О новом подходе к расчёту и визуализации информационных множеств в задачах динамического поиска

Павел Воронин
Факультет Вычислительной Математики и Кибернетики,
Московский Государственный Университет, Москва, Россия
pavel.voronin@gmail.com

Аннотация

Предложен новый общий алгоритм расчёта и визуализации информационных множеств в задачах динамического поиска, основанный на использовании полей расстояний. Подробно описана его реализация для двух- и трёхмерного случаев, где удалось добиться интерактивной скорости работы.

Ключевые слова: *динамический поиск, поля расстояний.*

1. ВВЕДЕНИЕ

Существует масса постановок задач поиска, мотивированных различными практическими задачами. Все они, так или иначе, формализуют задачу получения неизвестной информации об искомом объекте. Например, в монографии [1] задача поиска – это дифференциальная игра с неполной информацией, а в работе [2] авторы исследуют игры поиска на графах.

Отдельный большой класс представляют собой задачи поиска объектов на заданном множестве. Воспользуемся классификацией из статьи [3] и напомним основные понятия: поисковая область, объекты поиска, ищущие объекты и условия обнаружения.

Областью поиска чаще всего является подмножество некоторого пространства, не обязательно связное и ограниченное, и вообще говоря, изменяющееся во времени.

Объекты поиска, статичные или подвижные, лежат в области поиска. Для простоты их можно считать точечными. В различных задачах объекты поиска могут способствовать нахождению, не препятствовать ему или же деятельно сопротивляться.

Ищущие объекты – это подвижные точки, в задачу которых входит отыскание такого обхода области поиска, при котором будут обнаружены все объекты поиска или же получено доказательство, что ни одного искомого объекта в заданной области не было.

Из возможных *условий обнаружения* наиболее распространены: а) приближение одного из ищущих к искомому на заданное расстояние; б) попадание искомого объекта на линию прямой видимости одного из ищущих (подобные задачи рассматриваются в областях сложной топологии, например, на плоскости с вырезами – см. [3]).

В зависимости от задачи, поведение искомых объектов может быть разным. Объекты могут быть неподвижными или перемещающимися, причём движение может быть детерминированным (принадлежать какому-то классу, см. [4]), стохастическим (см. [5]) или уклоняющимся (когда объект активно препятствует нахождению). Могут отличаться и информационные возможности объектов (их осведомлённость о выбранной стратегии и параметрах движения противодействующей стороны). Мы будем рассматривать случай, когда искомые объекты активно уклоняются от ищущих, местоположение, скорость и стратегия движения которых в каждый момент времени им известны, – в то время как ищущие знают только максимальную скорость искомых; ищущим необходимо выбрать такую стратегию поведения, которая бы гарантировала нахождение искомых объектов в независимости от их действий. Эту же ситуацию с точки зрения ищущих можно интерпретировать иначе: ничего, кроме максимальной скорости противников, ни одной из сторон неизвестно – и, в дополнение к этому, ищущим неизвестно количество искомых объектов. Выбираемая стратегия должна гарантировать нахождение сколь угодно большого числа объектов.

Наиболее хорошо изучены задачи поиска уклоняющихся объектов в случае областей, представляющих собой подмножества евклидового пространства – двухмерного (например, в круге [3], монотонном многоугольнике [6], кольце [6] и даже на всей плоскости [3]) или трёхмерного (например, на поверхности цилиндра и тора [7] или в шаре [8]). Оказалось, что уже на таких простых областях аналитические методы исследования малопрактичны и громоздки. Во многом более эффективным видится разработанный в начале 1990-х Е.В. Шикиным, С.М. Губайдуллиным и А.Г. Чхартишвили геометрический подход. Общая идея методов этого типа заключается в построении вспомогательных *информационных множеств*, соответствующих сведениям, накапливаемых в процессе поиска ищущими и уклоняющимися объектами друг о друге. Обширный обзор работ по данной теме дан в работе [3].

Наиболее значимыми типами информационных множеств являются:

А) *Остаточные области* (рис. 1). Это такие подмножества области поиска, в которых уклоняющегося объекта быть не может (иначе он был бы обнаружен ищущими ранее). Обычно они образуются из-за превосходства ищущих в скорости или эффектов на границе поисковой области. Чем большая часть области поиска входит в остаточные, тем большей информацией о местоположении уклоняющего объекта обладает ищущий. А значит, его целью является увеличение остаточной области вплоть до момента, когда она

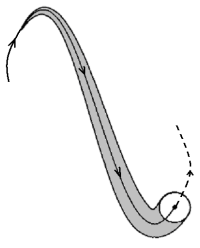

Рис. 1. Остаточная область

покроет всю область поиска.

Б) *Упреждающие области* (рис. 2), при нахождении в которых уклоняющийся объект не сможет избежать обнаружения в ближайшем будущем.

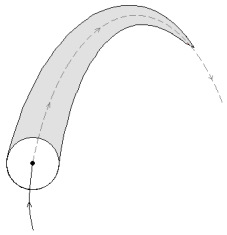

Рис. 2. Упреждающая область

Обычно с упреждающими областями работают при решении задач на уклонение: наиболее выгодная стратегия для объекта поиска состоит в том, чтобы активно избегать встречи с совокупной упреждающей областью ищущих.

