An Algorithmic and Experimental Study of Coordinated Networked Vehicles

Student: Timur Karatas

Advisor: Prof. Francesco Bullo

Department of Electrical and Computer Engineering

University of Illinois at Urbana-Champaign

Email: tkaratas@uiuc.edu

Completed on

This version March 3, 2004

Contents

1 Introduction

Recent decades have witnessed a significant improvement in networking, navigation, embedded computing, and miniaturization of electromechanical systems. Introduction of the standard IEEE 802.11b has made implementation of ad-hoc wireless networks possible. Navigation is improved by the removal of degradation of the Global Positioning System (GPS) signals in 2000. Furthermore, introduction of Wide-Area Augmentation System (WAAS) higher accuracy can be achieved by GPS, although WAAS is in experimental state. Through the improvement in manufacturing processes, it is inexpensive nowadays to purchase a whole computer system that fits into a single chip. This reduces space constraints and building smaller electromechanical systems becomes affordable, and easy to build.

Due to aforementioned developments, it is becoming possible for us to implement coordinated tasks by utilizing large number of robots. Their actions will utilize ad-hoc communication networks, satellite navigation, and on-board computing power. There are several advantages of using groups of agents. For instance, it increases robustness to failure of single agents or communication links. Additionally, it enables us to accomplishing tasks that are hard or impossible to implement by a single robot.

Although technology provides the physical components of such multi-vehicle networks, the potential benefits of such systems are not yet being realized in high-performance applications. As of today, the fundamental limitation is a lack of understanding of how to assemble and coordinate the individual vehicles into a coherent whole. In other words, there are no systematic methodologies to control large-scale, reliable, distributed systems such as a multi-vehicle network performing complex tasks.

Examples of complex tasks include coverage and surveillance problems such as optimal sensor placement for signal detection, optimal sensor tuning, minimum-time to intercept, and visibility. We can also add exploration and land mine detection, search and gradients identification problems (plume tracing), map building and mosaicing, target acquisition and identification. These problems are relevant in networks of devices of multiple scales all the way from tiny embedded sensors to small-to-medium size vehicles.

In this study we would like to contribute to these emerging technologies in two stages. In the first stage, we will build a theoretical background for control and coordination algorithms for groups of vehicles. The focus of this stage will be on autonomous vehicle networks performing distributed sensing tasks, where each vehicle acts as a tunable sensor. This problem is also referred to as *coverage control* for multi-vehicle systems. In the second stage, we will develop a testbed composed of a group of robotic vehicles. Furthermore we will implement and illustrate the ideas that we have introduced during the first stage.

The development of the coordination algorithms is the theoretic contribution. It is in an advanced state and appeared in [16]. The development of the testbed is the uncompleted work and it is based on well known engineering concepts and methods. Development of the testbed will enable us to illustrate the performance of the proposed algorithms. The contribution of developing the testbed comes from integration of multiple disciplines and technologies, such as, estimation, identification, non-linear control, path planning, real-time programming and embedded systems.

This document is organized as follows. Section 2 is a literature review that is relevant to our work. In Section 3 we develop distributed coordination algorithms for coverage control. In Section 4 we explain the components of the testbed. we propose the work that we plan to complete during this study in Section 5.

2 Literature Review

Mobile sensing networks

According to [18, 43, 57, 71], Working prototypes of active sensing networks have already been developed. In [57], launchable miniature mobile robots communicate through a preliminary wireless network. The vehicles are equipped with sensors for vibrations, acoustic, magnetic, and IR signals as well as an active CMOS module. This construction is based on groups, where each group has one central processing unit that is in charge of the high level control, and a number of very small units that is dependent on central processing unit.

A second system is suggested under the name of Autonomous Oceanographic Sampling Network, see [18]. In this case, underwater vehicles are envisioned measuring temperature, currents, and other distributed oceanographic signals. The vehicles communicate via an acoustic local area network and coordinate their motion in response to local sensing information and to evolving global data. This mobile sensing network is meant to provide the ability to sample the environment adaptively in space and time.

There are also tested systems for wireless multi vehicle systems that are under development, i.e., see [17, 19, 63]. The testbed in [63] consists of planar aerial vehicles (hovercraft) which communicate via a wireless network. In this setting the high level control is central. The setup in [17] is very similar to the one in [63], except that each vehicle has its own decision unit and uses central computer for navigation. In [19], each vehicle has a simple controller on board that receives input from a dedicated ground computer one for each.

Optimal sensor allocation and coverage problems

A basic problem that we consider in this study is that of characterizing and optimizing notions of quality-of-measurement provided by an adaptive sensor network in a dynamic environment. Within the context of this goal, we introduce a notion of *sensor coverage* that formalizes an optimal sensor placement problem. This spatial resource allocation problem is the subject of a discipline called locational optimization [22, 51, 52, 53, 65].

Because locational optimization problems are widely studied, a number of methods are indeed available to tackle coverage problems; see [23, 51, 52, 53]. However, most currently-available algorithms are not applicable to mobile sensing networks because they inherently assume a centralized computation for a limited size problem in a known static environment. This is not the case in multi-vehicle networks which, instead, rely on a distributed communication and computation architecture. Although an ad-hoc wireless network provides the ability to share some information, no global omniscient leader might be present to coordinate the group. The inherent spatially-distributed nature and limited communication capabilities of a mobile network invalidate classic approaches to algorithm design.

Distributed algorithms for cooperative control

During recent years, we see a significant amount of research focused on motion planning and coordination problems for multi-vehicle systems. Subjects include geometric patterns [2, 64, 66, 73], formation control [6, 21], gradient climbing [5], and conflict avoidance [67]. However, the distributed coordination laws for dynamic networks are being proposed just recently; e.g., see [36, 41, 54].

Along the line of behavior-based robotics, algorithms have been designed for sophisticated cooperative tasks; see [3, 6, 7, 12, 26, 58]. An example of coverage control is discussed in [35]. However, no formal results are currently available on how to design reactive control laws, check their correctness, and guarantee their optimality with respect to an aggregate objective.

