электронный журнал
МОЛОДЕЖНЫЙ
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО «Московский государственный технический университет им. Н.Э. Баумана»

УДК 004.93'1

Применение аппарата нечеткой логики при решении задачи поиска пути в неизвестном окружении

Бекасов Денис Евгеньевич,
ИУ7-82. МГТУ им. Н.Э. Баумана,
e-mail: bauman@bmstu.ru.

Введение

Одним из основополагающих понятий агентно-ориентированного подхода к изучению проблем Искусственного Интеллекта (ИИ) является понятие интеллектуального агента. Интеллектуальный агент получает результаты актов восприятия из окружающей среды и выполняет действия, причём интеллектуальный агент реализует функцию, которая преобразует последовательности актов восприятия в действия [1].

Одной из важнейших задач агента является задача планирования пути до некоторой цели. Особенно актуальным является вопрос ориентации в неизвестном окружении, т.к. пространство редко является статичным и не всегда имеется его формальное описание (карта).

1 Постановка задачи

Пусть имеется некоторый агент, способный обозревать ближайшее окружение. Для определенности будем считать, что это трехколесный робот, оборудованный датчиком расстояния (дальномером). Пусть имеется некоторое пространство, вмещающее в себя объекты трех типов: агента, цели и препятствия. Цель – это точка пространства, в которую агент должен попасть. Препятствия – совокупность точек пространства, через которые агент не способен осуществлять движение. Тогда задача формулируется следующим образом: агенту необходимо проложить путь от своей исходной позиции до заданной цели, огибая препятствия на пути. При этом ставятся ограничения на непрерывность и оптимальность пути, а сам агент представляется материальной точкой, положение которой точно задано координатами в пространстве.

Под оптимальным путем понимают наикратчайший возможный путь из исходной позиции до цели.

2 Обзор существующих подходов к решению задачи

Математически хорошо проработаны алгоритмы поиска пути в известном или частично известном окружении. Для этого используется дискретная математика (теория графов) и линейное программирование [5]. Задачи поиска кратчайшего пути в графе давно известны и изучены (например, алгоритм Дейкстры и его модификации). Кроме этого используют и другие алгоритмы (пересечения линий, волнового фронта, эвристические *A** и *D**).

Поиск пути в заранее неизвестном окружении формализовать гораздо сложнее, т.к. нет заранее составленной карты (формального описания окружения). Пространство не может быть представлено в виде четко определенной математической структуры и, соответственно, не может быть найден оптимальный путь в строгом смысле слова. Длина результирующего пути зависит от конкретного выбора на каждом шаге, т.к. неизвестно приведет ли сделанное решение к возникновению тупика (комбинации препятствий, выход из которой возможен только один – это возвращение на предыдущий шаг) или нет.

В робототехнике для задачи управления движения роботом существует термин «*Motion planning*»[11]. В рамках данной задачи выделяют следующие понятия:

- размерность рабочего пространства (*W*) – 2D, 3D и выше (управление сложными манипуляторами с большим количеством суставов).
- *C* или пространство конфигураций (число степеней свободы робота). Например, материальная точка на плоскости задаётся (x,y), плоское тело – (x,y,θ), трехмерное тело в трехмерном пространстве – (x,y,z,α,β,γ).

Все состояния агента характеризуются точкой в пространстве конфигураций. Вводят понятие пространства свободных конфигураций (C_{free}) – множество тех состояний агента, проекция которых на рабочее пространство не пересекается со множеством препятствий [4].

Алгоритмы управления роботом по способу представления пространства конфигураций разделяются на несколько категорий.

1 Сеточные алгоритмы.

Алгоритмы из данной группы используют в пространствах конфигураций малых размерностей. Каждая точка в *C* представляется узлом сетки. Агент, двигаясь по пути от начальной позиции до цели переходит таким образом из одного узла в другой, при условии его доступности. Для таких алгоритмов очень важным параметром является выбор разрешения сетки и ограничений на типы переходов (от этого зависит точность и быстродействие алгоритма)[4][11].