Информационные множества показали себя как крайне эффективный инструмент при решении задач динамического поиска как в двух-, так и трёхмерном случае (например, [6], [7], [8]) – и даже на плоскости Лобачевского [13].

Вместе с тем, попытки применить геометрический подход для случая нескольких ищущих объектов или невыпуклых поисковых областей сталкиваются с серьёзными трудностями. Дело в том, что структура информационных множеств в этих случаях значительно усложняется (см. рис. 3, работы [3], [6], [22]), и аналитические методы их построения становятся неэффективными.

Логичной видится высказанная в диссертации [6] идея перейти от геометрических методов к графическим: рассчитывать информационные множества на компьютере при помощи специальных численных методов, с последующей визуализацией. Действительно, наглядность и интерактивность процесса помогают быстрее и легче находить искомые траектории, а при достаточной скорости, точности и устойчивости процесс перебора траекторий из заданного класса и вовсе можно было бы автоматизировать.

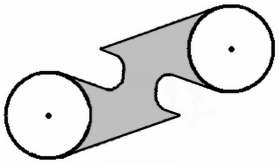

Рис. 3. Совместная упреждающая область

2. ОБЩАЯ СХЕМА ЧИСЛЕННЫХ МЕТОДОВ

Рассмотрим общую схему построения численных методов для расчёта остаточной области [6]. Для простоты начнём с двумерного случая, неограниченной области поиска, одного ищущего и одного уклоняющегося объекта, движущихся с постоянными скалярными скоростями.

Введём следующие обозначения. Пусть скорость уклоняющегося объекта равна U, а скорость ищущего объекта – V (будем считать, что $V > U$, то есть ищущий имеет преимущество по скорости). Пусть область обнаружения ищущего – круг радиуса R, а траектория его перемещения $P(t)$. Нам нужно рассчитать изменение переменной остаточной области $A(t)$ на отрезке времени от $t = 0$ до $t = T$.

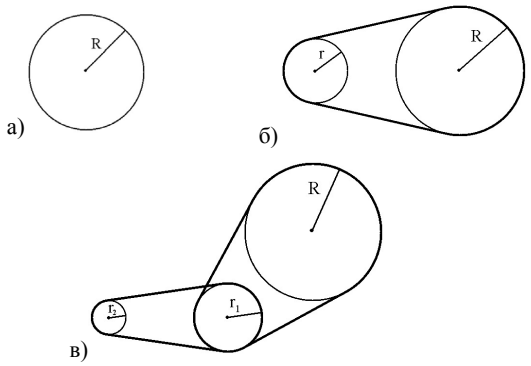

Рис. 4. Остаточные области A_0, A_1, A_2.

Для начала дискретизируем задачу. Приблизим заданную траекторию ломаной, проходящей через точки $P_0, P_1, \ldots, P_n$ – положения ищущего в моменты времени $t_0=0, t_1, \ldots, t_n=T$ ($t_i=i\cdot\Delta t$, $\Delta t=T/n$). Будем последовательно искать остаточные области $A_0, A_1, \ldots, A_n=A$, $A_i=A(t_i)$. На рис. 4 указаны первые три остаточные области: A_0, A_1 и A_2.

Область A_0, начальное состояние нашей системы, – это круг радиуса R с центром в точке P_0, $B(P_0; R)$ (рис. 4(а)): в начальный момент времени t_0 об уклоняющемся объекте можно сказать только то, что он находится не ближе, чем на расстоянии радиуса обнаружения от ищущего объекта.

Область A_1 устроена немного сложнее. За время Δt ищущий объект переместился в точку P_1. Значит, A_1 состоит, как минимум, из $B(P_1; R)$. Однако, поскольку скорость уклоняющегося объекта меньше скорости ищущего, то A_1 принадлежат ещё и круги, в которые уклоняющийся объект никак не мог успеть добраться:

$$B(P(t_0 + \tau);\ max(R - \tau \cdot U, 0)),\ 0 \le \tau \le \Delta t.$$

Как несложно проверить [14], объединение всех таких кругов даёт фигуру, изображённую на рис. 4(б) (подобные фигуры будем обозначать как $A(C_1, C_2, R_1, R_2)$, где C_i – центры, а R_i – радиусы образующих кругов; R_1 может равняться нулю). Итак, $A_1 = A(P_0, P_1, r, R)$, $r = R - \Delta U$, $\Delta U = \Delta t \cdot U$.

За время Δt круг $A_0=B(P_0; R)$ сузился до круга $B(P_0; r)$. Легко видеть, что для фигур типа $A(C_1, C_2, R_1, R_2)$ сужение также даёт фигуру того же типа. Имеем (рис. 4(в)):

$$A_2 = A(P_0, P_1, r - \Delta U, r) \bigcup A(P_1, P_2, r, R).$$

Теперь легко понять общую формулу

$$\begin{cases} A_0 = B(P_0; R), \\ A_{k+1} = Offset(A_k, -\Delta U) \bigcup A(P_k, P_{k+1}, r, R). \end{cases} \quad (1)$$

Здесь $Offset(A, -r)$ – сужение (offsetting) множества A на полосу шириной r. (Эту операцию иногда ещё называют «r-расширением» с отрицательным r.)

