ОЦСАНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

**Е.В. Первушин**

# Применение конечных автоматов для улучшения характеристик нейроподобных сетей

Руководитель – И.М. Аничкин

Санкт-Петербург

2007

# ОГЛАВЛЕНИЕ

## *ВВЕДЕНИЕ*

Нейронные сети возникли из исследований в области искусственного интеллекта – из попыток воспроизвести способность биологических нервных систем обучаться и исправлять ошибки, моделируя низкоуровневую структуру мозга. Основной областью исследований по искусственному интеллекту в 60-е – 80-е годы были экспертные системы. Такие системы основывались на высокоуровневом моделировании процесса мышления (в частности, на представлении, что процесс мышления построен на манипуляциях с символами). Скоро стало ясно, что подобные системы, хотя и могут принести пользу в некоторых областях, не охватывают некоторые ключевые аспекты человеческого интеллекта.

Искусственные нейронные сети являются электронными моделями нейронной структуры мозга, который, главным образом, учится на опыте. Для работы с нейронной сетью требуется предварительное обучение – корректировка весов связей, в результате которой каждое входное воздействие приводит к формированию соответствующего выходного сигнала. После обучения сеть может применять полученные навыки к новым входным сигналам.

Вычисления в нейронных сетях существенно отличаются от традиционных, в силу высокой параллельности их можно рассматривать как коллективное явление.

Естественные нейронные сети  доказывают, что множество проблем, не поддающиеся решению традиционными методами, могут быть эффективно решены с их помощью – задачи, которые можно представить в общем виде как задачи построения отображения X –> Y, где X и Y – соответственно пространства входных и выходных признаков.

Системы на основе искусственных нейронных сетей позволяют с успехом решать проблемы распознавания образов, выполнения прогнозов, оптимизации, ассоциативной памяти и управления.

**Классификация образов.** Задача состоит в определении принадлежности входного образа (например, языкового сигнала или рукописного символа), представленного вектором признаков к одному или нескольким предварительно определенным классам. К известным приложениям относятся распознавание букв, распознавание языка, классификация сигнала электрокардиограммы, классификация клеток крови.

**Кластеризация/категоризация.** При решении задачи кластеризации, обучающее множество не имеет меток классов. Алгоритм кластеризации основан на подобии образов и помещает похожие образы в один кластер. Известны случаи применения кластеризации для добычи знаний, сжатия данных и исследования свойств данных.

**Аппроксимация функций.** Обучающая выборка (($x_1$,$y_1$), ($x_2$,$y_2$)..., ($x_n$,$y_n$)) (пары данных вход-выход), генерируется неизвестной функцией F, искаженной шумом. Задача аппроксимации состоит в нахождении неизвестной функции F. Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования.

**Предвидение/прогноз.** В последовательные моменты времени $t_1$, $t_2$,..., $t_n$ заданы n дискретных отсчетов $y(t_1)$, $y(t_2)$, ..., $y(t_n)$. Задача состоит в предвидении значения $y(t_{n+1})$ в следующий момент времени $t_{n+1}$. Предвидение/прогноз имеют большое значение для принятия решений в бизнесе, науке и технике (предвидение цен на фондовой бирже, прогноз погоды).

**Оптимизация.** Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться

как проблемы оптимизации. Задачей алгоритма оптимизации является нахождение такого решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию.

**Память, адресуемая по смыслу.** В традиционных компьютерах обращение к памяти доступно только с помощью адреса, который не зависит от содержания памяти. Более того, если допущена ошибка в вычислении адреса, то может быть найденная совсем другая информация. Ассоциативная память или память, адресуемая по смыслу, доступна по указанию заданного содержания. Содержимое памяти может быть вызвано даже по частичному входу или поврежденном содержании. Ассоциативная память может быть использована в мультимедийных информационных базах данных.

**Управление.** Рассмотрим динамическую систему, заданную совокупностью {u(t), y(t)}, где u(t) – входное управляющее воздействие, а y(t) – выход системы в момент времени t. В системах управления с эталонной моделью целью управления является расчет такого входного воздействия u(t), при котором система действует по желательной траектории, заданной эталонной моделью. Примером является оптимальное управление двигателем и разумное поведение робота (повернуться направо или налево, двигаться вперед и т.д.) для того, чтобы достичь цели.

Несмотря на преимущества нейронных сетей в отдельных областях над традиционными вычислениями, существующие нейронные сети не являются совершенными решениями. Они обучаются и могут делать «ошибки». Кроме того, нельзя гарантировать, что разработанная сеть будет оптимальной сетью.

Также необходимо отметить, что тренированная нейронная сеть вычисляет результат по конкретному набору исходных данных, не учитывая динамику и изменение этих данных.

Существует ряд задач, где помимо мгновенного слепка входных данных, требуется анализировать предысторию возникновения такого набора. Для таких задач могут использоваться нейронные сети с обратной связью, например, сети Хопфилда (рис. 1).

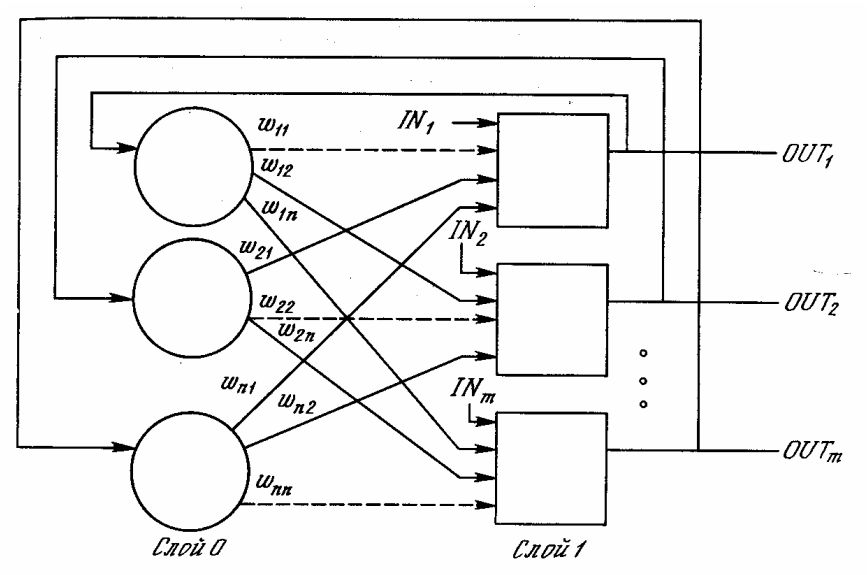


Рис. 1. Сеть Хопфилда

Такие сети имеют пути, передающие сигналы от выходов ко входам. В них отклик является динамическим. После приложения нового входа вычисляется выход, который передаваясь по сети обратной связи, модифицирует вход. Затем выход повторно вычисляется, и процесс повторяется снова и снова. Для устойчивой сети последовательные итерации приводят к все меньшим изменениям выхода, пока, в конце концов, выход не становится постоянным. Для многих сетей процесс никогда не заканчивается. Такие сети называют неустойчивыми.

Эти сети могут обладать памятью, для того чтобы хранить предысторию, но для них трудно предсказать будет ли выход сети устойчивым [1].

Для устранения этого недостатка в настоящей работе для задач, в которых требуется хранить предысторию, предлагается комбинировать нейронные сети и конечные автоматы. Конечные автоматы, благодаря наличию в них памяти, могут сохранять необходимые параметры и свойства для дальнейшего их анализа нейронной сетью. При помощи автоматов выделяются параметры, которые невозможно измерить для одного конкретного входного образа, но которые видны при обзоре всей динамики происходящих процессов. Эти параметры обрабатываются нейронной сетью, которая на их основе принимает решение.

Входной слой нейронной сети связан в таком случае не с конкретными мгновенными характеристиками процессов, происходящих в системе, а с выделенными характеристиками всей последовательности образов, по которой необходимо принимать решение — формировать выходной вектор. Распознавание образов в такой структуре превращается в распознавание динамических образов.

## *1. Постановка задачи*

Типичная нейронная сеть может использоваться для распознавания изображений. Цвет каждой из точек входного изображения передается на нейроны входного слоя нейронной сети. При этом с помощью весовых коэффициентов и функции активации вычисляется вектор выходных сигналов (рис. 2)

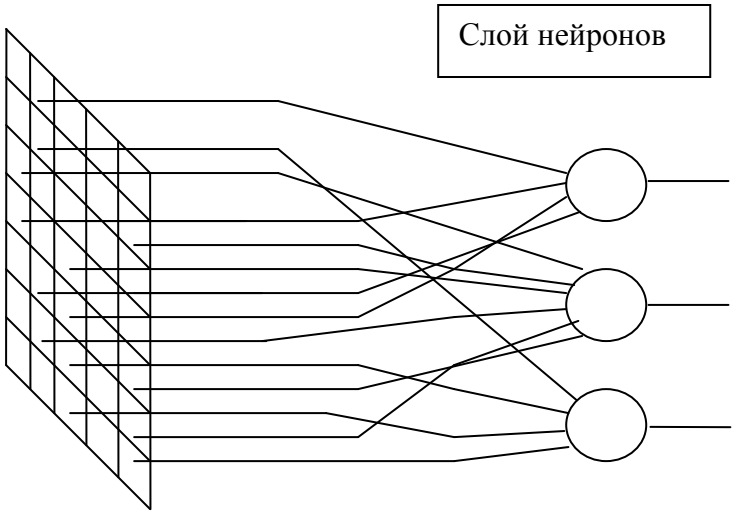


Рис. 2. Нейронная сеть для распознавания изображений

Круг задач, решаемых такой нейронной сетью, ограничен распознаванием статических изображений (без учета предыстории изменений). Нейронная сеть вычисляет результат по конкретному набору исходных данных, не учитывая динамику и изменение этих данных.

В тех задачах, где требуется учитывать динамику процесса, следует применять другие структуры, например, описанную в данной работе комбинацию нейронной сети и набора конечных автоматов.

При помощи автоматов выделяются параметры, которые невозможно измерить для одного конкретного входного образа, но которые видны при обзоре всей динамики происходящих процессов. Эти параметры могут обрабатываться нейронной сетью, которая на их основе может принимать решения.

С каждой из точек входного изображения связан конечный автомат, на который подаются события, определяемые изменениями, происходящими в этой точке. Каждый автомат, в свою очередь, связан с нейронной сетью, на входы которой он отправляет сигналы. Нейронная сеть, распознавая различные конфигурации состояний конечных автоматов, распознает динамические образы (рис.3).

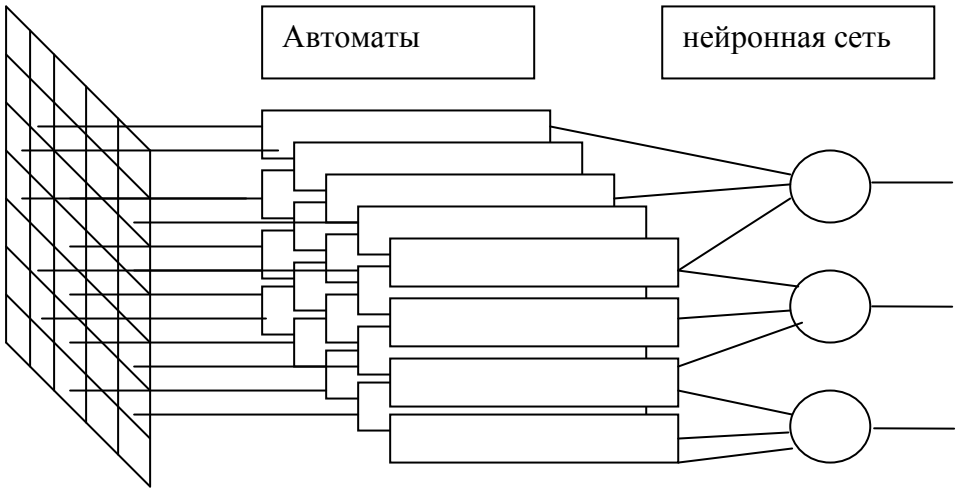


Рис 3. Схематическое изображение взаимодействий

Одной из задач, которые может решать такая структура, является отслеживание движения объекта.

Первым примером, рассмотренным в настоящей работе, является создание детектора движущихся объектов. Детектор движения обнаруживает движущиеся объекты в кадре и определяет направление их движения. В каждом детекторе, наряду с сенсорными, выполняются вычислительные операции.

Такие системы применяются для охраны защищенных объектов и охранной сигнализации, автоматического включения света, для фильтрации видеоинформации, контроля движения транспортных средств. Существующие коммерческие видеодетекторы движения, например, *Bosch VMD01* с технологией определения направления движения

*AutoTrack* [2], аналоговый видеодетектор *VTD MOTION* [3], *VideoIQ* компании *General Electric* – это дорогостоящие продукты. Они поставляются с системой видеокамер и обрабатывающих процессоров, и имеют закрытые алгоритмы работы.

Типичные требования, предъявляемые в видеодетекторам движения:

- возможность настройки детектора на размер движущегося объекта, фиксация перемещений объектов одного типа (например, автомобилей) и игнорирование объектов другого типа (людей, животных);
- возможность работы детектора в реальном времени;
- устойчивая работа в широком диапазоне внешних условий;
- минимальное количество ложных срабатываний на шум регистрирующей и передающей аппаратуры;
- минимальное количество срабатываний на объекты, не представляющие опасности для охраняемого периметра/объекта (качающиеся деревья, листва и т.д.);
- возможность фиксаций как быстрых, так и медленных перемещений объектов.
- работа в плохо освещенных помещениях;
- высокая скорость обновления информации.

Пусть дано клеточное поле некоторой размерности, на котором расположены движущиеся объекты. Требуется построить алгоритм, выделяющий определенный движущийся объект, размеры которого соответствуют предварительным заданным оценкам. При этом фон может быть зашумлен случайными помехами.

Демонстрационным примером, использующим описанную структуру, является разумное управление автомобилем в потоке транспорта.

Имеется выделенный объект (своя машина), над которым доступно управление – изменение скорости и направления движения. Необходимо как можно более долгое время избегать столкновений с другими объектами. Есть возможность наблюдения за другими объектами в некоторой прямоугольной области вокруг своей машины – в машине находится примитивное устройство «радар», которое может определять наличие или отсутствие другого объекта в некоторой части пространства вокруг себя.

В системе координат, связанной со своей машиной, область разбита на клетки. Для каждой из клеток в любой момент времени можно получить булево значение, показывающее находится ли в области, за которую отвечает данная клетка, часть машины (занята ли ней хотя бы половина клетки). На рис. 4 показаны четыре машины и связанные с ними клетки.

