11 INTRODUCTION TO ROBOT PROGRAMMING

11.1 Robot Programming Fundamentals

The fundamental steps involved in robot programming are:

1) Define the task and break it down into sub-tasks.
2) Describe the sub-tasks in detail.
3) Define the location points required for each subtask.
4) Convert the information from steps 1-3 into program instructions.
5) If possible use robot simulation software to check your program.
6) Run your program with the robot's speed set to be slow. This will allow you to carefully monitor the robot's actions and to safely stop it if necessary.
7) Based on steps 5-6 revise your program if necessary.

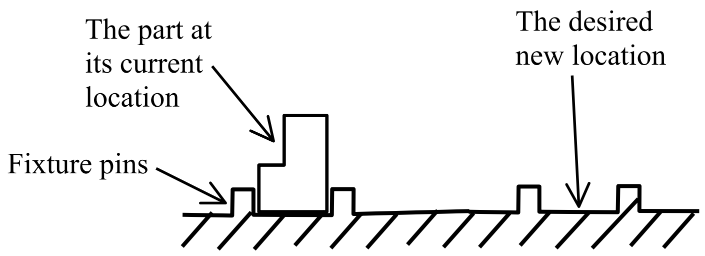

Figure 11.1 A simple pick and place operation.

For example, let's say the task is a simple pick and place operation (recall section 1.7). We want to move a part from its current location to a new location as illustrated in Figure 11.1. We assume the robot is equipped with a gripper capable of grasping this part. This task definition can be broken into the sub-tasks:

a) move the gripper to the vicinity of the "pickup location",
b) pickup the part,
c) move the gripper (now holding the part) to the vicinity of the "drop-off location" and
d) drop-off the part.

Regarding step 2) of the programming procedure, sub-task (a) requires moving to a location above the pickup location with the gripper centred over the part, and aligned with the part. Sub-task (b) requires opening the gripper, moving it down so that its jaws surround the part, closing the gripper, and moving it upwards until it is clear of the fixture pins. Sub-task (c) requires moving the gripper until the part is centred above the drop-off location. Sub-task (d) requires moving the gripper downwards until the part is at its drop-off location, opening the gripper to release the part, and moving the gripper upwards until it is above the part.

Regarding step 3) of the programming procedure, there are two methods for generating the point locations. With "on-line programming" or "teach programming" each location is generated by manually moving the robot until the desired position and orientation of the tool is reached, and then storing the current position/orientation of the arm (This process known as "teaching"). This method has the advantages:

- The robot does not have to be very accurate, it only has to have good repeatability.
- CAD models of the robot and its environment are not required.

and the disadvantages:

- Production must be stopped to allow the program locations to be taught. This can be lengthy (and expensive) if the task is complex (*e.g.* arc welding).
- The programmer may have to stand close to the robot while teaching it the locations, making safety an issue.

The other option is termed "off-line programming". With this method the locations are generated automatically from CAD models of the robot and its environment (this includes parts, fixtures, conveyors, etc.). Off-line programming has the advantages:

- Production does not have to be stopped to allow the point locations to be generated.
- The programmed motions of the robot (and other equipment in the workcell) can be simulated for safe debugging.
- No need to stand close to robot.

and the disadvantages:

- The robot must be accurate.
- Accurate CAD models of the robot and its environment are required.
- Expensive software is required.

In industry, some companies are starting to use off-line programming but on-line programming is much more common.

Before we can continue with step 4), we must learn the instructions for a specific robot language.

11.2 The V+ Programming Language

The first widely used robot was manufactured by Unimation. This robot was programmed using a language called "VAL". This was superseded by "VAL-II", and more recently by "V+". The V+ language is used by new robots made by the companies Adept and Staubli. Many modern robot languages were also derived from VAL and are similar to it. For these reasons only the V+ language will be covered in this course.

The V+ language is an interpreted language (it executes like Matlab and not like compiled languages such as C). This makes debugging easier. It also allows most program instructions to be run one at a time using a command prompt.

11.2.1 Commonly Used V+ Language Instructions

Robot motion instructions:
APPRO <location>, <distance>

Moves the tool to the position and orientation described by <location> offset along the resultant z axis of the tool by <distance> mm. A positive <distance> sets the tool back from the specified location while a negative <distance> sets the tool forward. The motion is performed in joint space.

APPROS <location>, <distance>
Same as APPRO except that the motion is performed in Cartesian space (*i.e.* straight line motion).

BREAK
This instruction interrupts the program execution until the robot reaches its current desired destination. In doing so it also prevents the robot controller from blending the motion segments together.

DELAY <time>
Causes the robot motion to stop for the specified period of time in seconds.

DEPART <distance>
Moves the tool by <distance> mm along the tool z axis. A positive <distance> moves the tool back while a negative <distance> moves the tool forward.

DEPARTS <distance>
Same as DEPART except that the motion is performed in Cartesian space (*i.e.* straight line motion).

HERE <location>
Stores the current location of the robot in the variable <location>.