Полученная формула легко обобщается на случай нескольких ищущих объектов. Пусть i-му объекту соответствует траектория $P^{(i)}$. Тогда

$$\begin{cases} A_0 = \bigcup_{i=1}^{N} B(P_0^{(i)}; R), \\ A_{k+1} = Offset(A_k, -\Delta U) \bigcup_{i=1}^{N} A(P_k^{(i)}, P_{k+1}^{(i)}, r, R). \end{cases} \quad (2)$$

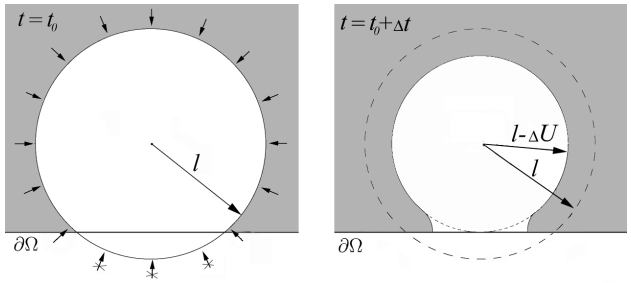

Рис. 5. Неравномерное сокращение остаточной области вблизи границы.

Предположим теперь, что область поиска Ω ограничена. В этом случае нам необходимо учесть, что со стороны границы области поиска $\partial\Omega$ уклоняющийся объект двигаться не может, а значит, сокращение остаточной области здесь будет идти медленнее (рис. 5).

Тем самым, формулы (2) принимают вид:

$$\begin{cases} A_0 = \Omega \cap \left(\bigcup_{i=1}^{N} B\left(P_0^{(i)}; R\right) \right), \\ A_{k+1} = \left(\Omega \setminus Offset\left(\Omega \setminus A_k, +\Delta U\right)\right) \cup \left(\Omega \cap \left(\bigcup_{i=1}^{N} A\left(P_k^{(i)}, P_{k+1}^{(i)}, r, R\right) \right) \right). \end{cases} \quad (3)$$

Здесь первое слагаемое – сужение, учитывающее влияние границы, а второе – та часть текущей области обзора ищущих, которая не выходит за пределы области поиска.

Нетрудно заметить, что формулы (1) – (3) не содержат операций, которые были бы характерны исключительно для плоскости. И действительно, все они остаются верными как для трёхмерных поисковых задач, так и для задач поиска на поверхностях (формально трёхмерных, но фактически двухмерных со своей внутренней геометрией). При этом под *B(P;R)* понимаются, соответственно, шар и геодезический круг, под $A(C_1, C_2, R_1, R_2)$ – трёхмерный и геодезический аналоги фигуры на рис. 4, а под *Offset(A, ±r)* – трёхмерное и геодезическое *r*-расширения.

Итак, мы выяснили, что для широкого класса поисковых задач на плоскости и в пространстве возможно приближённое рекуррентное вычисление информационных множеств по формулам (1) – (3). Теперь нам необходимо выбрать в каждом случае такое программное представление множеств, которое позволяло бы эффективно проводить четыре базовые операции: объединение, пересечение, разность и *r*-расширение.

3. B-АЛГОРИТМ

В работе [6] автор концентрирует своё внимание на двухмерном случае. Предложенный в ней алгоритм (далее будем называть его «*B*-алгоритм») основан на представлении поискового множества в виде набора отрезков, аппроксимирующих его границу (так называемое граничное представление, *B-Rep*). Расчёт *B-Rep* для A_{k+1} по *B-Rep* для A_k ведётся по формуле (3).

Поскольку сами по себе базовые операции выводят за пределы класса *B-Rep*-представимых множеств (см. рис. 6), то вместо них необходимо использовать их регуляризованные приближения [15].

Для борьбы с ошибками округления используется особое представление вершин и отрезков [16], позволяющее вести все вычисления в целочисленной арифметике.

В работе [6] приводится также ряд алгоритмических и программных оптимизаций, которые в совокупности позволяют достичь интерактивных скоростей расчёта и визуализации при высокой точности результатов.

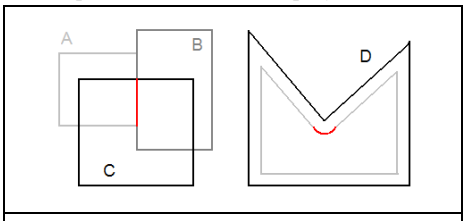

Рис. 6. $C \setminus \left(A \cap B\right)$ содержит разрез, *Offset(D,-s)* – дугу окружности

3.1 Анализ *B*-алгоритма

B-алгоритм обладает рядом достоинств (свобода от ошибок округления, интерактивная скорость работы), однако его применение натыкается на существенные препятствия:

1) Он сложен для понимания и реализации.
2) Ещё труднее будет реализовывать его обобщение на трёхмерный случай. Более того, учитывая сложность базовых операций, которые при этом обязательно понадобятся (в частности, вычисление пересечений треугольников), реализация интерактивной системы видится крайне труднодостижимой.
3) Неясно, можно ли обобщить его на случай поиска на поверхностях.