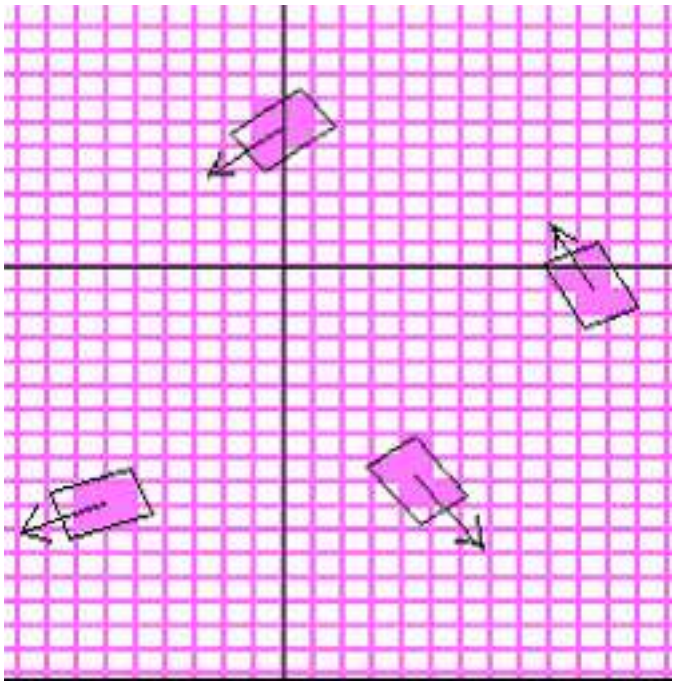

Рис. 4. Машины и связанные с ними клетки

Скорость своей машины можно изменять: ускорять или замедлять в определенных пределах. Также можно с определенной скоростью выполнять поворот. В любой момент можно подать команду изменения скорости или поворота на заданный угол, и эта команда будет исполнена на следующем шаге.

Обработка происходит пошагово. Поэтому все ограничения могут заданы для каждого из тактов.

Скорости всех машинок считаются ограниченными. Временной такт выбирается достаточно коротким. Скорости машин измеряются в ячейках за один такт. Можно считать, что все скорости меньше единицы, с тем, чтобы за один такт перемещение каждого объекта происходило не более чем на одну клетку.

Измеряя скорости движения отдельных машинок, можно с большей точностью предсказывать дальнейшее их движение и анализировать представляемую ими опасность с тем, чтобы избегать столкновений с ними.

Цель этого примера – создать систему автоматов, анализирующих динамику всех клеток, измеряющих скорости и опасности и создающих входной сигнал на нейронную сеть. Создать нейронную сеть, принимающую решение о повороте или изменении скорости.

### *2. Детектор движения*

Программный детектор движения – это программное обеспечение, которое применяется с целью обнаружения движения в потоке видеокадров и трассировки движущихся объектов.

Разработка детектора движения заключается в исследовании и применении методов обработки растровых изображений и обнаружения на них движущихся объектов. Это обеспечивает:

- возможность работы детектора в реальном времени;

- устойчивую работу в широком диапазоне внешних условий;
- минимальное количество ложных срабатываний на шум регистрирующей и передающей аппаратуры;
- минимальное количество срабатываний на объекты, которые не представляют опасности для охраняемого периметра/объекта (качающиеся деревья, листва и т.д.).

Наиболее общим подходом для определения изменений между двумя кадрами изображения (образами) $f(x, y, t_i)$ и $f(x, y, t_j)$, взятыми соответственно в моменты времени $t_i$ и $t_j$, основывается на сравнении соответствующих точек этих двух образов. Для этого применяется процедура, заключающаяся в формировании, так называемой, разности образов [4].

Предположим, что имеется эталонный образ, имеющий только стационарные компоненты. Если сравним этот образ с таким же образом, имеющим движущиеся объекты, то разность двух образов получается в результате вычеркивания стационарных компонент (оставляются только ненулевые записи, которые соответствуют нестационарным компонентам изображения).

Разность между двумя кадрами изображения, взятыми в моменты времени $t_i$ и $t_j$, можно определить следующим образом:

$$d_{ij}(x,y) = \begin{cases} 1, если \mid f(x,y,ti) - f(x,y,tj) \mid > 0 \\ 0, противном\_случае \end{cases},$$

где $\theta$ — значение порогового уровня. $d_{ij}(x, y)$ принимает значение, равное единице, для пространственных координат (x, y) только в том случае, если два образа в точке с этими координатами существенно различаются по интенсивности, что определяется значением порогового уровня $\theta$.

При анализе движущегося образа все пикселы изображений разности $d_{ij}(x, y)$, имеющие значение, равное единице, рассматриваются

как результат движения объекта. Этот подход приметим только в том случае, если два образа зарегистрированы и освещенность имеет относительно постоянную величину в пределах границ, устанавливаемых пороговым уровнем θ. На практике записи в $d_{ij}(x, y)$, имеющие значение, равное единице, часто появляются в результате действия шума. Обычно на разности двух кадров изображения такие значения выглядят как изолированные точки. Для их устранения применяется простой подход, заключающийся в формировании четырех или восьмисвязных областей из единиц в $d_{ij}(x, y)$. Затем пренебрегают любой областью с числом записей, меньшим заранее заданного. При этом можно не распознать малые и/или медленно движущиеся объекты. Однако это увеличивает вероятность того, что остающиеся записи в разности двух кадров изображения действительно соответствуют движению.

Разность кадров из-за шума часто содержит изолированные записи. Несмотря на то, что число таких записей может быть сокращено или полностью ликвидировано в результате анализа связности пороговых уровней, этот процесс может также привести к потере изображений малых или медленно движущихся объектов.

Для достижения связности иногда используется метод согласования блоков [5]. Изображение делится на большое число блоков (обычно — прямоугольных). Каждый блок одного кадра (блок поиска) сравнивается с блоками следующего кадра, и оценивается вероятность того, что они являются фрагментами одного и того же участка изображения. Если в следующем кадре находится блок, подобный блоку поиска первого кадра, то делается вывод, что блоки находятся в разных местах кадра вследствии движения (рис.5).

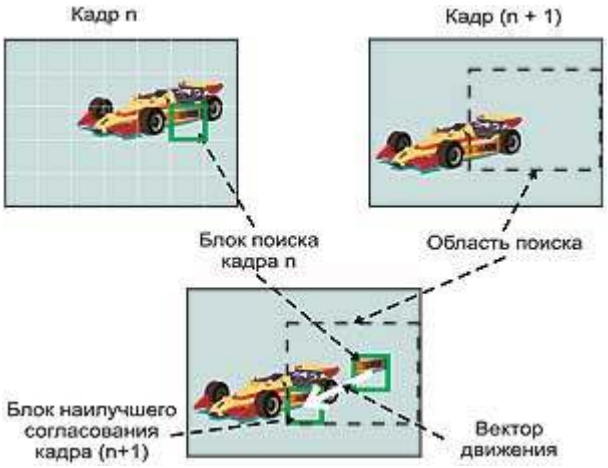


Рис. 5. Иллюстрация метода согласованных блоков

Критерием наилучшего согласования может быть минимизация некоторой меры различий в величине сигналов двух блоков: квадратичное отклонение, отклонение по абсолютной величине и т.п. Затем вычисляются компоненты смещения блока из исходного положения до того, которое наилучшим образом согласуется с блоком поиска, что дает величину и направление вектора движения. Обычно поиск похожих блоков выполняется не в пределах всего изображения, а в рамках некоторой области, окружающей исходный блок, копия которого должна быть найдена. Размеры области и блока поиска сильно влияют на качественные показатели системы оценки векторов движения.

Точность оценки вектора находится на уровне одного элемента, поскольку обнаружение согласования блоков основано на сравнении значений элементов изображения. Размеры области поиска определяют максимальную величину вектора движения, которая может быть измерена, поскольку этот параметр ограничивает максимальную скорость перемещения объекта, которую возможно зафиксировать. Например, область поиска может быть равной квадрату размером 64×64 точки, что является удовлетворительным во многих случаях.

Размеры блока поиска прямо влияют на разрешение и вероятность ложного согласования. Когда блок поиска представляет собой квадрат

16×16 точек, он содержит довольно значительную часть изображения. Если определено, что блоки согласуются – похожи друг на друга, то это с высокой вероятностью говорит о том, что результат оценки действительно отражает смещение объекта. Возможность ложного согласования мала. Однако для каждого блока из 256 точек находится всего один вектор – разрешение невелико. Для малых блоков с размерами 2×2 достигается высокое разрешение, но частота ошибок становится недопустимо большой, поскольку для блоков такого размера велика вероятность ложных согласований. Известно, что в задачах компрессии оптимальным блоком, при котором достигается компромисс между разрешением и частотой ошибок, является квадрат размером 8×8 точек. Однако для целей преобразования стандартов ни разрешение, ни частота ошибок, соответствующие такому блоку, не являются приемлемыми. Это означает, что метод согласования блоков для преобразования стандартов требует доработки.

### *3. Автоматный подход*

В [6] предложена парадигма автоматного программирования, в рамках которой построение и реализация алгоритма выполняется в виде конечного автомата, для которого базовым является понятие «состояние». В указанной работе данный подход был использован применительно к системам логического управления, в которых входные и выходные воздействия являются двоичными переменными.

Предлагается использовать состояния автоматов в качестве входных воздействий на нейронную сеть.

На первом этапе построения искусственной нейронной сети осуществляется отбор входных переменных, влияющих на принятие решения. Классическая тренированная нейронная сеть способна принимать решение, основываясь на наборах входных переменных, и не

учитывает историю их изменений. В свою очередь, система автоматов содержит значимую информацию о динамике процессов в своих состояниях.

Комбинирование автомата и нейронной сети, использующей в качестве входов состояния автоматов, позволяет использовать накопленную автоматами информацию для принятия решения нейронной сетью (рис. 6).

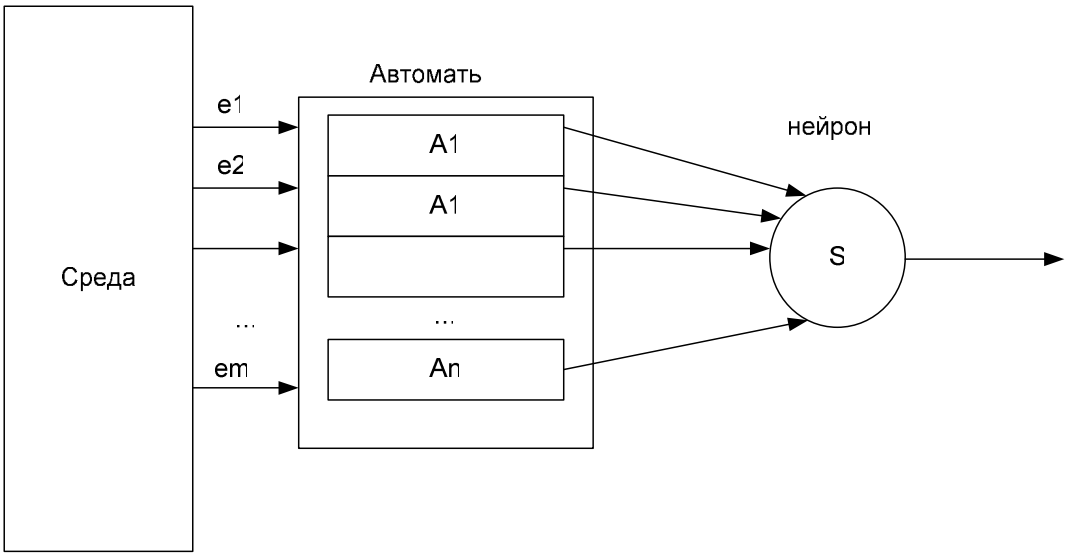


Рис.6. Комбинация автоматов и нейронной сети

### *4. Внешняя среда*

Для задачи построения детектора движений внешним воздействием является изображение, полученное от камеры.

Изображение представлено в виде серых полутоновых оттенков. На каждом шаге на изображение добавляется случайный шум (например, дождь, снег). На заднем плане происходит передвижение некоторого объекта (рис. 6)

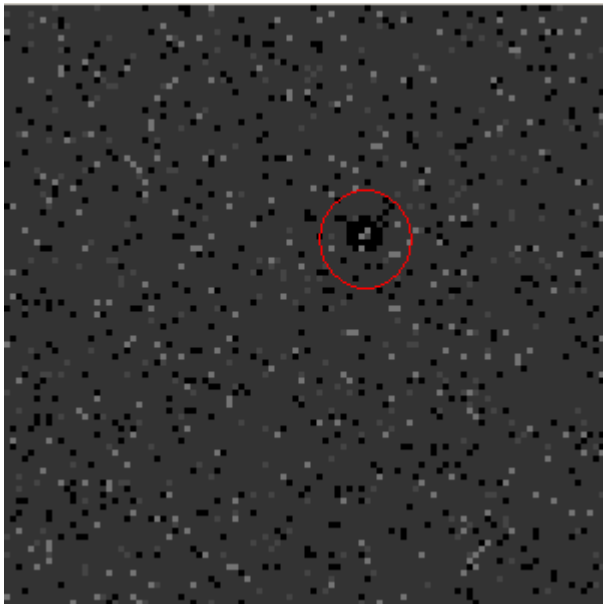

Рис. 6. Входное изображение. Обведен движущийся объект

Необходимо по изменениям точек определять, где находится движущийся объект и направление его движения.

С каждой из точек изображения связан автомат, на который отправляются сообщения об изменениях цвета точки. На этот автомат отправляются события, характеризующие поточечные изменения изображения, полученного с камеры.

## *5. Автомат A1. Состояние клетки*

Каждой точке входного изображения с видеокамеры соответствует свой экземпляр автомата *A1*. Он предназначен для хранения динамики каждой отдельной точки. События на этот автомат отправляются внешней средой. Автомат фиксирует изменения цвета изображения на каждом шаге работы.

### 5.1. Словесное описание

Автомат *A1* предназначен для хранения динамики каждой отдельной точки. В каждой точке находится по экземпляру такого автомата. Выделяются четыре состояния:

0 – над клеткой не находится ничего;

1 – на данном шаге над клеткой появился объект;

2 – объект движется над точкой;

3 – объект только выехал с клетки.

После построения решетки, на каждый из автоматов *A1*, соответствующих каждой из клеток, посылается событие $e_0$ или $e_1$, в зависимости от нового состояния исследуемых клеток.

События $e_0$ или $e_1$ посылаются на автомат внешней средой на каждом такте. При этом

$e_0$ – на текущем шаге в точке произошло изменение цвета;

$e_1$ – на текущем шаге изменения цвета не произошло.

Внутренние переменные:

$x_1$ – текущее значение цвета;

$x_2$ – значение цвета «фона», корректируется в процессе работы.

Воздействие $z_1$ – установить значение «фона» $x_2$, равным текущему значению цвета.

При случайном изменении цвета точки, соответствующий ей автомат переходит в состояние 1. В дальнейшем, если автомат возвращается в предыдущее состояние, по цвету, соответствующему «фону», автомат переходит в состояние 3.

Данный автомат не гарантирует исчезновения случайных помех. По принципу работы он похож на поиск разности кадров, описанный в главе 3, с тем исключением, что он имеет возможность переходить и находиться в состоянии «активно». Это специально выделенное состояние характеризует местонахождение и передвижение объекта в рассматриваемой точке, и тем самым, более точно хранит изменения, связанные с клеткой.

### 5.2. Граф переходов

На рис. 7 приведен граф переходов автомата *A1*.

