Efficient Dynamic Tracking Control System for Mobile Network Agents

D.Neelakant
Department of Computer Science and Engineering
Kamala institute of tech & sci, Singapuram Huzurabad ,
Dist:Karimnager,Andhra Pradesh, India-505468

Anjaneyulu. S
Department of Computer Science and Engineering
Balaji Institute of Tech& Sci,Narsampet,Warangal
Andhra Pradesh, India-506331

M.Srinivas
Associate Professor & HoD
Department of Computer Science and Engineering
Balaji Institute of Tech& Sci,Narsampet,Warangal
Andhra Pradesh, India-506331

Abstract- This paper presents a complete tracking of high-performance decentralized control design that permit mobile agents with dynamic distributed networked sensing capabilities to track the desired routs(trajectory) , identify what information must be distributed to each agent for tracking, and develop methods to minimize the communication needed for the trajectory information distribution.(This paper I have written/modified for submition of my M.tech thesis)

Keywords - dynamical networks, tracking

I .INTRODUCTION

The several modern applications, teams of autonomous agents with distributed sensing and/or communication capabilities are required to cooperatively complete a complex task. Typically, such controllers use a two loop structure(nonlinear , linear controller), where the outer loop enervates reference signals based on tracking errors and the inner loop uses the reference signals to improve the dynamics . For instance, teams of autonomous vehicles, which sense relative positions, may need to follow a "lawn-mower" pattern to search a minefield. Similarly, a bank of antennas may need to follow a path in a coordinated fashion. At first these tracking problems for networks of communicating/sensing agents seem no more challenging than automated tracking problems for single devices. Seemingly, we could distribute to each agent in the network its desired path, which the agent could then independently. However, many of the tracking problems that our group has encountered—in applications ranging from air traffic management to sensor fusion and vehicle control .The Linear Parameter Varying (LPV) controllers can overcome these problems by systematically incorporating information about variation of vehicle dynamics with scheduling variables.

Fundamentally, what makes these problems challenging is that, due to cost or security or complexity constraints, individual agents do not have sophisticated enough observation capabilities to independently know where they are, and so move as they wish in their environment. Instead, each agent depends in an essential way on sensing of or communication with other agents to be able to follow desired paths .

Second viewpoint, we have found that each agent fundamentally needs to sense/receive information about other agents simply to operate, i.e., coordination is needed for task completion. For tracking tasks in particular, this fundamental need to use sensed information also implies that information about other agents' desired trajectories must be communicated for informed use of the sensed information. In this paper, we explains the controller

architecture ,different methodology for tracking control in networks whose agents depend on distributed ensing/communication capabilities to complete desired tasks.

Due to the volume and performance requirement, motion control in electronic manufacturing requires high speed and precision. In this paper, we present a systematic approach to improve the performance of point-to-point motion of a positioning system by designing a collision free path from starting to destination point.

The motivation of this paper is to create a dynamic tracking Control frame work that achieve high performance and remains relatively easy to design and implement. The following approach to study to tracking control in communicating-agent networks.
Our overall approach of control design is summarized below:
Chapter II: Verify system linearity for small amplitude inputs using LTI model (Linear time invariant).
Chapter III :Implementation of Dynamic Control and Refinement model.
Chapter IV: Introduced a new swarm optimization (PSO) algorithm for identifying collision free paths.
Chapter V: Interpolation of the obtained collision-free path, which is solved using a radial basis function neural network (RBFNN), and trajectory generation, based on the interpolated path.
Chapter VI**:**Advanced Algorithms for Motion Planning Problem

II. LINEAR TIME INVARIANT (LTI) MODEL

Linear time invariant (LTI) model identification based on input/output responses may be performed in either time or frequency domains. The time domain approach is adversely affected by high frequency noise. Therefore we decided to use the frequency domain subspace identification method [1], [2]. The experimental transfer function is first obtained by a sine sweep (with input amplitude chosen small enough to avoid saturation but large enough to increase signal-to-noise ratio).

In system identification, it is important to consider possible input/output delays. If the experimental system contains a pure delay, direct application of LTI identification will often approximate the delay with non-minimum phase zeros and additional poles. The approximation can be avoided by including a pure time delay in the identification procedure by time shifting the output signal relative to the input signal before performing the identification. Including the delay in this manner can result in identified models of lower order, better agreement between simulated and experimental responses (no artificial undershoot), and significantly less non minimum phase behavior in the identified model. For the inverse dynamics controller, the non-minimum phase zeros can compromise the tracking performance, while the pure delay simply leads to a time shift in the response. The gain and phase comparison between the experiment frequency and time step responses are shown in Fig. 1.