По-видимому, причина этих проблем – в выборе *B-Rep* в качестве представления информационных множеств. Основным плюсом *B-Rep* является скорость его рендеринга на любой современной видеокарте. Однако в случае рассматриваемой задачи граница множества обычно состоит из нескольких сотен, максимум тысяч примитивов. Основное время уходит не на вывод границы на экран, а на её расчёт, то есть на вычисление результатов базовых операций. Поэтому основным критерием выбора представления для поисковых множеств должны быть простота и эффективность именно их реализации.

Руководствуясь этими соображениями, мы разработали новый алгоритм, основывающийся на полях расстояний.

4. ПОЛЯ РАССТОЯНИЙ

В качестве представления для информационных множеств мы предлагаем использовать неявное их задание, известное как функциональное представление (Function Representation, *F-Rep* [17]) или поле расстояний (Distance Field, *DF* [9]). Хотя операции их инициализации и рендеринга по времени сравнительно затратны, зато многие операции редактирования (включая теоретико-множественные операции и *r*-расширение) просчитываются чрезвычайно быстро [9], [17].

Итак, вместо заданного замкнутого геометрического объекта будем рассматривать порождаемое им в пространстве скалярное поле, значение которого в произвольной точке

равно расстоянию от этой точки до границы объекта, взятому со знаком плюс, если точка лежит вне объекта, и со знаком минус в противном случае. (Например, для круга радиусом единица с центром в начале координат *DF* будет задаваться формулой $d(x,y)=1-\sqrt{x^2+y^2}$.)

При этом граница породившего *DF* тела *C* будет в точности совпадать с нулевой изолинией *DF*:

$$\partial C = ISO_0 DF = \{(x,y): d(x,y)=0\}.$$

Более того, тем же способом для *C* можно вычислить *r*-расширение:

$$Offset(C,r)=ISO_r DF=\{(x,y): d(x,y)=r\}=ISO_0(DF-r).$$

Пусть теперь у нас есть поля расстояний, порождённые двумя телами *A* и *B*. Тогда верны следующие формулы:

$$A\cup B = ISO_0\left(\min\left(DF_A, DF_B\right)\right),$$

$$A\cap B = ISO_0\left(\max\left(DF_A, DF_B\right)\right),$$

$$\overline{A} = ISO_0\left(-DF_A\right),$$

$$A\setminus B = A\cap\overline{B} = ISO_0\left(\max\left(DF_A, -DF_B\right)\right).$$

Для дополнительной устойчивости вместо минимума и максимума можно использовать более гладкие *R*-функции высокого порядка [17].

Заметим также, что хотя

$$DF_{Offset(C,r)}=DF_C-r,$$

$$DF_{\overline{A}}=-DF_A,$$

но

$$DF_{A\cup B}\neq\min\left(DF_A, DF_B\right),$$

$$DF_{A\cap B}\neq\max\left(DF_A, DF_B\right),$$

$$DF_{A\setminus B}\neq\max\left(DF_A, -DF_B\right).$$

4.1 *DF*-сетки

Ясно, что, в отличие от шара, для достаточно сложных тел *DF* явного представления не имеет (рис. 7), поэтому возникает проблема выбора подходящего описания. Наиболее распространённым подходом является хранение замеров значения поля в узлах некоторой сетки.

Инициализация значений поля в узлах сетки – это слабое место полей расстояний и, вообще говоря, может потребовать значительных вычислительных затрат. Например, если тело задано как список из *N* отрезков своей границы, а в сетке *M* узлов, то при полном переборе нам понадобится *M*·*N* раз вычислять расстояние от точки до отрезка.

Если же у нас уже есть сетка с замерами, то мы можем извлечь из неё массу полезной информации:

1) Значение поля в произвольной точке восстанавливается интерполяцией значений поля в вершинах клетки, которой эта точка принадлежит.

2) Градиент поля в любой точке считается либо численно, либо аналитически как градиент интерполянты.

3) Форму породившего *DF* тела можно восстановить по замерам на сетке, вычислив в каждой клетке некоторое приближение нулевой изолинии поля.

Рис. 7. Пример двумерного поля расстояний (яркость пропорциональна модулю значения поля).

4.1.1 2D-сетки

Рассмотрим простой пример: квадратную сетку, параллельную осям координат. По аналогии с трёхмерным методом Marching Cubes [11], будем строить в каждой клетке линейное приближение изолинии (или, при желании, ограниченного ею тела). С точностью до симметрии, клетка может быть лишь в одной из шести конфигураций (рис. 8а).

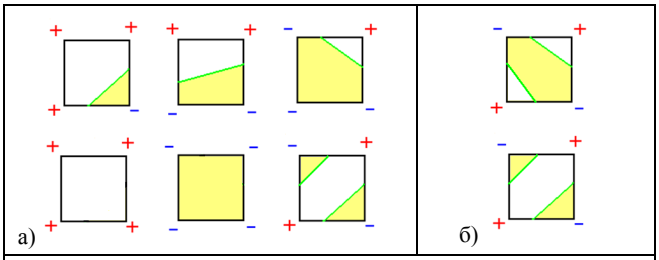

Рис. 8. Marching Quads, 6 возможных конфигураций + два варианта последней конфигурации

Последняя из этих конфигураций представляет особую проблему: выбор связности для неё произволен (рис. 8б), что зачастую приводит к несогласованности топологии искомого тела и его реконструкции. Как и в трёхмерном случае [18], с этой проблемой можно бороться по-разному: повышать степень интерполянты, учитывать для проблемных клеток конфигурацию соседей и, наконец, самое простое – перейти от квадратной сетки к треугольной. И действительно: треугольник может находиться лишь в четырёх возможных конфигурациях, и все они однозначны (рис. 9).