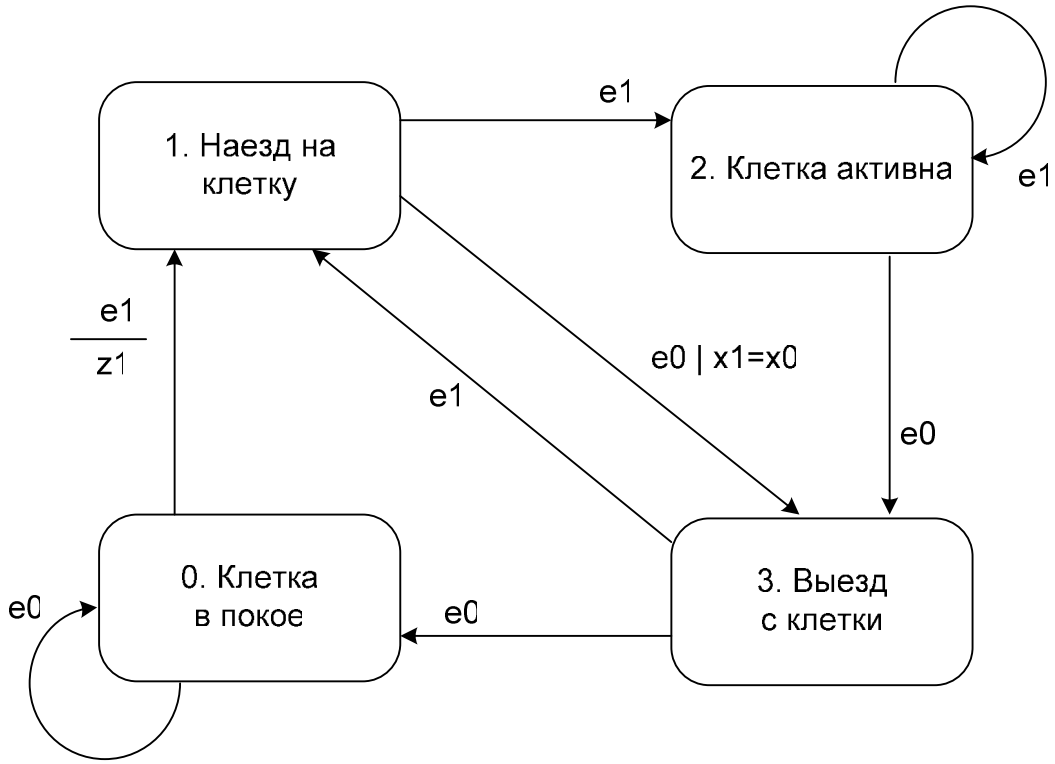


Рис.7. Граф переходов автомата *A1*

## *6. Нейронная сеть детектора движения*

Для определения направления и скорости движения объекта на основе данных от автоматов используется нейронная сеть.

Вычисление скорости происходит только в момент, когда на клетку въезжает движущийся объект, и соответствующий клетке автомат *A1* находится в состоянии 1. Таким образом, рассматривается лишь передний фронт движения. В те моменты, когда объект уже проезжает над точкой, вычисление скорости должно быть связано с определением размеров объекта и других его характеристик (например, раскраски), и, поэтому в них скорость не вычисляется.

Вычисленные скорости по фронту движения суммируются, и вычисляется среднее значение. На рис. 8 показана картина состояний автоматов *A1*.

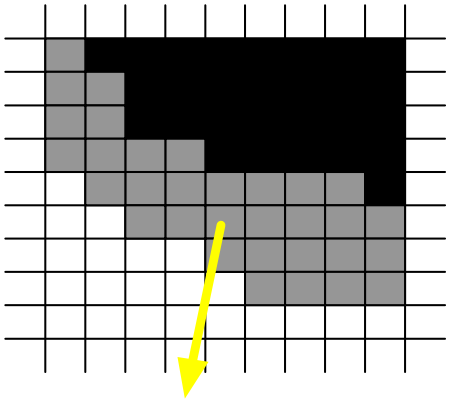

Рис. 8. Состояния окружающих клеток

Черным цветом заполнены клетки, автоматы *A1* которых находятся в состоянии 2 – над которыми находится движущийся объект, и он находился на них на предыдущем шаге. Серым цветом заполнены клетки, автоматы *A1*которых находятся в состоянии 1 – те, на которые произошел наезд движущегося объекта. Более серым цветом показаны пустые клетки в нулевом состоянии.

Связное множество таких клеток, автоматы *A1* которых находятся в состоянии 1, являются передним фронтом движения объекта.

На каждом шаге для каждой клетки фронта выделяется квадрат заданного размера (в демонстрационной программе сторона такого квадрата – семь точек). Из состояний автоматов *A1* этих соседних клеток формируется входной вектор нейронной сети, и активируется процесс вычисления. Нейронная сеть по входному вектору, собранному из состояний соседних точек, вычисляет скорость объекта в данной точке.

Вычисление нейронной сети применяется одинаково ко всем точкам переднего фронта движения объекта. В результате вычисляется набор скоростей.

Следующим шагом работы является вычисление среднего значение скоростей точек по переднему фронту движения. На рис. 9 показан сдвиг шарообразного объекта и вектора скорости, которые вычисляются по фронту его движения.

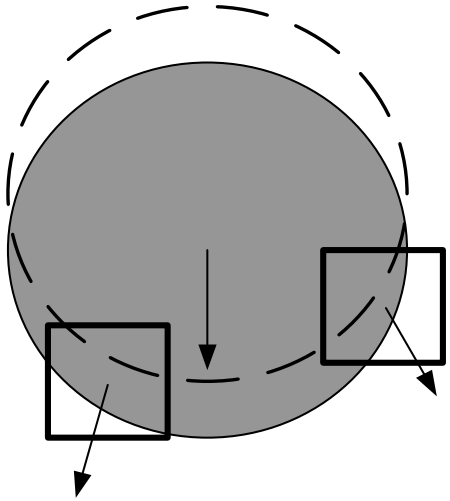

Рис. 9. Сдвиг объекта вниз

Серым цветом обозначено новое местоположение объекта, а пунктиром – местоположение на предыдущем такте работы. Квадратами обведены две точки, автоматы *A1* которых находятся в состоянии 1 – те, на которые произошел наезд объекта.

На основании информации о соседних точках при помощи нейронной сети вычисляются вектора скоростей. Затем вектора скоростей суммируются, и вычисляется средняя скорость, которая далее считается скоростью движения самого объекта.

### 6.1. Конфигурация нейронной сети

Не существует строго определенной процедуры для определения количества нейронов и количества слоев в сети.

Чем больше количество нейронов и слоев, тем шире возможности сети, тем медленнее она обучается и работает и тем более нелинейной может быть зависимость вход-выход.

Количество нейронов и слоев связано:

- со сложностью задачи;
- с количеством данных для обучения;
- с требуемым количеством входов и выходов сети;
- с имеющимися ресурсами: памятью и быстродействием машины, на которой моделируется сеть.

Попытки записать эмпирические формулы для числа слоев и нейронов, оказались малоэффективными, и применимость формул оказалась очень ограниченной.

Если в сети слишком мало нейронов или слоев:

- сеть не обучится, и ошибка при ее работе останется большой;
- на выходе сети не будут передаваться резкие колебания аппроксимируемой функции y(x).

Превышение требуемого количества нейронов тоже мешает работе сети.

Если нейронов или слоев слишком много:

- быстродействие будет низким, а памяти потребуется много — на фон-неймановских ЭВМ;
- сеть переобучится: выходной вектор будет передавать незначительные и несущественные детали в изучаемой зависимости y(x), например, шум или ошибочные данные;
- зависимость выхода от входа окажется резко нелинейной: выходной вектор будет существенно и непредсказуемо изменяться при малом изменении входного вектора x;
- сеть будет неспособна к обобщению: в области, где нет или мало известных точек функции y(x), выходной вектор будет случаен и непредсказуем. Ответ сети (вектор) не будет адекватен решаемой задаче.

В рассматриваемой задаче для определения скорости движения объекта в каждой точке для избежания лишнего влияния удаленных точек и для уменьшения быстродействия, рассматриваются только ближние соседи точки в квадрате 7x7.

Для вычисления скорости рассматриваются только те точки, на которые на текущем шаге, согласно состоянию автомата *A1*, был совершен наезд движущегося объекта. Таким образом, если объект проходит прямо над точкой (рис. 10), и скорость движущегося объекта больше стороны рассматриваемого квадрата, то в пределах этого квадрата невозможно определить величину скорости движущегося объекта.

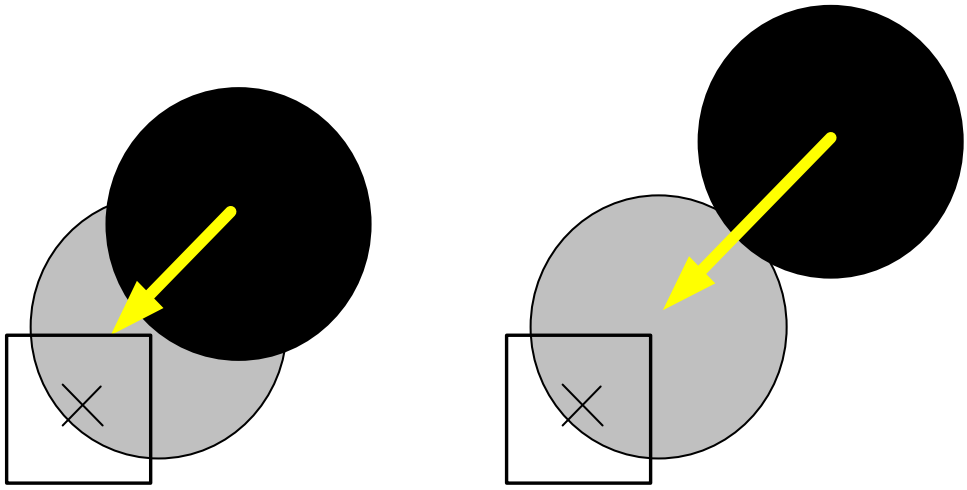

Рис 10. Два случая быстродвижущихся объектов, воспринимаемые в пределах квадрата как одинаковые

Таким образом, получается верхняя оценка определяемой скорости. В дальнейшем эталонные значения скорости будут также нормироваться по ней, для того чтобы они получались в пределах (-1; 1).

Входной вектор нейронной сети формируется из состояний автоматов соседних точек внутри рассматриваемого квадрата. Важны только состояния 1 и 2 – фронт и точки, над которыми объект продолжает двигаться. Входы нейронной сети будем считать принимающими значения 0 или 1. Для каждой соседней точки выделим по два входа нейронной сети.

Первый вход равен 1, если автомат соответствующей точки находится в состоянии «1», и равен 0 – в противном случае. Второй вход равен 1, если ее автомат находится в состоянии «2», и 0 – в противном случае (рис. 11).

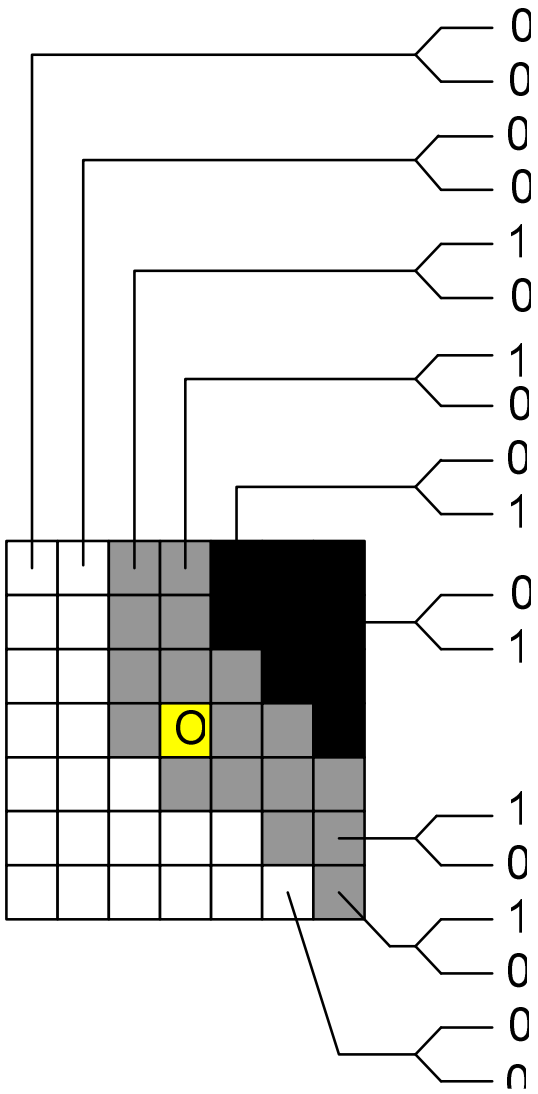


Рис. 11. Выходы с автоматов на нейронную сеть

На выходе нейронной сети формируются два значения – вертикальная и горизонтальная составляющая скорости объекта. Принимаемые значения должны находиться в интервале ( -1; 1 ).

В соответствии с таким требованием принята сигмоидальная активационная функция гиперболический тангенс y = th(s) (рис. 12)

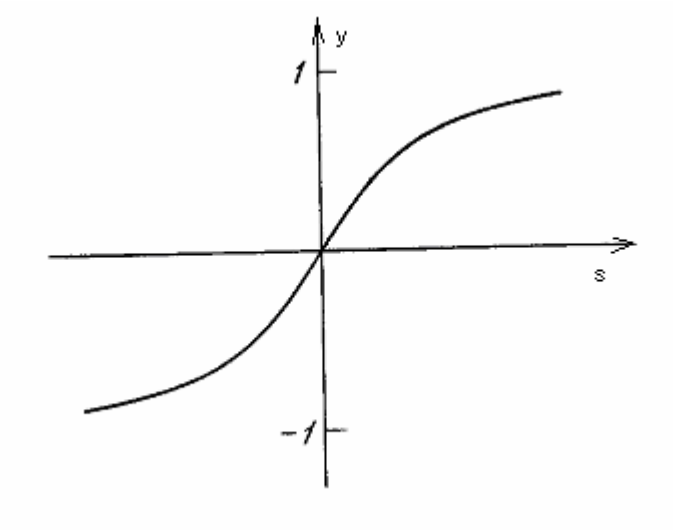


Рис. 12. Активационная функция

В соответствии с рекомендациями, приведенными в [9], выбрана структура нейронной сети с одним промежуточным слоем и двумя выходами (рис. 13).