OPEN, CLOSE
Causes the gripper to open/close during the next motion segment.

OPENI, CLOSEI
Causes the gripper to open/close immediately and the motion stop after the current move.

MOVE <location>
Causes the robot to move to the specified location using joint space motion.

MOVES <location>
Causes the robot to move to the specified location using LSPB Cartesian space motion.

SPEED <speed> ALWAYS
Sets the motion speed to be a percentage of the maximum.

Other instructions:

SIG <signal_number>
Returns the logical state of the specified digital input. This may be used to read signals from other hardware devices (e.g. safety devices, part presence sensors, conveyors, PLCs, etc.)

SIGNAL <signal_number>
Sets the specified digital output signal as follows:
- output to change = abs(signal_number)
- if signal_number>0 then turn output on
- if signal_number<0 then turn output off.

This command may be used to communicate with other hardware devices such as conveyors or PLCs.

WAIT <logic_condition>
Suspends program execution until the specified condition becomes true.

11.2.2 Example Programs

Example 11.1

We'll start by completing the simple pick and place example. It is assumed that the "pickup" and "drop-off" locations have been stored in the location variables "pickup_point" and "dropoff_point". These could have been created either by off-line or on-line programming. With V+, on-line programming may be done by manually moving the robot to the desired location and saving this location by typing the "HERE <location>" instruction at the command prompt, *e.g.* >HERE pickup_point

Recalling the simple pick and place operation, the first task is to move the gripper to a point directly above the part at its pickup location. Next the gripper should be opened and the moved down so its jaws surround the part. This is accomplished by the program instructions:

```
APPROS pickup_point, 100
OPENI
MOVES pickup_point
```

To close the gripper and pick up the part:

```
CLOSEI
DEPARTS 50
```

Note that the 100 mm and 50 mm values used with the APPROS and DEPARTS instructions are dependent on the size of the part and the size of the fixture pins, respectively.

Next we want to move the gripper to a spot above the drop-off point, move downwards, release the part, and finally back away. The corresponding instructions are:

```
APPROS dropoff_point, 50
MOVES dropoff_point
OPENI
DEPARTS 100
```

The complete V+ program is written:

```
PROGRAM example1
    SPEED 50 ALWAYS
    APPROS pickup_point, 100
    OPENI
    MOVES pickup_point
    CLOSEI
    DEPARTS 50
    APPROS dropoff_point, 50
    MOVES dropoff_point
    OPENI
    DEPARTS 100
END
```

Note that the SPEED instruction has been used (line 2) to set the motion speed to 50% of maximum.

Example 11.2

In this example we'll compare programs for continuous motion vs. discontinuous motion. We assume that trajectory points named A, B and C have already been defined. If blended Cartesian space straight line motion is desired from A to C the program would be simply:

```
PROGRAM example2a
   MOVES A
   MOVES B
   MOVES C
END
```

The resulting motion would be continuous and B is a via point.

Alternatively, if we wish to stop at point B we use the BREAK command as follows:

```
PROGRAM example2b
   MOVES A
   MOVES B
   BREAK
   TYPE "Got to point B"
   MOVES C
END
```

Example 11.3

In this example a robot is being used to transfer parts from a conveyor to another machine. The robot must wait for the conveyor to stop in the right position before it can pickup the part. The conveyor must also wait for the part to be picked up before it can start moving again. Digital input and output signals are used for this communication. The pickup and drop-off locations have been previously defined.

```
PROGRAM example3               Comments
   SPEED 30 ALWAYS             Sets speed to be 30% of max.
   SIGNAL -3                   Set the signal to the conveyor to be false (output)
   APPROS pickup_point, 75     Move robot to a "ready to grip" position
   OPENI                       Wait for motion to complete and then open gripper
   WAIT SIG(5)                 Wait for the ok signal from the conveyor (input)
   MOVES pickup_point          Move gripper down to part
   CLOSEI                      Wait for motion to complete and then close gripper
   DEPARTS 150                 Lift part off conveyor
   BREAK                       Wait until the part is clear of the conveyor
   SIGNAL 3                    Send true restart signal to the conveyor (output).
   APPROS dropoff_point, 50    Approach the drop-off point
   MOVES dropoff_point         Move to the drop-off point
   OPENI                       Wait for motion to complete and then release the part
   DEPARTS 100                 Back away 100 mm
END
```

Note that this program only picks up a single part. However, the robot controller can be set to run this program as many times as desired (including indefinitely).