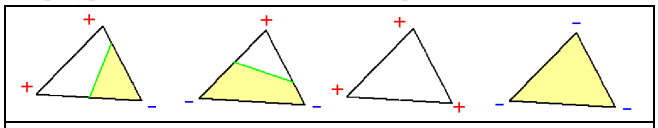

Рис. 9. Marching Triangles

Из всевозможных треугольных сеток, покрывающих плоскость, оптимальной для нас является, очевидно, сетка из одинаковых правильных треугольников, гарантирующая равномерное качество приближения по всей области.

4.1.2 3D-сетки

Кубические сетки классического метода Marching Cubes подвержены топологической несогласованности в ещё большей степени, чем плоские квадратные: помимо разрывов, могут также образовываться дырки в поверхности – причём вероятность такого сценария при восстановлении тел сложной структуры довольно высока. Поэтому здесь нам также желательно перейти к более простым, тетраэдральным сеткам [23], [24], [25]. У тетраэдра всего две конфигурации, требующие триангуляции, и обе лишены неоднозначности (рис. 10).

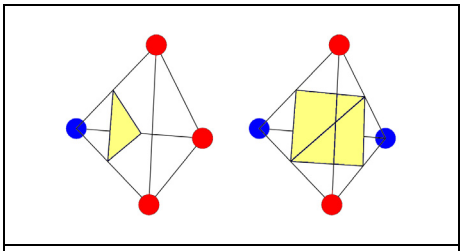

Рис. 10. Marching tetrahedra

Проблема в том, что, в отличие от правильного треугольника, правильный тетраэдр не замощает пространства. С другой стороны, как показала практика, все сетки, составленные из тетраэдров с сильно отличающимися длинами рёбер, дают неудовлетворительное качество восстановления и рендеринга – в частности, очень плохо показали себя все сетки, основанные на разбиении куба (Канейро, Скалы, MT6 [19]).

Мы остановили свой выбор на сетке, составленной из копий единственного известного на данный момент тетраэдра, заполняющего трёхмерное пространство. У него одна пара скрещенных рёбер длины $2a$ и три пары длины $a\sqrt{3}$. Искомую сетку можно получить подразбиением сетки из ромбических додекаэдров: каждый из них разбивается на 24 базовых тетраэдра [20].

5. *DF*-АЛГОРИТМ

Посмотрим теперь подробнее, как именно ведутся вычисления при помощи полей расстояний.

Для простоты возьмём формулу (1) и перепишем её, выразив все действия через операции над полями расстояний:

$$A_{k+1} = ISO_0(DF_{A_k} + \Delta U) \bigcup A(P_k, P_{k+1}, r, R),$$
$$A_{k+1} = ISO_0\left(\min\left(DF_{A_k} + \Delta U, DF_{A(P_k,P_{k+1},r,R)}\right)\right).$$

Получаем такую последовательность действий:

$$1) DF_{old} = DF_{A_k} + \Delta U,$$
$$2) DF_{upd} = DF_{A(P_k,P_{k+1},r,R)},$$
$$3) DF_{new} = \min(DF_{old}, DF_{upd}),$$
$$4) DF_{A_{k+1}} = DF_{ISO_0 DF_{new}}.$$

Отметим, что этот алгоритм, как и дальнейшие рассуждения данного параграфа, практически не зависит от размерности задачи или типа сетки.

5.1 Линейная сложность

На первый взгляд, в описанном выше алгоритме только первый и третий пункты выполняются за линейное от числа узлов в сетке время. Второй и четвёртый представляют собой построение *DF* по заданному объекту, то есть наиболее медленный этап работы с полем. Но и здесь можно обойтись вычислениями линейной сложности.

Во-первых, поле для $A(P_k, P_{k+1}, r, R)$ можно выписать в явном виде. Как видно из рис. 11, и для плоского, и для трёхмерного варианта этой фигуры всё пространство делится двумя конусами на три части. В двух из них понадобится вычислять расстояние от точки до окружности (сферы), а в третьей – от точки до равнобедренного треугольника (конуса). Это означает, что *DF* для $A(P_k, P_{k+1}, r, R)$ вычисляется за один проход по сетке.

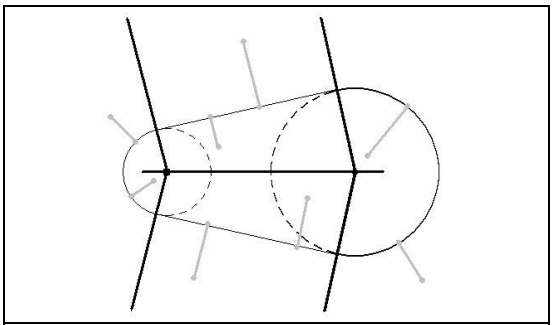

Рис. 11. Вычисление расстояния до $A(P_k, P_{k+1}, r, R)$.