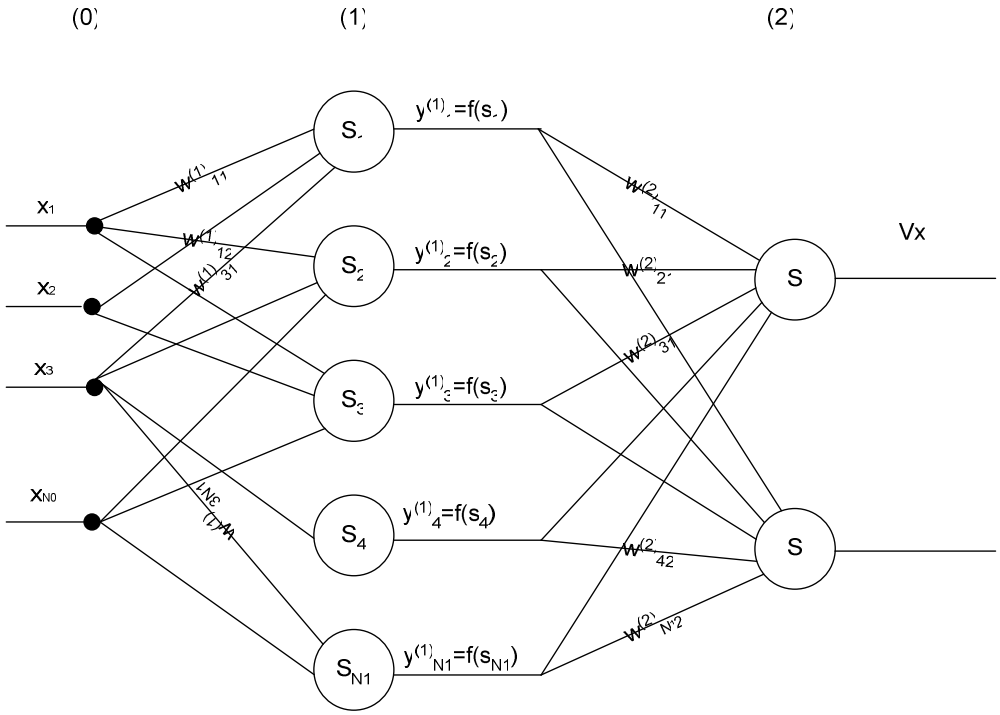


Рис. 13. Структура нейронной сети

Количество нейронов в промежуточном слое выбирается равным удвоенному количеству входов.

Примеры в обучающей выборке делятся на две группы: обучающая и контрольная (для того чтобы избежать переобучения). При

переобучении увеличивается выборка или увеличивается количество нейронов в промежуточном слое. Прямое нахождение числа нейронов в скрытом слое является задачей большой трудоемкости.

Обозначим $w_{ij}^{(n)}$ весовой коэффициент *j*-го нейрона *n*-го слоя для связи с *i*-м нейроном *(n-1)*-го слоя;

$y_p^{(N)}$ – выход N-го выходного слоя сети для p-го нейрона;

$x_p$ – входной слой нейронной сети.

$N_0$, $N_1$, $N_2$ – количество нейронов нулевого, первого и второго слоев, $N_2 = 2$, $N_0 = 2*7^2$ – количество входов.

При этом

$$s_p^{(1)} = \sum_{i=1}^{N0} x_i w_{ip}^{(1)}$$

$y_p^{(1)} = th(s_p^{(1)})$ – выходные значения первого слоя; (1)

$$s_p^{(2)} = \sum_{i=1}^{N0} y_i^{(1)} w_{ip}^{(2)}$$

$y_p^{(2)} = th(s_p^{(2)})$ – выходные значения второго слоя; (2)

Здесь $y^{(2)}_1$ и $y^{(2)}_2$ являются значениями горизонтальной и вертикальной проекций скорости и выходными значениями второго слоя и всей нейронной сети.

### 6.2. Обучение нейронной сети

Существует три вида обучения.

**Обучение с учителем.** При этом сети предъявляется набор обучающих примеров. Каждый обучающий пример представляют собой пару: вектор входных значений и желаемый выход сети. В ходе обучения весовые коэффициенты подбираются таким образом, чтобы по этим входам давать выходы, максимально близкие к правильным.

**Обучение с поощрением.** При этом сети не указывается точное значение желаемого выхода, однако, ей выставляется оценка, определяющая хорошо она поработала или плохо.

**Обучение без учителя.** Сети предъявляются некоторые входные векторы и в ходе их обработки в ней происходят некоторые процессы самоорганизации, приводящие к тому, что сеть становится способной решать какую-то задачу.

В рассматриваемой нейронной сети можно создать ряд обучающих примеров с известной скоростью, и на них можно подбирать весовые коэффициенты таким образом, чтобы нейронная сеть давала по результатам работы максимально близкие значения к эталонным.

Для этого создается клеточное поле без помех. В центре поля выделяется клетка, для которой высчитываются скорости (рис. 14).

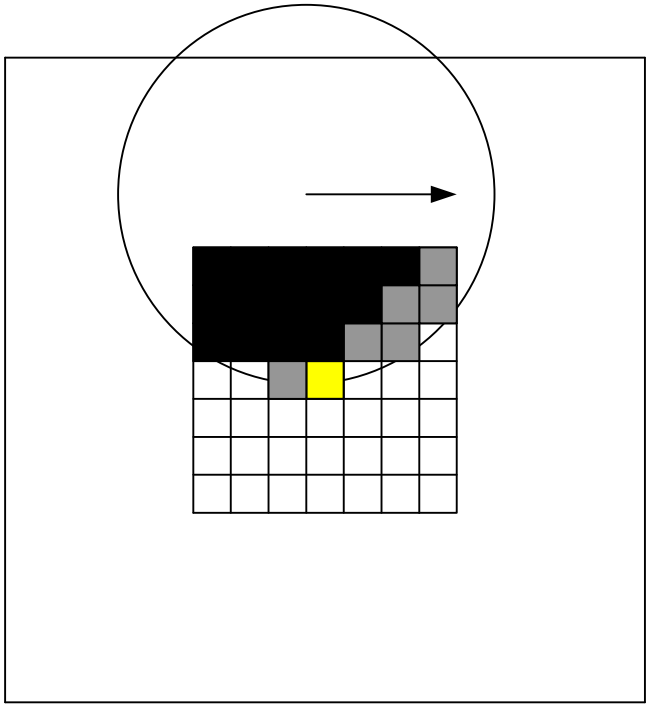

Рис. 14. Пробные запуски объектов

В разных направлениях моделируется запуск шарообразных объектов до момента их прохождения над выделенной клеткой. В этот момент нейронная сеть вычисляет значение скорости, и оно сравнивается

с реальным. В зависимости от величины разности между реальным и вычисленным значениями, корректируются весовые коэффициенты нейронной сети.

На пробных запусках объектов варьируются следующие величины:

- размеры объекта;
- начальная точка;
- направление движения объекта;
- скорость движения объекта.

Тем самым получаем ряд примеров, которые делятся на две группы. На первой группе примеров происходит обучение нейронной сети, на второй группе происходит ее тестирование. Размеры групп выбираются примерно равными. Время настройки сети производится таким образом, чтобы не происходило переобучения сети, и она давала примерно равные результаты на тренировочном и тестовом наборах.

Если результаты признаются неудовлетворительными, то происходит увеличение размеров промежуточного слоя и увеличение тестового набора (дополнительные пробные запуски).

Для настройки нейронной сети используется алгоритм обратного распространения. Это один из вариантов обучения с учителем. Первоначальные значения весовых коэффициентов устанавливаются случайными. Строится обучающее множество, состоящее из пар вход сети – желаемый выход $\{X, D\}$. Через $Y$ обозначается реальное выходное значение нашей сети, которое в начале практически случайно из-за случайности весовых коэффициентов.

Обучение состоит в том, чтобы подобрать весовые коэффициенты таким образом, чтобы минимизировать некоторую целевую функцию. В качестве целевой функции рассмотрим сумму квадратов ошибок сети на примерах из обучающего множества.

$$E(w) = \sum_{j,p} (y_{j,p}^{(N)} - d_{j,p})^2 , \qquad (3)$$

где $y_{j,p}^{(N)}$ реальный выход $N$-го выходного слоя сети для $p$-го нейрона на $j$-м обучающем примере, $d_{j,p}$ желаемый выход. При этом минимизировав такой функционал, получается решение по методу наименьших квадратов.

Поскольку весовые коэффициенты в зависимость $y_{j,p}^{(N)}(x)$ входят нелинейно, воспользуемся для нахождения минимума методом наискорейшего спуска. При этом на каждом шаге обучения производится изменение весовых коэффициентов по формуле

$$\Delta w_{ij}^{(n)} = -\eta \cdot \frac{\partial E}{\partial w_{ij}^{(n)}} , \qquad (4)$$

где $w_{ij}^{(n)}$ весовой коэффициент $j$-го нейрона $n$-го слоя для связи с $i$-м нейроном *(n-1)*-го слоя. Параметр $\eta$ называется параметром скорости обучения.

Таким образом, требуется определить частные производные целевой функции $E$ по всем весовым коэффициентам сети. Согласно правилам дифференцирования сложной функции

$$\frac{\partial E}{\partial w_{ij}^{(n)}} = \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{dy_j^{(n)}}{ds_j^{(n)}} \cdot \frac{\partial s_j^{(n)}}{\partial w_{ij}^{(n)}} , \qquad (5)$$

где $y_j^{(n)}$ – выход, а $s_j^{(n)}$ – взвешенная сума входов $j$-го нейрона $n$-го слоя. Можно вычислить $\frac{dy_j^{(n)}}{ds_j^{(n)}}$ . Для гиперболического тангенса эта величина будет равняться

$$\frac{dy_j^{(n)}}{ds_j^{(n)}} = 1 - th^2 s = 1 - y_j^{(n)2} . \qquad (6)$$

Третий сомножитель $\partial s_j^{(n)} / \partial w_{ij}^{(n)}$ – это выход $i$-го нейрона *(n-1)*-го слоя:

$$\frac{\partial s_j^{(n)}}{\partial w_{ij}^{(n)}} = y_i^{(n-1)}. \qquad (7)$$

Частные производные целевой функции по весам нейронов выходного слоя теперь можно вычислить. Производя дифференцирование (6) по $y_j^{(N)}$ и учитывая (5) и (7), получим:

$$\frac{\partial E}{\partial w_{ij}^{(N)}} = (y_j^{(N)} - d_j) \cdot \frac{dy_j^{(N)}}{ds_j^{(N)}} \cdot y_i^{(N-1)}. \qquad (8)$$

Введем обозначение

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{dy_j^{(n)}}{ds_j^{(n)}}. \qquad (9)$$

Тогда для нейронов выходного слоя справедливо соотношение:

$$\delta_j^{(N)} = (y_j^{(N)} - d_j) \cdot \frac{dy_j^{(n)}}{ds_j^{(n)}}. \qquad (10)$$

Для весовых коэффициентов нейронов внутренних слоев невозможно сразу записать, чему равен первый сомножитель из (9). Однако его можно представить следующим образом:

$$\frac{\partial E}{\partial y_j^{(n)}} = \sum_k \frac{\partial E}{\partial y_k^{(n+1)}} \cdot \frac{dy_k^{(n+1)}}{ds_k^{(n+1)}} \cdot \frac{\partial s_k^{(n+1)}}{\partial y_j^{(n)}} = \sum_k \frac{\partial E}{\partial y_k^{(n+1)}} \cdot \frac{dy_k^{(n+1)}}{ds_k^{(n+1)}} \cdot w_{jk}^{(n+1)}. \qquad (11)$$

В этой формуле первые два сомножителя есть не что иное, как $\delta_k^{(n+1)}$. Таким образом, с помощью (11) можно выражать величины $\delta_j^{(n)}$ для нейронов *n*-го слоя через $\delta_k^{(n+1)}$ для нейронов *(n+1)*-го. Поскольку для последнего слоя $\delta_j^{(N)}$ вычисляется по (10), то остальные значения вычисляются с помощью рекурсивной формулы

$$\delta_j^{(n)} = \left[ \sum_k \delta_k^{(n+1)} \cdot w_{jk}^{(n+1)} \right] \cdot \frac{dy_j}{ds_j}. \qquad (12)$$

Таким образом, получаются значения $\delta_j^{(n)}$ для всех нейронов всех слоев.

Окончательно формула (4) для модификации весовых коэффициентов может быть записана в виде:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot y_i^{(n-1)}. \qquad (13)$$

Полный алгоритм обучения нейронной сети с помощью алгоритма обратного распространения строиться следующим образом.

Всем весовым коэффициентам сети присваиваются случайные начальные значения. При этом сеть осуществляет какое-то случайное преобразование входных сигналов, и значения целевой функции (3) будут велики.

На вход сети подается один из входных векторов из обучающего множества. Вычисляются выходные значения сети, запоминая при этом выходные значения каждого из нейронов.

Рассчитываются по формуле (10) значения $\delta_j^{(N)}$. Затем с помощью рекурсивной формулы (12) подсчитываются все остальные $\delta_j^{(n)}$, и, наконец, с помощью (13) – изменение весовых коэффициентов сети.

Корректируются веса сети:

$$w_{ij}^{(n)} = w_{ij}^{(n)} + \Delta w_{ij}^{(n)}. \qquad (14)$$

Рассчитывается целевая функция (3). Если она удовлетворительно мала, то считаем сеть успешно обучившейся. Иначе возвращаемся на шаг 2.

### 6.3. Вычисление скорости объекта

После вычисления при помощи нейронной сети скорости во всех точках переднего фронта движения объекта, производится вычисление среднего арифметического этих скоростей (рис. 15)

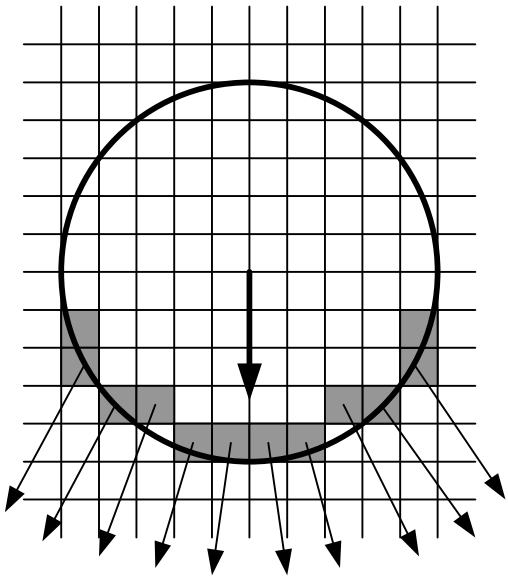

Рис. 15. Скорости в точках переднего фронта движения

Сложность всех вычислений зависит от количества точек входного изображения линейно O(N).

Переход автомата происходит за O(1).

Вычисление в нейронной сети производится за $N_0*N_1+N_1*N_2$ шагов, где $N_0$, $N_1$, $N_2$ – количества нейронов соответствующих слоев нейронной сети. Эти значения в рассматриваемом примере являются фиксированными. Вычисления производятся лишь для точек переднего фронта движения объекта, а их количество зависит от степени динамичности сцены и размеров движущихся объектов.

## *7. Результат работы детектора движения*

На основе глав 4–6 построена программа, детектирования движения объектов, которая приложена к документации.

Построена и натренирована нейронная сеть, определяющая движения в пределах квадрата 7x7 точек.

Вычислено значение $E = \frac{\sum_{j,p}(y_{j,p}^{(N)} - d_{j,p})^2}{N}$, равное суммарному по всем примерам квадратичному отклонению, деленному на количество примеров.

Построено 2084 тестовых примеров.

Среднее квадратичное отклонение на тренировочной выборке: 0.0181002.

Среднее квадратичное отклонение на тестовой выборке: 0.037955 (скорость измеряется в пределах от -1 до 1).

На рис. 15 показаны два последовательных шага работы демонстрационной программы.