The studies in [10, 68] discuss distributed asynchronous algorithms as networking algorithms, rate and flow control, and gradient descent flows, from a numerical optimization viewpoint. Typically, both of these references consider networks with fixed topology, and do not address algorithms over ad-hoc dynamically changing networks. Another common assumption is that any time an agent communicates its location, it broadcasts it to every other agent in the network. In our setting, this would require a non-distributed communication set-up.

Estimation, path planning and control for a single robot

The problem of estimating state of a system from noisy sensor information has been widely studied in literature. When the system dynamics and observer models are linear, the classical approach is to calculate the minimum mean square error estimate by the *Kalman Filter* [37]. Several suitable extensions have been sought to the Kalman filter for the cases of nonlinear system dynamics and observers [20, 31, 59, 60, 69]. The most widely used approach is Extended Kalman filtering (EKF), which attempts to cope with nonlinearities simply by linearization, see [20, 59] for a comprehensive study.

It is well known that the optimal solution to the nonlinear filtering problem is infinite dimensional [44]. A variety of suboptimal approaches has been developed, broadly classified as Monte Carlo methods and analytical approximations [31, 60, 69]. [69] introduces unscented Kalman filter (UKF) to address the limitations of EKF and other suboptimal approaches. UKF is based on a deterministic sampling approach to capture the mean and covariance estimates with a minimal set of sample points. Additionally, some recent papers investigate general observation processes where sampling is randomly spaced in time [48].

Various numerical techniques deal with trajectory and path planning problems. In numerical optimal control, the optimal open-loop inputs and the resulting trajectories are often obtained through nonlinear programming. Because the optimization problem is infinite dimensional, various forms of transcription are used to cast the variational problem into a nonlinear program; see [33] for an early reference on the collocation method, and [11, 24, 70] for various recent surveys. Within the context of robotic path planning, the most successful solution are randomized methods, e.g., see the recent overview article [45]. Specific examples of algorithms include randomized potential field [9] and probabilistic roadmaps [38].

In [25] optimal trajectory generation for classes of differentially flat systems with auxiliary constraints has been studied. The constraints are satisfied through linearization and offline optimization. [55] investigates fuel-optimal trajectory generation subject to avoidance requirements via mixed integer linear programming (MILP). To use MIPL, only linear models can be considered. [27, 28] study randomized path planning algorithms for dynamical systems in the presence of fixed and moving obstacles. This work differs from the original RRT approach by relying on a precomputed lookup-table of optimal cost-to-go function as distance measure.

Embedded hardware and software

Text [14] covers designing, building and debugging embedded hardware. Information on MicroChip PIC, Atmel AVR and Motorola 68000 series processors, as well as architectures of digital signal processors are included. Additionally, the book gives information about serial communication protocols, such as SPI, I^2C, RS-232, RS-422, and networking protocols such as, RS-485, CAN and Ethernet. The books [34, 72] also discuss embedded hardware, focusing on x86-, StrongARM-, and PowerPC-based target boards. Information on peripheral interfacing via serial, USB, and parallel ports, and memory mapped I/O is also given. References [39, 56, 61] introduce avionics, embedded systems for navigation, and their components.

There are various books that discuss development of software for embedded systems. Books [34, 72] give information on building embedded systems utilizing the Linux operating system. Details include downloading and setting up development and debugging tools, building a kernel and a root file system, manipulating storage devices, setting up boot loader and networking. Text [4] covers threaded programming, including the details of synchronization and management of threads and POSIX standard. Text [29] gives information on POSIX.4, real time programming standard on UNIX operating systems. Text [62] discusses interprocess communications, including pipes, FIFOs, message queues, mutexes, semaphores, and shared memory.

3 Development of Distributed Algorithms for Coverage Control

3.1 Locational Optimization

In this section we describe a collection of known facts about locational optimization problem. Along our study, we interchangeably refer to the elements of the network as sensors, agents, vehicles, or robots. We let $\mathbb{R}_+$ be the set of nonnegative real numbers, $\mathbb{N}$ be the set of positive natural numbers and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

Let Q be a convex polytope in $\mathbb{R}^N$ including its interior, and let $\|\cdot\|$ denote the Euclidean distance function. We call a map $\phi : Q \to \mathbb{R}_+$ a *distribution density function* if it represents a measure of information or probability that some event take place over Q. One example of a distribution density function can be seen in Figure 1. Let $P = [p_1, \ldots, p_n]$ be the *location of n sensors*, each moving in the space Q. Because of noise and loss of signal strength, the *sensing performance* at point q taken from ith sensor at the position p_i degrades with the distance $\|q - p_i\|$ between q and p_i. Let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be a non-decreasing differentiable function to describe this degradation. Therefore, $f(\|q - p_i\|)$ provides a quantitative assessment of how poor the sensing performance is.

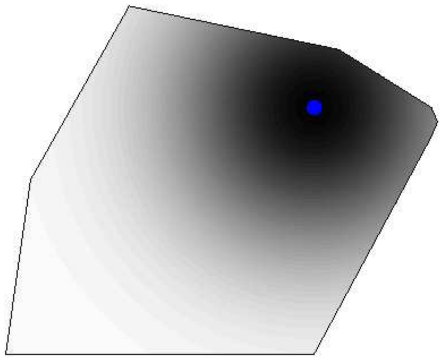


Figure 1: Contour plot on a polygonal environment of the Gaussian density function $\phi = \exp(-x^2 - y^2)$.

A *partition* of Q is defined as a collection of n polytopes $\mathcal{W} = \{W_1, \ldots, W_n\}$ with disjoint interiors whose union is Q. We say that two partitions $\mathcal{W}$ and $\mathcal{W}'$ are equal if W_i and W_i' only differ by a set of ϕ-measure zero, for all $i \in \{1, \ldots, n\}$.

We consider the task of minimizing the locational optimization function

$$\mathcal{H}(P, \mathcal{W}) = \sum_{i=1}^{n} \int_{W_i} f(\|q - p_i\|)\phi(q)dq, \tag{1}$$

where we assume that the ith sensor is responsible for measurements over its "dominance region" W_i. The function $\mathcal{H}$, thus the optimization, has to be minimized with respect to both the sensors location P, and the assignment of the dominance regions $\mathcal{W}$ to achieve a local minim. This problem is referred to as a *facility location* and in particular as a continuous p-median problem in [22].

