РОБОТЫ

Робот - это соединение программного обеспечения с оборудованием. Программное обеспечение является интеллектом, стоящим за оборудованием, и этот интеллект отличает робота от других форм автоматизации.

Существует два типа роботов. Первый тип включает, индустриальные (промышленные) сборочные роботы с фиксированным местом нахождения, типа тех, которые используются для сборки автомобилей. Этот вид робота должен функционировать только в высоко управляемой среде, которая точно разработана для него.

Второй тип - это автономные роботы. Они предназначены для работы в реальном мире. Создание истинно автономного робота требует решения ряда самых сложных задач ИИ. По этой причине автономные роботы представляют интерес для специалистов в области ИИ.

Автономный робот существенно сложнее промышленного, потому что он должен быть намного "умнее". Автономный робот работает в неуправляемой окружающей среде (т.е. не созданной специально человеком и не поддерживаемой его действиями) реального мира, поэтому ему требуется набор различных навыков, которые промышленному роботу не нужны. Например, автономный робот должен уметь видеть и слышать и понимать естественный язык, то есть распознавать то, что этот язык передает. Обеспечение робота этими способностями уже довольно сложное дело. А кроме этого, робот должен быть способным решать задачи, что является возможно наиболее трудным делом с точки зрения программирования. Робот должен уметь приспосабливаться к различным ситуациям, потому что не возможно заранее запрограммировать робота на все возможные случаи.

Кроме технических проблем существуют правовые вопросы, касающиеся автономных роботов. Если робот, например, совершает преступление, случайно или преднамеренно, то является ли робот виновным или виноват его владелец? Возможно виновной стороной является изготовитель робота или лицо, которое запрограммировало его. Кроме того, имеет ли автономный робот какие-нибудь права или он просто механический раб? Эти вопросы необходимо решить.

1. Рука робота

Фактически все промышленные роботы - это просто руки робота с зажимами на конце. Рука робота - это основной действующий элемент любого робота, как промышленного, так и автономного. Необходимо понять несколько сложных задач, с которыми обычно сталкиваются при управлении рукой робота. Например, когда человек берет стакан с водой, перемещение кажется легким и без каких-либо мысленных усилий. Однако, на самом деле это сложная задача, которая требует координации отдельных мускулов. Для младенца требуется несколько месяцев, чтобы научиться делать одно и то же действие.

В основном рука робота моделирует человеческую руку. Большинство рук роботов имеют шесть осей движения (шесть степеней свободы), это обеспечивает самую большую свободу перемещений. На рисунке 1 показан эскиз руки с шестью осями. Каждая ось (сочленение сустав), использует собственный двигатель или, в случае больших рук, гидравлический цилиндр.

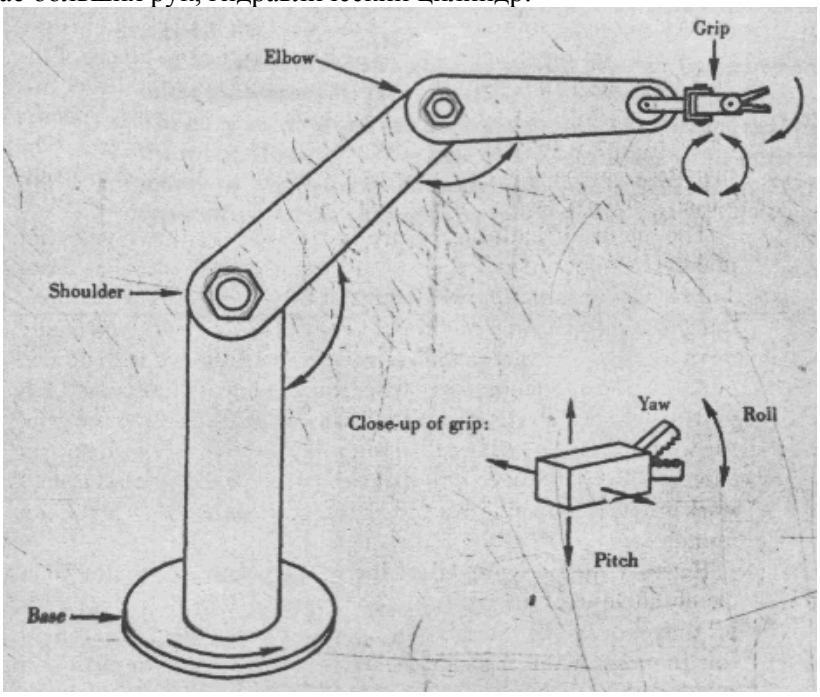

Рис. 1. Эскиз руки робота с шестью осями

Как показано на рисунке 1, рука робота с шестью осями фактически состоит из двухкоординатных систем. Основа и две части руки позволяют перемещать ее в пределах системы координат X-Y-Z. Поэтому, при достижении любой точки в пространстве требуется использования пяти соединений руки. Перемещение по шестой оси - вращение зажима, влияет только на ориентацию.