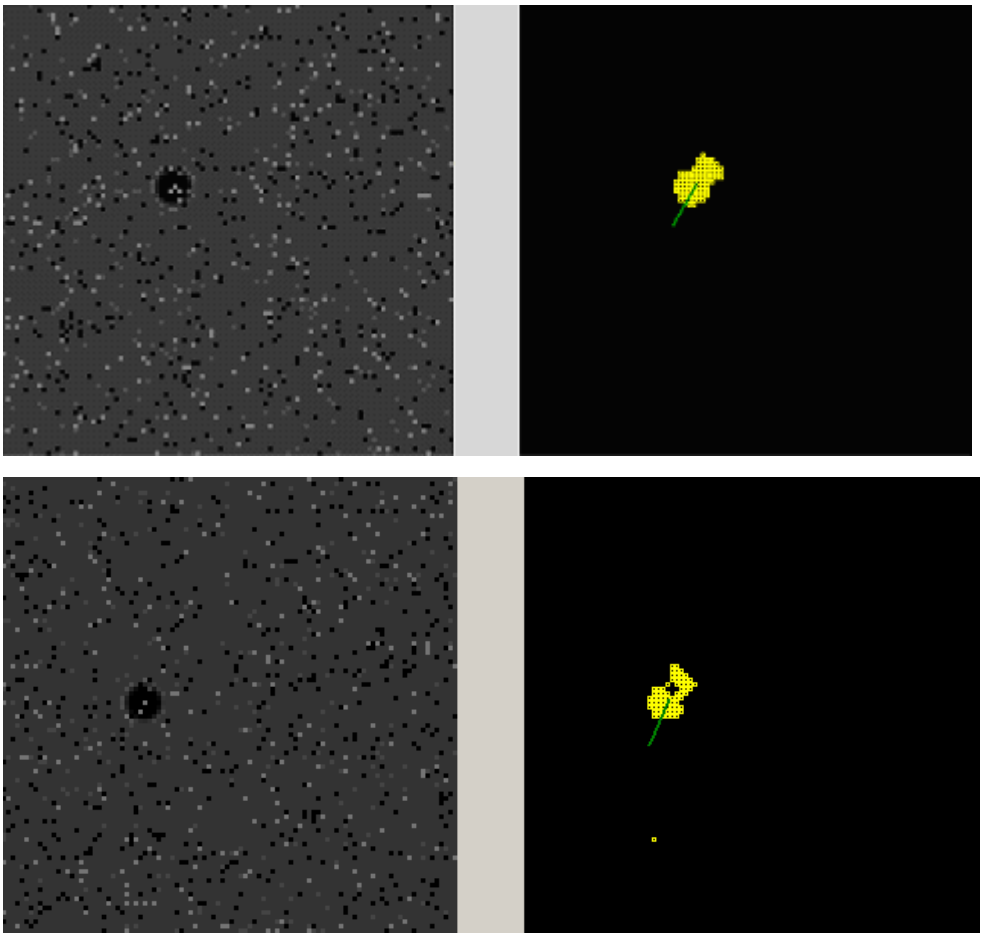

Рис.15. Шаг работы детектора движения

В демонстрационной программе по зашумленному полю происходит движение объекта. При этом улавливаются контуры и направление движения объекта.

На поле находится изменяющий размеры, скорость и направление объект. Через равные промежутки времени производится замер его скорости V и сравнивается с реальным значением d. Считается среднее квадратичное отклонение ошибки как

$$E = \frac{\sum (v_x - d_x)^2 + (v_y - d_y)^2}{N}.$$

На выборке из 250 замеров, среднее квадратичное отклонение составило 0.0203245961.

Ниже приведен пример анализа движущегося автомобиля с реальными кадрами, полученными с видеокамеры и обработанными с помощью предложенного подхода (рис. 16).

Рис. 16. Пример работы детектор движений

Из рассмотрения рис. 16 следует что предложенная система не уступает по функциональности коммерческим аналогам.

## *8. Демонстрационный пример работающей технологии*

Демонстрационным примером, использующим описанную структуру, является разумное управление автомобилем в потоке транспорта.

Имеется выделенный объект (своя машина), над которым доступно управление – изменение скорости и направления движения. Необходимо как можно более долгое время избегать столкновений с другими объектами. Есть возможность наблюдения за другими объектами в некоторой прямоугольной области вокруг своей машины – в машине находится примитивное устройство «радар», которое может определять наличие или отсутствие другого объекта в некоторой части пространства вокруг себя.

В системе координат, связанной со своей машиной, область разбита на клетки. Для каждой из клеток в любой момент времени можно получить булево значение, показывающее, находится ли в области, за которую отвечает данная клетка, часть машины (занята ли ней хотя бы половина клетки). На рис. 17 показаны четыре машинки и связанные с ними клетки.

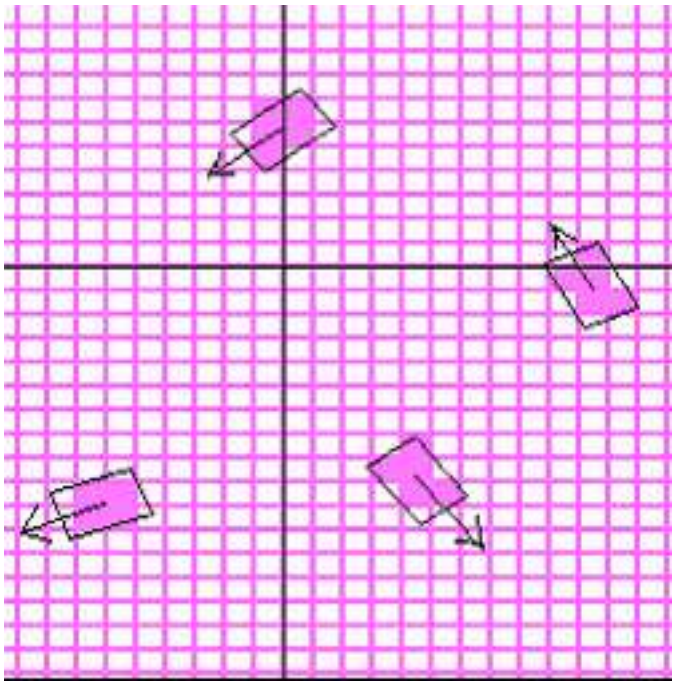

Рис. 17. Машины и связанные с ними клетки

Скорость своей машины можно изменять: ускорять или замедлять в определенных пределах. Также можно с определенной скоростью выполнять поворот. В любой момент можно подать команду изменения скорости или поворота на заданный угол, и эта команда будет исполнена на следующем шаге.

Обработка информации происходит пошагово. Поэтому все ограничения могут заданы для каждого из тактов.

Скорости всех машинок считаются ограничены. Временной такт выбирается достаточно коротким. Скорости машин измеряются в ячейках за один такт, и можно считать, что все скорости меньше единицы, с тем, чтобы за один такт перемещение каждого объекта происходило не более чем на одну клетку.

Измеряя скорости движения отдельных машинок, можно с большей точностью предсказывать дальнейшее их движение и анализировать представляемую ими опасность с тем, чтобы избегать столкновений с ними.

## *9. Внешняя среда для машинок*

Внешняя среда представляет собой поле, на котором расположены прямоугольники – машины. Они могут двигаться или оставаться на месте. Отдельным цветом выделена своя машина. При необходимости слежения за траекториями отдельных машин можно задавать им возможность оставлять след по мере движения. Поле представляет собой тор, когда движущийся объект выезжает за пределы области, он появляется с противоположной ее стороны. Машины представлены в демонстрационной программе объектом *Car*, свойствами которого являются матрица преобразований координат, скорость, размеры.

Изменяя свойства отдельных машин, например, искусственно увеличивая их длину и устанавливая нулевую скорость, можно организовать появление таких объектов, как стенки или препятствия. На каждом шаге происходит изменение координат движущихся объектов в соответствии с их направлениями движений и скоростями.

На рис.18 изображены одиннадцать машинок (одна из которых является «своей»), каждая из которых движется в своем направлении.

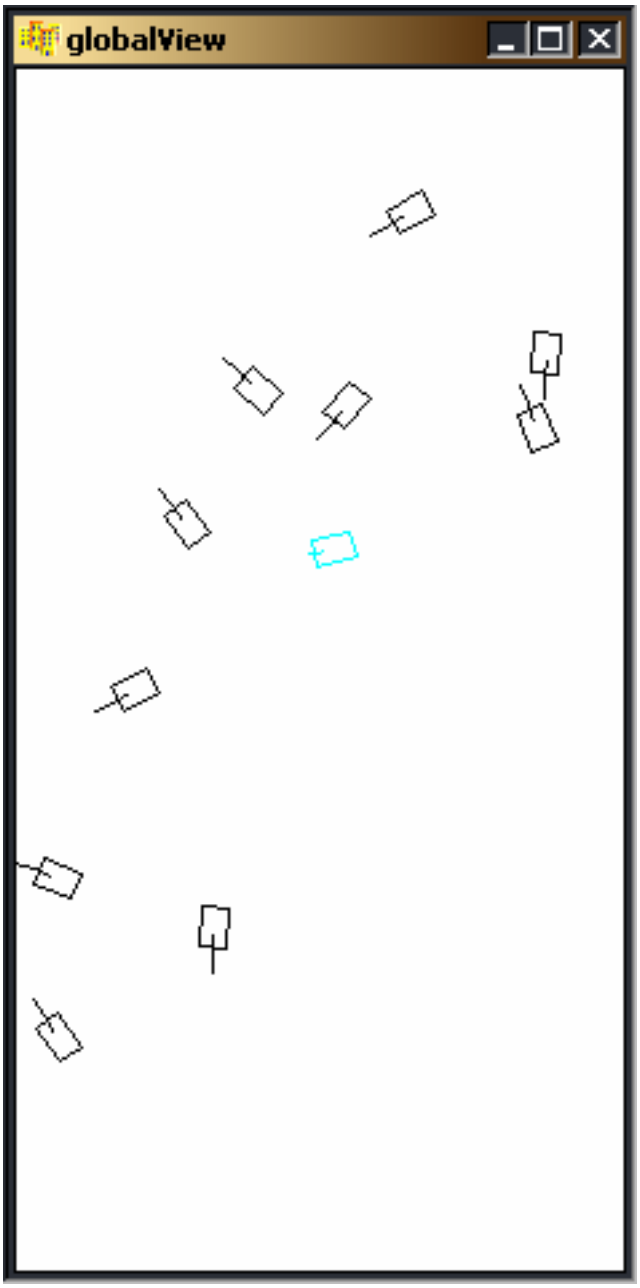


Рис.18. Вид машин во внешней среде

### *10. Локальная решетка*

После перемещения всех объектов на очередном шаге, их координаты переводятся в локальные координаты выделенного объекта. В этих координатах своя машина расположена в точке (0; 0), соответственно, приближение любой другой машины к началу координат является нежелательным и потенциально опасным (угроза столкновения).

Для каждой из клеток рассчитывается, какая часть машины расположена в ее пределах, и если это значение занимает больше половины клетки, соответствующее значение этой клетки устанавливается равным «1». В противном случае, клетка помечается значением «0» как пустая.

Для вычисления площади фигуры, образованной пересечением квадратной клетки и фигуры, которую образует машина, в клетку равномерно заносятся N точек. После этого определяется, какая часть всех точек находится внутри фигуры, из которой составлена машина. Этот процесс повторяется для каждой из клеток, на которых может находиться исследуемый объект – на квадрате от минимальных до максимальных координат.

Используемый размер решетки в демонстрационной программе – 40 на 80 клеток.

## *11. Автомат A1. Состояние клетки*

### 11.1. Словесное описание

Автомат *A1* предназначен для хранения динамики каждой отдельной клетки. Автомат может находиться в одном из четырех состояний:

0 – над клеткой не находится ничего;

1 – на данном шаге над клеткой появился объект;

2 – объект находится над клеткой более одного шага;

3 – объект только что выехал с клетки.

После построения решетки, на каждый из автоматов *A1*, соответствующих каждой из клеток, посылается событие $e_0$ или $e_1$, в зависимости от нового состояния исследуемых клеток.

События $e_0$ или $e_1$ посылаются на автомат внешней средой на каждом такте;

$e_0$ – на текущем шаге над клеткой не находится ничего;

$e_1$ – на текущем шаге над клеткой находится объект.

### 11.2. Граф переходов

Граф переходов автомата приведен на рис. 19.

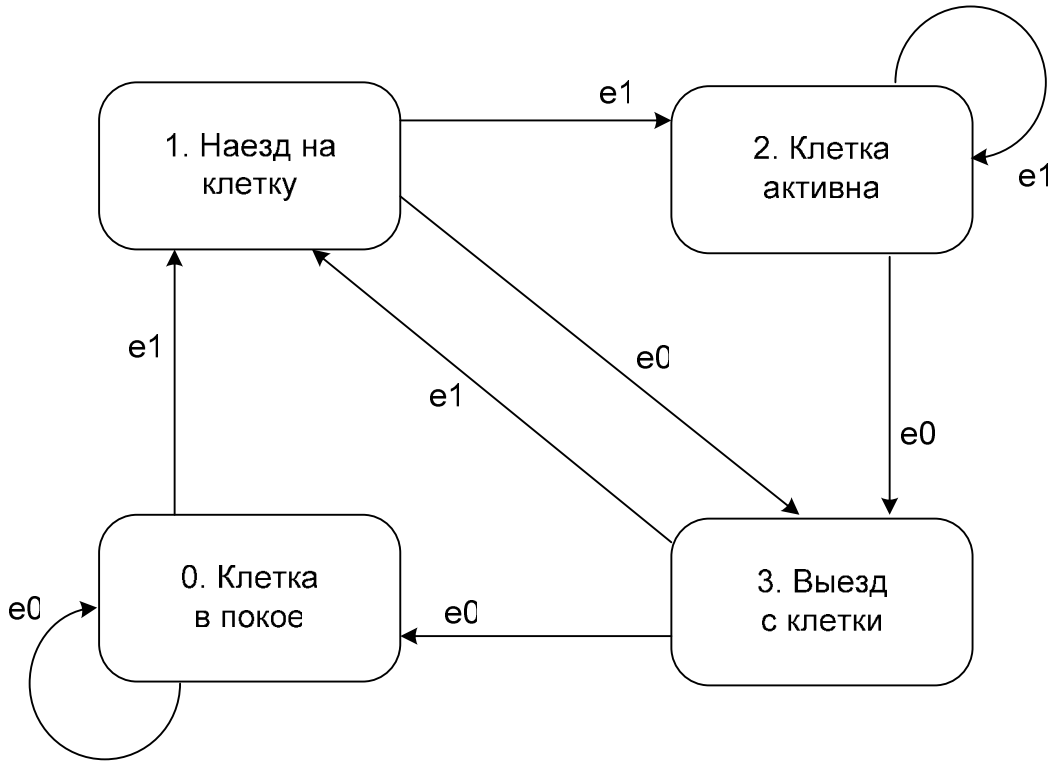


Рис.19. Граф переходов автомата *A1*

## *12. Автомат для измерения скорости*

### 12.1. Словесное описание

Для того, чтобы измерять скорость прямолинейно движущегося объекта, было решено применить автомат, так как объекты, проецирующиеся на решетку имеют дискретные сдвиги на каждом такте работы. Эти сдвиги могут соответствовать входным воздействиям

автомата. Требуется измерять именно мгновенную, а не среднюю скорость, так как именно мгновенная является значимой для определения угрозы, которую несет движущийся объект.

Допустим точка движется прямолинейно в одном направлении со скоростью 0 <= V <= 1.