Во-вторых, для эффективной реализации четвёртого шага необходимо иметь в виду два факта:

а) Для рендеринга нужно знать точное значение DF_{A_k} только в узлах клеток, непосредственно содержащих участки нулевой изоповерхности. Для остальных узлов достаточно правильного знака;

б) Для расчёта $DF_{A_{k+1}}$ нужны точные значения DF_{A_k} только в клетках, содержащих $(-\Delta U)$-изоповерхность, от остальных достаточно верного знака.

Сказанное означает, что на четвёртом шаге нам нужно вычислять значения поля только в узлах сетки, принадлежащих пограничным или внутренним клеткам, лежащим не далее, чем в ΔU от границы.

Отсюда же следует, что при расчёте *DF* в любой заданной точке мы можем искать минимум расстояния от неё не до всей границы порождающего объекта, а только до тех её участков, которые принадлежат удалённым от этой точки не более чем на ΔU клеткам.

На практике для получения адекватных результатов сетку и разбиение временного ряда берут такими, что $\Delta U \le l$, где l – характерная длина ребра сетки. Это, в свою очередь, означает, что в большинстве случаев ближайшим участком границы для любого исследуемого узла будет один из участков, содержащихся в соседних с ним клетках, а значение поля в узле определится путём нескольких проверок (рис. 12).

Количество вычислений можно сократить и дальше:

1) Для нашего алгоритма нам нужно уметь вычислять только сужение объектов, но не их расширение. Следовательно, в узлах клеток, целиком лежащих вне порождающего *DF* объекта, достаточно знать знак поля.

2) Пересчитывать значения поля в узлах клеток, содержащих границу, не имеет смысла, так как именно из этих значений приближение изоповерхности и получено.

Подводя итог, получим следующую схему четвёртого шага:

для каждого узла P

если $DF_{new}(P) < 0$

$$DF_{A_{k+1}}(P) = -\infty$$

иначе

$$DF_{A_{k+1}}(P) = +\infty$$

для каждой клетки C

если C содержит ISO_0DF_{new}

S = приближение ISO_0DF_{new} внутри C

для каждого узла I клетки C

$$DF_{A_{k+1}}(I) = DF_{new}(I)$$

для каждого соседнего с C узла N

$$DF_{A_{k+1}}(N) = \min\left(DF_{A_{k+1}}(N), dist(N,S)\right)$$

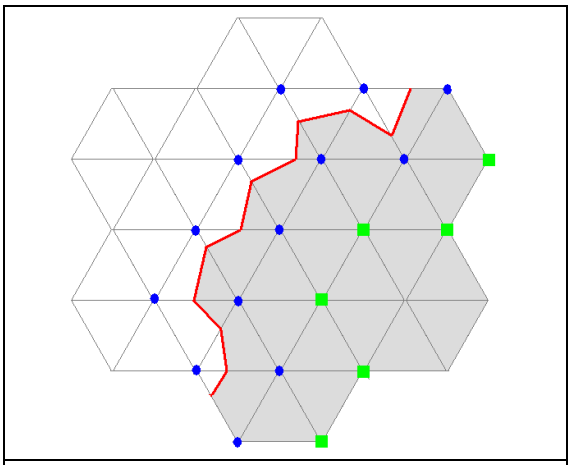

Рис. 12. Значения в узлах, помеченных кружочком, копируются из DF_{new}; в помеченных квадратиком – вычисляются как минимум расстояний до приближений ISO_0DF_{new} в соседних клетках.

Линейную сложность описанным вычислениям гарантирует предпросчитанная на этапе построения сетки информация о связности: для каждой клетки известен список соседних вершин (находящихся от неё на расстоянии $\le l$). Благодаря тому, что мы используем почти регулярные сетки, число найденных таким образом соседей оказывается небольшим и почти всюду одинаковым.

5.2 Граница области поиска

Помимо тех усложнений в вычислениях, что отражены в формуле (3), учёт границы поисковой области привносит ещё одну проблему. Посмотрим, что происходит, когда мы вычисляем пересечение двух тел, заданных своими *DF* на одной и той же сетке (рис. 13а). Из-за того, что мы храним только значения *DF* в узлах сетки, результат вычислений вблизи точек пересечения тел будет значительно искажён. Проблема гораздо серьёзнее, чем кажется на первый взгляд: в случае нашего алгоритма такое «обрубание» происходит на каждом кадре, и, накапливаясь, практически полностью нивелирует эффект взаимодействия остаточной области с границей области поиска.

Решение подсказывает особый случай: если граница хотя бы одного из объектов проходит только по рёбрам сетки, то результат пересечения будет именно таким, как нам бы хотелось (рис. 13б).

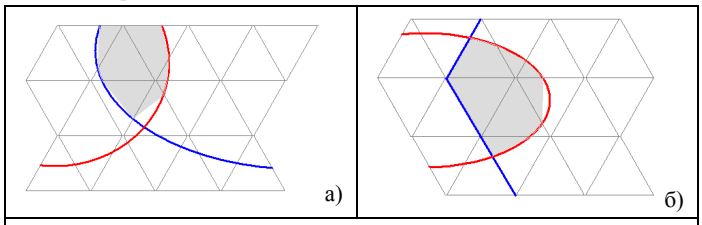

Рис. 13. Пересечение двух областей, заданных DF.