If the positions of any two agents are interchanged, along with their associated regions of dominance, the value of the locational optimization function $\mathcal{H}$ is not affected. More precisely, let Σ_n denote the discrete group of permutations of n elements, then $\mathcal{H}(p_1, \dots, p_n, W_1, \dots, W_n) = \mathcal{H}(p_{\sigma(1)}, \dots, p_{\sigma(n)}, W_{\sigma(1)}, \dots, W_{\sigma(n)})$ for all $\sigma \in \Sigma_n$. To eliminate this discrete redundancy, one could take natural action of Σ_n on Q^n, and consider Q^n/Σ_n as the configuration space for the position P of the n vehicles.

3.2 Voronoi Partitions

If the sensors' locations are fixed, the optimal partition of Q is the *Voronoi partition* $\mathcal{V}(P) = \{V_1, \dots, V_n\}$ generated by the points $(p_1, \dots, p_n)$,

$$V_i = \{q \in Q \mid \|q - p_i\| \leq \|q - p_j\|, \forall j \neq i\},$$

see [16]. The set of regions $\{V_1, \dots, V_n\}$ is called the Voronoi diagram for the generators $\{p_1, \dots, p_n\}$. The two Voronoi regions V_i and V_j are called adjacent if they share an edge. In this case p_i is called a *(Voronoi) neighbor* of p_j. The set of indexes of the Voronoi neighbors of p_i is denoted by $\mathcal{N}(i)$. Clearly, $j \in \mathcal{N}(i)$ if and only if $i \in \mathcal{N}(j)$. We also define the (i, j)-face as $\Delta_{ij} = V_i \cap V_j$. Voronoi diagrams can be defined with respect to various distance functions, e.g., the 1-, 2-, s-, and ∞-norm over $Q = \mathbb{R}^m$, see [42]. For a comprehensive treatment on Voronoi diagrams see [52]. Some useful facts about the Euclidean setting are the following: if Q is a convex polytope in a N-dimensional Euclidean space, the boundary of each V_i is the union of $(N-1)$-dimensional convex polytopes.

For the following, we will denote the optimization problem (1) as,

$$\mathcal{H}_{\mathcal{V}}(P) = \mathcal{H}(P, \mathcal{V}(P)).$$

By using the definition of the Voronoi partition, we have $\min_{i\in\{1,\dots,n\}} f(\|q-p_i\|) = f(\|q-p_j\|)$ for all $q \in V_j$. Therefore,

$$\mathcal{H}_{\mathcal{V}}(P) = \int_Q \min_{i\in\{1,\dots,n\}} f(\|q-p_i\|)\phi(q)dq\,, \qquad (2)$$
$$= E_{(Q,\phi)}\left[\min_{i\in\{1,\dots,n\}} f(\|q-p_i\|)\right],$$

that is, the locational optimization function can be interpreted as an expected value composed with a min operation. This is the usual way in which the problem is presented in the facility location and operations research literature [22]. One can show [23] that

$$\frac{\partial \mathcal{H}_{\mathcal{V}}}{\partial p_i}(P) = \frac{\partial \mathcal{H}}{\partial p_i}(P, \mathcal{V}(P)) = \int_{V_i} \frac{\partial}{\partial p_i} f\left(\|q-p_i\|\right)\phi(q)dq, \qquad (3)$$

i.e., the partial derivative of $\mathcal{H}_{\mathcal{V}}$ with respect to the ith sensor only depends on its own position and the position of its Voronoi neighbors. Therefore the computation of the derivative of $\mathcal{H}_{\mathcal{V}}$ with respect to the sensors' location is decentralized *in the sense of Voronoi*. Furthermore, the Voronoi partition $\mathcal{V}$ depends at least continuously on $P = (p_1,\dots,p_n)$, the function $\mathcal{H}_{\mathcal{V}}$ is at least continuously differentiable.

3.3 Centroidal Voronoi Partitions

Let us recall some basic quantities associated to a region $V \subset \mathbb{R}^N$ and a mass density function ρ. The (generalized) mass, centroid (or center of mass), and polar moment of inertia are defined as

$$M_V = \int_V \rho(q)\,dq, \quad C_V = \frac{1}{M_V}\int_V q\,\rho(q)\,dq,$$
$$J_{V,p} = \int_V \|q-p\|^2\,\rho(q)\,dq.$$

Additionally, by the parallel axis theorem, one can write,

$$J_{V,p} = J_{V,C_V} + M_V\,\|p - C_V\|^2 \qquad (4)$$

where $J_{V,C_V} \in \mathbb{R}_+$ is defined as the polar moment of inertia of the region V about its centroid C_V.

Let us consider again the locational optimization problem (1), and suppose now we are strictly interested in the setting

$$\mathcal{H}(P, \mathcal{W}) = \sum_{i=1}^{n} \int_{W_i} \|q-p_i\|^2 \phi(q)dq, \qquad (5)$$

that is, we assume $f(\|q-p_i\|) = \|q-p_i\|^2$. The parallel axis theorem leads to simplifications for both the function $\mathcal{H}_\mathcal{V}$ and its partial derivative:

$$\begin{aligned}\mathcal{H}_\mathcal{V}(P) &= \sum_{i=1}^{n} J_{V_i,C_{V_i}} + \sum_{i=1}^{n} M_{V_i} \|p_i - C_{V_i}\|^2 \\ \frac{\partial \mathcal{H}_\mathcal{V}}{\partial p_i}(P) &= 2M_{V_i}(p_i - C_{V_i}).\end{aligned}$$

Here the mass density function is $\rho = \phi$. It is convenient to define

$$\mathcal{H}_{\mathcal{V},1} = \sum_{i=1}^{n} J_{V_i,C_{V_i}}\,, \quad \mathcal{H}_{\mathcal{V},2} = \sum_{i=1}^{n} M_{V_i}\|p_i - C_{V_i}\|^2\,.$$

Therefore, the local minimum points for both of the location optimization function $\mathcal{H}_\mathcal{V}$ are *centroids* of their Voronoi cells. Equivalently,

$$C_{V_i} = \operatorname{argmin}_{p_i} \mathcal{H}_\mathcal{V}(P). \tag{6}$$

The partitions and points from (6) for $\mathcal{H}$ are called *centroidal Voronoi partitions.* We will refer to a sensors' configuration as a *centroidal Voronoi configuration* if it gives rise to a centroidal Voronoi partition.