При дискретизации значения пройденного ей расстояния, за каждую единицу времени точка может перейти в соседнюю клетку, либо остаться в текущей. В соответствии с этим генерируются сигналы $e_1$ или $e_0$:

$e_0$ – точка осталась в пределах текущей ячейки;

$e_1$ – точка сдвинулась в соседнюю ячейку, целочисленная составляющая координаты увеличилась.

Таким образом, прямолинейное движение в одном направлении может быть описано последовательностью этих событий.

Примеры:

$e_1\ e_1\ e_0\ e_1\ e_1\ e_0\ e_1\ e_1\ e_0\ \ldots$ – скорость V = 2/3;

$e_0\ e_0\ e_0\ e_1\ e_0\ e_0\ e_1\ \ldots$ – скорость V = 2/7.

Построим автомат для измерения мгновенной скорости по последовательности воздействий e0 и e1. События e1 и e0 генерируются с постоянным временным интервалом.

Состояния рассматриваемого автомата соответствуют определенным скоростям движения:

0 – скорость равна нулю;

1..6 – скорость равна 1/6;

7..11 – скорость равна 1/5;

12..15 – скорость равна 1/4;

16..18 – скорость равна 1/3;

19..20 – скорость равна 1/2;

21..23 – скорость равна 2/3;

24..27 – скорость равна 3/4;

28..32 – скорость равна 4/5.

Если скорость объекта находится между этими значениями, то состояние автомата будет изменяться между ближайшими к нему значениями. В таблице значения расстояния и состояний для V=0.4.

Таблица

| Номер | Расстояние | Целая часть | Событие | Состояние | Скорость |
|---|---|---|---|---|---|
| 0 | 0.4 | 0 | e0 | 0 | 0 |
| 1 | 0.8 | 0 | e0 | 0 | 0 |
| 2 | 1.2 | 1 | e1 | 6 | 0.16666 |
| 3 | 1.6 | 1 | e0 | 5 | 0.16666 |
| 4 | 2 | 2 | e1 | 11 | 0.2 |
| 5 | 2.4 | 2 | e0 | 10 | 0.2 |
| 6 | 2.8 | 2 | e0 | 9 | 0.2 |
| 7 | 3.2 | 3 | e1 | 15 | 0.25 |
| 8 | 3.6 | 3 | e0 | 14 | 0.25 |
| 9 | 4 | 4 | e1 | 18 | 0.3333 |
| 10 | 4.4 | 4 | e0 | 17 | 0.3333 |
| 11 | 4.8 | 4 | e0 | 16 | 0.3333 |
| 12 | 5.2 | 5 | e1 | 20 | 0.5 |
| 13 | 5.6 | 5 | e0 | 19 | 0.5 |
| 14 | 6 | 6 | e1 | 20 | 0.5 |
| 15 | 6.4 | 6 | e0 | 19 | 0.5 |
| 16 | 6.8 | 6 | e0 | 16 | 0.3333 |

В рассматриваемом примере автомат будет по очереди принимать значения 1/3 и 1/2, так как эти значения являются близкими к 0.4. Автомат

принимает эти значения спустя девять тактов после начала движения на расстоянии трех клеток от начала.

При резком изменении скорости такой автомат изменит свое состояние в соответствии с новой скоростью.

Для использования автомата необходимо на каждой единицы времени посылать один из сигналов $e_0$ или $e_1$.

Минимальное значение скорости 1/6. Если автомат движется с меньшей скоростью, он будет находиться в состоянии 1. Переход в состояние 0 не производится в связи с тем, что даже очень медленно движущийся объект представляет собой угрозу, которая может быть не замечена, если автомат будет переводиться в состояние 0.

Для измерения скорости движения на плоскости используются **четыре таких автомата**, по одному на каждое из направлений:

*A10* – автомат, измеряющий скорость вверх;

*A11* – автомат, измеряющий скорость вправо;

*A12* – автомат, измеряющий скорость вниз;

*A13* – автомат, измеряющий скорость влево.

Один из автоматов *A10* и *A12* всегда находится в состоянии 0, также как один из автоматов *A11* и *A13*, так как направления, за которые отвечают эти автоматы, противоположны. Для принудительного перевода автомата для измерения скорости в состояние 0 используется специальное событие $e_{-1}$. Таким образом, каждая клетка хранит по одному экземпляру автомата *A1* и по четыре экземпляра автомата для измерения скорости.

Дополнительное событие $e_{-1}$, которое переводит автомат на значение скорости 0, используется для гарантированного перевода автомата, измеряющего скорость, в нулевое состояние, в случае движения в направлении противоположному тому, за которое он отвечает.

Например, при резком изменении направления движения справа налево, событие $e_{-1}$ будет подано на автомат *A11* для того, чтобы перевести его в состояние 0. При дальнейшем движении влево за измерение горизонтальной составляющей скорости будет отвечать автомат *A13*.

## 12.2. Граф переходов

Граф переходов автомата приведен на рис. 20.

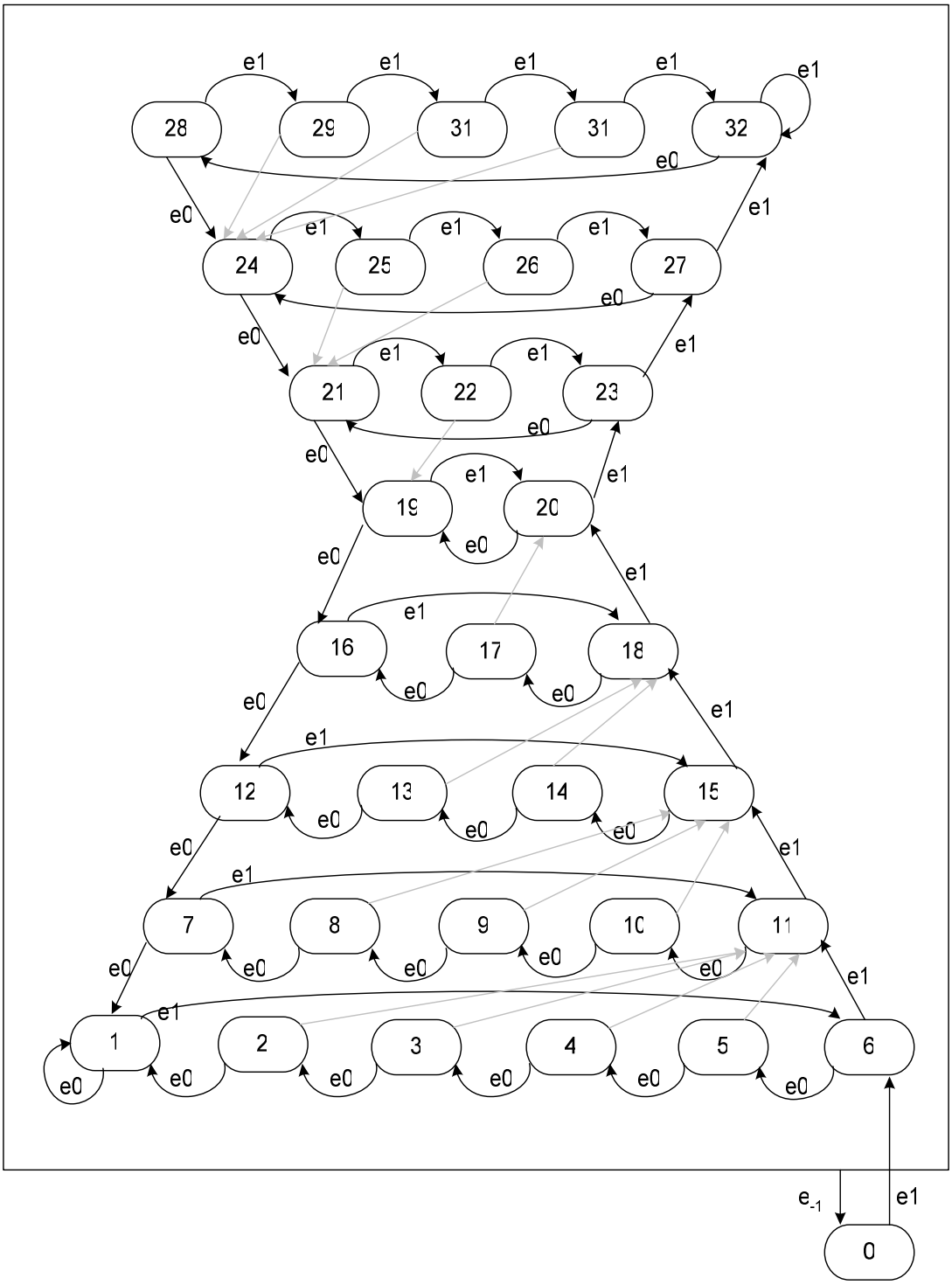


Рис.20. Граф переходов автомата измерения скорости

## *13. Автомат A2*

### 13.1. Словесное описание

Автомат *A2* обходит все поле, посещая каждую клетку по одному разу, для обхода всех ее соседей с целью сбора информации о текущем направлении и скорости движения. Движение начинается с левой нижней клетки. Переменная $x_1$ определяет, является ли текущая ячейка крайней, и значение этой переменной определяется внешней средой.

Воздействия $z_1..z_4$ сдвигают указатель на ячейку (рис. 21) в одном из четырех направлений:

z1 – перейти на ячейку вверх;

z2– перейти на ячейку справа;

z3– перейти на ячейку вниз;

z4– перейти на ячейку влево.

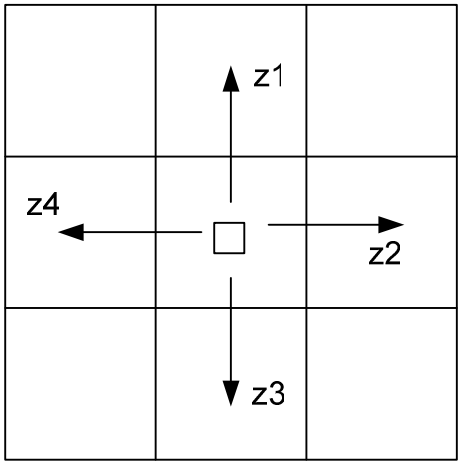


Рис.21. Направления сдвигов событий z1..z4

Автомат начинает движение слева направо, находясь в состоянии 0, и последовательно обходит все соседние ячейки, обрабатывая значения скоростей в этих ячейках и устанавливая значение скорости в текущей ячейке.

Когда автомат *A2* достигает правого края поля (это определяется по значению переменной `x1)`, он поднимается вверх на одну клетку и начинает движение влево (рис. 22).

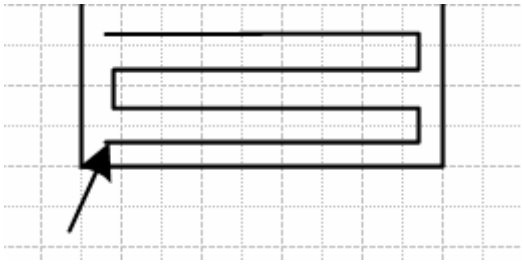

Рис. 22. Обход всех клеток решетки

Направление обхода соседних клеток зависит от направления движения (вправо или влево) автомата и определяется состоянием, в котором он находится:

0 – автомат находится в обрабатываемой клетке и движется слева направо;

1..8 – обход соседних клеток от той, в которой он находился в состоянии 0;

10 – автомат находится в обрабатываемой клетке и движется справа налево;

11..18 – обход соседних клеток от той, в которой он находился в состоянии 10;

На рис. 23, 24 показаны направления обходов соседних клеток.

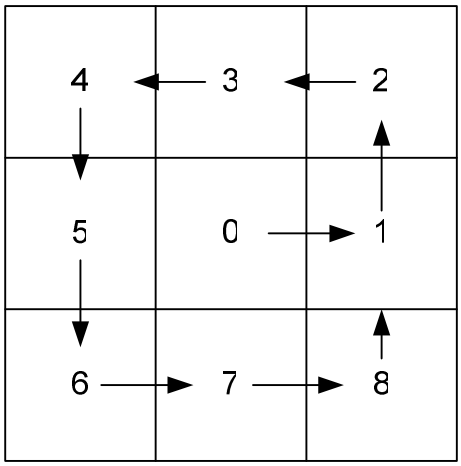


Рис.23. Обход клеток при движении слева направо

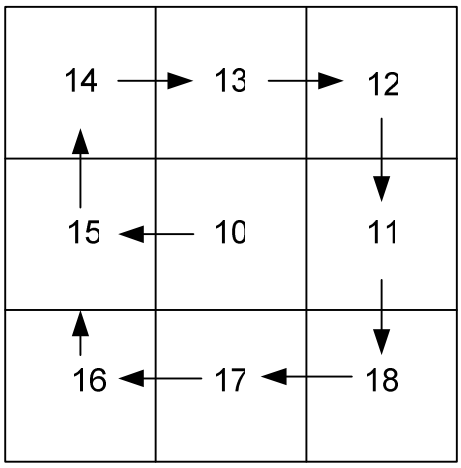


Рис.24. Обход клеток при движении слева направо

В клетках, соответствующих состояниям 1, 3, 5, 7, 11, 13, 15, 17, которые являются соседними в прямых направлениях от обрабатываемой, выполняются действия `z11`, `z12`, `z13`, `z14` которые предназначены для коррекции скорости текущей обрабатываемой клетки. Эти действия описаны ниже.

Событие `z100` предназначено для окончательной установки нового значения скорости в обрабатываемой клетки.

## 13.2. Граф переходов

Граф переходов автомата приведен на рис. 25.

