WARSAW UNIVERSITY OF TECHNOLOGY

ROBOT PROGRAMMING METHODS

05 June 2017

DEBALEENA MISRA
EMARO 174/EM

Robot Programming Methods
Prof. Cezary Zieliński

1. INTERNAL STRUCTURE OF THE AGENT

The agent is an embodied agent, capable of perceiving its environment through the receptors (floor and wall sensors) and acts upon the environment by means of an end-effector realised by a 6-dof manipulator. It is integrated with an internal control sub-system and a transmission buffer to achieve the goal tasks. A schematic representation is shown below:

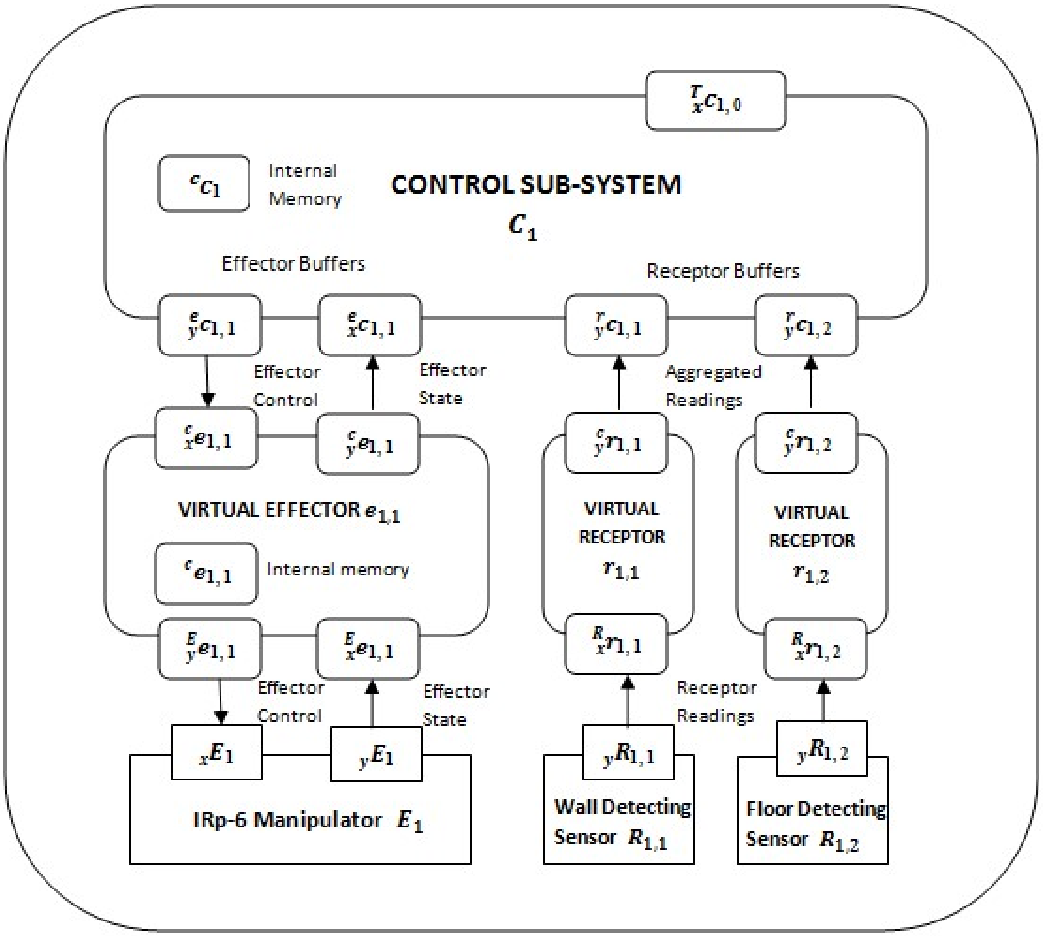


Figure 1: Internal Structure of the embodied agent

The agent is composed of three types of entities:

- Virtual effector processing the readings of the 6-dof manipulator and controlling the manipulator motors
- Virtual receptor composing the readings of the touch sensors $(R_{1,1}, R_{1,2})$
- Control subsystem responsible for all the processing involved in solving the task.

The control subsystem communicates with other agent which is the user or the agent that handles the wireless input from the user.

2. SAMPLING RATES OF THE AGENT'S SUB-SYSTEMS

Sampling rate is the rate at which samples of an analog signal is taken to convert to digital form, and is expressed as samples per second, or Hertz. In our task, we have binary touch sensors as receptors and a 6-dof manipulator as the effector.