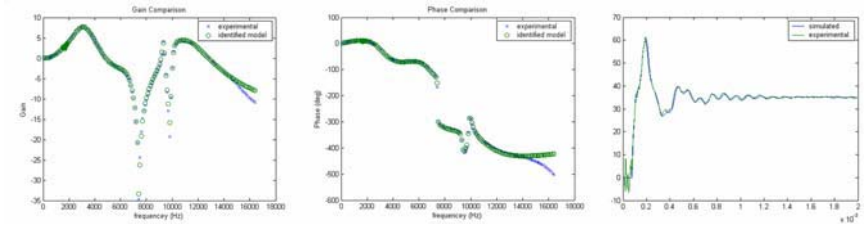

Fig. 1. Frequency Gain/Phase and Time Step Response Comparison between Experimental Data and Identified Model

III. DYNAMICS CONTROL AND REFINEMENT

In this chapter to use the identified model, G, to construct dynamics filter G^{+} ,by replacing the unstable zeros by the stable mirror images and inverting the transfer function. We then apply the inverse dynamics filter to the desired output y_{des} to generate the command input: u = $G^{+}\, y_{des}$.

A trajectory generator is used to generate $y_{des}(t)$ based on the position, velocity, and acceleration constraints.

To avoid unbounded problems, we use a half-sine profile for acceleration and deceleration. The experimental result of the 500μm move is shown in Figure 2.

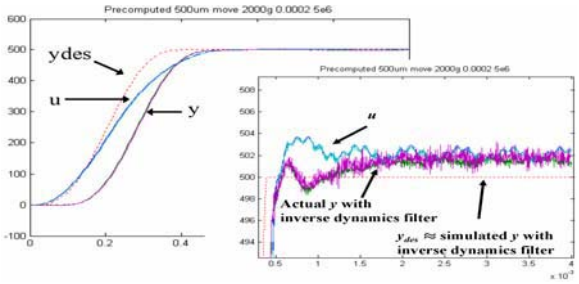

Fig. 2 dynamics filter

Large tracking errors near the entry into and inside the settling zone are observed. Clearly, using inverse dynamics filter alone does not meet the desired performance specification.

To correct for this mismatch between experimental response and model we modify the input to u = $G^{-1} y_{des} + \Delta u$. where Δu is obtained using the iterative refinement algorithm based on the output tracking error . Using the complete trajectory tracking error to iteratively update the command input is known as iterative learning control (ILC). This concept was originally introduced in robot tracking control. For this study, we apply the gradient descent approach with the nominal LTI model, G, as the approximate gradient. The basic algorithm is summarized below:

Algorithm : Given $\underline{y^*} = \{y^*(t_i) : i \in 0,1,\ldots..N\}$ and $u_o = \{u_o(t_i) : i \in 0,1,\ldots..N\}$.

1) Apply $\underline{u}$ to the physical system and obtain the output sequence $\underline{y} = \{y(t_i) : i \in 0,1,\ldots..N\}$.
2) Update $\underline{u}$ by adding following term

$$\Delta \underline{u} = -\alpha G^* (\underline{y} - \underline{y}^*) \qquad (1)$$

Where G^* is the adjoint of G, and α may be set as a sufficiently small constant or found by using a line search (which would require additional runs).

3) Iterate until $\|y - y^*\|$ or $\|\Delta u\|$ becomes sufficiently small.

The key step in the above algorithm is the update equation (1). Let the state space parameters of G be (A,B,C,D). The adjoint G^* is given by $(-A^T, -C^T, B^T, B^T)$ but it must propagate backward in time from the zero state. To implement $G^*\Delta y$ we first reverse Δy backwards in time, filter it forward in time through the filter. The result of iterative refinement for the move length 500μm is shown below fig 3

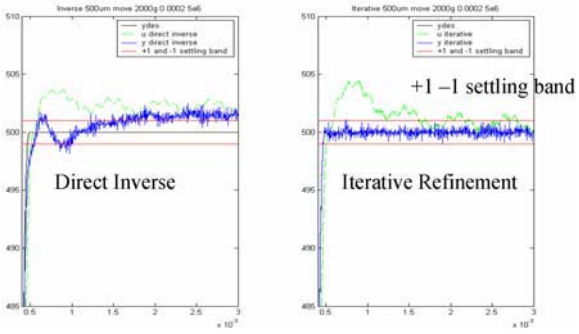

Fig 3 Experimental results of applying iterative refinement to 500μm.

