

Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Кафедра компьютерных технологий

А. П. Ковшаров

Оптимизация параметров стратегий поиска объектов на море

Бакалаврская работа

Научный руководитель: А. С. Ковалев

Санкт-Петербург
2015

Содержание

Содержание	3
Введение	5
Глава 1. Постановка задачи	7
1.1 Задача построения маршрута поиска в общем случае	7
1.1.1 Расширения задачи коммивояжера	7
1.1.2 Распределение частиц	9
1.1.3 Формулировка задачи построения маршрута поиска в общем случае	10
1.2 Задача построения маршрута поиска стратегией “Параллельное галсирование”	11
1.2.1 Стратегии поиска	11
1.2.2 Формулировка задачи построения маршрута поиска стратегией “Параллельное галсирование”	12
1.3 Задача симуляции прохождения маршрута	13
Глава 2. Симуляция эволюции распределения	15
2.1 Примеры моделей и распределений	15
2.1.1 Разновидности начальных распределений	15
2.1.2 Разновидности моделей изменения распределения	16
2.2 Процесс симуляции изменения распределения и прохождения маршрута	18
2.2.1 Симуляция изменения распределения	18
2.2.2 Симуляция сбора частиц средством поиска	19
2.3 Корректировка области симуляции с течением времени	20
2.4 Статистика прохождения маршрута	22

Глава 3. Алгоритм построения маршрутов согласно стратегии “Параллельное галсирование”	24
3.1 Алгоритм построения маршрута при фиксированном рас- пределении	24
3.2 Корректировка маршрута	24

Введение

Основной целью данной работы является разработка эффективного метода построения маршрутов поиска объектов согласно общепринятым стратегиям поиска. Метод должен работать с большим количеством различных моделей изменения распределения вероятности обнаружения объекта со временем.

Существующие подходы строят весь маршрут исходя из статичных данных в начальный момент времени — информации о начальном распределении и модели его изменения. Таким образом главным недостатком существующих подходов является необходимость разработки нового алгоритма для каждой модели изменения распределения. Учитывая тот факт, что подобрать правильную модель, хорошо приближающую реальность, крайне непросто, возникает необходимость разработки алгоритма, работающего единообразно на широком классе различных моделей. Основная идея рассматриваемого подхода — использование симулятора для получения информации о распределении в любой момент времени при построении маршрута. Таким образом при планировании пользователь в первую очередь выберет модель, как можно лучше приближающую реальность в данном случае, а после запустит алгоритм построения маршрута.

Таким образом задачами данной работы являются разработка инструмента симуляции изменения вероятности во время прохождения маршрута и алгоритма построения маршрута при заданных условиях. В данной работе будет рассмотрен алгоритм построения маршрута согласно стратегии “Параллельное галсирование”.

Рассчетная задача может быть использована оператором как вспомогательное средство при планировании маршрутов поиска объектов, по-

терпевших бедствие, в реальной жизни. Визуализированный процесс изменения распределения может пригодиться при решении какие области следует исследовать более подробно, а какие можно пропустить.

Концепт данной задачи был продемонстрирован на одной из выставок. К задаче был проявлен интерес и было решено внедрить ее в комплекс расчетных морских задач.

В главе 1 решаемая задача будет рассмотрена более подробно. Описаны классы маршрутов, получаемые при использовании стратегии “Параллельное галсирование”. Будут приведены особенности задачи, которые отличают ее от классической задачи коммивояжера и делают невозможным использование ранее разработанных методов решения для решения исходной задачи в общем случае.

В главе 2 будут рассмотрены вопросы, связанные с разработкой симулятора на CUDA. Обозначены предоставляемые симулятором сервисы.

В главе 3 будет описан алгоритм построения маршрутов согласно стратегии “Параллельное галсирование”. В первую очередь будет рассмотрен подход с динамическим программированием для определения глобального маршрута при фиксированном распределении. Далее будет рассмотрена корректировка маршрута согласно изменениям распределения. В конце будут рассмотрены локальные оптимизации маршрута для повышения его эффективности.

Глава 1. Постановка задачи

1.1. ЗАДАЧА ПОСТРОЕНИЯ МАРШРУТА ПОИСКА В ОБЩЕМ СЛУЧАЕ

1.1.1. Расширения задачи коммивояжера

В классической формулировке задача коммивояжера (traveling salesman problem, TSP) звучит так: Дан взвешенный граф необходимо найти цикл, минимального веса, посещающий все его вершины. Евклидовым TSP — называется частный случай TSP, когда весами ребер являются расстояния на плоскости. Задача построения маршрута поиска объекта на первый взгляд очень похожа на задачу коммивояжера. Однако в реальности особенности задачи оказываются существенными:

- Средство поиска в один момент времени посещает несколько вершин, а именно все вершины попадающие в круг определенного радиуса с центром в его текущем положении. Существует расширение TSP под названием GTSP или обобщенная задача коммивояжера, которое решает следующую задачу: Дан взвешенный граф и разбиение его вершин на **непересекающиеся** множества, необходимо найти цикл минимального веса, посещающий хотя бы одну вершину из каждого множества. Существует сведение данной задачи к TSP не увеличивающее размерности и доказывающее ее NP -полноту. Но к сожалению данное расширение неприменимо к нашей задаче, так как множества могут пересекаться.
- Зачастую все вершины посетить физически невозможно. Соответственно выделяются вершины в которых более вероятно обнаружить объект. Сопоставим вершине v величину p_v — вероятность обнару-

жить объект в этой вершине. $p_{path} = \sum_{v \in path} p_v$. На практике длина путей с $p_{path} \geq 0.99$ может превышать длину путей с $p_{path} \geq 0.9$ в десятки раз. То есть длина пути растет экспоненциально в зависимости от p_{path} . Следовательно необходимым параметром задачи становится максимальная длина пути (или время поиска с физической точки зрения). Известно обобщение Profit Based TSP: каждой вершине сопоставляется значение p_v , при посещении вершины к сумме призов добавляется $p_v - t_v$, где t_v —время посещения, необходимо составить маршрут с наибольшей суммой призов. К сожалению наша задача и здесь сравнительно более общая, так как величины призов могут изменяться нелинейно.

- Распределение p_v действительно может быть не статично по времени и изменяться согласно заданной модели. Следовательно время и расстояние не взаимозаменяемы в поставленной задаче и оптимальное значение p_{path} может быть различно если мы фиксируем один из параметров. В поставленной задаче будет фиксированно время. Однако все модели изменения обладают свойством: распределение изменяется непрерывно, перераспределяясь не превышая фиксированную скорость, не появляется извне и не исчезает (будем считать что при посещении призы “собираются” и исчезают). Соответственно $p_{v,t} = \sum_{pos(q_i)=t} q_i$ и приз q_i в вершине v мы можем собрать лишь в какие-то промежутки времени. Обобщение Time Windows TSP решают соответствующую задачу: вершину v — можно посетить лишь во время $[t_{v,l}; t_{v,r}]$. Проблема сведения к этому обобщению в том, что значительно увеличивается количество вершин, в частности новая вершина будет сопоставлена j -му моменту, когда приз q_i оказался в вершине v .

- Другое обобщение для случая с движущимися вершинами — Kinetic TSP или Moving Targets TSP. Это классическое евклидово TSP, в котором вершины движутся в фиксированном направлении. Однако ранее были рассмотрены только случаи где вершины движутся с фиксированной скоростью или коммивояжер должны возвращаться в стартовую вершину после посещения каждой. В исходной задаче скорость и направления перемещения призов могут изменяться. Кроме того сведение вновь значительно увеличивает число вершин.
- Фактически до текущего момента времени задача формулировалась в непрерывном случае. При разработке алгоритма для ЭВМ необходимо провести некоторую дискретизацию. Необходимое количество вершин для двумерной задачи растет квадратично и соответственно приемлемое значение может достигать миллионов вершин. Такое количество вершин велико даже для самых быстрых реализаций приближенных решений TSP.
- В реальной жизни невозможно искать согласно произвольной траектории из-за технических ограничений на передвижение поискового средства. Помимо этого недопустимо применение не общепризнанных, зарекомендовавших себя стратегий поиска.

1.1.2. Распределение частиц

Использование понятия распределение вероятности обнаружения противника в дальнейшем будет не совсем удобно, так как одна из основных частей задачи “сбор” вероятности — не является корректной операцией над вероятностями.

Введем альтернативное понятие — распределение “частиц”. Частица $\pi : \Pi$ — гипотеза изначального расположения объекта и его поведения в дальнейшем. Вес частицы w_π — вероятность осуществления именно этой

гипотезы. $pos(\pi, t)$ — положение частицы π в момент времени t . Сумма весов всех частиц изначально равна единице. Расширение понятия веса частиц на непрерывный случай аналогично расширению для вероятностного пространства. “Собрать частицу” — проверить гипотезу. После проверки частица исчезает и больше не может быть собрана. Распределение частиц $f(dS, t, path)$ — функция, ставящая в соответствие области и времени сумму весов частиц находящихся в этой области в заданное время, с учетом частиц собранных средством поиска к данному моменту. С течением времени частицы могут перемещаться в любом направлении с ограниченной скоростью, однако не могут появляться из ниоткуда или исчезать иным способом, кроме сбора их средством поиска. Сумма собранных частиц фактически равна априорной вероятности обнаружить объект, имеющий заданное начальное распределение вероятности и закон его изменения.

1.1.3. Формулировка задачи построения маршрута поиска в общем случае

Задано распределение частиц $f(dS, t, path)$, t_{search} — время поиска и параметры средства поиска: pos_0 — начальная позиция, r — радиус обнаружения и v_{max} — максимальная скорость передвижения. Необходимо найти маршрут $path(t)$ — определяющий позицию средства поиска в любой момент времени, максимизирующий сумму весов собранных частиц sum_{res} . Более формально $sum_{res} = 1 - \int_S f(dS, t_{search}, path)$, $path(0) = pos_0$, $|path'(t)| \leq v_{max}$, собраны те и только те частицы π для которых $\exists \tau \leq t_{search} ||pos(\pi, \tau) - path(\tau)|| \leq r$.