Наибольшая трудность при управлении перемещением такого типа руки не в достижении точности движений - оборудование позаботится об этом, а в координировании действий шести соединений. Например, представим, что необходимо, чтобы рука робота попала во внутрь частично открытого окна, и что робот первоначально стоит с вытянутой рукой в сторону от окна. Если перемещать каждый сустав независимо, начиная с основы и до зажима, то окно, скорее всего, будет разбито, потому что вращение робота с вытянутой рукой приведет к удару в окно. Чтобы работа происходила должным образом, все соединения должны работать согласованно, как и человеческая рука.

Однако, только одновременным перемещением всех составляющих руки нельзя полностью решить все задачи. Например, часто необходимо для робота протянуть зажим к предмету по прямой линии, чтобы достать его внутри узкого объекта. Выполнение этого действия требует использования довольно сложной тригонометрии. На рисунке 2 показано графическое представление этой ситуации. Для перемещения зажима в точку C', суставы A и B должны одновременно быть в движении. Движение сустава A заставляет руку вытягиваться, а также двигаться вниз. Это требует компенсации с помощью движения сустава B. Кроме того, длины участков руки могут быть различными, и тогда суставам A и B придется двигаться с различными скоростями, чтобы получилось переместить руку именно по прямой линии. Задачи того же типа, приходится решать в более широком масштабе, если, робот должен полностью изменить свою ориентацию в пределах ограниченной пространственной площадки. Например, много компенсирующих движений может потребоваться, если робот должен работать вблизи угла.

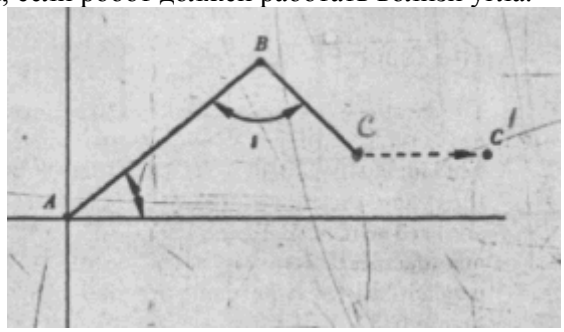

Рис. 2. Робот протягивает зажим к предмету по прямой линии

Вариантом задачи движения по прямой линии является случай, когда робот должен переместить зажим по горизонтальной прямой перед собой. Единственный способ, которым это можно выполнить - поворот основания робота. Однако, поскольку основание вращается, то другие суставы должны компенсировать это движение. Все эти типы задач можно решать с помощью применения относительно простых тригонометрических преобразований, но вычисления должны быть выполнены в режиме реального времени.

2. Промышленный робот

В области робототехники, основные усилия прилагались к созданию и улучшению промышленных сборочных роботов. Поскольку эти роботы используются в управляемой среде (среде специально созданной и поддерживаемой человеком), то они могут быть существенно менее разумными, чем автономные роботы. В настоящем и обозримом будущем промышленные роботы могут исполнять только те задачи, на решение которых они были явно запрограммированы. Есть два способа обучения робота новым навыкам: 1) их можно обучить, используя обучающий терминал или 2) они могут быть запрограммированы, с использованием языка управления роботом.

2.1. Обучающий терминал

Наиболее общий метод программирования робота состоит в том, чтобы решить новую задачу с помощью обучающего терминала. Обучающий терминал это ручной пульт управления, который позволяет оператору приводить в движение различные суставы робота. (Это подобно пультам управления, которые используются для игрушек-моделей автомобилей, лодочек и самолетов.)

