Thommen Karimpanal George
1000889

30.508: OPTIMIZATION AND CONTROL

FINAL REPORT: OPTIMAL PATH PLANNING

AIM:

To derive and implement an optimal path planning algorithm for a known environment. The optimization is performed with respect to the robot's power consumption, navigation and obstacle avoidance behaviours. The algorithm could be tested out using the Evobot platform, and information on the environment costs is obtained from the overhead Optitrack motion capture cameras.

1.0 MOTIVATION:

A fundamental challenge for an autonomous mobile robot is to navigate from point A to point B in an environment in the best possible manner, avoiding obstacles, and at the same time, ensuring that the trajectory it follows is optimal with respect to one or more quantities, which are decided by the designer/programmer. The Evobots platform (Figure 1 and Figure 2) is a swarm of micro-robots which will be used to explore the capabilities of evolutionary control, with the ultimate objective of achieving a robust, adaptive and autonomous swarm. The task for such robots may require them to spend days, weeks or even months in outdoor environments, where once the on-board power source becomes depleted, the only source of power will be the solar energy they are able to collect from the panels (not shown in figure 1) mounted on them. Since robot motion is the most energy expensive process in the robot, it is essential to ensure that whenever the robots decide to move through the environment, it needs to do so in an energy-optimal manner. An optimal path planner could be mathematically derived and tested out on the Evobots, with the primary aim of minimizing energy consumption. Additional requirements, such as the smoothness of the trajectory could also be included. The motion tracking cameras at Temasek laboratories at SUTD could be made use of in order to provide the robot with feedback about the environment layout. In the future, this would not be required, as the robots would have an internal model of the environment through SLAM (Simultaneous Localization and Mapping) algorithms. It would also be interesting to investigate how evolutionary control algorithms would achieve this path planning objective. The mathematical solution to optimal path planning could later on be compared with the evolutionary solution to reveal specific advantages of each approach.

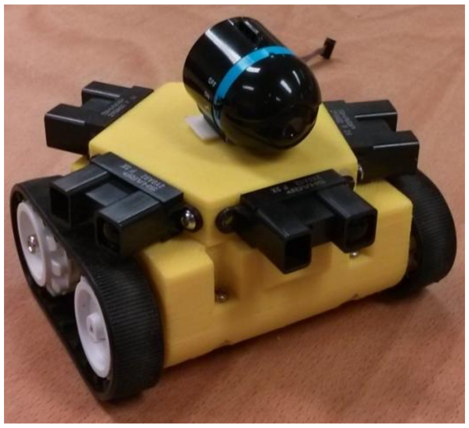


Figure 1: The Evobot platform

2.0 FORMULATION:

The optimal path planning problem described above can be formulated as a constrained optimization problem. The fundamental aim of this path planning algorithm is to minimize the energy spent by the robot in moving from point A to point B. This would thus be a component in the objective function. The algorithm would also need to take into consideration the obstacles in the environment in order to plan the optimal path. This would be a second component in the objective function, and would be embedded into the objective function through a term that provides the environment costs. A third component in the objective function would be a terminal constraint which will ensure that the robot moves through the environment and ends up at the required final position (point B). This is the terminal cost. The model of the robot would also be known by deriving the state space equations that govern the dynamics of the robot, and this will be treated as a constraint in the formulation of the optimization problem. An additional constraint is on the inputs that can be given to the robot. The motors on either side of the robot are controlled using PWM. Since, the PWM duty cycles can vary only between 0 and 1 (-1 to 0 for motion in the reverse direction), this imposes a constraint on the input signal. The formulation of the objective function shown below:

$$\min_{u\in U} J(u) = \int_0^T P(x(t))dt + \int_{x0}^{xT} E(x(t))\ dx + D\big(T, x(T)\big)$$

Subject to: $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{t}, \boldsymbol{x}, \boldsymbol{u})\ and\ \ x \in X = \{x(t) \in C^1[0,T]^n : x(0) = x_0\}$

$and\ u \in U = \{u(t) \in C[0,T]^m\}\ and -\mathbf{1} \le \boldsymbol{u_i} \le \mathbf{1}\ where\ 1 \le i \le 2\ (2\ PWM\ inputs)\ and\ u_3 = 1$
(These constraints apply to the simplified model discussed in section 3.0)

Where P(x(t)) is the power consumed by the robot as a function of the states. This will be estimated from the model of the robot. D(T,x(T)) is a term which specifies the final position (it adds terminal cost to the objective function), E(x(t)) is a function which specifies the environment costs. The environment costs will be derived from an image processing algorithm as mentioned in section 4.0. The dynamics of the system is specified by the equation $\dot{x} = f(t, x, u)$. The exact form of the equation will be derived using the Bond Graph technique. The resulting dynamic model will be treated as a constraint.