1.2. ЗАДАЧА ПОСТРОЕНИЯ МАРШРУТА ПОИСКА СТРАТЕГИЕЙ “ПАРАЛЛЕЛЬНОЕ ГАЛСИРОВАНИЕ”

1.2.1. Стратегии поиска

Однако помимо того, что решения общей задачи сложно найти, построенные маршруты могут быть практически неприменимы из-за ограничений на передвижение средства поиска. В реальной жизни поиск объектов осуществляется общепринятыми стратегиями, такими как “заданный маршрут”, “гребенка”, “расширяющийся квадрат”, “параллельное галсирование”(рис. 1.1). **Галс** — линия пути воздушного судна от поворота до поворота.



Рис. 1.1: Стратегии поиска

Согласно [приказу об утверждении правил проведения авиационных поисково-спасательных работ]:

“Поиск способом “Заданный маршрут” выполняется по линии заданного пути, проходящего вдоль участка маршрута воздушного судна, потер-

певшего бедствие.”

“Способ “Гребенка” заключается в одновременном обследовании района поиска группой воздушных судов путем совместного полета по параллельным прямолинейным маршрутам на интервалах, составляющих примерно 75% визуальной видимости или дальности действия поисковой аппаратуры.”

“Поиск [способом “Расширяющийся квадрат”] состоит в обследовании одиночным воздушным судном района вокруг известной точки, в которой предполагается нахождение потерпевшего бедствия экипажа.”

“Поиск способом “Параллельное галсирование” применяется при недостаточном количестве имеющихся поисково-спасательных воздушных судов и для обследования значительной площади. ... При этом способе район поиска разделен на несколько участков поиска (полос), которые просматриваются одновременно несколькими одиночными воздушными судами или последовательно одним воздушным судном. ... Расстояние между галсами устанавливается таким же, как и интервал между воздушными судами при поиске способом “Гребенка”. Для сокращения количества разворотов прямолинейные участки галсов целесообразно ориентировать вдоль полос обследования.” Следуют отметить, что для разных местностей рекомендуется разное расстояние между галсами.

В данной работе будет рассмотрено построение маршрута поиска одиночным самолетом согласно стратегии “Параллельное галсирование”.

1.2.2. Формулировка задачи построения маршрута поиска стратегией “Параллельное галсирование”

Задано распределение частиц $f(dS, t, path)$; t_{search} — время поиска; параметры средства поиска: pos_0 — начальная позиция, r — радиус обнаружения и v_{max} — максимальная скорость передвижения; параметры стратегии поиска: l — прямая параллельная направлению гал-



Рис. 1.2: Параметры “параллельного галсирования”

сов(полосе обследования). Необходимо найти маршрут $path(t)$ — определяющий позицию средства поиска в любой момент времени, максимизирующий сумму весов собранных частиц sum_{res} . $path(t)$ должен состоять из прямолинейных отрезков, удовлетворяющих стратегии поиска “параллельное галсирование” с галсами параллельными прямой l . $path(t)$ должен быть задан количеством галсов n , а также последовательностями чисел $l_i, i \in 1..n$ — величина проекции на прямую l (может быть отрицательной), $h_i, i \in 1..(n-1)$ — расстояние между галсами с номерами i и $i+1$. (рис. 1.2)

$$\sum_{i=1}^{n-1} (|l_i| + h_i) + |l_n| \leq v_{max} \cdot t_{search}.$$

Средство поиска перемещается по пути со скоростью v_{max} . Более формально $sum_{res} = 1 - \int_S f(dS, t_{search}, path)$, $path(0) = pos_0$, $|path'(t)| = v_{max}$, собраны те и только те частицы π для которых $\exists \tau \leq t_{search} ||pos(\pi, \tau) - path(\tau)|| \leq r$.

1.3. ЗАДАЧА СИМУЛЯЦИИ ПРОХОЖДЕНИЯ МАРШРУТА

Для визуализации распределений использована аналогия с температурой: Наиболее вероятные участки обнаружения противника отображаются белым цветом (наиболее горячие), менее вероятные красным, самые маловероятные синим (рис. 1.3). Промежуточные состояния изображены градиентом между этими цветами.