Классический пример такого алгоритма – *A**. Он является эвристическим уточнением алгоритма Дейкстры. В отличие от оригинала перебираются не все вершины, а только наиболее вероятные. Для этого учитывают не только стоимость прохождения по пути от текущей вершины к соседней, но и эвристическую функцию приоритета (в простейшем случае – это расстояние до цели), что значительно сужает область поиска. Таким образом, при обходе препятствия агентом, агент будет выбирать наиболее выгодные, с точки зрения заданной эвристики, позиции [1][5][6].

Использование данных подходов при поиске пути в неизвестном окружении не является оптимальным решением, т.к. подразумевает постоянное перестроение сетки и перерасчет пути.

2 Геометрические алгоритмы.

- Использование графа видимости (графа взаимовидимых позиций). Вершинами графа являются доступные позиции в пространстве, а ребрами – доступные переходы между позициями (если переход между позициями возможен, то между соответствующими вершинами есть ребро). Поиск пути осуществляется с помощью алгоритмов поиска в графах (например, Дейкстры). Строится в пространствах с полигональными препятствиями. При перемещении в неизвестном окружении, агент может по ходу движения достраивать граф видимости, добавляя новые вершины по мере того, как они будут открываться. Данный подход может быть использован, если у агента имеется достаточно широкий угол обзора.

- Использование декомпозиции пространства конфигураций на ячейки. При этом строится граф связей, содержащий в качестве вершин ячейки. Вершины графа соединены ребром тогда и только тогда, когда соответствующие ячейки смежны. Алгоритмы построения такого графа делятся на точные и приближенные. Точные алгоритмы проводят точную декомпозицию свободных зон (при этом получаются нерегулярные ячейки), а приближенные аппроксимируют эти зоны ячейками простой формы [4][5].
- Использование суммы Минковского. Сумма Минковского – это сложение 2 множеств путем складывания всех векторов в обоих множествах. В данном случае речь идет о вычислении пространства C путем сложения множества точек препятствий и множества точек агента. Применительно к движению в неизвестном окружении может применяться только для локальных корректировок при обходе препятствия по контуру.

3 Алгоритмы с использованием потенциального поля.

Движение агента рассматривается как движение точки в потенциальном поле, где потенциал каждой точки поля складывается из её приближенности к цели и удаления от препятствий. Траектория такой точки и есть путь до цели. Достоинством алгоритма является малая вычислительная стоимость, однако существует вероятность попадания агента в локальный минимум. Для того чтобы избежать этого, часто используют комбинацию данного алгоритма с разбиением Вороного, где за узлы разбиения принимают все локальные минимумы [4].

4 Алгоритмы на основе выборки.

Из C_{free} выбирается N конфигураций, которые используются как контрольные точки для создания плана движения (*roadmap*)[4]. План движения соединяет две точки, если отрезок, их соединяющий, полностью лежит в C_{free}. Для построения пути в пункт назначения из начальной точки, их добавляют в план движения. Если план движения не соединяет их, то в него добавляются дополнительные контрольные точки.

Данный алгоритм часто используют в многомерных пространствах конфигураций, т.к. скорость его работы не зависит от размерности пространства. Если рассматривать его применительно к двумерному случаю, то составленный план движения будет являться графом видимости, описанным выше. В этом случае в графе видимости достаточно оставить только те ребра, которые являются касательными к препятствиям в обеих точках.

При движении в неизвестном окружении агент должен по мере продвижения проверять достижимость очередной контрольной точки и в случае необходимости добавлять новые в зависимости от узнаваемых условий.

Несложно заметить, что практически все подходы в конечном итоге можно свести к задаче на поиск в графе. А значит, для решения применимы все подходы, используемые в подобных случаях: алгоритм волнового фронта, алгоритм Дейкстры, его эвристические модификации (A*, D*), симплексный метод, динамическое программирование.

Все рассмотренные выше алгоритмы и подходы в основном применяются для планирования пути при известном и частично известном окружении. Использование данных подходов при движении в неизвестном окружении в чистом виде либо невозможно, либо требует много вычислительных затрат. Для решения такой задачи необходимо или модифицировать их, или использовать специализированные подходы (например, алгоритмы *Bug*[3]). Такие алгоритмы по сути представляют из себя различные стратегии выбора решения при той или иной конфигурации пространства и используют некоторые идеи описанных выше алгоритмов

(приоритеты точек, эвристические оценки, графы видимостей). Самое распространенное семейство таких подходов – это *Bug*-алгоритмы. Ниже приведены несколько их модификаций.