Обучающий терминал связан с роботом через компьютер, управляющий роботом. Если необходимо научить робота выполнять определенное действие, то обучающий терминал позволяет направлять движение робота по траектории, которая обеспечивает выполнение задачи. Каждое движение любого сустава компьютер сразу же записывает как новое положение. После того, как задание выполнено – обучение заканчивается и теперь робот может выполнять эту работу самостоятельно, без дальнейшей помощи.

2.2. Язык управления роботами

Обучающий терминал - превосходный метод для обучения робота простым задачам типа сварки и покраски. Но когда работа становится более сложной и синхронизация движений становится более важной, или же, если робот должен распознавать и отвечать различным образом в различных возможных ситуациях, то обучающий терминал быстро теряет свою эффективность. По этой причине были разработаны языки управления роботами.

Язык управления роботами (ЯУР) - компьютерный язык, который позволяет разрабатывать программы управления роботом. В дополнение к обычному перечню команд, таких как операторы циклов или условные операторы, язык также включает команды управления движением робота. Команды управления движениями отличают ЯУР от множества других универсальных языков программирования. ЯУР имеет встроенную базу данных, которая содержит пространственную информацию относительно каждого перемещения, которое робот может выполнить.

ЯУР не предназначен для замены обучающего терминала, а скорее дополняет его. Следовательно, ЯУР должен иметь возможность связи с обучающим терминалом. Типичный метод обучения состоит в том, что, используя обучающий терминал, роботу передается необходимая пространственная информация, а затем используется ЯУР для составления предписания, как робот должен применять эту информацию. В общем случае, каждому отдельному положению приписывают символьное имя, по которому программа сможет обращаться к нему.

Типичный ЯУР имеет синтаксис, подобный языку BASIC. Хотя он не очень хорош, но этот синтаксис легок для освоения. (Но уже разработаны более структурированные, паскаль-подобные языки). Первый и наиболее широко используемый ЯУР - VAL, он был разработан компанией Unimation Corporation. Чтоб получить представление о VAL, рассмотрим пример. Его можно использовать его для описания действий робота, который удаляют коробки с конвейерной ленты.

```
10 WAIT 2        //робот ожидает, пока по линии 2 не поступит сигнал, означающий, что предмет на
                 //конвейере и находится в области действия робота
  MOVE POS1      //робот перемещает руку в позицию над конвейером (имя позиции POS1)
  MOVE POS2      // робот опускает руку на предмет, лежащий на конвейере
  CLOSEI         //робот закрывает зажим (берет предмет с конвейера)
  MOVE BOX1      // робот перемещает предмет в коробку (имя позиции BOX1)
  OPENI          //робот открывает зажим и оставляет предмет в коробке
 GOTO 10         //переход в состояние ожидания подхода следующего предмета на конвейере
```

Отрабатывая эту программу, робот будет ждать, пока по линии номер 2 не поступит сигнал, означающий, что предмет на конвейере и находится в области действия робота. Затем, робот будет двигаться к предмету, опустит открытый зажим к его вершине. Затем закроет зажим, переместит предмет с конвейера и выпустит предмет, открыв зажим. Далее осуществляется циклический переход к ожиданию появления следующего предмета. Три положения POS1, POS2 и BOX1 - символьные имена местоположений, которые были введены при помощи обучающего терминала.

Все промышленные роботы обеспечены некоторыми способами доступа к входным и выходным сигналам синхронизации; например, команда WAIT в примере использует только входной сигнал 2 (линия 2). Промышленные роботы вообще не имеют никаких встроенных датчиков. Если робот нуждается в некотором виде датчиков, то они выполняются в виде самостоятельного прибора, который просто соединяется с роботом посредством линии входных сообщений. Программист обязан должным образом интерпретировать значение сигнала, поступающего с такой линий.

4. Имитатор робота

Совсем не обязательно иметь робота, чтобы проводить эксперименты в области робототехники. Оборудование, из которого состоит робот, несущественно, потому что поведение робота определяет программное обеспечение. Без программного обеспечения робот - просто дорогая "куча железа".

Здесь рассматривается довольно простой имитатор, который позволит увидеть, что такое программирование робота.

Имитатор представляет собой программу, которая содержит синтаксический анализатор ЯУР; базу данных для хранения пространственной информации, программы моделирования движения и отображения на экране; простой редактор, и ряд сервисных программ. Этот комплекс также включает распознаватель Дельта-D. Имитатор может стать отправной точкой для дальнейшего развития и расширения. Перед исследованием программы, необходимо понять, что она делает и каковы ее ограничения.

