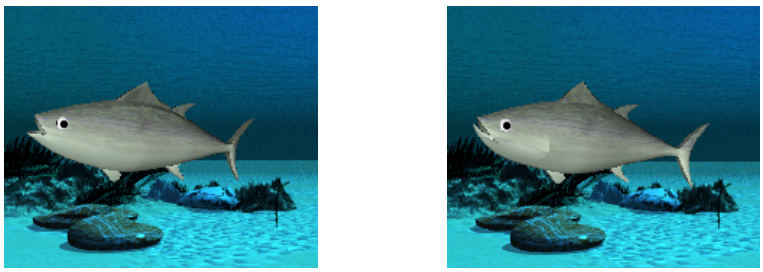


Рис. 7: «Плавающая рыба» («Fish»).

Другой пример анимации деформируемого объекта в реальном времени показан на рисунке 8. Сфера движется внутри куба, стенки которого деформируются при контакте со сферой. Использовалась C^2-непрерывная радиальная функция $\varphi_{1,3} = (1-r)_+^4(4r+1)$ (см. работу [1]). Подобные деформации в реальном времени могут быть использованы для моделирования эластичной среды в компьютерных играх. Несмотря на то, что в этом примере каждый раз проводился полный цикл вычислений, включая решение СЛАУ, для каждого кадра, была получена скорость 72 кадра/сек при разрешении 1024×768. Количество контрольных векторов и радиус поддержки определялись динамически в процессе симуляции.

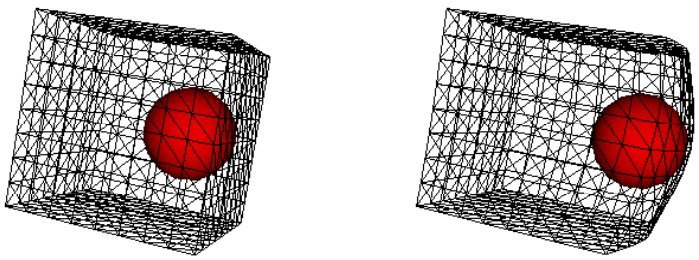


Рис. 8: «Прыгающий мяч» («Ball»).

Модель	Число вершин	Число граней	Число векторов	Радиус	Скорость (кадр/сек.)
«Face» (рис. 1)	6158	7024	32	0.2	107
«Fish» (рис. 7)	975	1119	92	0.3	632
«Ball» (рис. 8)	386	768	дин.	дин.	72

Таблица 1: Результаты тестов: Athlon 1Ghz, 650MB RAM.

7 Заключение

В данной работе был представлен подход к решению задачи анимации деформируемых объектов, основанный на использовании базиса радиальных функций с компактным носителем. Экспериментальные результаты показывают, что данных подход позволяет получать гладкие локальные и правдоподобные деформации, проводя все вычисления в режиме реального времени.

Выбор множества контрольных векторов для задания деформации является достаточно трудоемкой задачей. Поэтому возможность изменения области поверженной деформации путём изменения радиуса поддержки существенно облегчает работу.

Средняя скорость анимации, как следует из таблицы 1, достаточно высока. Таким образом, представленный алгоритм может быть использован в компьютерных играх.

Список литературы

[1] H. Wendland. Piecewise polynomial, positive defined and compactly supported radial functions of minimal degree. *AICM*, 4:389–396, 1995.

[2] N. Kojekine, V. Savchenko, D. Berzin, and I. Hagiwara. Software tools for compactly supported radial basis functions. In *Computer Graphics and Imaging, Proc. IASTED CGIM'2000*, pages 234–239, 2001.

[3] V. Savchenko and A. Pasko. Transformation of functionally defined shapes by extended space mappings. *The Visual Computer*, 14:257–270, 1998.

[4] T.W. Sederberg and S.R. Parry. Free-form deformation of solid geometric models. *Computer Graphics*, 20(4):151–160, 1986.

[5] S. Coquillart. Extended free-form deformation: a sculpting tool for 3d geometric modeling. *Computer Graphics*, 24(4):187–196, 1990.

[6] S. Coquillart and P. Jancen. Animated free-form deformation: an interactive animation technique. *Computer Graphics*, 25(4):23–26, 1991.

[7] W.M. Hsu, G.F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. *Computer Graphics*, 26(2):177–184, 1992.

[8] P. Borrel and D. Bechmann. Deformation of n-dimensional objects. *International Journal of Computational Geometry and Applications*, 1(4):427–453, 1991.

[9] P. Borrel and A. Rappoport. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics*, 13(2):137–155, 1994.

[10] D. Bechmann. Space deformation models survey. *Computers & Graphics*, 18(4):571–586, 1994.

[11] J.H. Ahlberg, E.N. Nilson, and J.L. Walsh. The theory of splines and their applications. *Academic Press, New York*, 1967.