1 Алгоритм поворота (*Bug0*).

Агент движется по прямой линии до цели, а при возникновении препятствия запоминает свою позицию и движется влево вдоль препятствия в поисках «прохода». После этого он продолжает свой путь (либо возвращаясь на старую траекторию, либо генерируя новую исходя из текущего положения). Если агент вернулся в точку выбора, то он начинает движение вправо. Алгоритм можно модифицировать, выбирая сторону поворота на основе реальной ситуации – например, с помощью некоторой оценки возможной длины препятствия в обе стороны. Алгоритм не универсален, т.к. возможны случаи, когда агент не находит путь до цели [2].

2 *Bug1*

Агент движется по направлению к цели до встречи с препятствием. После этого запоминает эту позицию, поворачивает налево и продолжает путь вдоль препятствия до возврата в позицию встречи с препятствием. Если наиближайшая точка к цели совпадает с точкой встречи с препятствием, то пути до цели не существует. Если нет, то агент отправляется в эту точку и начинает движение к цели по прямой. Алгоритм всегда находит путь, если он существует [2][3].

3 *Bug2*

Модификация *Bug1*, созданная с целью исключить необходимость обхода всей границы препятствия. Отличие состоит в том, что агент двигается вдоль препятствия до пересечения с изначальной траекторией движения, а не в поисках самой близкой точки к цели, что уменьшает проходимый путь. Однако чистый алгоритм может дать неверный результат при объезде невыпуклых препятствий, поэтому обычно используют его модификации, учитывающие эту особенность[2][3].

4 *Tangent Bug*

Предыдущие алгоритмы не учитывали способности агента дистанционного определения препятствий. Агент двигался до цели «на ощупь», что значительно понижало оптимальность выбранного пути. Для учета возможности определения препятствий на расстоянии был разработан алгоритм, строящий локальные графы касательных. Алгоритм использует следующие понятия: точка разрыва (препятствие/граница видимости) и интервал непрерывности (определяется двумя точками разрыва).

Агент движется в сторону цели по прямой до тех пор, пока не зафиксирует препятствие на пути (прямая будет пересекать интервал непрерывности). После этого агент начинает движение к одной из точек разрыва, для которой эвристическая оценка минимальна (например, к той, сумма расстояний от которой до робота и до цели минимальна). В процессе движения агент получает новые точки разрыва и движется к ним до тех пор, пока эвристическая оценка не перестанет уменьшаться (т.е. не достигнет локального минимума). После этого агент двигается вдоль границы препятствия, сохраняя направление. Покидает границу, когда препятствие больше не мешает проходу к цели[3].

Данный подход позволяет найти более оптимальный путь, чем предыдущие, однако требует более сложной организации системы машинного зрения, если речь идет о роботах.

3 Применение нечеткой логики.

Все алгоритмы поиска пути, названные ранее, рассматривают путь к цели как некоторую ломаную линию, где узлы – это точки принятия решения о повороте на определенный угол и движении дальше по прямой. Некоторые из рассмотренных

алгоритмов дают хорошее приближение к оптимальному пути (т.е. их длина их пути стремится к длине наикратчайшего возможного пути до цели), однако для повышения оптимальности эти траектории следует сгладить. Сглаживание траектории во-первых позволит сократить путь, а во-вторых обеспечит плавность движения агента без резких изменений скорости и временных задержек на повороты в узлах. Задача сглаживания пути при известном окружении может решаться на этапе планирования траектории, а при движении в неизвестном окружении эту задачу необходимо решать в реальном времени.