Example 11.4

This example was adapted from the book by Klafter *et al.* [1]. The robot is required to pick up blocks from the location defined by "PICKUP" and then stack four of them on top of each other. Please see Figure 10.2 for an illustration. This is a basic form of the palletizing operation we discussed in section 1. Note that the height of each block is 75 mm. The program is fairly straightforward, so only a few lines will be described in detail. In line 8, location B is made equal to the location "DEPOSIT" using the "SET" command. DEPOSIT is the drop-off location for the first block in the stack. In lines 9 and 24 a "for-next" loop is created. In line 23, location B is shifted by 75 mm in the Z direction using the "SHIFT" command. This allows the blocks to be stacked properly.

```
      PROGRAM STACK
1.        REMARK
2.        REMARK   THIS PROGRAM PICKS UP PARTS FROM A FIXED
3.        REMARK   LOCATION CALLED "PICKUP", THEN DEPOSITS THEM AT A LOCATION
4.        REMARK   CALLED "B". IT IS ASSUMED THAT FOUR PARTS ARE TO BE STACKED
5.        REMARK   ON TOP OF ONE ANOTHER.
6.        REMARK
7.        OPENI
8.        SET B=DEPOSIT
9.        FOR COUNT=1 TO 4
10.           APPROS PICKUP, 200.00
11.           MOVES PICKUP
12.           CLOSEI
13.           DEPARTS 200.00
14.           APPRO B, 200.00
15.           MOVES B
16.           OPENI
17.           DEPARTS 200.00
18.           REMARK      COUNT INDICATES THE TOTAL NUMBER OF ITEMS STACKED
19.           TYPE COUNT
20.           REMARK      MOVE THE LOCATION OF B UP BY 75 mm TO ACCOUNT
21.           REMARK      FOR THE HEIGHT OF EACH BLOCK
22.           SHIFT B BY 0.0, 0.0, 75.0
23.       END
24.       TYPE  "END OF STACK PROGRAM"
      END
```

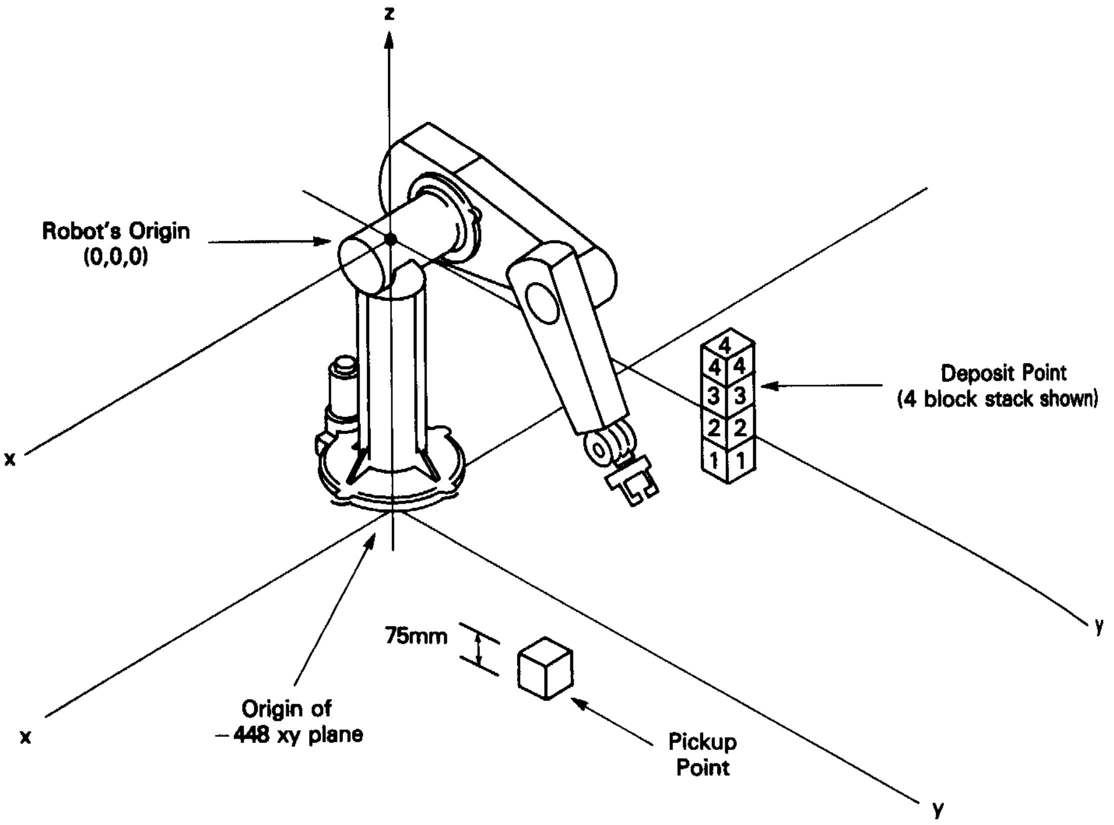

Figure 11.2 The robot and its environment used for example 11.4 [1].

11.3 Concluding Remarks

This section is meant to introduce only the fundamentals of robot programming. Robot programming is a complex job since it involves the control of the robot, its end effector and other workcell devices. The motions of these machines are a function of time and must be carefully coordinated to prevent failures from happening. In addition, worker safety must be ensured by following appropriate safety guidelines. Robot manufacturers offer courses for learning robot programming in greater detail.

References

1. R.D. Klafter, T.A. Chmielewski, M. Negin, "Robotic Engineering: An Integrated Approach", Prentice Hall, 1989.