3.4 Continuous and Discrete Time Lloyd Descent

In this section, we introduce algorithms to compute the location of sensors that minimize the cost $\mathcal{H}$. We first discuss the Lloyd algorithm in quantization theory. Then, we will treat them in two categories. In section 3.4.1, we propose a continuous-time version of the classic Lloyd algorithm. In section 3.4.2, we present a family of algorithms in discrete-time. In both settings, we show that the proposed algorithms are *gradient descent flows.*

Lloyd algorithm in quantization theory [32, 47] is usually presented as follows: given the location of n agents, $p_1, \ldots, p_n$, (i) construct the Voronoi partition corresponding to $P = (p_1, \ldots, p_n)$; (ii) compute the mass centroids of the Voronoi regions found in step (i). Set the new location of the agents to these centroids; and return to step (i).

3.4.1 A continuous-time Lloyd algorithm

Assume that the sensors location obeys the dynamical behavior,

$$\dot{p}_i = u_i.$$

Let $\mathcal{H}_{\mathcal{V}}$ be a cost function to be minimized and impose that the location p_i follows a gradient descent. Equivalently, consider $\mathcal{H}_{\mathcal{V}}$ as a Lyapunov function and stabilize the multi-vehicle system to one of its local minima via dissipative control. Formally, we set

$$u_i = -k_{\text{prop}}(p_i - C_{V_i}), \tag{7}$$

where $k_{\text{prop}} \in \mathbb{R}_+$. Further, we assume that the partition $\mathcal{V}(P) = \{V_1, \dots, V_n\}$ is continuously updated.

Proposition 3.1 (Continuous-time Lloyd descent). *For the closed-loop system induced by equation* (7)*, the sensors location converges asymptotically to the set of critical points of* $\mathcal{H}_{\mathcal{V}}$*, i.e., the set of centroidal Voronoi configurations on* Q*. Assuming this set is finite, the sensors location converges to a centroidal Voronoi configuration.*

Proof. Under the control law (7), we have

$$\frac{d}{dt}\mathcal{H}_{\mathcal{V}}(P(t)) \;=\; \sum_{i=1}^{n} \frac{\partial \mathcal{H}_{\mathcal{V}}}{\partial p_i}\,\dot{p}_i \;=\; -2k_{\text{prop}} \sum_{i=1}^{n} M_{V_i} \|p_i - C_{V_i}\|^2 \;=\; -2k_{\text{prop}}\mathcal{H}_{\mathcal{V},2}(P(t)).$$

By LaSalle's principle [40], the sensors location converges to the largest invariant set contained in $\mathcal{H}_{\mathcal{V},2}^{-1}(0)$, which is precisely the set of centroidal Voronoi configurations. Since this set is clearly invariant for (7), we get the stated result. If $\mathcal{H}_{\mathcal{V},2}^{-1}(0)$ consists of a finite collection of points, then $P(t)$ converges to one of them by continuity of $\mathcal{H}_{\mathcal{V}}$. □

3.4.2 A family of discrete-time Lloyd algorithms

Let T be a continuous mapping $T : Q^n \to Q^n$ providing the following two properties,

(i) for all $i \in \{1, \dots, n\}$, $\|T_i(P) - C_{V_i(P)}\| \leq \|p_i - C_{V_i(P)}\|$, where T_i denotes the ith component of T,

(ii) if P is not centroidal, then there exists at least one j, such that $\|T_j(P) - C_{V_j(P)}\| < \|p_j - C_{V_j(P)}\|$.

Property (i) assures that, the agents of the network do not increase their distance to its corresponding centroid, if they moved according to T. Property (ii) assures that at least one robot moves at each iteration and strictly approaches the centroid of its Voronoi region. Therefore, by properties (i) and (ii), the fixed points of T are the set of centroidal Voronoi configurations.

Proposition 3.2 (Discrete-time Lloyd descent). *Let $T : Q^n \to Q^n$ be a continuous mapping satisfying properties (a) and (b). Let $P_0 \in Q^n$ denote the initial sensors' location. Then, the sequence $\{T^m(P_0) \mid m \in \mathbb{N}\}$ converges to the set of centroidal Voronoi configurations. If this set is finite, then the sequence $\{T^m(P_0) \mid m \in \mathbb{N}\}$ converges to a centroidal Voronoi configuration.*

Proof. Consider $\mathcal{H}_{\mathcal{V}} : Q^n \to \mathbb{R}_+$ as an objective function for the algorithm T. Using the parallel axis theorem, $\mathcal{H}(P, \mathcal{W}) = \sum_{i=1}^n J_{W_i, C_{W_i}} + \sum_{i=1}^n M_{W_i} \|p_i - C_{W_i}\|^2$, and therefore

$$\mathcal{H}(P', \mathcal{W}) \leq \mathcal{H}(P, \mathcal{W}) , \tag{8}$$

as long as $\|p_i' - C_{W_i}\| \leq \|p_i - C_{W_i}\|$ for all $i \in \{1, \dots, n\}$, with strict inequality if for any i, $\|p_i' - C_{W_i}\| < \|p_i - C_{W_i}\|$. In particular, $\mathcal{H}(C_{\mathcal{W}}, \mathcal{W}) \leq \mathcal{H}(P, \mathcal{W})$, with strict inequality if $P \neq C_{\mathcal{W}}$, where $C_{\mathcal{W}}$ denotes the set of centroids of the partition $\mathcal{W}$. Moreover, since the Voronoi partition is the optimal one for fixed P, we also have

$$\mathcal{H}(P, \mathcal{V}(P)) \leq \mathcal{H}(P, \mathcal{W}) , \tag{9}$$

with strict inequality if $\mathcal{W} \neq \mathcal{V}(P)$.