Для получения сглаженной траектории, приближенной к оптимальной, а так же для организации естественного управления сложной технической системой, такой как мобильный робот, можно использовать аппарат нечеткой логики. Общей предпосылкой для применения нечетких систем управления является, с одной стороны, наличие неопределенности, связанной как с отсутствием информации, так и сложностью системы и невозможностью или нецелесообразностью ее описания традиционными методами и, с другой – наличие объекта, необходимых управляющих воздействий, возмущений и т.п., а также наличие информации качественного характера [8].

В теории нечетких множеств любая величина задается некоторым множеством её возможных значений, характеризующихся той или иной степенью принадлежности (с помощью так называемой функции принадлежности) объекту, описываемому этим нечетким множеством. Функция принадлежности может принимать значения от 1 («полностью принадлежит») до 0 («полностью не принадлежит»). Для описания объектов и явлений реального мира с позиции теории нечетких множеств используют понятие лингвистической переменной [9].

Кроме того, помимо удобной формализации сложной обработки данных, нечеткий подход позволяет достигнуть сразу несколько дополнительных целей, а именно:

- использование нечеткой модели дает возможность задания поведения агента в виде одной единой базы знаний, что унифицирует решение и предоставляет возможность удобных модификаций в будущем;
- применительно к движению агента нечеткая схема управления может быть построена таким образом, что на выходе она будет выдавать готовые управляющие воздействия, без необходимости их обработки (т.е. не угол разворота, а, например, скорости двигателей).

Если рассматривать задачу создания нечеткой системы управления агентом применительно к рассматриваемой структуре робота (трехколесный двухприводный мобильный робот, оснащенный ультразвуковым датчиком расстояния на вращающейся платформе), то её можно обозначить следующим образом:

- входные характеристики: положение робота, положение цели, расстояния до препятствий по необходимым направлениям;
- выходные характеристики: управляющие воздействия на двигатели.

Лингвистические переменные формируются из указанных характеристик, функции принадлежности должны быть сформулированы так, чтобы обеспечить плавный переход между классами ситуаций, а база правил должна включать в себя правила для определения базового направления и правила корректировки направления в зависимости от обнаруживаемых препятствий на пути.

4 Нечеткий алгоритм достижения цели

Исходя из указанных выше рассуждений был предложен нечеткий алгоритм поиска пути. Идея алгоритма состоит в следующем: после начальной инициализации агент запоминает координаты цели и свое начальное состояние. Он переводит их в

понятие лингвистических переменных и с помощью базы правил определяет направление на цель. После этого по полученному направлению (+/- 45 градусов) он проверяет наличие препятствий, переводит результат в значения лингвистических переменных и, опять используя базу знаний, получает значения корректировок. После этого он формирует угловую характеристику скорости двигателей и фазифицирует её. Функция принадлежности данной лингвистической переменной подобрана таким образом, что получившиеся в результате фазификации степени принадлежности двум термам и равны скоростям двух двигателей. Полученные значения посылаются на контроллер двигателей и агент снова производит те же действия. Данный алгоритм повторяется до тех пор, пока цель не будет достигнута. На рисунке 4.1 представлена схема данного алгоритма, а в следующем разделе представлено описание соответствующей базы знаний.