The optimization problem formulated above is a constrained one, with constraints on the inputs. Direct methods could be made use of to solve this problem. In particular, the method of state and control parameterization could be made use of to solve this problem and find the set of optimal control inputs. The justification for this choice of method will be provided in the section on implementation.

3.0 ROBOT MODEL:

The robot is modeled using the bond graph technique. In the model, the robot is treated as a two point masses connected together, with the two masses concentrated at either side of the robot as shown in figure 2.

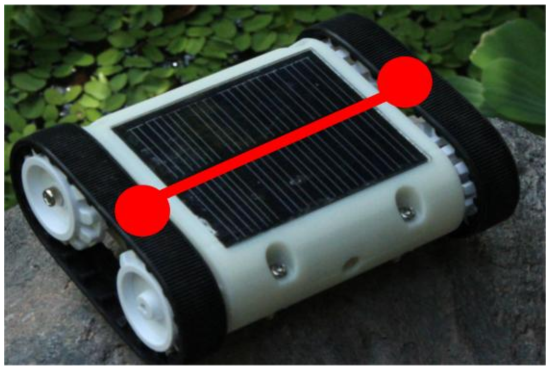


Figure 2: The evobot platform treated as a 2 point mass

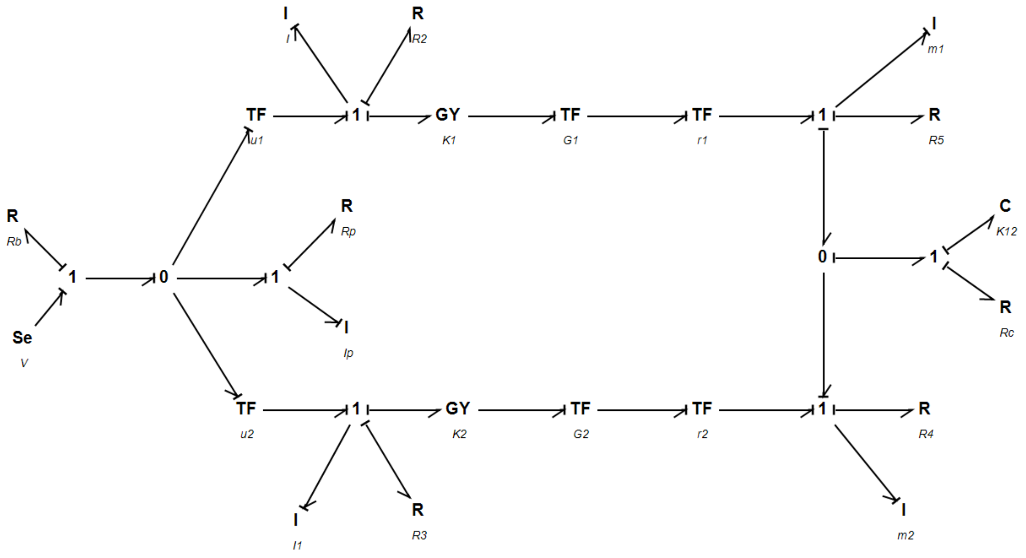


Figure 3: Bond Graph Model of the robot

The bond graph model of the system is shown in figure 3. In the model, a voltage source in series with an internal resistance drives the two motors, while at the same time, powering up the other components and sensors in the robot, which are treated as parasitic components. The voltages to the motors are modulated as per their respective PWM duty cycles. The motor is modeled as a gyrator element with some inductive behavior. The motors in the robot also contain a gearbox which provides a speed reduction of 1:100, which is also modeled. The rotational motion of the motor is converted to translational motion using a transformer element whose modulus is equal to the radius of the wheel. Each motor is assumed to be attached to half the total mass of the robot. The motion of both 'halves' of the robot are coupled together with mechanical coupling (a stiffness and damper element). The equations derived from the bond graph are shown below:

$$\begin{bmatrix} \dot{I_1} \\ \dot{I_2} \\ \dot{v_1} \\ \dot{v_2} \\ \dot{\Theta} \\ \dot{I_p} \end{bmatrix} = \begin{bmatrix} (-\frac{R_b}{L{u_1}^2} - R/L) & -R_b/Lu_1u_2 & -K/GrL & 0 & 0 & -R_b/Lu_1 \\ -R_b/Lu_1u_2 & (-\frac{R_b}{L{u_2}^2} - R/L) & 0 & -K/GrL & 0 & -R_b/Lu_1 \\ 2K/Grm & 0 & (-2R_f - 2R_c)/m & 2R_c/m & -2K_{12}/m & 0 \\ 0 & 2K/Grm & 2R_c/m & (-2R_f - 2R_c)/m & -2K_{12}/m & 0 \\ 0 & 0 & 1/l & -1/l & 0 & 0 \\ -R_b/L_pu_1 & -R_b/L_pu_2 & 0 & 0 & 0 & (-R_b/L_p - R_p/L_p) \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ v_1 \\ v_2 \\ \Theta \\ I_p \end{bmatrix} + \begin{bmatrix} 1/Lu_1 \\ 1/Lu_2 \\ 0 \\ 0 \\ 0 \\ 1/L_p \end{bmatrix} [V]$$

Where :

I_1- Current from one of the sources
I_2- Current from the other source
v_1- Velocity of a point on the rim of the wheel/velocity of one side of the robot
v_2- Velocity of a point on the rim of the other wheel/velocity of the other side of the robot
ϴ – Heading/ orientation
I_p- Parasitic Current (includes IR and optical flow sensors)
L_p- Parasitic Inductance
R_p- Parasitic Resistance
R_b – Internal resistance of the battery (0.5Ω)
R- Resistance of motor windings (2 Ω)
L – Inductance of each motor (0.0015H)
K – K value of the motors (0.005V/rad/s)
G – Gear Ratio of motors (1/100)
r – Radius of the wheels (0.02m)
l – equivalent length of the robot (0.15)
m – equivalent mass of the robot (0.15kg)
V- Battery Voltage (3.8V)
u_1- PWM ratio to first motor
u_2- PWM ratio to second motor
R_f- Translational Frictional Damping Constant (7Ns/m)
R_c – Damping constant of the coupling (1Ns/m)
K_{12} – Stiffness of the coupling (1Ns/m)

The model can be simplified by treating the two halves of the robot as independent dynamic systems and by separately accounting for the parasitic losses. With this approach, it would be required to use three inputs, two PWM inputs u_1 and u_2 (corresponding voltages V_1= u_1V and V_2= u_2V) for each of the two motors, and one constant input 1(corresponding voltage 1*V=V) that acts directly on the parasitic elements. The corresponding bond graph models for each side and for the parasitic elements are shown in figures 4 and 5 respectively.

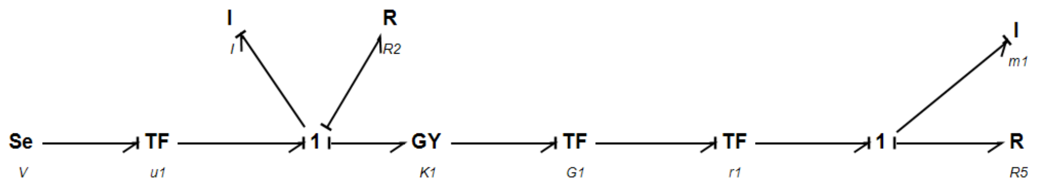


Figure 4: Model of one side of the robot

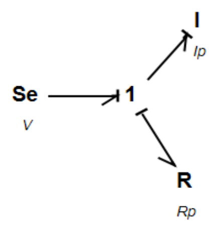


Figure 5: Model of the parasitic effects in the robot

The system equations can be derived from these models and can be coupled together to obtain an alternative, simplified model as shown below:

$$\begin{bmatrix} \dot{I_1} \\ \dot{I_2} \\ \dot{v_1} \\ \dot{v_2} \\ \dot{\theta} \\ \dot{I_p} \end{bmatrix} = \begin{bmatrix} -R/L & 0 & -K/GrL & 0 & 0 & 0 \\ 0 & -R/L & 0 & -K/GrL & 0 & 0 \\ 2K/Grm & 0 & -2R_f/m & 0 & 0 & 0 \\ 0 & 2K/Grm & 0 & -2R_f/m & 0 & 0 \\ 0 & 0 & 1/l & -1/l & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -R_p/L_p \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ v_1 \\ v_2 \\ \theta \\ I_p \end{bmatrix} + \begin{bmatrix} V/L & 0 & 0 \\ 0 & V/L & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & V/L_p \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ 1 \end{bmatrix}$$

Either of the above mentioned models could be used for solving the optimization problem at hand. In both the models, two additional states were added, one each to determine the x and y positions of the robot. This can easily be computed as the velocities v_1 and v_2 are known.