Assuming a maximum speed of 1m/s for the end effector and a maximum acceleration-deceleration of 50m/s, the end effector will have a stopping distance of 1cm. Also assuming, the touch sensors have a range of 2cm, then the control system needs to be able to react within 1cm at maximum speed or in less than 10ms. At least 2 samples are needed within this period, or $\frac{2\ samples}{10ms} = 200$ samples/s.

We can define a sampling rate of 250 samples/s or a sampling period of 4ms for all subsystems

Thus, ${}^{c}T = {}^{r}T = {}^{e}T = 4ms$

3. GENERAL BEHAVIOUR OF THE AGENT SUB-SYSTEMS

The control loop for the agent is closed to the environment. The sensory information from the real receptors $R_{j,l}$ (floor and wall detecting sensors) are aggregated by the virtual receptors $r_{j,l}$ and transmitted to the control sub-system that generates suitable commands to the virtual effectors e_j that translate them into signals to drive the real effector E_j (manipulator).

So, the virtual receptors r_j and virtual effectors e_j basically convert the raw sensor readings and motor commands into a form that is required by the control sub-system to perform the task.

All the sub-systems use communication buffers to transmit or receive information to/from other components.

– Behaviour of the Virtual Receptors

The virtual receptors present to the control sub-system the environment information in a form that is appropriate for control purposes. Information aggregation is the fundamental responsibility of the virtual receptors, in extraction of required information from complex receptors, and gathering and processing readings obtained from them.

In the given task, there are five real receptors that are sub-divided into two categories of binary touch sensors: (a) Wall-detecting, and (b) Floor-detecting. These touch sensors give binary information (contact or non-contact). They can be operated by the use of high or low bits. Thus, we can assume each of them to be of one bit in an 8 bit register. The 3 most significant bits will be of no use to us.

– Behaviour o f the Virtual Effectors

The virtual effectors act as a link between the control sub-system and the real effectors acting on the environment. It performs a two-fold task:

(a) The control sub-system acquires proprioceptive information from the end-effector manipulator, in a form that maybe used in the control algorithm. So, the virtual effectors transform the encoder readings into a homogenous matrix representing the position and orientation of the end-effector wrt a world coordinate frame.

(b) The control values as output from the control sub-system also appears as homogenous matrices which are transformed into PWM ratios that maybe implemented by the effector hardware, i.e the 6-dof manipulator.

Some computations might need values from the earlier control cycles, thus internal memory is also involved.

4. DATA STRUCTURES (BUFFERS) WITH THE CONTROL SUB-SYSTEM

${}^{T}_{x}c_{1,\,0}$ - Commands received (SET or START/STOP) in the form of a single word

Buffers received from/to the virtual Effector

${}^{e}_{y}c_{1,\,1}\left[{}^{0}_{E}T_{\,d,c}\,, n_d\right]$ – Desired relative Cartesian pose, and number of virtual effector steps in which the pose will be attained

${}^{e}_{x}c_{1,\,1}\,[{}^{0}_{E}T\;{}_{c}\,]$ - Current absolute Cartesian pose

Buffers from virtual receptors

${}^{r}_{x}c_{1,\,1}\,[S_1, S_2, S_3, S_4]$ — Binary status of the wall-detecting sensors

${}^{r}_{x}c_{1,\,2}[S_5]$ — Binary status of the floor sensors

Internal memory of the control sub-system

${}^{c}C_1\,[\,{}^{0}_{M}T]$ - Absolute pose of maze

${}^{c}C_1\left[\,{}^{E}_{p}T_{\,d}\right]$ - Desired pose of a point wrt the end-effector

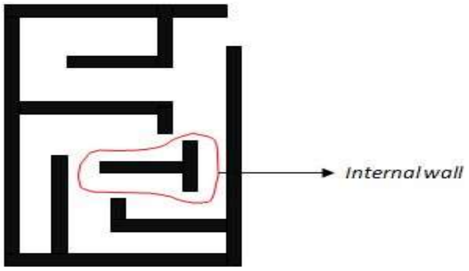
${}^{c}C_1\,[w(i,j)]$ - The weight matrix and i, j are the indices of each position on the map.