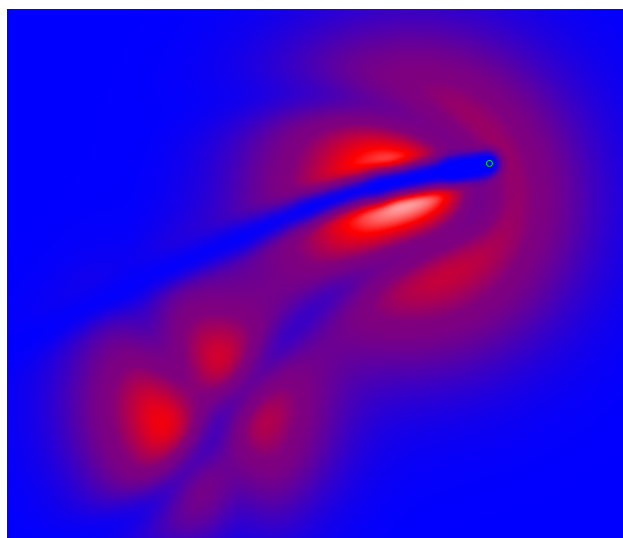


Рис. 1.3: Пример визуализации

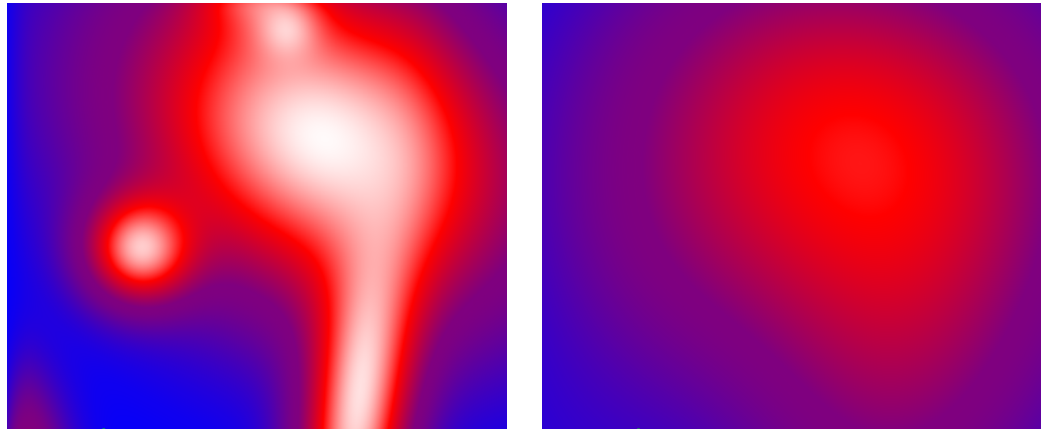
Инструмент, проводящий симуляцию изменения распределения вероятности в условиях сбора ее средством поиска, помимо визуализации должен предоставлять статистические данные для возможности его использования в алгоритмах оптимизирующих параметры маршрутов поиска. Размеры данных, которые должны быть обработаны в реальном времени оказываются слишком велики и использование лишь мощностей CPU оказывается недостаточным. Поэтому для разработки были задействованы мощности GPU и технология CUDA.

Благодаря тому, что большинство моделей изменения распределения можно представить **ядром** (Kernel, Convolution Matrix, матрица свертки — небольшая матрица используемая для создания блюра, резкости и тому подобных эффектов, посредством свертки с изображением) и фактически производить Image Convolution (свертку изображения) с заданным ядром на каждой итерации — задача хорошо подходит для архитектуры CUDA. Более сложные модели удастся симулировать посредством нескольких запусков с различными ядрами и последующей их композицией.

Более подробно подход для симуляции описан в главе 2.

Глава 2. Симуляция эволюции распределения

2.1. ПРИМЕРЫ МОДЕЛЕЙ И РАСПРЕДЕЛЕНИЙ



(а) Композиция стандартных распределений для задания начального распределения (b) Изменения распределения согласно модели случайного блуждания

Рис. 2.1: Пример симуляции изменения распределения без маршрута

2.1.1. Разновидности начальных распределений

Начальное распределение частиц фактически может быть представлено любой двумерной функцией, интеграл которой по всей плоскости равен единице. Однако для большинства применений кажется достаточным задание начального распределения как композиции стандартных двумерных распределений (рис. 2.1(а)), таких как равномерное распределение в произвольной области или нормальное распределение задаваемое эллипсом, содержащим 3σ вероятности и тому подобное. Инструмент симуляции предоставляет удобный инструмент их задания.

На протяжении всей симуляции распределение хранится в текстуре (матрице). Значение, которое записано в каждом пикселе, означает сум-

марный вес частиц, находящихся под этим пикселем, если его нарисовать в мире. Пиксель текстуры имеет достаточно малый размер при рисовании в мире, чтобы достаточно точно различать положения различных частиц, при этом достаточно большой, чтобы текстура покрывающая всю вероятность влезла в память GPU, и ее обработка занимала приемлимое время.

2.1.2. Разновидности моделей изменения распределения

Изменения распределений происходят с фиксированным временем дискретизации Δt . Для симуляции изменений в большинстве моделей используется свертка с ядром с периодом Δt .

Процесс свертки с ядром осуществляется следующим образом, пусть $D_t, D_{t+\Delta t}$ — последовательные распределения, а K — ядро размера $(2k + 1) \times (2k + 1)$ (Пусть обращение к несуществующим элементам возвращает 0):

$$D_{t+\Delta t}[i][j] = \sum_{di=-k}^k \sum_{dj=-k}^k D_t[i + di][j + dj] \cdot K[di][dj] \quad (2.1)$$

Размер матрицы свертки выбирается такого размера, чтобы дать возможность симулировать передвижение в произвольном направлении с достаточной точностью. На данный момент выбрано значение $k = 5$.