Обратим внимание на то, что в формуле (3) во всех теоретико-множественных операциях одним из операндов являет поисковая область Ω. Это означает, что для решения проблемы достаточно подразбить исходную сетку так, чтобы рёбра новой сетки достаточно хорошо приближали границу Ω. На плоскости в качестве такого подразбиения удобнее всего взять триангуляцию внутренности нулевой изоповерхности поля расстояний, заданного Ω. Более того, поскольку результатом вычислений на каждом кадре всегда будет какая-то область *внутри* Ω, то все треугольники вне Ω можно отбросить. Как видно из рис. 14, такой процесс порождает неравносторонние и даже почти вырожденные треугольники, но их число невелико, и в целом качество сетки остаётся достаточно высоким.

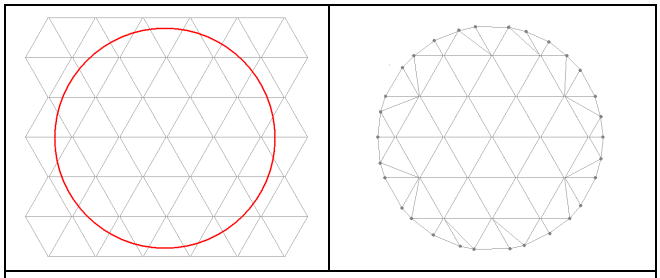

Рис. 14. Адаптация сетки к поисковой области.

6. ПРИМЕРЫ РАБОТЫ

На основе описанных выше алгоритмов была написана программа, в реальном времени рассчитывающая и визуализирующая при помощи OpenGL поисковые множества для плоской и трёхмерной поисковой задач. Область поиска, радиус обнаружения, количество объектов, их скорости и траектории задаются пользователем.

На рис. 15 показано несколько примеров двумерных остаточных областей, рассчитанных по описанному выше алгоритму. На рис. 16 изображён пример работы алгоритма в случае двух ищущих и ограниченной области поиска.

Алгоритм успешно справляется с учётом обоих этих факторов.

На рис. 17 приводится несколько примеров остаточных областей, рассчитанных в трёхмерном случае. Как видно из рисунка, качество поверхности вблизи особенностей не очень высоко. В будущем предполагается для большей устойчивости извлечения изоповерхностей использовать регуляризованные алгоритмы [18].

Программа тестировалась на компьютере следующей конфигурации: процессор Intel Pentium 4 (3 ГГц), 1.5 Гб оперативной памяти, видеокарта NVidia GeForce FX 5600. Для двумерной задачи анимация велась со скоростью от 60 до 200 кадров в секунду (в среднем, 75-100), для трёхмерной – от 20 до 40 кадров в секунду (в среднем, 25-30).

7. ЗАКЛЮЧЕНИЕ

В работе представлен новый подход к расчёту и визуализации информационных множеств в задачах динамического поиска.

На основе полей расстояний разработан алгоритм для численного расчёта остаточных областей в широком классе задач. Воспользовавшись временной и пространственной когерентностью данных, удалось сделать сложность алгоритма линейной от количества узлов сетки.

На основе описанного алгоритма для двух- и трёхмерных поисковых задач реализована система, визуализирующая информационные множества в реальном времени.

8. СПИСОК ЛИТЕРАТУРЫ

[1] Айзекс Р. *«Дифференциальные игры»,* М., Мир, 1967.

[2] Петросян Л.А., Зенкевич Н.А. «*Оптимальный поиск в условиях конфликта*», Л., 1987.

[3] Чхартишвили А.Г., Шикин Е.В. «*Динамический поиск объектов. Геометрический взгляд на проблему*», Фундаментальная и прикладная математика, 1995, т.1, вып. 4.

[4] Ким Д.П. «*Методы поиска и преследования подвижных объектов*», М., Наука, 1989.

[5] Хеллман О. «*Введение в теорию оптимального поиска*», М., 1989.

[6] Березин С.Б. «*Графический подход к исследованию и решению задач динамического поиска*», М., 2002.

[7] Губайдуллин С.М., Шикин Е.В. *«Следящие области на цилиндре и на торе»*, Вестник московского университета, сер. 15, Вычислительная математика и кибернетика, 1992, №2.

[8] Скворцов А.А. *«Динамический поиск в трёхмерных выпуклых областях»*, М., Деп. в ВИНИТИ, №2985-В95.

[9] B. Payne and A. Toga *"Distance field manipulation of surface models",* Computer Graphics and Applications, 1992.

[10] S.F. Frisken et al *"Adaptively Sampled Distance Fields: A general Representation of Shape for Computer Graphics"*, Computer graphics and interactive techniques, 2000.

[11] W.E. Lorensen, H.E. Cline *"Marching Cubes: A high resolution 3D surface construction algorithm"*, CGIT, 1987.

[12] B. Vrolijk, C.P. Botha, F. H. Post *"Fast time-dependent isosurface extraction and rendering"*, SCCG, 2004.

[13] Чхартишвили А.Г., Шикин Е.В. «*Следящая область в пространстве Лобачевского»*, Вестник Московского университета, сер. 1, Математика. Механика. 1994, N2.

[14] Чхартишвили А.Г. *«Метод следящих областей в задачах динамического поиска»*, М., 1994.