Assuming that the agent builds a map of the maze then the control subsystem needs to remember previous positions that the end effector has been to, where walls have been detected etc. If the map is a weighted matrix then the memory must store positions and a weight associated to those positions

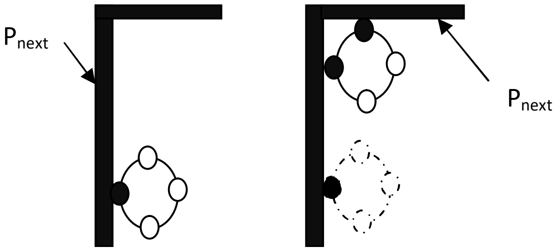
5. TRANSITION FUNCTIONS AND TERMINAL CONDITIONS

ALGORITHM

A simple wall-following algorithm is used for enabling the robot manipulator to move from initial position to the goal exit location. Upon receiving the initialisation START commands, the manipulator first moves to the given starting location. Using the maze data and current manipulator location, the point on the nearest outer wall is computed and the end-effector is moved to this location. It may be noted that it is essential to find the nearest *outer* wall to the robot, since wall-following an *internal* wall would result in the robot moving round and round about it in circles infinitely. Thus, all internal walls must be avoided. An example maze below highlights a possible case of an internal wall.



Once an (outer) wall has been detected, the robot starts to follow it by moving parallel to it by sampling points on the wall at regular intervals. The successive points on the wall along which the end-effector should move, depends on the readings of the binary wall-sensors. For example, if two consecutive wall-sensors are activated, it indicates a corner is reached, so now it simply moves either left (or right, depending on which two sensors are activated) and continues the motion along the wall. Some example cases of motion along the wall using wall-sensor information are shown below:



Depending on which sensor(s) is/are active, the desired next point on the wall is found and the end-effector moves there. Since there are no cul-de-sacs within the maze, following an (outer) wall till the end will always eventually result in reaching the exit.

BEHAVIOUR OF THE AGENT

The robot behaviour control software is thus defined in terms of the following states:

- IDLE STATE (B_1) : Robot is always in this idle state, before the initial coordinates are established (SET not given) and robot has not moved to initial position (START not given). Robot returns to this state when goal is reached, or STOP command given.

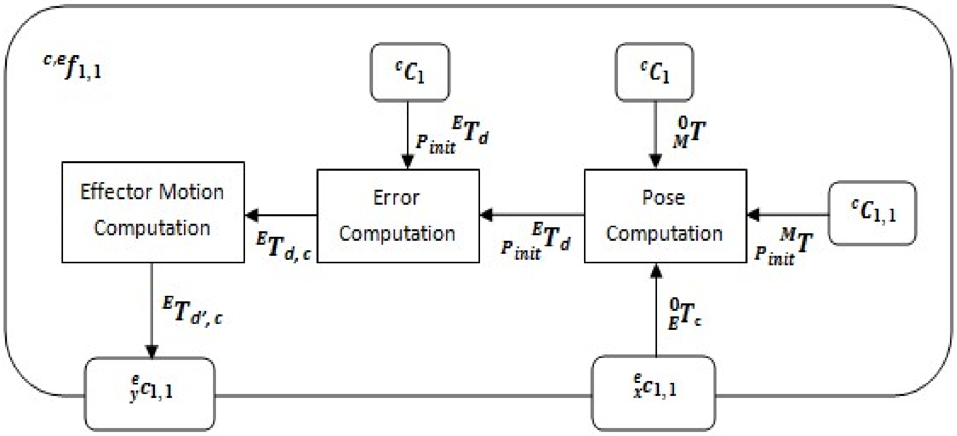
- INITIAL STATE (B_2) : Robot moves to the initial position (location and pose) on receiving the START command after the initial coordinates are established with SET. Then the state changes to B_3. This B_2 state is interrupted when a STOP is received.

- FIND CLOSEST (OUTER) WALL (B_3) : Closest (outer) wall is found using the maze information and robot's current positions, and the end-effector is moved in the direction to that wall. Upon sensing a wall, at-least one of the circumference sensors are activated and the state changes to B_4 state. This B_3 state is interrupted when a STOP is received.

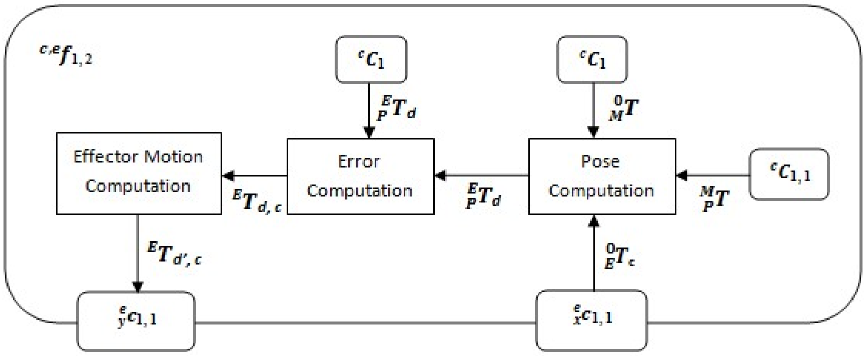
- COMPUTE DIRECTION AND MOVE PARALLEL TO WALL (B_4) : When either of the four circumference sensors is activated due to contact with a wall, this state is entered. It is assumed that the control subsystem divides the maze in nodes or cells and builds a weight map of the maze following a wall and probing each next cell. The robot now moves along the wall and follows it till the end and reaches the exit. Robot exits this state when goal is reached and the end-effector location is outside the maze, or when interrupted by a STOP command.