IV. PARTICLE SWARM OPTIMIZATION(PSO)

PSO algorithm is multi-agent evolutionary search technique. The space of solution is searched with multiple particles, whereby every particle is directed on the basis of its own experience and the experience of the whole swarm. Basic variables are position of particle, which represents the potential solution, velocity of the particle,

which represents the change of position in current iteration and fitness function, which is the measure of success of the particle.

Let $x_i(k)$ and $v_i(k)$ denote the position and velocity of *i*-th particle in *k*-th iteration, respectively. Algorithm can be described by following steps:

1. **Problem definition:**
   Allowable position and velocity ranges $[V_{min}, V_{max}]$ and $[X_{min}, X_{max}]$, respectively, swarm size *N*, measure of success for every particle fitness function $F_i(k)$; value of this function measures the success of the *i*-th particle;
2. **Algorithm initialization:**
   Positions and velocities of particles are initialized with uniform random numbers from $[V_{min}, V_{max}]$ and $[X_{min}, X_{max}]$, respectively, i.e.,

$$x_i(0) = X_{min} + \chi_1(X_{max} - X_{min}) \text{ and } v_i(0) = V_{min} + \chi_2(V_{max} - V_{min}) \quad (2)$$

   where χ_1 and χ_2 are uniform random numbers from [0,1];

3. **Fitness function evaluation:**
   For every particle in swarm, the following variables are evaluated: fitness function, self-best position $P_i(k)$ and global-best position $P_g(k)$;

4. **Velocity correction:**

$$V_i(k+1) = wV_i(k) + c_1\gamma_1[P_i(k) - (3)\, x_i(k)] + c_2\gamma_2[P_g(k) - x_i(k)] \quad (3)$$

   Where c_1 and c_2 denote self-confidence and swarm-confidence parameters, respectively, while w stands for inertia factor and γ_1, γ_2 are random numbers from [0,1]. Inertia factor determines the effect of current motion on a future motion. Large values of this parameter leads to global search, while small values leads to fine, local search, which is suitable when algorithm converges. Thus, variable value of inertia is used, such that inertia starts from large value, and decreases as algorithm iterates. Also self-confidence and swarm-confidence factors should be variable. Self-experience should have dominant effect on particle motion at the beginning of the algorithm, while later, swarm experience should prevail. Particles velocities must stay inside allowable interval $[V_{min}, V_{max}]$;

5. **Position correction:**

$$x_i(k+1) = x_i(k) + v_i(k+1) \quad (4)$$

   Position must stay inside allowable interval $[V_{min}, V_{max}]$;

6. **Termination of algorithm:**
   Algorithm terminates when maximum number of iterations is reached, or good enough value of fitness function. erformance of algorithm heavily depends on particles diversity. It is preferred that swarm consists of diverse particles at the beginning of algorithm. Later, as algorithm iterate, diversity should decrease, in order to finely converge to optimum. It is necessary to allow passing through detected optimum, in order to avoid stucking in it, because it can be local minima.

V. RADIAL BASIS FUNCTION NEURAL NETWORK (RBFNN)

Radial basis function neural networks (RBFNN) are three-layer neural networks. Their structure is shown on Fig. 4. These networks are widely used for nonlinear function approximation, as well as multilayer perceptrons (MLPs). Although they cannot achieve the accuracy of the MLP networks, their advantage over MLP is in much faster

training. For achieving the same accuracy as MLP, RBFNN is usually more complex, i.e., it has more nodes in the hidden layer.