Время Δt выбирается исходя из размера ядра и физического размера пикселя. Если размер пикселя $h \times h$, максимальная скорость перемещения объекта v_{max} тогда (2.2).

$$\Delta t = \frac{k \cdot h}{v_{max}} \quad (2.2)$$

- **Модель случайных блужданий**

Простейший пример модели изменения распределения — модель случайных блужданий (рис. 2.1(b)). Ядро для модели случайных блужданий представляет собой матрицу со значениями равными

площади пересечения текущей ячейки с кругом вписанным в квадрат размера $2kh \times 2kh$. Ячейки соответственно отнормированы для равенства суммы единице.

Посредством изменения ядра (замены круга на кольца), можно изменять интенсивность блужданий.

- **Модель движения в одном направлении**

Пусть заданы последовательности $\alpha_i, p_i, 1 \leq i \leq n$ — углов и вероятностей движения в направлении каждого из них. Для симулирования заданной модели можно использовать n экземпляров ядер и текстур для симуляции движения распределений соответственно в каждом из направлений. На выходе для получения итогового распределения достаточно скомбинировать полученные результаты с соответствующими весами p_i .

- **Модель притяжения(отталкивания) к заданным точкам**

Иногда уместной оказывается модель избегания некоторых точек (например точек вблизи текущей позиции поискового средства, если объект является вражеским и не желает быть обнаруженным) или напротив приближения к ним (например в случае если объект знает ближайшее положение суши, он может стремиться двигаться в ее направлении). Здесь удобно воспользоваться аналогией с электрическими зарядами, однако действие электрических сил для создания ускорения не кажется естественным в данной задаче. Значение функции аналогичной силе Кулона будет задавать направление и скорость (вместо ускорения) движения при нахождении в данной точке. Таким образом каждая из точек имеет свой заряд q_i , позицию pos_i , радиус действия R_i и закон убывания влияния, для примера используем Кулоновские r^{-2} . Пусть максимальная скорость перемещения объекта v_{max} . Таким образом вектор скорости объекта в

точке p равен (2.5).

$$\chi_i(p) = \begin{cases} 1 & \text{если } (p - pos_i)^2 \leq R_i \\ 0 & \text{иначе} \end{cases} \quad (2.3)$$

$$v' = \sum_{i=1}^n \frac{\chi_i(p) \cdot q_i \cdot (pos_i - p)}{|pos_i - p|^3} \quad (2.4)$$

$$v = \min(|v'|, v_{max}) \cdot \frac{v'}{|v'|} \quad (2.5)$$

При реализации данной модели, во время свертки ядро в каждой конкретной ячейке будет различным. Таким образом приходится работать с текстурой скоростей из глобальной памяти вместо ядра в константной памяти как в модели со случайными блужданиями, что в несколько раз замедляет время симуляции.

2.2. ПРОЦЕСС СИМУЛЯЦИИ ИЗМЕНЕНИЯ РАСПРЕДЕЛЕНИЯ И ПРОХОЖДЕНИЯ МАРШРУТА

2.2.1. Симуляция изменения распределения

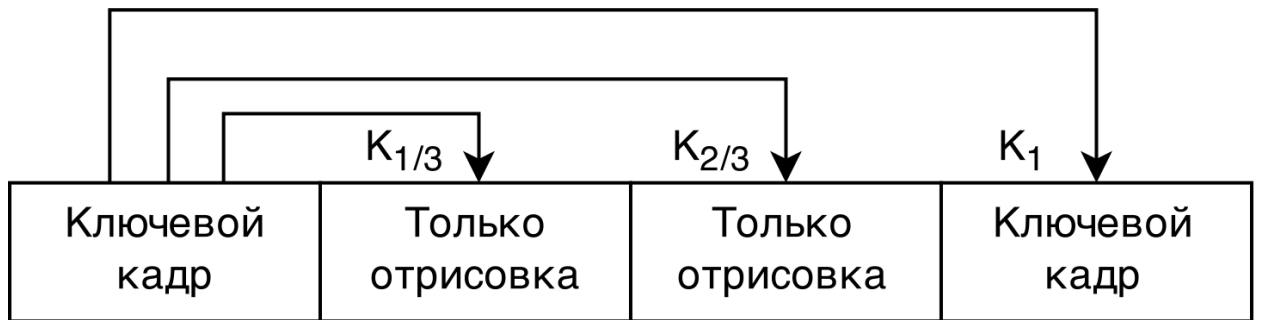


Рис. 2.2: Схема применения частичных ядер

Симуляция изменения распределения производится путем применения свертки текущего распределения с ядром с промежутком времени

Δt_{frame} . Если $\frac{1}{\Delta t} \geq 60FPS$, где Δt — время с которым необходимо применять заданное ядро, тогда $\Delta t_{frame} = \Delta t$. Иначе время применения ядра слишком велико и не обеспечивает необходимой динамичности.