[15] Скворцов А.А. *«Алгоритмическая реализация теоретико-множественных операций на плоскости»*, М., Деп. в ВИНИТИ 10 июля 2000 г., №1915-В00.

[16] K. Sugihara, M. Iri *"A Solid Modelling System Free from Topological Inconsistency"*, Journal of Informational Processing, Vol. 12, No. 4, 1989.

[17] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. "*Function representation in geometric modeling: concepts, implementation and applications",* The Visual Computer, 11(8):429–446, 1995.

[18] G. Treece, R. Prager, A. Gee. *"Regularised Marching Tetrahedra: improved iso–surface extraction",* Computers & Graphics, 23(4):583–598, 1999.

[19] А. Семенихин, А. Игнатенко *«Сравнительный анализ методов интерактивной триангуляции сеточных функций»*, http://cgm.graphicon.ru

[20] J. Frost, P. Cagle "An Amazing, Space Filling, Non-regular Tetrahedron", http://mathforum.org/pcmi/hstp/resources/dodeca/index.html

[21] R. Kimmel, J. A. Sethian *"Computing Geodesic Paths on Manifolds"*, Proceedings of National Academy of Sciences 95, 15,8431-8435, (July) 1998.

[22] Ларин П.М. *«Следящая область пары отрезков»*, Фундаментальная и прикладная математика, 1998, Т.4, Вып.2, С.559-566.

[23] S. L. Chan and E. O. Purisima *"A new tetrahedral tesselation scheme for isosurface generation"*, Computers & Graphics, 22(1):83–90, February 1998.

[24] T. Theußl, T. Möller, M. E. Gröller *"Optimal regular volume sampling"*. Proceedings of IEEE Visualization 2001.

[25] K. Levinski, A.Sourin *"Interactive Polygonisation for Function-based Shape Modelling"*, Eurographics 2002.

Об авторе

Павел Воронин – аспирант факультета Вычислительной Математики и Кибернетики Московского Государственного Университета им. М.В. Ломоносова.

E-mail: pavel.voronin@gmail.com

On a new approach to computation and visualization of information sets in dynamic search

Abstract

This article presents a new general distance fields based algorithm for computation and visualization of information sets in dynamic search problems. Details on both two- and three-dimensional versions are given. In both cases interactive framerate was achieved.

About the author

Pavel Voronin is a Ph.D. student at Lomonosov Moscow State University, Department of Computational Mathematics and Cybernetics, e-mail: pavel.voronin@gmail.com

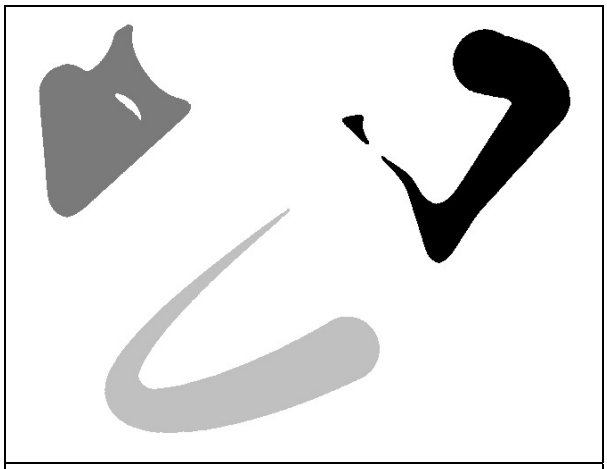

Рис. 15. Остаточные области на плоскости.

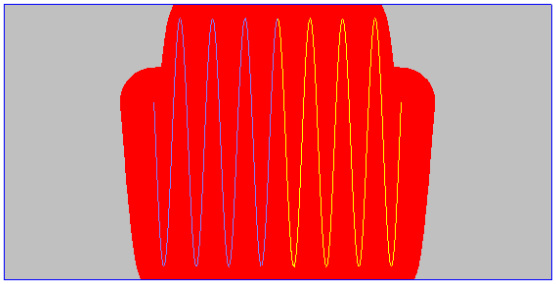

Рис. 16. Совместная остаточная область двух ищущих в прямоугольнике

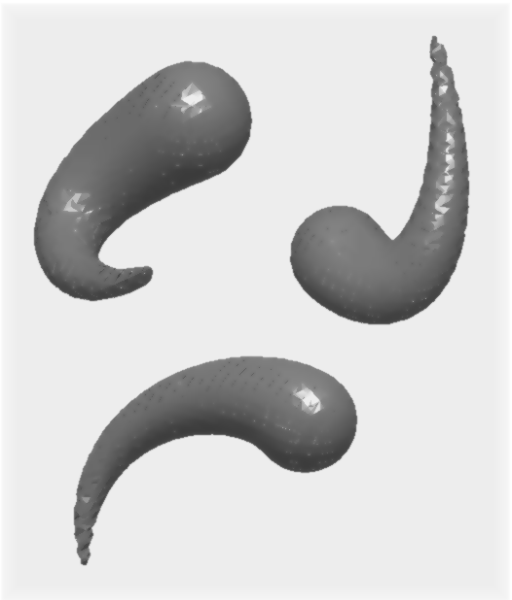

Рис. 17. Остаточные области в пространстве