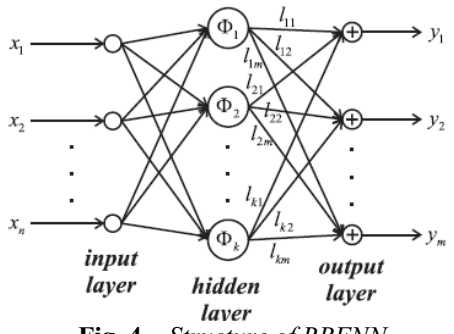

Fig. 4 – *Structure of RBFNN*

Let $x = [x_1 \dots x_n]^T$ and $y = [y_1 \dots y_m]^T$ denote input and output vectors, respectively. Activation function of hidden neurons is:

$$\Phi_i(x) = e^{(\|x - w_i\|^2)/2\sigma^2}, \quad i = 1,\dots, n \tag{5}$$

Where $\|.\|$ denotes Euclidean norm. Activation function $\Phi_i(x)$ is centred in vector i w, while σ denotes spread of the function. Output layer is linear, so output of the network is linear combination of the hidden layer outputs:

$$y_j = \sum_{i=1}^{k} l_{ij} \Phi_i(x), \quad j = 1,\dots.m \tag{6}$$

It can be seen from (5) that activation of hidden layer neuron i is the strongest when $x = w_i$, because $\Phi_i(x) = 1$. Activation decreases when input departs from vector w_i. Basic idea is to divide input space onto k overlapping regions, while every hidden neuron will be active only in one region, i.e. some neighbourhood of w_i. If region width σ is too small, network generalizes poorly, while for large values of this parameter, interpolation can be coarse.

Network can be trained such that approximation error on the training set is zero. This can be impractical, because size of the hidden layer is equal to the size of the training set. Thus, training algorithm should gradually increase the size of hidden layer until desired value of approximation error or maximum number of hidden neurons is reached.

VI .ADVANCED ALGORITHMS FOR MOTION PLANNING PROBLEM

Path generation using PSO algorithm

It is assumed that the positions of obstacles are known and static. The goal is generation of collision free path from starting to destination point, so that the path is as short as possible. This task will be solved using PSO algorithm. Generated path is given as an array of two-dimensional points, so the obtained path is not smooth. Robot has fixed maximum step size, i.e. maximum distance between current and next point V_r^{max}. Increase of this parameter speeds up the algorithm, but decreases the path smoothness and decreases the possibility of algorithm to be get stuck in complex scenarios with large number of close obstacles. It is also assumed that all obstacles are circular and there is no overlapping between obstacles, although they can touch each other, but not more than two. Sizes of all obstacles are increased for the dimension of mobile robot. The experimental results is shown in fig 5

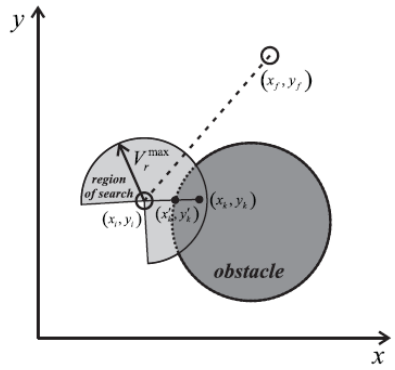

Fig 5 path generation using PSO

Region of search is illustrated on above fig Let (x_i , y_i) denotes optimal point generated by PSO in previous iteration, which represents the center of search region in current iteration, while (x_f , y_f) denotes destination point.. Region of search is circular sector with central angle of 270°, symmetric relative to line joining points (x_i , y_i) and (x_f , y_f). On this way, algorithm always progresses in sense that every new point is closer to destination than previous one. This solution gives better results than circle. On the other hand this solution is better than half–circle, because it could happen that optimal solution lies in the corner of chosen search region, which is not covered by the half–circle.

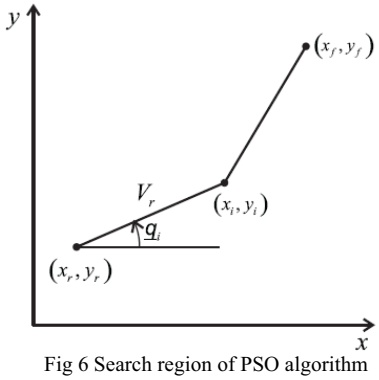

Fig 6 Search region of PSO algorithm

More complex situation arises when the region of search collides with obstacles. *In this case it is necessary to eliminate all points that are located at the intersection of search region and obstacles*. These situations can be avoided on two ways. The first way is to mark these points as inadequate by giving them large positive value of fitness function. This solution is simple, but in this way population loses some particles, i.e., artificially reduces the size of the population. The other way is to move points that lie inside the obstacles to the obstacle edge (see Fig. 6). On this way algorithm will move particles into the allowable part of search region, and there is no loss of population particles.