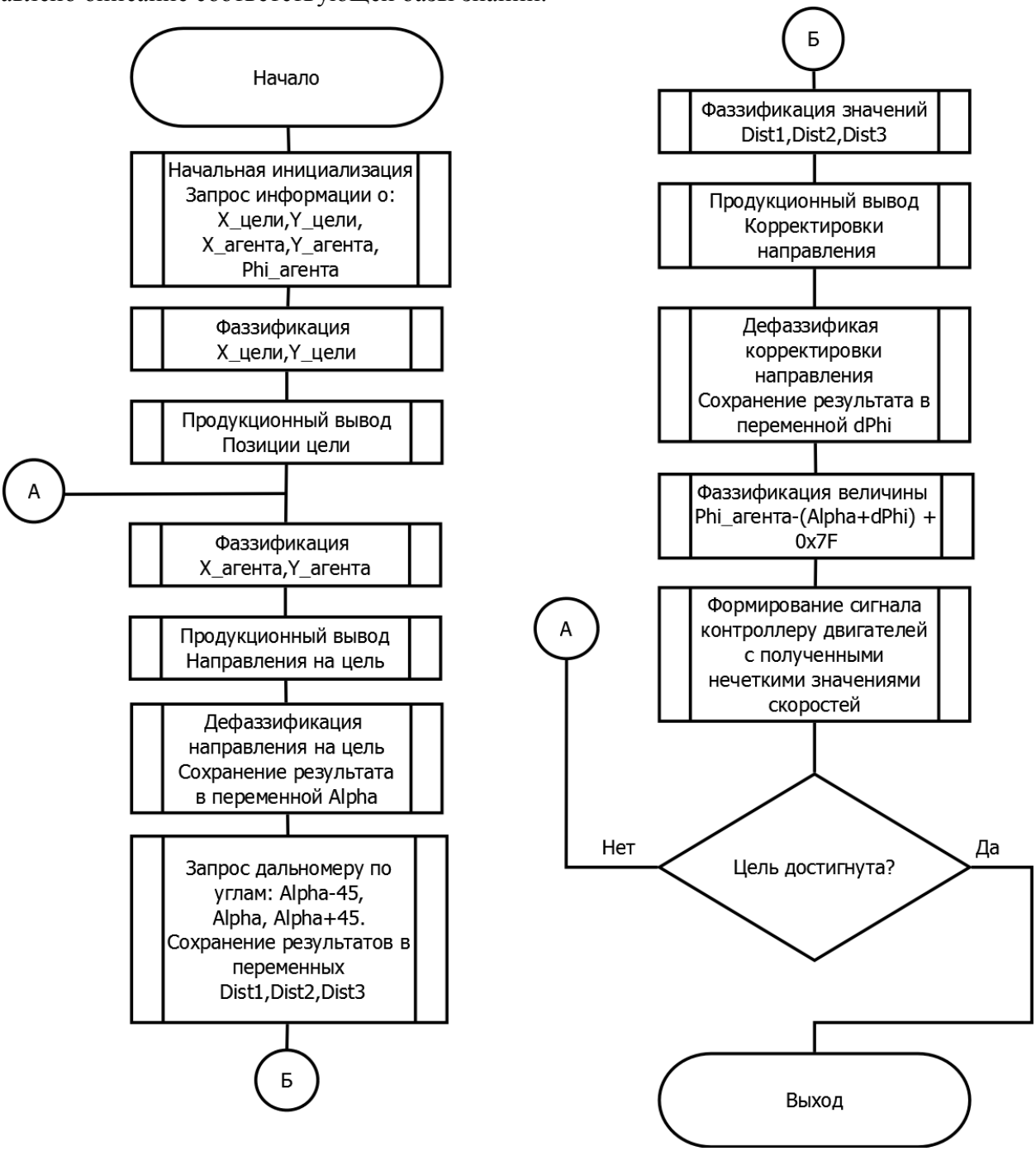

Рисунок 4.1 - Схема нечеткого алгоритма поиска пути.

5 База знаний предложенного алгоритма

В описанном в предыдущем разделе алгоритме введены различные лингвистические переменные. Ниже следует их описание с функциями принадлежности. Описание дано применительно к реальной реализации на 8-битном микроконтроллере.

1 X и Y - задают координаты в пространстве задачи.

Пространство задачи ограничено по X и по Y интервалом значений [0,255]. Нечеткое множество задается на 3 термах – «лево», «прямо», «право». Функция принадлежности представлена на рисунке 5.1.

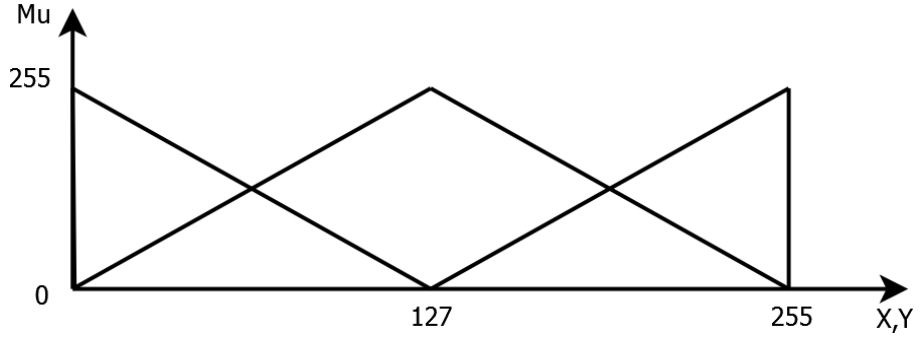

Рис. 5.1 – Функция принадлежности лингвистических переменных X и Y.