Для обеспечения актуальной информации в промежутках между ключевыми кадрами, идущими с интервалом Δt , необходимо добавить m промежуточных, так чтобы $\Delta t_{frame} = \frac{\Delta t}{m+1} \geq 60FPS$. Таким образом из оригинального ядра K_1 нам необходимо получить m промежуточных с некоторой точностью приближающих реальность $K_{\frac{1}{m+1}}, K_{\frac{2}{m+1}}, \dots, K_{\frac{m}{m+1}}$. Посредством применения заданных ядер к последнему ключевому кадру, будут получены приближенные распределения в промежуточные моменты времени (рис. 2.2), однако полученные с помощью этих ядер текстуры не будут использованы для построения следующих, что избавляет симулятор от накопления погрешности от применения приближенных ядер.

Для получения ядра $K_{\frac{i}{m+1}}$ необходимо элементы ядра K_1 домножить на $\alpha = 1 + c_0 - c_0^{\frac{i}{m+1}}$, где $c_0 = K_1[k][k]$, после чего добавить остаток $1 - \alpha$ к стационарной ячейке ядра $K_{\frac{i}{m+1}}[k][k]$.

2.2.2. Симуляция сбора частиц средством поиска

Сбор частиц производится с интервалом Δt_{frame} в текстуре соответствующей последнему на данный момент ключевому кадру. После сбора к ключевому кадру применяются ядра, рассчитывающие перемещение оставшихся частиц, для получения кадра, соответствующего текущему времени.

Сбор частиц осуществляется посредством домножения значения каждого пикселя на долю его площади, не покрытой зоной видимости ни в один из моментов времени в течении текущего интервала Δt_{frame} . То есть подразумевается, что вес частицы распределен равномерно по площади пикселя.

Для получения суммарной зоны видимости во все моменты времени

текущего интервала маршрут средства поиска делится на прямолинейные участки. После чего на каждом участке берется объединение $n = \beta \frac{dist}{r}$, взятых через равные промежутки, зон видимости, где $dist$ — длина прямолинейного участка, r — радиус видимости средства поиска, $beta$ — коэффициент точности края. Выбрано $\beta = 100$, что для всех разумных размеров текстур дает достаточно точные края объединения зон видимости.

2.3. КОРРЕКТИРОВКА ОБЛАСТИ СИМУЛЯЦИИ С ТЕЧЕНИЕМ ВРЕМЕНИ

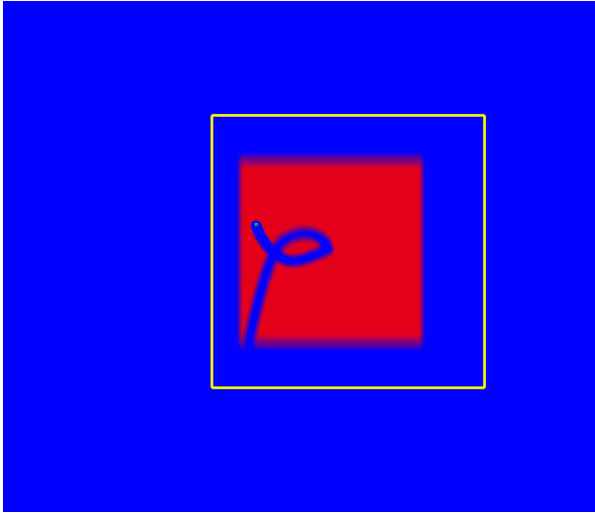


Рис. 2.3: Начальная область симуляции

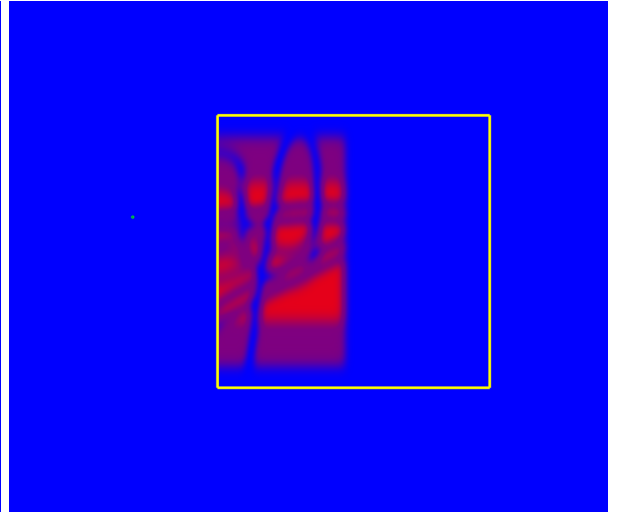


Рис. 2.4: Выход распределения за границы области, без корректировки

Одна из сторон прямоугольника $rect$, охватывающего интересующий район поиска, параллельна прямой галсирования (в случае построения маршрутов стратегией “Параллельное галсирование”). В начальный момент времени район задается вручную пользователем (рис. 2.3). Частицы, оказавшиеся снаружи начального района учитываться не будут.