Now, because of property (a) of T, inequality (8) yields

$$\mathcal{H}(T(P), \mathcal{V}(P)) \leq \mathcal{H}(P, \mathcal{V}(P)) = \mathcal{H}_{\mathcal{V}}(P) ,$$

and the inequality is strict if P is not centroidal by property (b) of T. In addition,

$$\mathcal{H}_{\mathcal{V}}(T(P)) = \mathcal{H}(T(P), \mathcal{V}(T(P))) \leq \mathcal{H}(T(P), \mathcal{V}(P)) ,$$

because of (9). Hence, $\mathcal{H}_{\mathcal{V}}(T(P)) \leq \mathcal{H}_{\mathcal{V}}(P)$, and the inequality is strict if P is not centroidal. Therefore, $\mathcal{H}_{\mathcal{V}}$ is a descent function for the algorithm T. The theorem follows from the global convergence theorem from [40], and continuity of T. ☐

3.4.3 Computations over polygons with uniform density

In this section, we study some of the closed-form expression for the control laws introduced above. Assume the Voronoi region V_i is a convex polygon (i.e., a polytope in $\mathbb{R}^2$) with N_i vertexes labeled $\{(x_0, y_0), \dots, (x_{N_i-1}, y_{N_i-1})\}$ such as in Fig. 2. It is convenient to define $(x_{N_i}, y_{N_i}) = (x_0, y_0)$. We also assume that the density function is $\phi(q) = 1$.

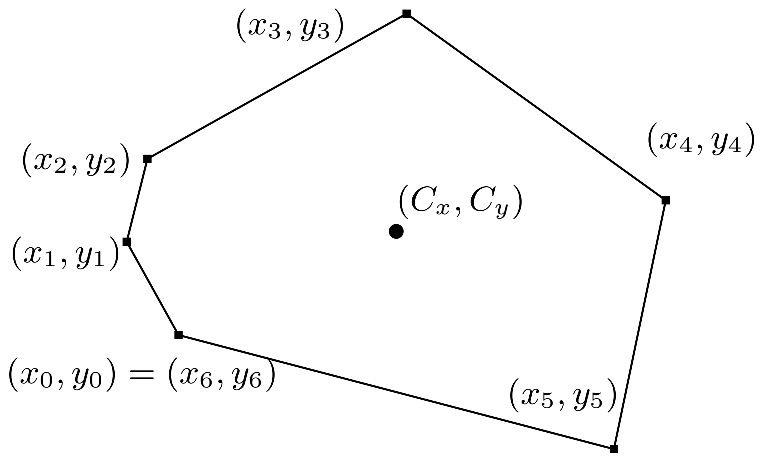


Figure 2: Notation conventions for a convex polygon.

By evaluating the integrals corresponding to mass, center of mass and moment of inertia, one can obtain the following closed-form expressions

$$\begin{aligned} M_{V_i} &= \frac{1}{2} \sum_{k=0}^{N_i-1} (x_k y_{k+1} - x_{k+1} y_k) \\ C_{V_i,x} &= \frac{1}{6 M_{V_i}} \sum_{k=0}^{N_i-1} (x_k + x_{k+1})(x_k y_{k+1} - x_{k+1} y_k) \\ C_{V_i,y} &= \frac{1}{6 M_{V_i}} \sum_{k=0}^{N_i-1} (y_k + y_{k+1})(x_k y_{k+1} - x_{k+1} y_k) \,. \end{aligned} \tag{10}$$

To present a simple formula for the polar moment of inertia, let $\bar{x}_k = x_k - C_{V_i,x}$ and $\bar{y}_k = y_k - C_{V_i,y}$, for $k \in \{0, \ldots, N_i - 1\}$. Then, the polar moment of inertia of a polygon about its centroid, $J_{V_i,C}$ becomes

$$J_{V_i,C_{V_i}} = \frac{1}{12} \sum_{k=0}^{N_i-1} (\bar{x}_k \bar{y}_{k+1} - \bar{x}_{k+1} \bar{y}_k) \;\cdot (\bar{x}_k^2 + \bar{x}_k x_{k+1} + \bar{x}_{k+1}^2 + \bar{y}_k^2 + \bar{y}_k \bar{y}_{k+1} + \bar{y}_{k+1}^2) \,.$$

The proof of these formulas is based on decomposing the polygon into the union of disjoint triangles. We refer to [15] for analog expressions over $\mathbb{R}^N$. We would like to note that equation (10) for a polygon's centroid leads to a closed-form *algebraic* expression for the control law in equation (7) as a function of the neighboring vehicles' location.

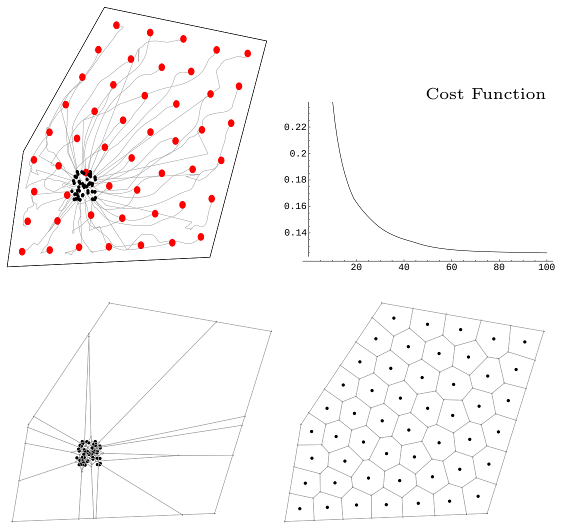


Figure 3: Uniform distribution of sensors obtained by 50 vehicles in a polygonal environment. The vehicles' initial positions are in a tight group in the lower left corner and their final positions are optimally distributed.

3.4.4 Numerical simulations

In this section we display some simulation results to show the performance of the continuous-time Lloyd algorithm. The algorithm is implemented in `Mathematica` as a single centralized program. Figure 3 illustrates the result when the distribution density function is uniform.

Figure 4 displays the result when the distribution density function is an inverse exponential about the location shown by the large dot. In both simulations the number of agents were 50. One can see agents' initial locations in the lower left corner, final locations in the lower right corner, the cost function vs. time in the upper right corner, and the path generated in the upper left corner.