[12] J. Dushon. *Constructive Theory of Functions of Several Variables*, chapter Splines Minimizing Rotation Invariants Semi-Norms in Sobolev Spaces, pages 85–100. Springer-Verlag, 1976.

[13] В.А. Василенко. *Сплайн-функции: Теория, Алгоритмы, Программы.* Новосибирск, Наука, 1983.

[14] R.M. Bolle and B.C. Vemuri. On three-dimensional surface reconstruction methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(1):1–13, 1991.

[15] G. Greiner. *Wavelets, Images and Surface Fitting*, chapter Surface Construction Based on Variational Principles, pages 277–286. AL Peters Ltd., 1994.

[16] F.L. Bookstein. Principal warps: Thin plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Mashine Intelligence*, 11(6):567–585, 1989.

[17] F.L. Bookstein. *Morphometric Tools for Landmark Data.* Cambridge University Press, 1991.

[18] F.L. Bookstein. Two shape metrics for biomedical outline data: Bending energy, procrustes distance, and the biometrical modeling of shape phenomena. In *Proc. Shape Modeling Conference (SMIA'97)*, pages 110–120, 1997.

[19] P. Litwinovicz and L. Williams. Animating images with drawing. In *Computer Graphics (Proceedings of SIGGRAPH'94)*, pages 409–412, 1994.

[20] J.C. Carr, W.R. Fright, and R.K. Beatson. Surface interpolation with radial basis functions for medical imaging. *IEEE Transaction on Medical Imaging*, 16(1):96–107, 1997.

[21] R.K. Beatson and W.A. Light. Fast evaluation of radial basis functions: Methods for 2-d polyharmonic splines. Technical Report 119, Mathematics Department Univ. of Canterbury, New Zealand, 1994.

[22] W. Light. *Wavelets, Images and Surface Fitting*, chapter Using Radial Functions on Compact Domains, pages 351–370. AL Peters Ltd., 1994.

[23] J.C. Carr, T.J. Mitchell, R.K. Beatson, J.B. Cherrie, W.R. Fright, B.C. McCallumm, and T.R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Computer Graphics, Proceedings SIGGRAPH'2001*, pages 67–76, 2001.

[24] L. Greengard and V. Rokhlin. A fast algorithm for particle simulation. *Journal of Computational Physics*, 73:325–348, 1997.

[25] V. Savchenko and L.Schmitt. Reconstructing occlusal surfaces of teeth using a genetic algorithm with simulated annealing type selection. In *Proceeding of 6th ACM Symposium on Solid Modeling and Application*, pages 39–46, 2001.

[26] V.V. Savchenko, A.A. Pasko, T.L. Kunii, and A.V. Savchenko. *Multimedia Modeling*, chapter Feature based sculpting of functionally defined 3D geometric objects, pages 341–348. Towards Information Superhighway, 1995.

[27] B. Morse, T.S. Yoo, P. Rheingans, D.T. Chen, and K.R. Subramanian. Interpolating implicit surfaces from scattered surface data using complactly supported radial basis functions. In *Proceedings of the Shape Modeling Conference*, pages 89–98. Genova, Italy, 2001.

[28] S. Lee, G. Wolberg, and S.Y. Shin. Scattered data interpolation with multilevel b-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):228–244, 1997.

[29] D. Thalmann, J. Shen, and E. Chauvineau. Fast realistic human body deformations for animation and vr applications. In *Computer Graphics International*, pages 166–174, 1999.

[30] P. Fua, R. Plankers, and D. Thalmann. From synthesis to analysis: Fitting human animation models to image data. *Computer Graphics International*, pages 4–11, 1999.

[31] Y. Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *SIGGRAPH'95 Proceedings*, pages 191–198, 1995.

[32] J. Bloomenthal and C. Lim. Skeletal methods of shape manipulation. In *Proceedings of International Conference on Shape Modeling and Applications*, pages 44–47, 1997.

[33] A. Verroust and F. Lazarus. Extracting skeletal curves from 3d scattered data. *Proceedings of International Conference on Shape Modeling and Applications*, pages 194–202, 1999.

[34] H. Wendland. On the smoothness of positive definite and radial functions. *Journal of Computational and Applied Mathematics*, 101:177–188, 1999.

[35] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1986.

[36] A. Jennings. A compact storage scheme for the solution of symmetric linear simultaneous equations. *Computational Journal*, 9:281–285, 1966.

[37] W.H. Press, S.A. Teukolsky, T. Vetterling, and B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1997.

[38] The visualization toolkit textbook and open source c++ library, with tcl, python, and java bindings. http://www.kitware.com/vtk.html, 2001.

[39] K. Shoernake. Animating rotation with quaternion calculus. In *ACM SIGGRAPH 1987*, 1987.