4.1. Описание имитатора робота

Среда имитатора робота представляет собой мир, в котором есть только четыре объекта: квадрат, два треугольника, и сам робот. Остальная часть экрана - пустое место. Квадрат, треугольники и

распознаватель Дельта-Д - те же самые, что рассматривались в распознавании образов - здесь они входят в состав имитатора. Робот представлен на экране символом #. (Так как графический режим программа не использует, то ее можно запускать на любом компьютере.) Робот может идти куда угодно, если на его пути не встречается никаких предметов. Квадрат и треугольники - его препятствия, и робот должен их обходить.

Робот может ощутить препятствие, находясь в непосредственной близости, и распознавать треугольники и квадрат. Имитатор управляется с помощью следующего меню:

Редактирование программы (**Edit**)
Обучение робота (**Teach**)
Запуск программы (**Run**)
Выход (**Quit**)

Существует два способа программирования робота. Первый - применение непосредственно ЯУР и второй - использование обучающего терминала. ЯУР простой, но легко расширяемый и состоит из следующих команд:

moveto <строка столбец>
move <направление>
movep <название точки>
findt
finds
goto <название метки>
ifsense <направление> then <оператор>

Команда "moveto".

По команде moveto робот должен идти в точку на экране с координатами X,Y. Верхний левый угол экрана имеет координаты 0,0. Это означает, что строки пронумерованы от 0 до 24, а столбцы от 0 до 79. Например, команда

moveto 60 12

заставляет робота передвинуться в позицию 60,12. Заметим, что запятая не используется как разделитель между числами-аргументами функции; Вы должны использовать пробел.

Команда "move".

Команда move заставляет робота переместиться на одну позицию в направлении, определенном аргументом. В качестве значений аргумента используются ключевые слова **left, right, up, down**. Например, команды

move up
move left

заставляют робота продвинуться на одну строку вверх и затем на одну позицию влево. Направления относительно экрана соответствуют тому как мы на него смотрим: верх экрана - top, а низ - down.

Команда «movep».

Команда movep сообщает роботу, что он должен переместиться в позицию, о которой он предварительно узнал, обучаясь посредством терминала. Все названия позиций начинаются с ключевого слова **point**, за которым следует целое число от 1 до 200, указывающее на номер позиции. Например, следующий фрагмент программы перемещает робота в соответствующие три позиции:

movep point1
movep point12
movep point2

Команды findt и finds.

Они сообщают роботу, что он должен найти треугольник или квадрат, соответственно. Чтоб это выполнить, робот использует распознаватель Дельта-D. Робот может найти предмет, только если он сам расположен выше этого предмета. После того, как он найдет предмет, робот расположится рядом с ним, на один столбец влево от предмета, в его верхнем левом углу.

Команда goto.

По команде goto осуществляется переход к одной из меток в программе. Например,

goto five

передает управление оператору с меткой five. Метки могут состоять из любой комбинации печатных символов, но они не должны совпадать ни с одним из ключевых слов ЯУР.

Команда ifsense.

Это единственный управляющий оператор (оператор перехода по условию), используемый в программе. Он определяет, есть ли объект справа, слева, выше, или ниже робота. Если объект есть, то робот выполняет действие, которое следует после **then**; если никакого объекта нет, то переход к выполнению следующей строки программы. Программист должен точно определить направление поиска, используя ключевые слова **up, down, left, right**. Например, этот фрагмент программы заставит робота двигаться вдоль стороны объекта:

```
Loop1 move down
    ifsense right  then  goto loop1
```

4.2. Редактор

Для подготовки и ввода программы на ЯУР используется редактор. Редактор прост: он позволяет, вводить символьные строки до тех пор, пока не будет введена пустая строка. Никаких других функций он не выполняет. Это сделано для того, чтобы сократить размер и так очень длинного варианта учебной программы. (Однако Вы можете самостоятельно добавить другие функции, или Вы может использовать отдельный редактор текста и создать процедуру загрузки файлов). Массив символов prog содержит программу, написанную на ЯУР.