3.5 Asynchronous Distributed Implementations

In this section we show an implementation of the Lloyd gradient algorithm in asynchronous distributed fashion. In Section 3.5.1 we describe our model for a network of robotic agents, Next, we provide a distributed algorithm for the local computation and maintenance of the Voronoi cells and communication radius. Finally, in Section 3.5.3 we propose a distributed

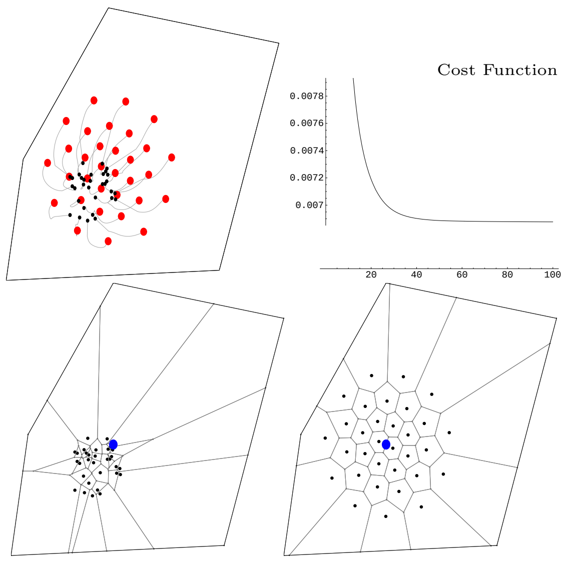


Figure 4: Non-uniform setting. The distribution density function has an inverse exponential about the location shown by the large circle in the bottom left and right figures.

asynchronous implementation of the Lloyd algorithm based on the gradient optimization algorithm as described in [68].

3.5.1 Modeling an asynchronous distributed network of mobile robotic agents

We start by introducing the notion of *robotic agent with computation, communication, sensing, and control capabilities* as the ith element of a network. Each vehicle has access to its unique identifier i. The ith agent occupies a location $p_i \in Q \subset \mathbb{R}^N$ and it is capable of moving in space, at any time $t \in \mathbb{R}_+$ for any period of time $\delta t \in \mathbb{R}_+$, according to a first order dynamics of the form:

$$\dot{p}_i(s) = u_i, \qquad \|u_i\| \leq 1\,, \quad \forall s \in [t, t+\delta t]. \tag{11}$$

Each agent has one processor, i.e. agent i is associated with processor i. The processors have the ability of allocating continuous and discrete states and performing operations on them. The processor i has access to the agent's location p_i and determines the *control pair* $(\delta t, u_i)$. Additionally, the processor i has access to a local clock $t_i \in \mathbb{R}_+$, and a *scheduling sequence*, i.e., an increasing sequence of times $\{T_{i,k} \in \mathbb{R}_+ \mid k \in \mathbb{N}_0\}$ such that $T_{i,0} = 0$ and $0 < t_{i,\min} < T_{i,k+1} - T_{i,k} < t_{i,\max}$. The processor i agent is capable of transmitting information to any other agent within a closed disk of radius $R_i \in \mathbb{R}_+$.

We assume the communication radius R_i to be a quantity controllable by the ith processor and the corresponding communication bandwidth to be limited. We represent the information flow between the agents by means of "send" (within specified radius R_i) and "receive" commands with a finite number of arguments. We also assume that all communication between agents and all sensing of agents locations to be always accurate and instantaneous.

Consider the closed-loop system formed by the evolution of a network of n agents, according to equation (11). The network evolution is said to be *Voronoi-distributed* if each $u_i(p_1, \ldots, p_n)$ can be written as a function of the form $u_i(p_i, p_{i_1}, \ldots, p_{i_m})$, with $i_k \in \mathcal{N}(i)$, $k \in \{1, \ldots, m\}$. It is well known that, for $n \geq 3$ there are at most $3n - 6$ neighborhood relationships in a planar Voronoi diagram [52]. Therefore, the number of Voronoi neighbors of each site is on average less than or equal to 6, and Voronoi-distributed algorithms lead to scalable networks. Finally, note that the set of indexes $\{i_1, \ldots, i_m\}$ for a specific generator p_i of a Voronoi-distributed dynamical system is not the same for all possible configurations and time instances of the network, i.e., the topology of the closed-loop system is *dynamic.*

3.5.2 Computation and maintenance of Voronoi cell and communication radius

A key requirement of the Lloyd algorithm presented in Section 3.4 is that each agent needs to know the relative location of each Voronoi neighbor to be able to compute its own Voronoi cell. The ability of locating neighbors plays a central role in numerous other algorithms, especially for localization, media access, routing, and power control in ad-hoc wireless communication networks, e.g. see [13, 30, 46]. In this study, the agents are able to obtain this information from the communication layer. In what follows, we provide a distributed asynchronous algorithm for the local computation and maintenance of Voronoi cells and communication radius.

Consider a robotic agent with communication capability. The processor i stores the information, such as position of the other agents in the state variable P^i. The objective is to determine the smallest distance R_i for vehicle i which provides sufficient information

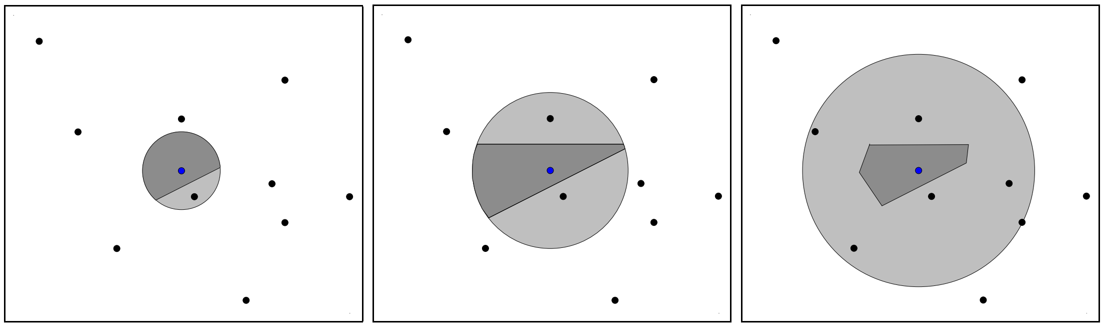


Figure 5: An execution (from left to right) of ADJUST COMMUNICATION RADIUS ALGORITHM: the sensing disk $\overline{B}(p_i, R_i)$ is in light gray, and the Voronoi cell estimate $W(p_i, R_i)$ is the darker gray region.