2 $P_{цели}$ и $P_{агента}$ – задают положения агента и цели в пространстве задачи.

Для экономии памяти в базе знаний, данные ЛП не имеют функций принадлежности, т.к. это промежуточные переменные, используемые в логическом выводе. Введены для более компактного задания правил в базе правил.

3 Alpha – направление до цели.

Угол задаётся в интервале от 0 до 2π. Данная лингвистическая переменная задается на 16-ти термах, каждый из которых представляет одно из возможных направлений движений робота. Функция принадлежности данной лингвистической переменной продемонстрирована на рисунке 5.2.

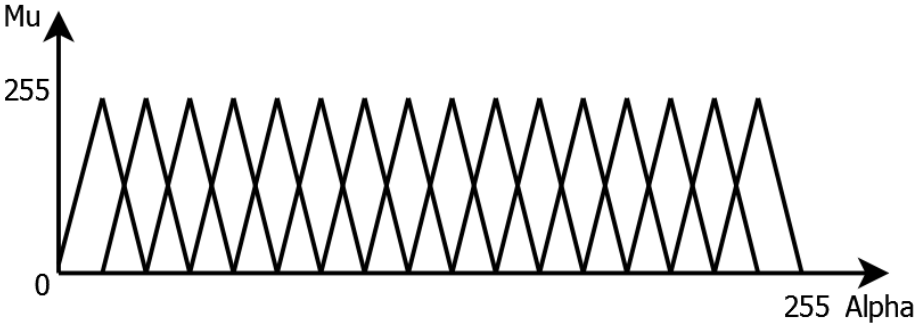

Рис. 5.2 – Функция принадлежности лингвистической переменной Alpha.

4 Dist1,Dist2,Dist3 – расстояния до препятствий по трем направлениям.

Задаются на трех термах: «близко», «средне», «далеко». От соотношения расстояний по трем направлениям зависит корректировка направления. Функция принадлежности представлена на рисунке 5.3.

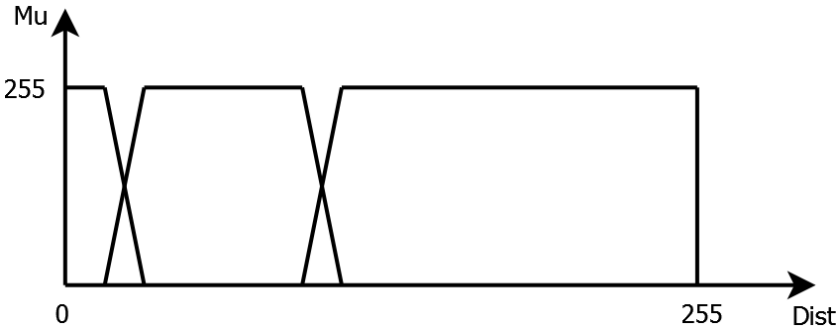

Рис. 5.3 – Функция принадлежности лингвистических перемнных Dist1, Dist2, Dist3.

5 dAlpha – корректировка направления.

Корректировка направления в зависимости от значений переменных Dist1, Dist2, Dist3. Корректировка заключается в измении угла на +/- 90 градусов. Переменная задается на 5 термах: «резко влево», «чуть влево», «прямо», «чуть вправо», «резко вправо». Функция принадлежности представлена на рисунке 5.4.