3.1MODEL VALIDATION

Both the robot models were validated by performing tests on the physical robot and measuring the electrical and mechanical characteristics. The test results are summarized below in Table 1

Turning tests:			
% of Max PWM voltage	**Radius while turning in cm (on floor mat)**	**Radius while turning in cm (on table surface)**	**Corresponding radius on simulation in cm**
25	12.5	11.5	12.5
50	20	19	22.9
75	58	55.5	53.5

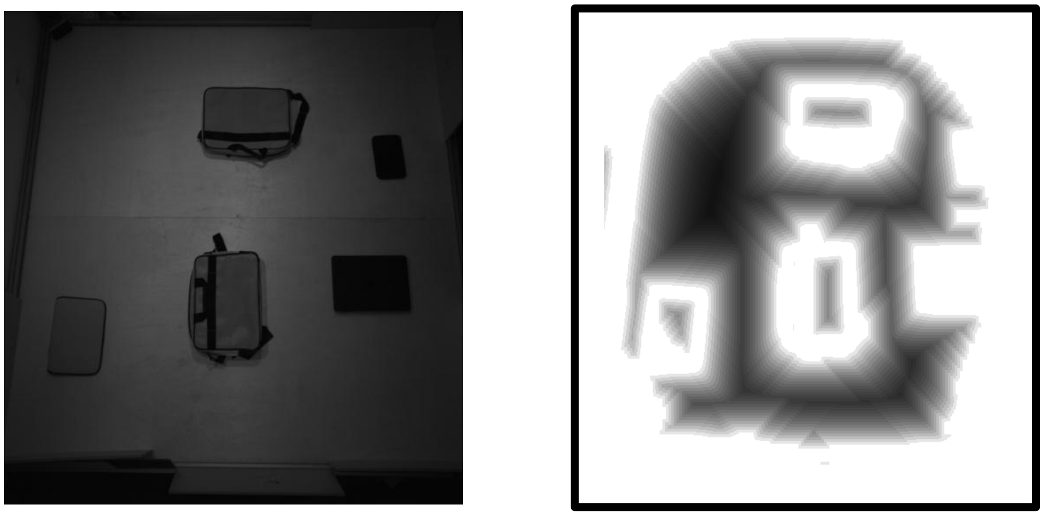
Straight Line Tests:		
% of max PWM voltage	**Actual Velocity (cm/s)**	**Velocity in simulation (cm/s)**
100	19.09	18.85
50	9.47	9.42
25	4.85	4.77
Electrical Tests:		
Model 1		
Condition:	**Average Measured Current (mA)**	**Current in Simulation (mA)**
Both motors on full speed	283	278
One Motor on full speed	240	233
Condition:	**Average Measured Current (mA)**	**Current in Simulation (mA)**
Model 2		

Both motors on full speed	283	288
One Motor on full speed	240	242

Table 1: Model Validation Results

4.0 Environment Cost Gradient:

The environment in which the robot moves consists of many obstacles which are to be avoided during the motion of the robot. The information of this environment cost is not very easy to obtain unless the robot possesses an internal representation of the environment through algorithms such as SLAM (Simultaneous Localization and Mapping). In this project, the robot is provided with the environment cost information using an image processing algorithm, which generates a cost gradient around the obstacles in the environment. The algorithm uses the image of the environment from the overhead motion capture cameras as an input. The image processing algorithm works as follows:
The input image is thresholded, so that the dark obstacles are isolated from the light background. Then the isolated obstacles are dilated multiple times, and each layer of dilation is assigned a value representing the pixel intensities. The value assigned is higher for the first dilation, and lower for the subsequent dilations. The assigned pixel intensity values can thus be used as a cost value as it decreases with decreasing distance from the obstacle. In other words, the algorithm produces a cost gradient around obstacles in the environment. A sample image and the cost gradient are shown below:



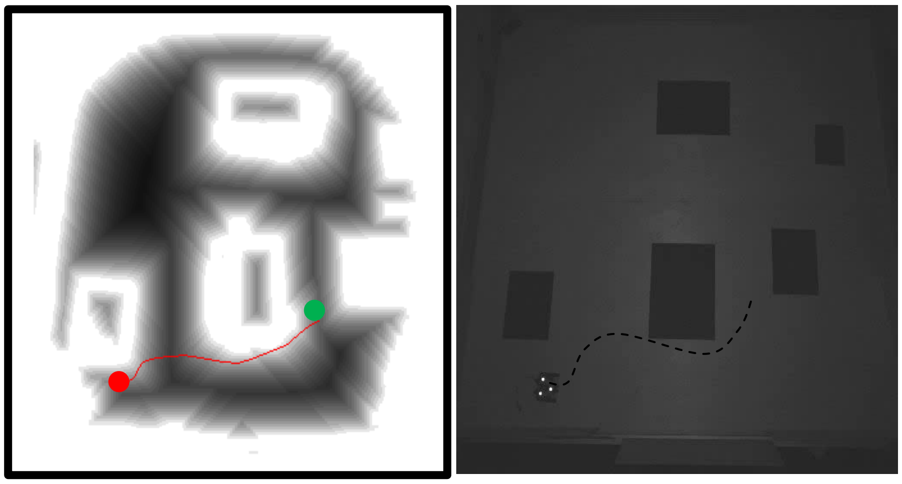
6(a) 6(b)

Figure 6: The figure shows a sample image (Figure 6(a)) and the resulting cost gradient (Figure 6(b)) generated by the algorithm (Dark objects are treated as obstacles)

5.0 Implementation:

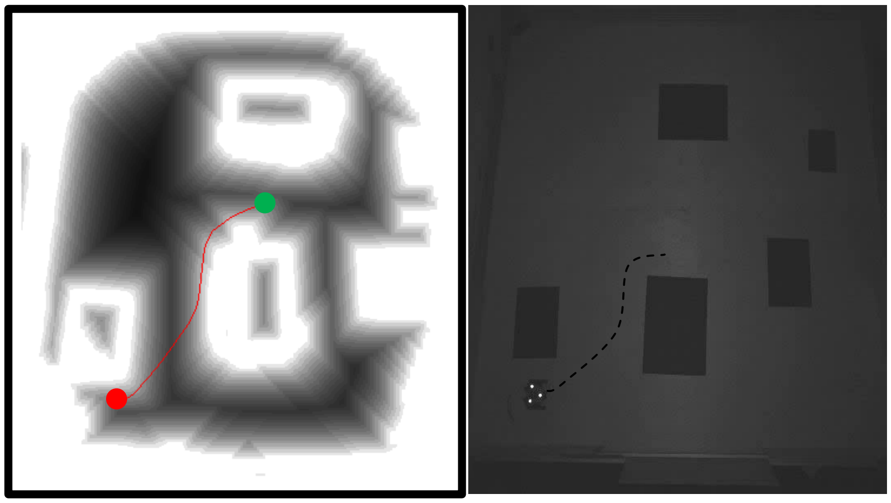
The problem was solved in simulation using the state and control parameterization method. This method was chosen as it was convenient to impose constraints on the control signals and also on the states. The states need to be constrained because the environment cost E(x(t)) is a function of the position of the robot. The states also include the (x,y) position of the robot, and these coordinates are limited by the size of the environment that is considered here. The control inputs generated from the problem solved in simulation were applied to the robot in an open loop implementation. The results can be seen below. Please refer to the supplementary material (attached with this report) for videos of the robot motion.