Разрешение текстуры выбирается таким образом, чтобы размер пикселя был квадратным (при необходимости немного изменяется выбранный район для симуляции), а количество пикселей было максималь-

но возможным, при котором можно обеспечить приемлимую производительность. Сейчас выбрано количество пикселей порядка $800^2 = 6.4 \cdot 10^5$.

С течением времени рассматриваемые частицы могут выходить за пределы текущего рассматриваемого района. Без корректировки района информация о местоположении этих частиц будет потеряна (рис. 2.4).

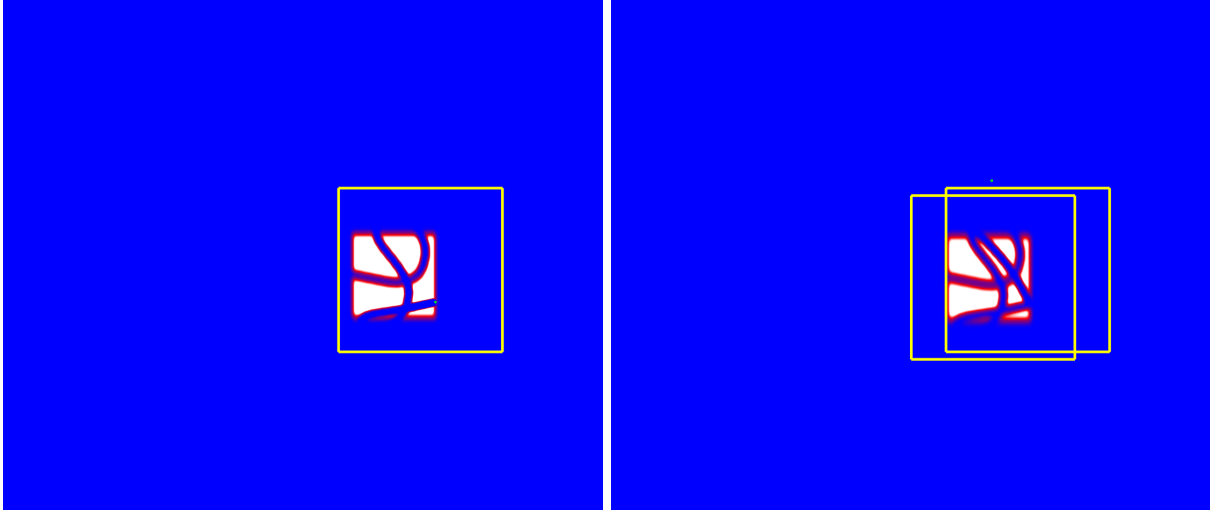


Рис. 2.5: Приближение распределения к границе области симуляции

Рис. 2.6: Изменение центра области симуляции согласно новому положению распределения

Корректировка рассматриваемого района осуществляется, когда в одном из пикселей на границе шириной $2k$, где k — параметр ядра, находятся частицы с суммарным весом более ϵ . (рис. 2.5) В реализации выбрано $\epsilon = 5 \cdot 10^{-3}$.

Для корректировки находится прямоугольник *rect*, охватывающий все пиксели суммарный вес в которых больше ϵ , причем длина и ширина прямоугольника содержат четное число пикселей. После чего выделяется новая текстура с центром, совпадающим с центром полученного прямоугольника (рис. 2.6) (благодаря четности, стороны пиксели новой и старой текстуры выровнены), после чего данные копируются в новую текстуру.

Так как разрешение новой текстуры всегда остается прежним, возможно необходимо увеличить прямоугольник покрываемый текстурой для

покрытия прямоугольника *rect*. Размер покрываемого прямоугольника последовательно увеличивается в два раза, равномерно расширяясь относительно центра, до тех пор пока не покроет прямоугольник *rect*, (рис. 2.7) после чего его положение немного корректируется, чтобы в каждый новый пиксель (если увеличили в 2^m раз) попало ровно 2^{2m} целых старых пикселей.



Рис. 2.7: Расширение и изменение центра области симуляции

2.4. СТАТИСТИКА ПРОХОЖДЕНИЯ МАРШРУТА

Симулятор предоставляет статистику о прохождении маршрута, по которой строятся графики прогресса поиска и поисковой производительности. Оба графика отображают зависимости от времени, поэтому изображены в одном окне с разными шкалами по оси ординат. (рис. 2.8)