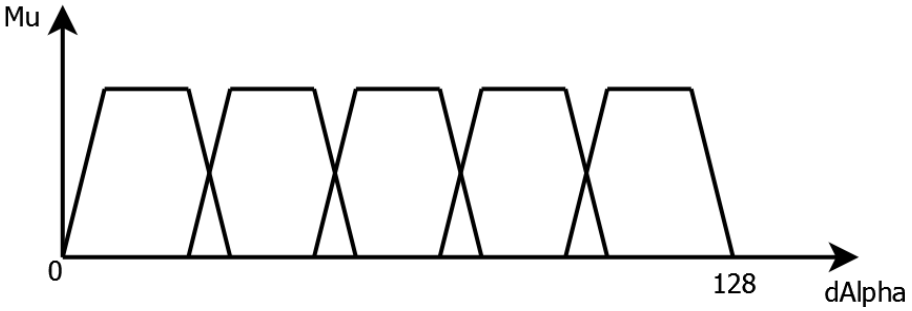

Рис. 5.4 – Функция принадлежности лингвичтической переменной dAlpha.

6 Moto – скорости для обоих моторов.

Эта переменная представляет из себя угловую характеристику Phi_агента – (Alpha+dPhi) +127, при фазификации которой получаются такие значения степеней принадлежности термам, которые могут напрямую использоваться как управляющие воздействия для скоростей 2-х двигателей. Переменная задается на двух термах: «левый двигатель» и «правый двигатель». Функция принадлежности представлена на рисунке 5.5.

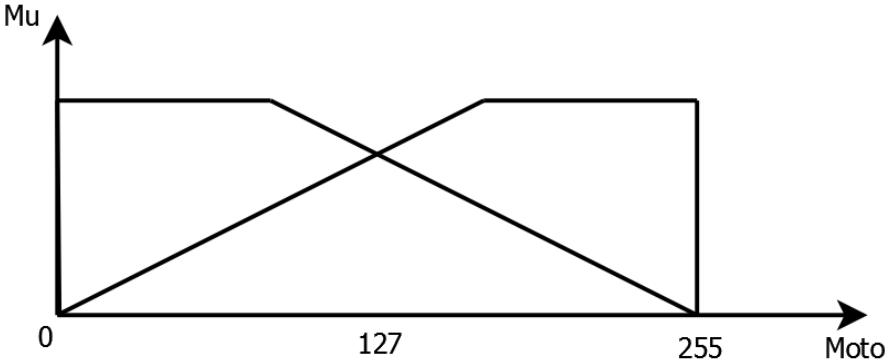

Рис. 5.5 – Функция принадлежности лингвистической переменной Moto.

База нечетких правил, используемая при выводе. Состоит из следующих смысловых блоков:

- Блок вывода позиции из значений координат.

Содержит правила вида: «Если X = «Лево», Y = «Лево», то P = «Левый нижний квадрат»»

- Блок вывода направления из значений позиций цели и агента.
  Содержит правила вида: «Если P_агента = «Левый нижний квадрат», P_цели = «Правый верхний квадрат», то Alpha = «направление на 45 градусов»»
- Блок вывода корректировки направления из значений препятствий.
  Содержит правила вида: «Если Dist1 = «Средне», Dist2 = «Близко», Dist3 = «Близко», то dAlpha = «Чуть влево»»

6 Результаты работы алгоритма

На рисунке 6.1 представлены результат работы предложенного алгоритма. Алгоритм был реализован на микроконтроллере, а тестирование проводилось на специально созданной компьютерной модели. Данная иллюстрация демонстрирует работоспособность предложенного решения и возможность его использования при решении практических задач.