Test 1:



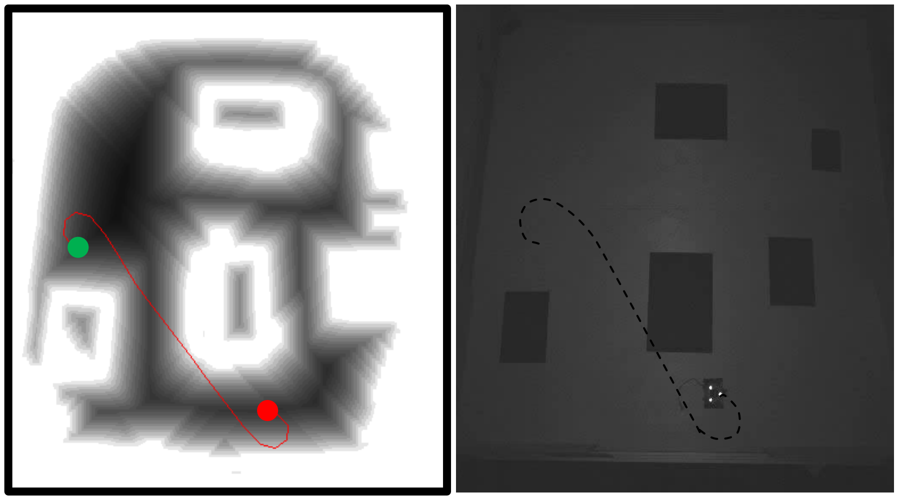
7(a)

Test 2:



7(b)

Test 3:



7(c)

Figure 7. Figures 7(a), 7(b) and 7(c) show the test results in simulation (on the left) and in reality (on the right)

As seen in the figure, open loop application of the optimal inputs generated by the matlab program to the robot results in the robot following a very similar trajectory in reality. There are a few differences in the simulated and actual paths, and their similarity is limited by the accuracy of the robot's dynamic model.

5.1 Implementation Issues

As seen from the test results in section 5.0, the implementation of optimal path planning worked quite well, even with just open loop control. However, there were a few issues to be noted with regards to the implementation. The first was that the cost function shown in section 2.0 had to be modified slightly to include weights on each term in the cost function. These weights represent the relative importance of each term in the objective function. The cost function was rewritten as:

$$J(x) = w_1 * \int_0^T p(x,u)dt + w_{2*} \int_{x0}^{xT} E(x)\,dx + w_3 * D(T, x(T))$$

Where w_1, w_2 *and* w_3 are weights associated with each term in the objective function. One has to manually tune these weight parameters to obtain a reasonable behaviour from the robot. For example, if the weight associated with the power term is too large, the program tries to optimize more to reduce the power term and neglects the other terms in the objective function. On the other hand, if weight associated with the environment cost is too large, the robot primarily tries to fulfill its obstacle avoidance requirement and neglects other terms such as the power term.

The second issue with the implementation is that one needs to have a reasonable guess with regards to the terminal time. If the terminal time is chosen as too large or too small, the robot's motion may not be ideal. For example, if the terminal time is chosen as too large, the robot moves through the environment aimlessly and then eventually reaches the goal at the terminal time. If the terminal time is too small, the robot tries to reach the final destination by following a straight line path from its initial position, neglecting other costs such as the environment costs. It is clear from this, that one needs to optimize the terminal time as well, in order to obtain a good path planning algorithm.

Another issue relates to the number of time steps chosen in simulation. In the videos attached, it is seen that the robot motion is quite jerky. This is because the number of time steps chosen was a small number (20 steps). Choosing a higher number of time steps would generate a smoother motion of the robot in the experiments, but would greatly increase the computational cost.

Apart from these issues, for future implementation, the possibility of using dynamic programming could be explored, as it provides a global minimum, as opposed to local minima in this implementation. This implementation also assumes that the environment is known, which is usually not the case in reality. In the future, Simultaneous Localization and Mapping (SLAM) algorithms could be implemented in order to create an internal model of the environment. Once the environment is known, a similar approach (with above mentioned improvements) can be used for optimal path planning.

REFERENCES:

[1] "Optimal and Efficient Path Planning for Partially-Known Environments", Antony Stentz in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '94)*, May, 1994, pp. 3310 - 3317.

[2] "RRT∗-Smart: Rapid convergence implementation of RRT∗ towards optimal solution" Nasir J., Malik U., Ayaz Y. and Hasan O. in Mechatronics and Automation (ICMA), 2012 International Conference on

[3] http://www.bondgraph.org/

[4] "Digital Image Processing", R.C. Gonzalez and R.E. Woods, Prentice Hall, Third Edition, 2008

[5] "System Dynamics: Modeling, Simulation, and Control of Mechatronic Systems", Dean C. Karnopp, Donald L. Margolis and Ronald C. Rosenberg, John Wiley & Sons Inc., 4th Edition, 2006