TRANSITION FUNCTIONS

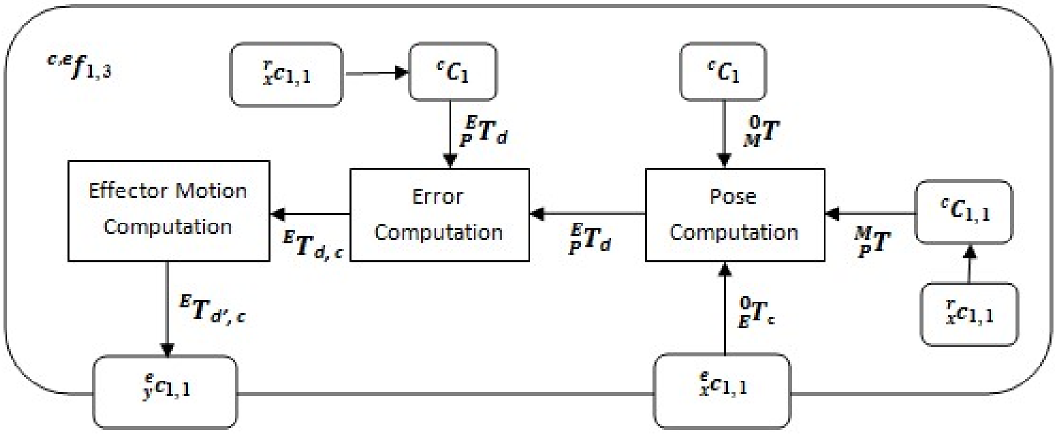
B_2 : MOVE TO INITIAL POSITION- Move to initial position given with START command



B_3 : MOVE TO NEAREST OUTER WALL - Move to point on nearest outer wall



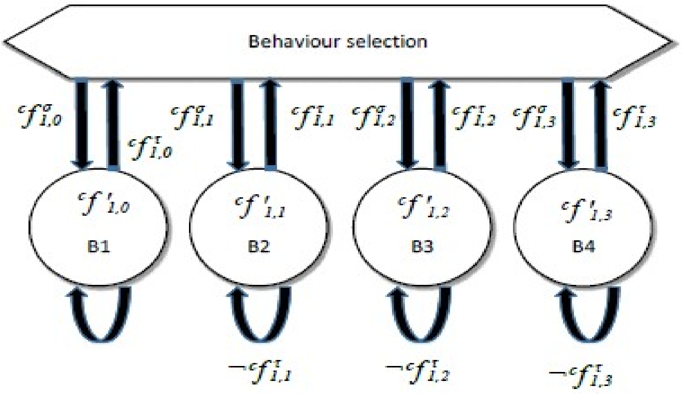
B_4 : MOVE PARALLEL TO WALL – Move to desired next point P on wall (computed using wall-sensor information)



TERMINAL CONDITIONS

- ${}^{c,e}f_{1,0}$: command = (c==START) Λ ($P_{curr} \neq P_{goal}$)
- ${}^{c,e}f_{1,1}$: command = (c==STOP) V ($P_{curr} = P_{init}$)
- ${}^{c,e}f_{1,2}$: command = (c==STOP) V ($S_1 V\ S_2\ V\ S_3 V\ S_4$)
- ${}^{c,e}f_{1,3}$: command = (c==STOP) V ($P_{curr} = P_{goal}$)