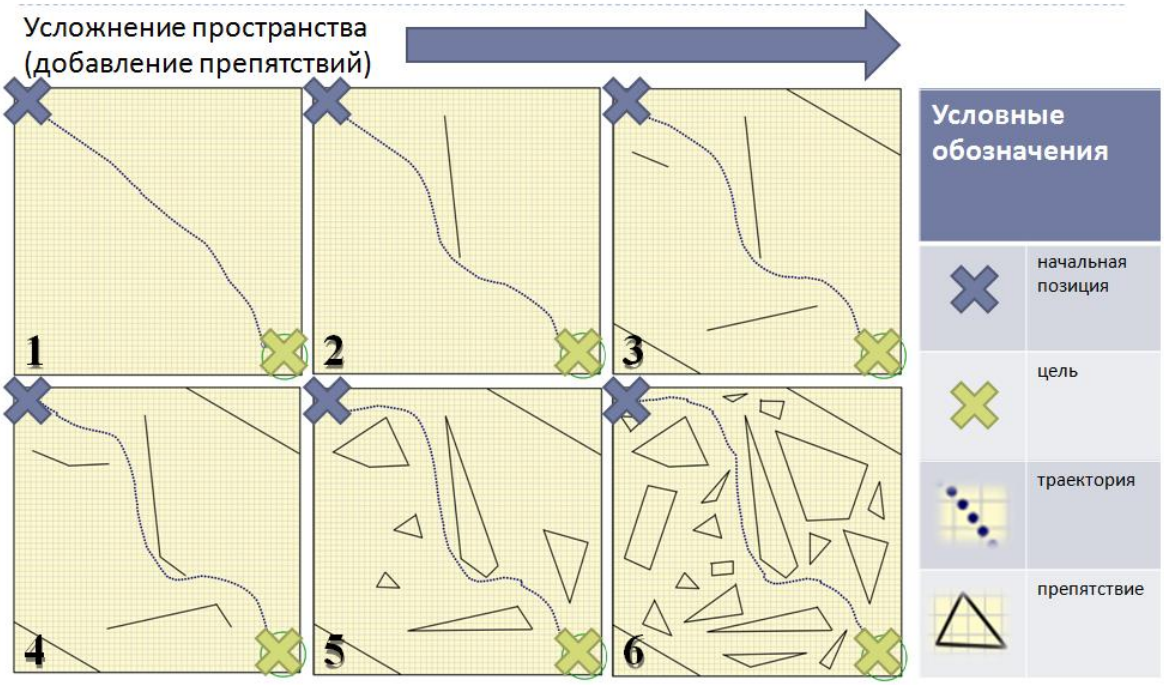

Рис. 6.1 – Пример работы алгоритма на последовательно усложняемом пространстве.

Заключение

В статье рассмотрены основные подходы к навигации мобильного робота в известном и неизвестном окружениях, а так же предложен пример реализации нечеткого алгоритма поиска пути. Используя преимущества таких алгоритмов как *Tangent Bug*, подобная система может улучшить оптимальность пути, сделать его траекторию более плавной, само управление – более естественным, а задание логики поведения агента – более удобным.

Список использованных источников

1 Рассел С., Норвиг П. Искусственный интеллект: современный подход. 2-е издание. М.: Издательский дом “Вильямс”, 2006. 1408 с.

2 Жуков С.Ю. Навигация интеллектуальных агентов в сложных синтетических пространствах : дис. канд. техн. наук. М., 2000

3 James N., An Analysis of Mobile Robot Navigation Algorithms in Unknown Environments: дис. PhD. School of Electrical, Electronic and Computer Engineering, Newcastle Univercity, 2010.

4 Lingelbach F., Path Planning using probabilistic cell decomposition: дис. PhD. Stockholm, 2005.

5 Jönsson F.M. An optimal pathfinder for vehicles in real-world digital terrain maps, The Royal Institute of Science: дис. PhD. School of Engineering Physics, Stockholm, 1997.

6 Stentz A. Optimal and Efficient Path Planning for Partially-Known Environments // Proceedings of the International Conference on Robotics and Automation, Carnegie Mellon University, Pittsburgh, 1994. С. 3310–3317.

7 Дуньюэ Ц. Управление мобильным роботом на основе нечетких моделей // Современные проблемы науки и образования. 2007. № 6. С. 115-121.

8 Гриняев С. Нечеткая логика в системах управления // Компьютерра. – 2001. – №38

9 Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры: Учеб. пособие для вузов . – 2-е изд., перераб. и доп. – М.: Изд-во МГТУ им. Н.Э.Баумана, 2004. – 400 с.: ил. – (Информатика в техническом университете).

10 Система управления движением // Интеллектуальный мобильный робот.URL. http://robot-rad.narod.ru/tactick.html (дата обращения 27.02.2012)

11 Motion planning // Wikipedia.URL. http://en.wikipedia.org/wiki/ Motion_planning (дата обращения 27.02.2012)