Particle (x_k , y_k) and centre of the search region (x_i , y_i), and for this new particle fitness function is evaluated. Particles in the PSO algorithm represent two-dimensional points in polar coordinates (radius with respect to the centre of the search area V_r and angle θ between *x*-axis and line joining particle and the centre of the search area). Next position should be obtained such that total length of the path is minimal and collision with obstacles is minimal. Path length *F*, from (x_r , y_r) to the destination (x_f , y_f) over (x_i , y_i) is

$$F = V_r + \sqrt{(x_r + V_r\cos\theta_i - x_f)^2} + \sqrt{(y_r + V_r\sin\theta_i - y_f)^2} \quad x_i = x_r + V_r\cos\theta_i \; y_i = y_r + V_r\sin\theta_i \; . \qquad (7)$$

Fitness function is weighted sum of path length *F*, given in (7) and additional term *P*(*i*) , which represents penalties if path leads over the obstacles.

This term has to be variable, taking into account size, position and orientation of the obstacles. Finally, fitness function fit_i is given by:

$$fit_i = w_1 F_i + w_2 \mathrm{P(i)},\ \mathrm{P(i)} = w_2 P_1(i) + w_3 P_2(i),$$

$$F_i = V_r + \sqrt{(x_r + V_r \cos\theta_i - x_f)^2} + \sqrt{(y_r + V_r \sin\theta_i - y_f)^2} \quad (8)$$

Penalization factor $P(i)$ consists of two factors $P_1(i)$, which represents penalization due to the existence of intersection between obstacle and path generated from current particle to the destination point, and $P_2(i)$, which represents penalization due to the existence of intersection between obstacle and path generated from particle given in the previous algorithm iteration to the current particle.

w_1, w_2, w_3 weight path length and path intersection with obstacles, respectively. Larger values of the w_1 will give a shorter path, which leads mobile robot very close to the obstacles, while larger values of w_2 and w_3 means less chance for collision between mobile robot and obstacles at the expense of longer path. It is recommended to choose larger values w_2 and w_3 in cases when algorithm works with larger values of the radius V_r^{max} , in order to ensure that generated path does not lead over the obstacles.

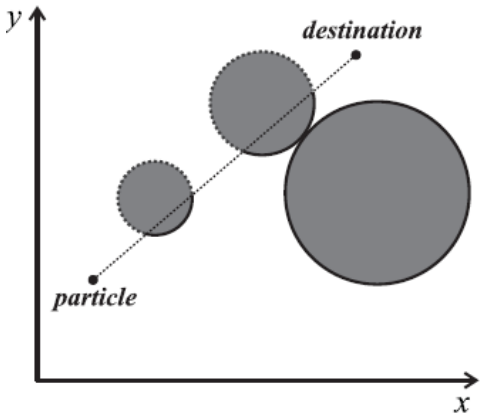

Fig 7 penalization factor evaluation for final result

Let us assume that there is obstacle between current and destination point, as shown on Fig. 3. Robot can circumvent obstacle from either side, but it is rational to select the shorter path. This implies that penalization factor should be shorter of these two paths. So, penalization factor can be defined as:

$$P_k(\mathrm{i}) = \min\left\{arc_j + \sum_{s=1}^{s} obim(s)\right\},\ \mathrm{j} \in \{1,2\}, \mathrm{k} \in \{1,2\} \quad (9)$$

Where arc_j denotes arc of the obstacle intersected by path, and $c(s)$ denotes circumference of adjacent obstacle.

VII .SIMULATION RESULTS

Proposed algorithm for motion planning of mobile robot is implemented in MATLAB and Java packages. The scenario with seven obstacles is adopted. In order to include robot dimensions, obstacles are enlarged with the dimension of mobile robot. It is assumed that robot width is $2b$ 40cm and wheel radius is r 8 cm. Maximum angular velocities of the wheels are $w_L^{max} = w_D^{max}$=16 rad/s It is assumed that therobot position and orientation measurements are corrupted with white Gaussian noise, which standard deviations are 1 cm and 1°, respectively Starting point is (2,4.5)m, while the destination point is (4.5,0.5)m. PSO algorithm searches space with the 30 particles in the swarm. Particles velocities are bounded on interval [–1,+1]. Search area radius is V_r^{max} = 0.25m. Choice of this parameter is critical. Small values lead to fine search, which produces smooth path with large number of points, but there is possibility of stacking between obstacles in complex scenarios, because algorithm has to choose between particles with similar quality. Larger values of V_r^{max} give the coarse path, but the possibility to be

get stuck is very small. The algorithm terminates after 100 iterations. Values of weights in fitness function are w_1=1, w_2=w_3=5.