to compute the Voronoi cell V_i. We start by noting that V_i is a superset of the convex set

$$W(p_i, R_i) = \overline{B}(p_i, R_i) \cap \big(\cap_{j:\|p_i - p_j\| \le R_i} S_{ij} \big), \tag{12}$$

where $\overline{B}(p_i, R_i) = \{q \in Q \mid \|q - p_i\| \le R_i\}$ and the half planes S_{ij} are

$$\{q \in \mathbb{R}^N \mid \|q - p_i\| \le \|q - p_j\|\}.$$

One can show that all Voronoi neighbors of p_i are within distance R_i from p_i, provided R_i is twice as large as the maximum distance between p_i and the vertexes of $W(p_i, R_i)$. Therefore, the minimum adequate communication radius is $R_{i,\min} = 2 \max_{q \in W(p_i, R_{i,\min})} \|p_i - q\|$. This argument guarantees the correctness of the algorithm outlined in Table 1, and visualized in Figure 5. Here, we assume that `update` involves two steps:

send $\big($"request to reply"$, p_i(t_i)\big)$ within radius R_i
receive $\big($"response"$, p_j\big)$ from all agents within radius R_i

Further, we require each agent to perform the following event-driven task: if the ith agent receives at any time t_i a "request to reply" message from the jth agent located at position p_j, it executes:

send $\big($"response"$, p_i(t_i)\big)$ within radius $\|p_i(t) - p_j\|$

3.5.3 Asynchronous distributed implementation of coverage control

In this section we present a modification to Lloyd algorithm for the solution of the optimization problem (1), so that the algorithm can be implemented by an asynchronous

Name:	ADJUST COMMUNICATION RADIUS ALGORITHM
Goal:	maintain communication radius for Voronoi cell calculation
Requires:	communication device with controllable radius R_i

At time t_i, local agent i performs:

```
1: initialize $R_i$, detect all $p_j$ within radius $R_i$
2: update $P^i(t_i)$, compute $W(p_i(t_i), R_i)$
3: while $R_i < 2\max_{q \in W(p_i(t_i),R_i)} \|p_i(t_i) - q\|$ do
4:     set $R_i := 2R_i$
5:     update $P^i(t_i)$, compute $W(p_i(t_i), R_i)$
6: end while
7: set $R_i := 2\max_{q \in W(p_i(t_i),R_i)} \|p_i(t_i) - q\|$
8: set $V_i := W_i(p_i(t_i), R_i)$
```

Table 1: Algorithm for adjusting communication radius

distributed network of robotic agents. For simplicity, we assume that at time 0 all clocks are synchronized and that each agent knows at 0 the exact location of every other agent. The algorithm COVERAGE ALGORITHM is illustrated in Table 2.

We have the following proposition, as a consequence of the results in [68, Theorem 3.1 and Corollary 3.1].

Proposition 3.3. *Let $P_0 \in Q^n$ denote the initial sensors location. Let $\{T_k\}$ be the sequence in increasing order of all the scheduling sequences of the agents of the network. Assume $\inf_k\{T_k - T_{k-1}\} > 0$. Then, there exists a sufficiently small $\delta_* > 0$ such that if $0 < \delta t \leq \delta_*$, the* COVERAGE ALGORITHM *converges to the set of critical points of $\mathcal{H}_{\mathcal{V}}$, that is, the set of centroidal Voronoi configurations.*

3.6 Geometric patterns and formation control

Here we suggest the use of decentralized coverage algorithms as formation control algorithms, and we present various density functions that lead the multi-vehicle network to predetermined geometric patterns. In particular, we present simple density functions that

Name:	Coverage algorithm
Goal:	distributed optimal agent location
Assumes:	$p_i(t+1) = p_i(t) + u_i$, $\|u_i\| \leq 1$
Requires:	(i) Voronoi cell computation
	(ii) centroid and mass computation
	(iii) positive real δ_0
	(iv) Adjust communication radius algorithm

For $i \in \{1, \dots, n\}$, ith agent performs at $t_i = T_{i,0} = 0$:

```
0: set $P^i(T_{i,0}) := (p_1^i(T_{i,0}), \dots, p_n^i(T_{i,0}))$
0: compute Voronoi region $V_i(T_{i,0})$
0: set $V_i := V_i(T_{i,0})$ and $R_i := 2 \max_{q \in V_i} \|p_i - q\|$
```

For $i \in \{1, \dots, n\}$, the ith agent performs at time $t_i = T_{i,k}$ either one of the following threads or both. For some $B_i \in \mathbb{N}$, we require that each thread is executed at least once every B_i steps of the scheduling sequence.

[Information thread]

```
1: run Adjust communication radius algorithm
```

[Control thread]

```
1: compute centroid $C_{V_i}$ and mass $M_{V_i}$ of $V_i$
2: apply control pair $\big(\delta_0,\ M_{V_i}(C_{V_i} - p_i(T_{i,k}))\big)$
```

Table 2: Coverage algorithm

lead to segments, ellipses, polygons, or uniform distributions inside convex environments.

Consider a planar environment, let k be a large positive gain, and denote $q = (x, y) \in Q \subset \mathbb{R}^2$. Let a, b, c be real numbers, consider the line $ax + by + c = 0$, and define the density function

$$\phi_{\text{line}}(q) = \exp(-k(ax + by + c)^2).$$

Similarly, let (x_c, y_c) be a reference point in $\mathbb{R}^2$, let a, b, r be positive scalars, consider the ellipse $a(x - x_c)^2 + b(y - y_c)^2 = r^2$, and define the density function

$$\phi_{\text{ellipse}}(q) = \exp\big(-k(a(x - x_c)^2 + b(y - y_c)^2 - r^2)^2\big).$$

Fig. 6 illustrates the performance of the closed-loop network corresponding to this density function. During the simulations, we observed that the convergence to the desired pattern was rather slow.