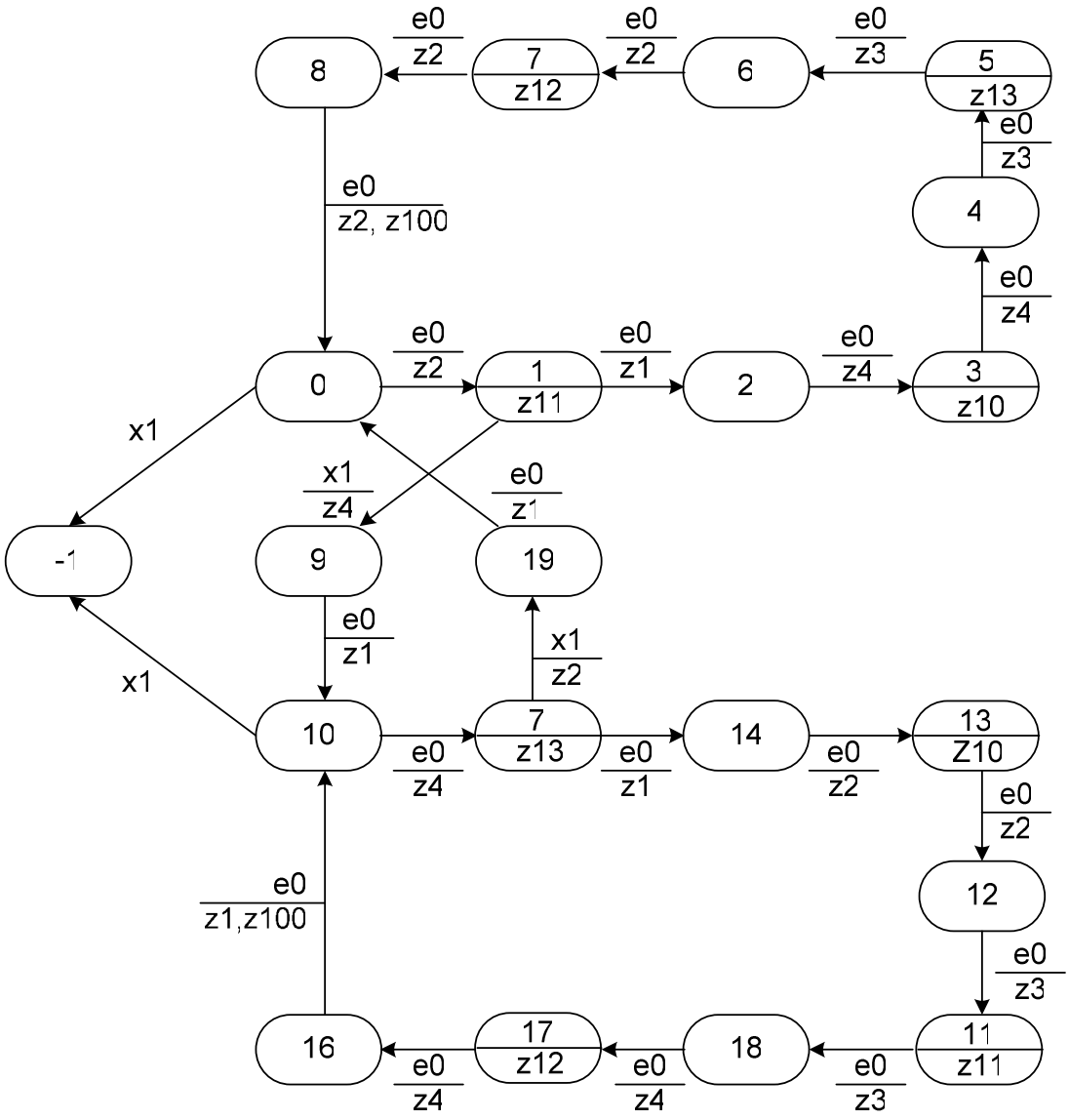


Рис.25. Граф переходов автомата *A2*

## 13.3. Определение скорости в клетке

Значение и направление скорости определяется по состояниям автоматов *A10*, *A11*, *A12*, *A13*, измеряющих горизонтальную и вертикальную составляющие скоростей.

На рис. 26 показано как определяется направление скорости движения в ячейке, основываясь на значениях состояний автоматов *A10…A13*.

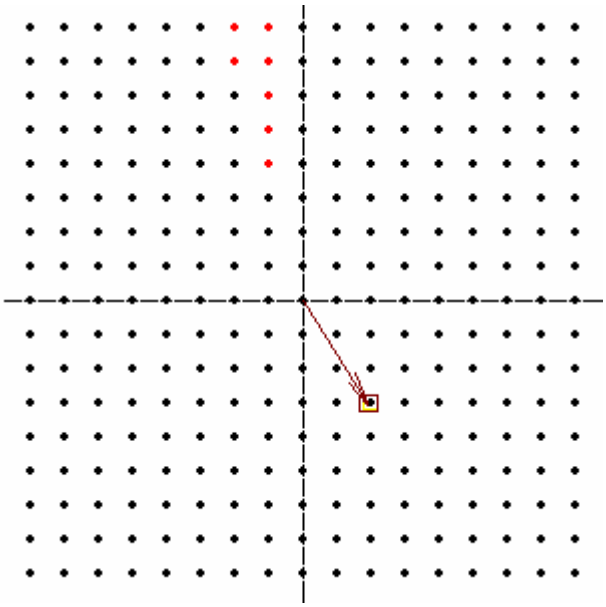

Рис. 26. Направление и значение скорости

Здесь:

- состояние автомата *A10* равно 0;
- состояние автомата *A11* находится в интервале 7...11, что соответствует значению скорости 1/5;
- состояние автомата *A12* находится в интервале 12...15, что соответствует значению скорости 1/4;
- состояние автомата *A13* равно 0.

Первый раз скорость определяется только для клеток, соответствующие автоматы *A1* которых на текущем такте работы перешли в состояние 1 – для тех клеток, на которые был совершен въезд машины (по фронту ее движения). Когда автомат *A2* доходит до такой ячейки, значение скорости в ней равно нулю, и оно должно быть установлено в зависимости от скоростей в соседних клетках. Среди соседних клеток выбираются те, автомат *A1* которых находится в состоянии 2 или 3 (на предыдущих шагах в этой клетке находился объект, значит, значение

скорости в ней определено). Далее считается среднее значение скорости в этих клетках, и по этому значению в рассматриваемой клетке устанавливается начальное состояние автоматов скоростей.

На следующем шаге посылаются события $e_1$ на те проекции скорости, с противоположной стороны которых находились занятые ячейки. На те вектора скорости, с противоположных сторон которых находились пустые клетки, посылаются событие $e_0$. Это событие посылается также на те клетки, которые находятся в состоянии 2 (те клетки, которые считаются оставшимися на своем месте).

Событие $e_{-1}$ отправляется на те клетки, автомат *A1* которых находится в состоянии 3 (эти клетки покинула сдвинувшаяся машина).

Рассмотрим эти правила (рис. 27).

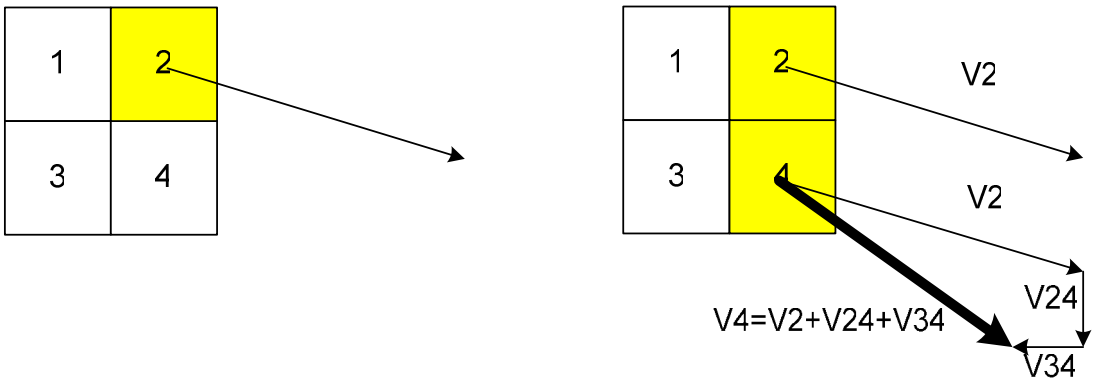


Рис. 27. Определение скорости клетки

Шаг 1: клетка 2 закрашена, остальные нет.

Шаг 2: клетки 2 и 4 закрашены, остальные нет.

При выполнении этих шагов, происходит сдвиг из клетки 2 в клетку 4. Таким образом, клетка 4 перешла в состояние 1, и ее скорость необходимо пересчитать.

Для этого сначала будут скопированы все составляющие скорости из клетки 2 в клетку 4 (V2).

Так как автоматам измерения скоростей требуется одно из событий $e_1$ или $e_0$ на каждом шаге работы, то анализируются произошедшие сдвиги.

В рассматриваемом примере над клеткой 4 расположена клетка 2, автомат *A1* которой находится в состоянии 2 или 3 ( на предыдущем шаге она была активна). При этом на автомат *A12*, который отвечает за направление скорости вниз, будет подано событие $e_1$ – воздействие клетки 2 на клетку 4, обозначенное как `V24` будет увеличивающим скорость.

Поскольку клетка 2 уже на предыдущем шаге была занята, то движение вверх отсутствует, и на автомат *A10*, который измеряет скорость движения вверх, подается событие $e_{-1}$, обнуляющее скорость.

За отправку этих событий отвечает действие `z10` автомата *A2*, которое генерируется, когда этот автомат находится в состояниях 2 или 3.

Поскольку больше прямых соседей нет, то на остальные три автомата для измерения скоростей подаются события $e_0$. Например, воздействие клетки 3 на клетку 4, обозначенное как `V34`, будет уменьшать скорость.

В качестве еще одного примера рассмотрим движение клетки наискосок – вправо вниз (рис. 28)

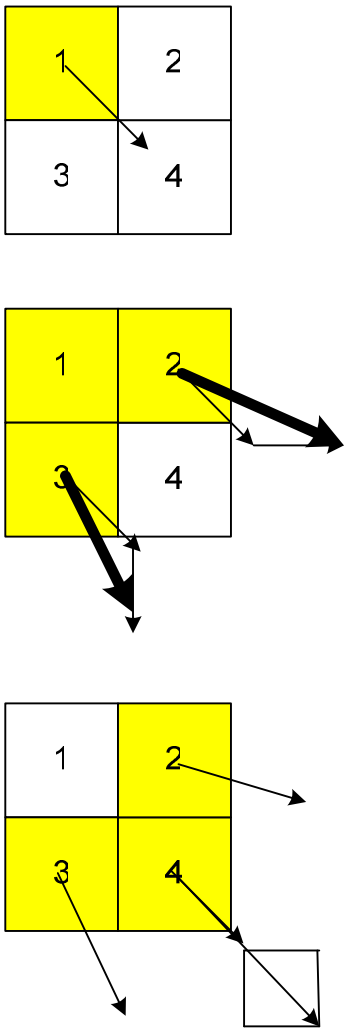


Рис. 28. Движение вправо-вниз

Это движение осуществляется следующим образом:

Шаг 1. Клетка 1 активна, направление вправо вниз.

Шаг 2. Активировались клетки 2 и 3. Их скорости копируются из клетки 1, но так как у клетки 2 есть активная соседняя клетка слева, на ее автомат *A11* посылается событие $e_1$, а у клетки 3 такое событие посылается на автомат *A12*.

Шаг 3. У клетки 4 направление и скорость копируются у соседей 2 и 3. После этого на оба автомата *A11* и *A12* подаются дополнительные воздействия $e_1$ ввиду того, что соответствующие соседи активны.

## *14. Вычисление опасности*

После того, как направление движения клетки и ее скорость вычислены, становится возможным определение опасности, которую данная клетка представляет.

Вокруг начала координат создается «квадрат опасности», перемещение внутрь которого чужих машинок является недопустимым. (В демонстрационной программе сторона такого квадрата равна четырем клеткам). Далее определяется ближайшая точка пересечения вектора скорости машины и сторон этого квадрата. Если такая точка существует, то машина движется в опасном направлении, и необходимо вычислить численное значение опасности.

Абсолютное значение скорости клетки определяется соотношением:

$V = \sqrt{V_x^2 + V_y^2}$ .

Вычисляется расстояние до «квадрата опасности» S из координат клетки, и точка пересечения вектора скорости и периметра опасного квадрата (рис. 29).

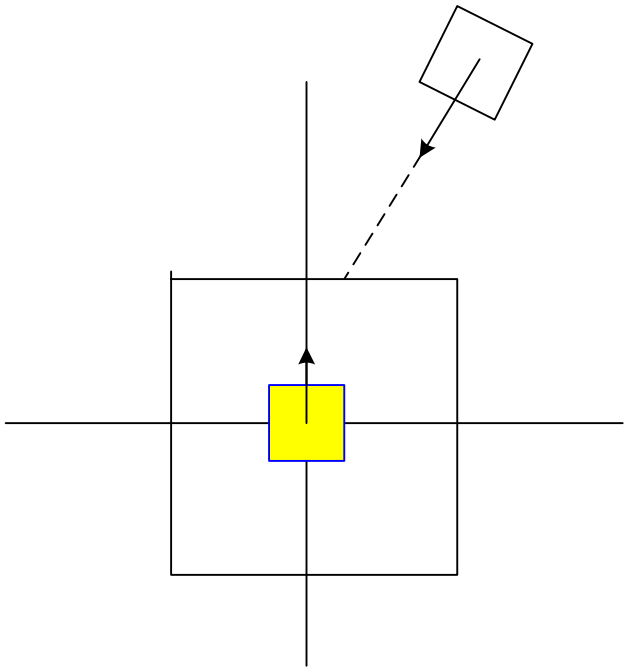

Рис.29. Вычисление опасности

Значение опасности D есть величина, обратная времени, за которое движущаяся клетка достигнет периметра:

D = V / S.

Опасность может проецироваться на непрерывное множество точек по периметру квадрата. Следующим шагом происходит дискретизация периметра «квадрата опасности». Он разбивается на некоторое число областей. В демонстрационной программе это число равно 12 (рис. 30).

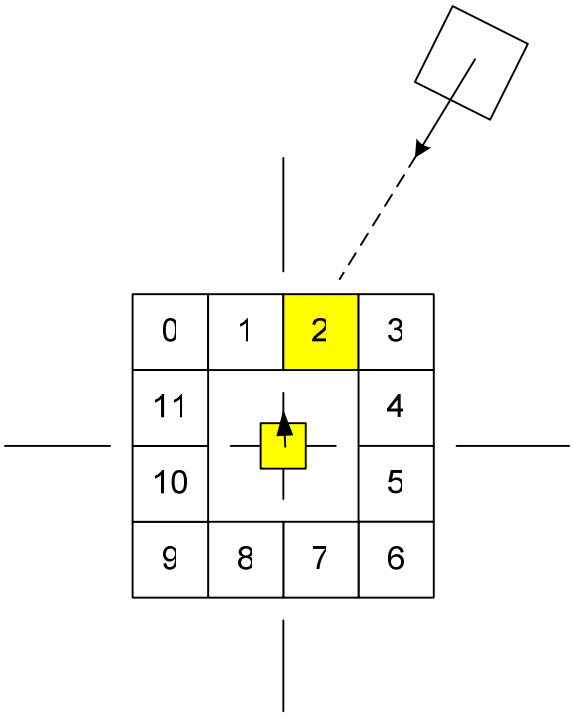


Рис. 30. Дискретизация опасного периметра

В ячейку 2 заносится рассчитанное значение опасности клетки D.

После обхода всех клеток рассчитывается максимум по всем «периметрам опасности». Функция максимум выбрана потому, что близко расположенная машина представляет большую опасность, чем крупный объект, расположенный дальше. Таким образом, получаем набор значений угроз d:

$$d_i = \max \frac{V_k}{Sk} \ .$$

Максимум берется по клеткам, представляющим угрозу для данной клетки, которые расположены на периметре «квадрата опасности».

Можно сказать, что происходит разбиение всей решетки на сектора, и по каждому из секторов считается максимальное значение угрозы.

Значения опасностей используются далее для формирования команды воздействия на свою машину для устранения опасности.

### *15. Алгоритм управления*

Для выработки команд для изменения скорости (с помощью педали газа и тормоза) и угла поворота (с помощью руля) строится простая нейронная сеть с линейной функцией активации, ограниченной сверху и снизу максимальными значениями скоростей и угла поворота.

При этом требуется определить два набора весовых коэффициентов $w^{(1)}$ и $w^{(2)}$: первый – для вычисления скорости, а второй – для угла поворота:

$$\Delta\upsilon = f_1(\sum_{i=1}^{n} d_i \cdot w^{(1)}{}_i);$$

$$\alpha = f_2(\sum_{i=1}^{n} d_i \cdot w^{(2)}{}_i) \ .$$

Устанавливаются случайные начальные малые значения векторов $w^{(1)}$ и $w^{(2)}$. Затем они корректируются на основе следующих правил.