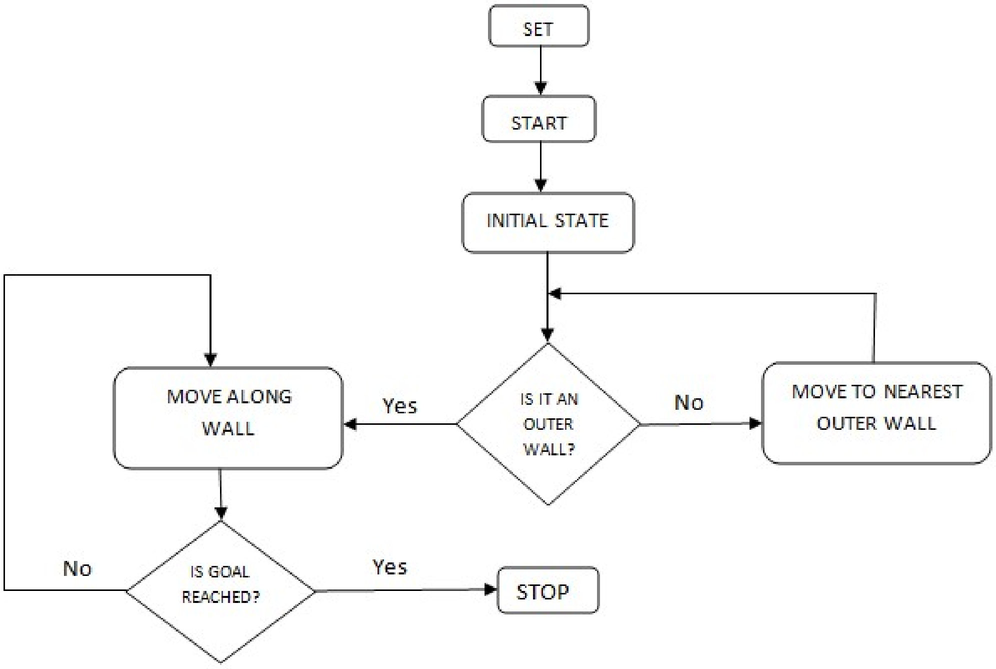
STRUCTURE OF CONTROL PROGRAM



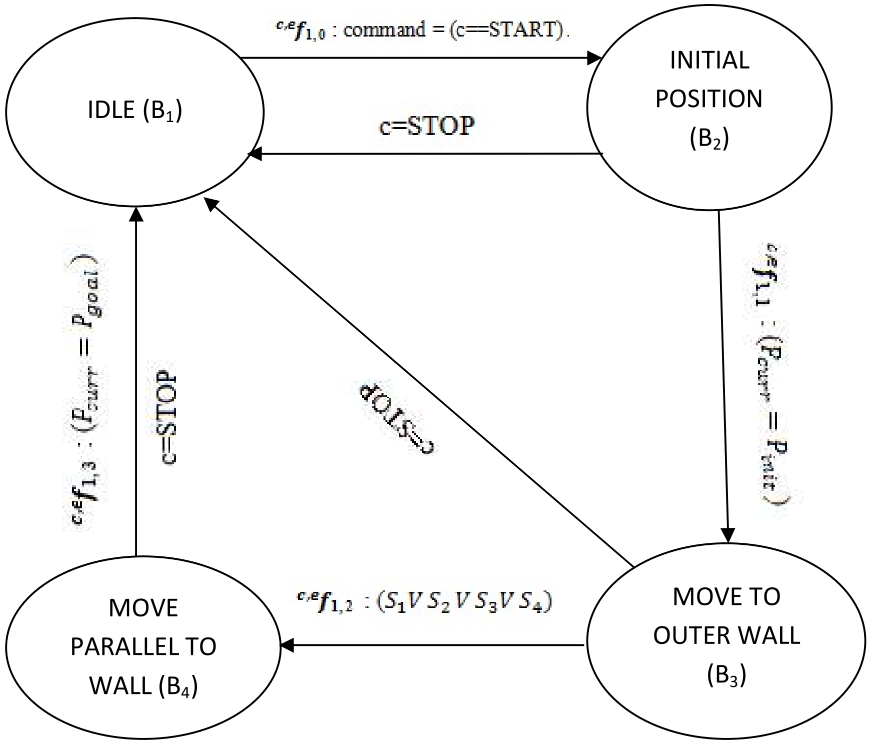
INITIAL CONDITIONS

- ${}^{c,e}\boldsymbol{f}_{1,0}$: command = (c=SET) V (c=STOP)
- ${}^{c,e}\boldsymbol{f}_{1,1}$: command = (c=START)
- ${}^{c,e}\boldsymbol{f}_{1,2}$: command = $(P_{curr} = P_{init})$
- ${}^{c,e}\boldsymbol{f}_{1,3}$: command = $(S_1 V\ S_2\ V\ S_3 V\ S_4)$

FLOWCHART



CONTROL FLOW


c,ef1,0 : command = (c==START).
IDLE (B1)
INITIAL POSITION (B2)
c=STOP
c,ef1,1 : (Pcurr = Pinit)
c=STOP
c,ef1,3 : (Pcurr = Pgoal)
c=STOP
c,ef1,2 : (S1V S2V S3V S4)
MOVE TO OUTER WALL (B3)
MOVE PARALLEL TO WALL (B4)