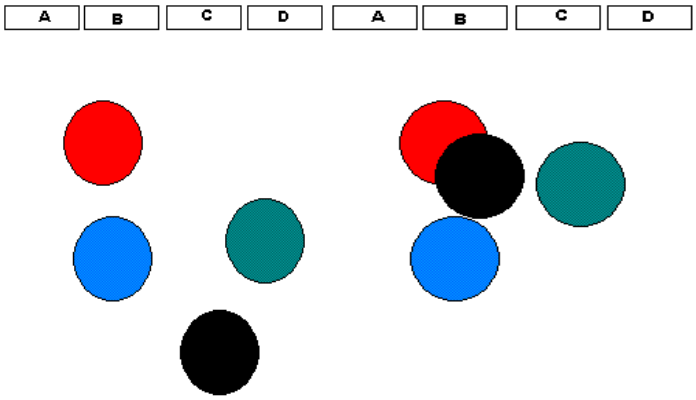

fig 8: Fig shows an experimental results using Java package

VIII.CONCLUSION

Solution of motion planning problem can be divided into generation of collision free path using PSO algorithm, interpolation of obtained path using RBFNN, trajectory generation based on interpolated path in this paper. In phases where obtained path is interpolated using RBFNN, it could happen that the generated path is not collision free, so this controller has a main goal to push mobile robot away from the obstacles. Although it is assumed that obstacles are circular, proposed method, with slight modifications, can be applied on obstacles of arbitrary shape. This approach can be applied in dynamic environments in which exist moving obstacles, due to action of obstacle avoidance. It can be also applied even in multi robot environments, with some modification of tracking control law. We expect to address design of advanced high-performance tracking controllers in future work; following references are used for recent work on designing high-performance decentralized controllers

REFERENCES

[1] S. Dutta: Obstacle Avoidance of Mobile Robot using PSO-based Neuro Fuzzy Technique, International Journal of Computer Science and Engineering, Vol. 2, No. 2, March 2010, pp. 301 – 304.
[2] J.A. Fax and R.M. Murray, "Information Flow and Cooperative Control of Vehicle Formations," IEEE Trans. Automatic Control, vol. 49, no. 9, pp. 1465-1476, Sept. 2004.
[3] M. Bowling, M. Veloso: Motion Control in Dynamic Multi – Robot Environments, IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA , USA, 8 – 9 Nov. 1999, pp. 168 – 173.
[4] R.O. Saber and R.M. Murray, "Consensus Problems in Networks of Agents with Switching Topology and Time-Delays," IEEE Trans. Automatic Control, vol. 49, no. 9, pp. 1520-1533, Sept. 2004.
[5] R. Olfati-Saber, J.A. Fax, and R.M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," Proc. IEEE, vol. 95, no. 1, pp. 215-233, Jan. 2007.
[6] M. Šušić, A. Ćosić, A. Ribić, D. Katić: An Approach for Intelligent Mobile Robot Motion Planning and Trajectory Tracking in Structured Static Environments, International Symposium on Intelligent Systems and Informatics, Subotica, Serbia, 8 – 10 Sept. 2011, pp. 17 – 22.
[7] Y. Wan, S. Roy, and A. Saberi, "A New Focus in the Science of Networks: Toward Methods for Design," Proc. Royal Soc. A, vol. 464, pp. 513-535, Mar. 2008.
[8] Y. Wan, S. Roy, A. Saberi, and A. Stoorvogel, "A Multiple Derivative and Multiple Delay Paradigm for Decentralized Controller Design," Proc. 48th IEEE Conf. Decision and Control (CDC 09), 2009.
[9] 9. Liang Chen, Sandip Roy, and Ali Saberi On the Information Flow Required for Tracking Control in Networks of Mobile Sensing Agents IEEE Trans, ON MOBILE COMPUTING, VOL. 10, NO. 5, APRIL 2011.
[10] J. Canny: The Complexity of Robot Motion Planning, MIT Press, Cambridge, MA, USA, 1988.