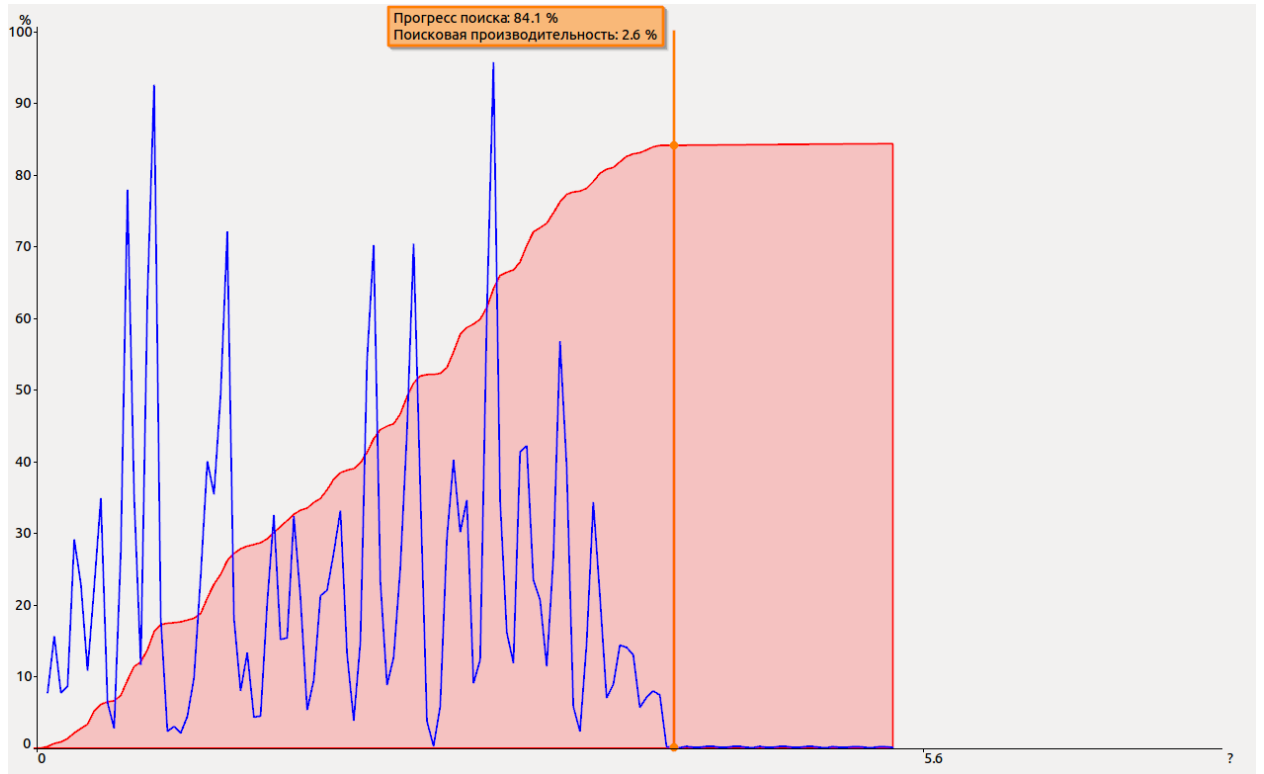


Рис. 2.8: Окно с отображением статистики

Прогресс поиска (красный график) — отображает текущий суммарный вес собранных частиц в процентах от общего веса в начальный момент времени.

Поисковая производительность (синий график) — отображает текущую эффективность поиска. Фактически является производной прогресса поиска по времени, однако также измеряется в процентах. Поисковая производительность $x\%$ — означает, что если мы будем искать с такой поисковой производительностью в течении всего времени поиска t_{search} , то сможем собрать $x\%$ веса частиц.

Глава 3. Алгоритм построения маршрутов согласно стратегии “Параллельное галсирование”

3.1. АЛГОРИТМ ПОСТРОЕНИЯ МАРШРУТА ПРИ ФИКСИРОВАННОМ РАСПРЕДЕЛЕНИИ

Текущий ход	Следующий ход	Собрано на текущем ходу	Новое состояние
L	L	$v[row][col][notCleared]$	$dp[cntHor + 1][row][col - 1][L][notCleared]$
L	LU	$v[row][col][notCleared]$ $cs[row][col][LDC]$	$dp[cntHor][row + 1][col][LU][0]$
R	R	$v[row][col][notCleared]$	$dp[cntHor + 1][row][col + 1][R][notCleared]$
R	RU	$v[row][col][notCleared]$ $cs[row][col][RDC]$	$dp[cntHor][row + 1][col][RU][0]$
LU	LU	$v[row][col][0]$ $hr[row][col]$	$dp[cntHor][row + 1][col][LU][notCleared + 1]$
LU	L	$v[row][col][0]$ $cs[row][col][RUC]$	$dp[cntHor + 1][row][col - 1][L][notCleared]$
LU	R	$v[row][col][0]$ $cs[row][col][LUC]$	$dp[cntHor + 1][row][col + 1][R][notCleared]$
RU	RU	$v[row][col][0]$ $hl[row][col]$	$dp[cntHor][row + 1][col][RU][notCleared + 1]$
RU	L	$v[row][col][0]$ $cs[row][col][RUC]$	$dp[cntHor + 1][row][col - 1][L][notCleared]$
RU	R	$v[row][col][0]$ $cs[row][col][LUC]$	$dp[cntHor + 1][row][col + 1][R][notCleared]$

Таблица 3.1: Переходы из состояния $dp[cntHor][row][col][curMove][notCleared]$

3.2. КОРРЕКТИРОВКА МАРШРУТА