4.3. Обучение Робота

Команда **teach** вызывает окружающую среду - экран, на котором расположены квадрат и треугольники. Робот помещен в начальное положение 0,0. Для перемещения робота можно использовать числовую клавиатуру или клавиши со стрелками. Если нажать клавишу INS, то компьютер сохраняет текущую позицию робота в пространственной базе данных и высвечивает сообщение

point<N> stored

в левом нижнем углу экрана. N - номер из интервала [1, 200]. Каждый раз при нажатии клавиши INS номер-название позиции будет увеличен на единицу. Так можно обучить робота (заложить в него) некоторому количеству поименованных пунктов, в которые затем он сможет направляться по имени позиции. Чтобы прекратить процесс обучение необходимо нажать клавишу END.

Поскольку обучающие процедуры используют стандартную функцию getch(), то необходимо использовать клавишу NUMLOCK, потому что getch() не может непосредственно считывать значения клавиш со стрелками. (Это требует обращения системы к BIOS - ненужное осложнение для уже сложной программы).

Функция teach(), показанная здесь, читает символы со вспомогательной клавиатуры. Таким образом, вспомогательная клавиатура выполняет роль обучающего терминала для моделирующего устройства робота. Вспомогательная функция get_point() возвращает индекс свободного участка в базе данных point.

Функция move() перемещает робота в позицию. Она возвращает 1 если перемещение возможно, и 0, если оно не возможно.

4.4. Запуск программы

Функция run() является главным циклом, который управляет вспомогательными процедурами, выполняющими операторы программы.

Каждый раз в цикле, get_token() возвращает лексему и соответствующий оператор выполняется. Get_token () - это лексический анализатор. Так как программа в оперативной памяти хранится в виде символьной строки, то лексема, равная '\0', означает, что компьютер достиг конца программы и цикл while завершается.

Когда имитатор сталкивается с командой moveto, то вызывается функция moveto_setup(), которая читает координаты точки назначения и подтверждает, что они находятся в пределах экрана. Затем вызывается moveto(), которая состоит из внешнего цикла while и двух внутренних циклов while, они выполняются, пока робот не достигнет предназначенного места. Сначала, робот перемещается по Y-координате, затем - по X-координате. Если никакие предметы не блокируют путь робота, то этот процесс прост. Однако, если робот должен обойти вокруг предмета, то программа вызывает функцию find_path_down() или find_path_right(), или обе. Они заставляют робота перемещаться вокруг предмета, проходя вниз или вверх, используя find_path_down() или, перемещаясь влево или вправо, используя find_path_right().

5. Использование моделирующего устройства.

Когда Вы запускаете программу-имитатора, то сначала выводится меню:

```
choose:  (E)dit, (R)un, (T)each, (Q)uit:
```

Чтобы ввести программу, нажмите E, и вводите ее операторы. Когда все команды введены, то введите пустую строку. Например, введите следующую программу:

```
moveto 70 12
moveto 0 0
finds
```

Введите пустую строку. Затем, в меню выберите опцию run. Появится изображение окружающей среды и Вы увидите робота, перемещающегося согласно инструкциям.

Когда Вы запустите программу, Вы увидите, что робот правильно идет вокруг предметов, которые находятся на его пути. Вы должны изучить функцию moveto, чтобы увидеть, как это выполняется. Если Вы внесете небольшие изменения в функции move(), как обозначено в комментарии, то робот будет оставлять след своих перемещений.

Вот другой пример: следующая программа заставит робота следовать по стороне квадрата.

```
  finds
10 move down
  ifsense right then goto 10
```

Пока объект остается справа от робота, он продолжает двигаться вниз.

Следующий шаг, который Вы должны сделать - пробовать обучить робота, используя команды обучения. Помните, что Вы должны ввести предназначенную для этого команду movep в вашу программу, чтобы использовать то, чему Вы обучили робота.

6. Расширение моделирующего устройства

В моделирующее устройство не включены другие функциональные возможности. Однако для вас не составит большого труда добавить их самостоятельно.

Особенно интересно добавить другие типы объектов, состоящие из одного символа, которые робот сможет подбирать. Это потребует добавления команд закрытия и открытия зажима, а также оператора, позволяющего роботу нащупать предмет. Можно заставить робота искать рассыпанные шарики. Можно добавить переменные в имитатор, создавая массив структур следующей формы:

```
struct variable {
      char name[10];
      int value;
} var[MAX];
```

Этот массив позволит Вам использовать простую команду IF и счетчики циклов. Однако, чтобы программа вычисляла сложные арифметические выражения, требуется использовать их анализатор.