Если опасность сзади, то в ячейках 5 – 10, следует увеличить скорость за счет увеличения коэффициентов $w^{(1)}{}_{5..10}$ .

Если опасность впереди, а угроза идет на ячейки 0 – 3, то следует уменьшить скорость за счет уменьшения соответствующих компонент $w^{(1)}$.

Если опасность слева, следует свернуть в противоположную сторону. При этом значения весов $w^{(2)}{}_{3..6}$ уменьшаются, а $w^{(2)}{}_{9..11}$ увеличиваются.

Дальше программа запускается в режиме обучения. При столкновениях соответствующие коэффициенты в ячейках опасности, с

которыми произошло само столкновение, увеличиваются по абсолютной величине.

Тренировка необходима для того, чтобы при небольшой угрозе машина научилась плавно обходить препятствия – без резких поворотов и резких изменений скорости.

Ниже приведены три примера (скриншота) использования описанного подхода.

Пример 1. Управляемая машина сворачивает с пути другой, движущейся ей наперерез, неуправляемой машины (рис. 31). Черным цветом показана траектория движения неуправляемой машины, а светлым – траектория управляемой.

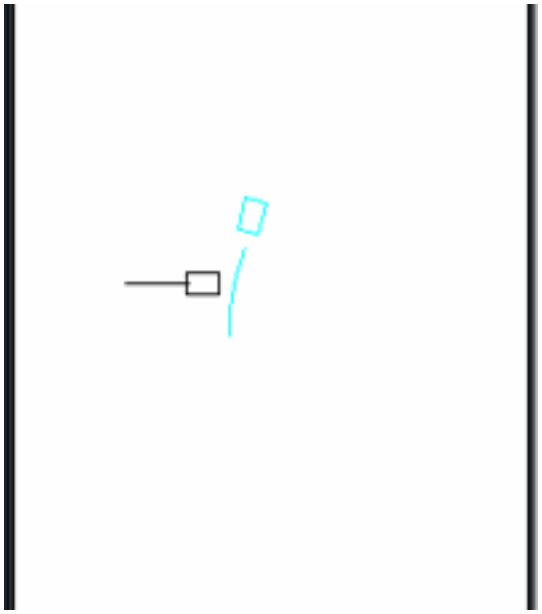

Рис.31. Пример движения

Пример 2. Управляемая машина замедляется и пропускает движущуюся наперерез неуправляемую машину (рис. 32).

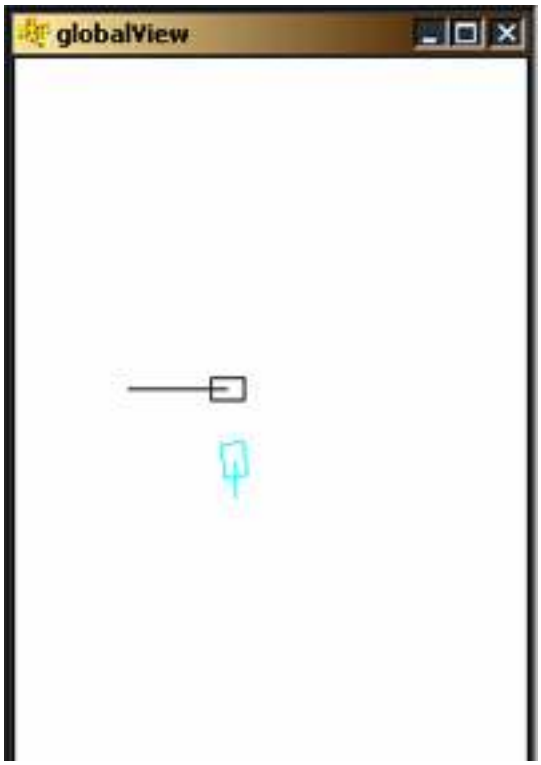


Рис. 32. Пример движения

Пример 4. Движущиеся параллельно управляемой машине неуправляемые машины не оказывают никакого влияния на нее (рис. 33).

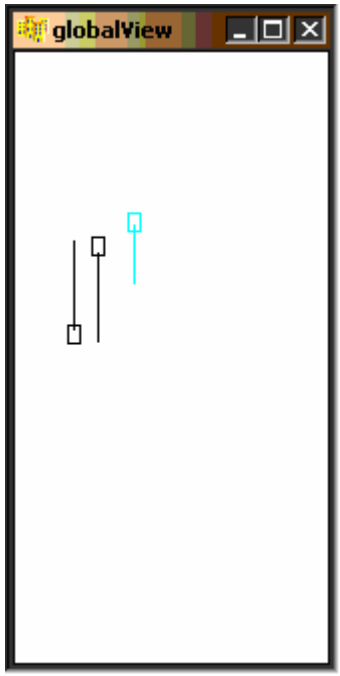


Рис. 33. Пример движения

## ВЫВОДЫ

1. В работе исследована возможность реализации некоторых элементов нейронной сети в виде конечных автоматов. Предложен вариант совместного использования конечных автоматов и нейронных сетей, при котором конечные автоматы находятся на входе нейронной сети и производят предварительный анализ входных данных.

Выполнено сравнение с традиционным использованием нейронных сетей.

Традиционно нейронная сеть используется, в том числе, и для распознавания изображений. Цвет каждой точки входного изображения передается на нейроны входного слоя нейронной сети, и через весовые коэффициенты и функцию активации вычисляется вектор выходных воздействий (рис. 30).

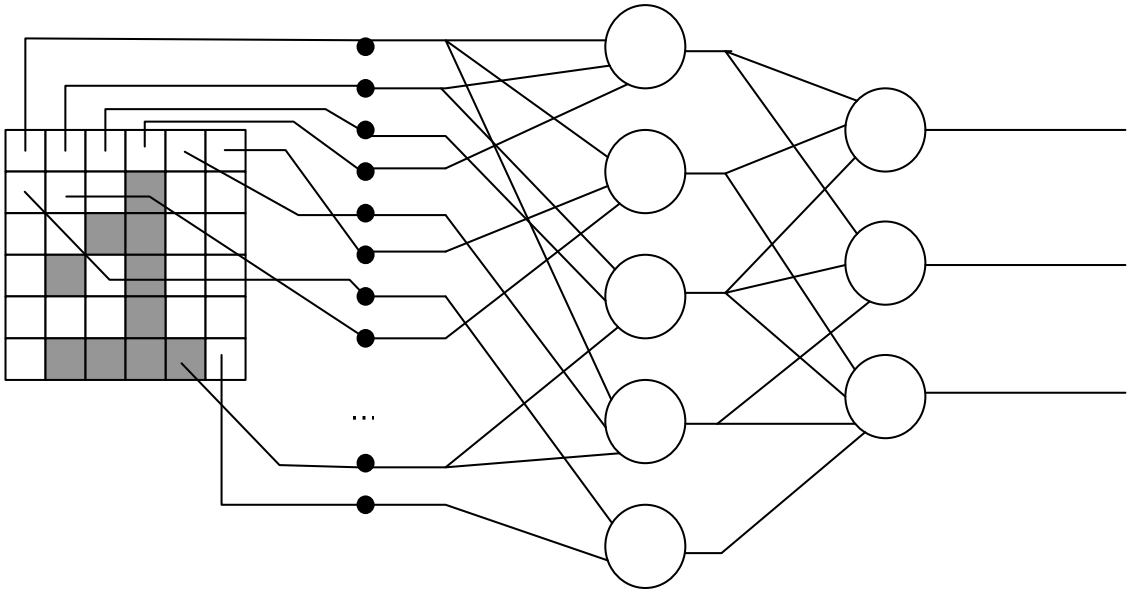


Рис. 30. Нейронная сеть для распознавания изображений

Круг задач, решаемых такой нейронной сетью, ограничен распознаванием статических изображений (без учета предыстории изменений). При этом нейронная сеть вычисляет результат по конкретному набору исходных данных, не учитывая динамику и изменение этих данных.

В тех задачах, в которых требуется учитывать динамику процесса, следует применять другие структуры, например, описанную в данной работе комбинацию нейронной сети и набора конечных автоматов.

В такой структуре при помощи автоматов выделяются параметры, которые невозможно измерить для одного конкретного входного образа, но которые видны проектанту при анализе динамики происходящих процессов. Эти параметры могут обрабатываться нейронной сетью, которая на их основе может принимать решения.

С каждой точкой входного изображения связан конечный автомат, на который подаются события, связанные с изменениями, происходящими в точке. Каждый автомат, в свою очередь, связан с нейронной сетью, на входы которой он отправляет сигналы. Нейронная сеть, распознавая различные конфигурации состояний конечных автоматов, позволяет выделить таким образом динамические образы (рис. 31).

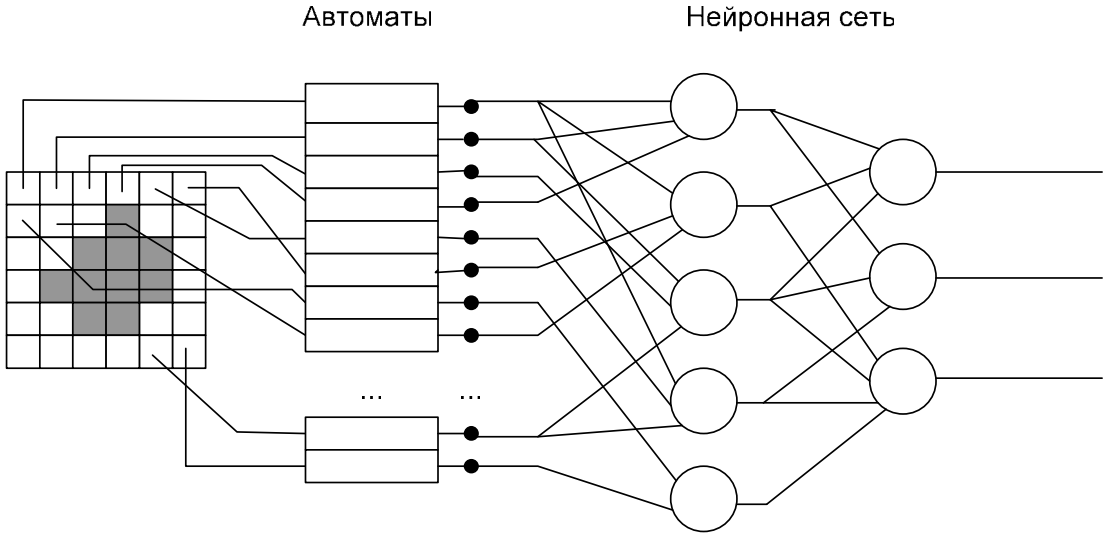


Рис 31. Схема взаимодействия

Целью построения такой структуры является улучшение характеристик нейронной сети, основанное на возможности конечных автоматов сохранять информацию о предыстории поступающих на вход данных.

Предложенный подход может быть использован в тех случаях, в которых возможностей нейронной сети по анализу статической информации на входе недостаточно, и требуется анализировать динамическую информацию, когда важно не только то, какая информация имеется на входе сети в данный момент, но и история ее изменения.

2. На основе подхода, предложенного в предыдущем пункте, создан пример, реализующий систему анализа движущихся изображений. Разработанная система позволяет выделять движущиеся объекты, определять их контуры, а также скорость и направление движения. В данном приложении **набор конечных автоматов** производит предварительный анализ изображения, выделяя контуры объекта, а **нейронная сеть** вычисляет скорость и направление его движения.

Промышленные аналоги применяются для охраны объектов, автоматического включения света, фильтрации видеоинформации, контроля движения транспортных средств и т.д. Это дорогостоящие продукты и они имеют закрытые алгоритмы работы.

Система анализа движущихся изображений, разработанная на основе предложенного метода построения нейронных сетей с применением конечных автоматов, не уступает аналогам в плане функциональности и при этом требует меньших вычислительных ресурсов для своей работы.

Разработанная система анализа движущихся изображений может быть использована в любых приложениях, в которых требуется анализ движущихся изображений, таких как: охрана защищенных объектов и охранные сигнализации, система контроля движения транспортных средств, система автоматического управления освещением, фильтрация видеоизображений.

3. Реализован пример «симулятора автомобильного движения», в котором система управления машиной анализирует ситуацию на дороге, используя разработанную систему, и управляет машиной так, чтобы избежать столкновений.

В данном примере **набор конечных автоматов** производит предварительную обработку дорожной ситуации – выделяются те машины, которые представляют угрозу выделенной, и высчитывается численное значение опасности. Полученный набор числовых значений опасностей поступает на вход **однослойной нейронной сети**, которая принимает решение об управлении машиной с целью избежания столкновения.

4. Созданы приложения, реализующие описанные примеры. Эти приложения опубликованы на сайте http://is.ifmo.ru/ в разделе «Проекты».

## *СПИСОК ЛИТЕРАТУРЫ*


1. *Hebb D. O.* Organization of behavior. New York: Science Editions. 1949.
2. *Новые разработки Bosch* в области обнаружения движения, статья, http://www.secnews.ru/foreign/1039.htm
3. *Системы видеонаблюдения для дома* и офиса, статья, http://videoglazok.ru/catalog/ustroistv/vtd_motion.htm
4. *Мокшин В.И., Петров А.А., Титов В.С., Якушенков Ю.Г.* Техническое зрение роботов. М: Машиностроение, 1990.
5. *Преобразование стандартов*: методы оценки векторов движения, статья, http://rus.625-net.ru/625/2006/08/theory.htm
6. *Шалыто А.А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
7. *Шалыто А.А., Туккель Н.И.* SWITCH-технология – автоматный подход к созданию программного обеспечения "реактивных" систем //Промышленные АСУ и контроллеры. 2000. № 10.
8. *Шалыто А.А.* Логическое управление. Методы аппаратной и программной реализации алгоритмов. СПб.: Наука, 2000.
9. *Заенцев И.В.* Нейронные сети: основные модели. Учебное пособие, Воронеж: Воронежский государственный университет, 1999.
10. *Варшавский В.И., Поспелов Д.А.* Оркестр играет без дирижера: размышления об эволюции некоторых технических систем и управлении ими. М.: Наука, 1984.
11. *Растригин Л.А.* Адаптация сложных систем. Рига: Зинатне, 1981.

12. *Барский А.Б.* Нейронные сети: распознавание, управление, принятие решений. М.: Финансы и статистика, 2004.
13. *Поспелов Д.А.* Моделирование рассуждений: Опыт анализа мыслительных актов. М.: Радио и связь, 1989.
14. *Шидловский С.В.* Автоматическое управление. Перестраиваемые структуры. Томск: Томский государственный университет, 2006.
15. *Туккель Н.И., Шалыто А.А.* Система управления танком для игры «Robocode». Объектно-ориентированное программирование с явным выделением состояний. Проектная документация. СПбГУ ИТМО. 2001. http://is.ifmo.ru/projects/tanks/
16. *Кузнецов Д.В., Шалыто А.А.* Система управления танком для игры «Robocode». Объектно-ориентированное программирование с явным выделением состояний. Вариант 2. Проектная документация. СПбГУ ИТМО. 2003. http://is.ifmo.ru/projects/robocode2