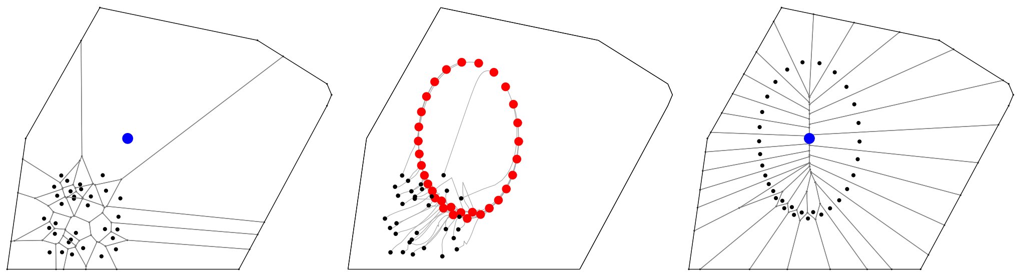


Figure 6: Coverage control for 32 vehicles with ϕ_{ellipse}. The parameter values are: $k = 500$, $a = 1.4$, $b = .6$, $x_c = y_c = 0$, $r^2 = .3$, and $k_{\text{prop}} = 1$.

Finally, define the smooth ramp function $\text{SR}_\ell(x) = x(\arctan(\ell x)/\pi + (1/2))$, and the density function

$$\phi_{\text{disk}}(q) = \exp(-k\,\text{SR}_\ell(a(x - x_c)^2 + b(y - y_c)^2 - r^2)).$$

This density function leads the multi-vehicle network to obtain a uniform distribution inside the ellipsoidal disk $a(x - x_c)^2 + b(y - y_c)^2 \leq r^2$. We illustrate this density function in Fig. 7.

It appears straightforward to generalize these types of density functions to the setting of arbitrary curves or shapes. The proposed algorithms are to be contrasted with the classic approach to formation control based on rigidly encoding the desired geometric pattern.

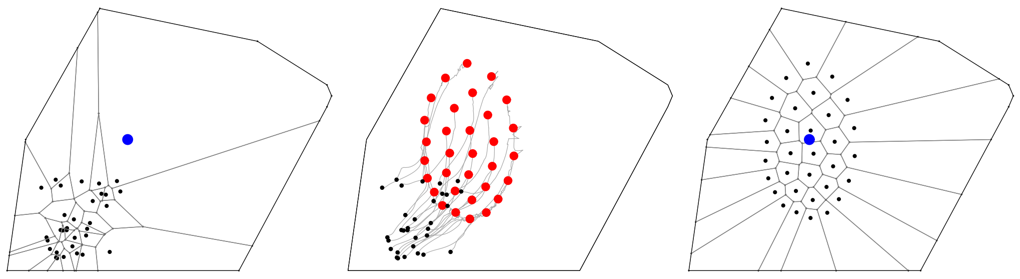


Figure 7: Coverage control for 32 vehicles to an ellipsoidal disk. The density function parameters are the same as in Fig. 6, and $\ell = 10$, $k_{\text{prop}} = 1$.

One disadvantage of the proposed approach is the requirement for a careful numerical computation of Voronoi diagrams and centroids. We refer to [64, 66] for previous work on algorithms for geometric patterns, and to [6, 21] for formation control algorithms.

4 Testbed Development

4.1 Mathematical Model of the Vehicle

To obtain a model for the vehicle behavior, we start with the simple kinematic model of a car, as shown in Figure 8, where we assume no slip. The state vector is $\mathbf{x} = [\ x_c\ y_c\ \theta\ ]^T$, where (x_c, y_c) are the coordinates of the center of the real axis in the inertial reference frame, and θ is the orientation of the vehicle with respect to the y-axis of the inertial reference frame.

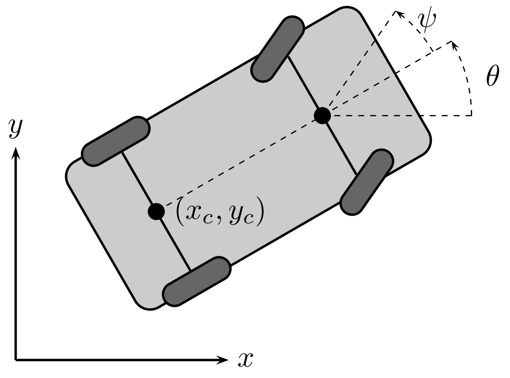


Figure 8: Input and state variables of a kinematic model of a car

We can now state the differential equation of the kinematic model as [49],

$$\dot{\mathbf{x}} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{1}{L}\tan(\psi)v \end{bmatrix} \tag{13}$$

where v is the speed and ψ is the steering angle of the car and they both are assumed to be inputs of the system.

Once the kinematic model has been established, we can add some complexity in order to model the real sytem with better accuracy. First, we will consider the effective voltage applied to the motors as an input rather than the vehicle speed. Thus, the motor torque(T_{motor}) equation is, [50]

$$\begin{aligned} T_{\text{motor}} &= K_{Ti_a} i_a(t) \\ &= \frac{K_{Ti_a}}{R_a}(e_a - e_b(t)) \\ T_{\text{motor}} &= \frac{K_{Ti_a}}{R_a}(e_a - K_{e_b\omega}\omega(t)) \end{aligned} \tag{14}$$

where i_a is the armature current, R_a is the armature resistance, e_a is the applied armature voltage, e_b is the back emf, ω is the angular velocity of the motor and K_{Ti_a}, $K_{e_b\omega}$ are system constants relating torque to current and back emf to angular speed of the motor.

Substituting angular speed with linear speed, incorporating the power train gear ratios into the motor torque equation, and lumping some constants, (14) becomes

$$F_{\text{car}} = K_{Fu}u(t) - K_{Fv}v(t) \tag{15}$$

In equation (15) F_{car} is the thrust of the motor on the vehicle, K_{Fv} is the constant accounting for decrease in force due to speeed of the vehicle, K_{Fu} is the constant relating thrust to effective input, and $u(t) \in [-1, 1]$ is the effective input.

We can further enhance the model by adding viscous friction and static friction terms due to mechanical interactions, by which the equation (15) becomes

$$F_{\text{car}} = K_{Fu}u(t) - K_{Fv}(u(t))v(t) \tag{16}$$

In equation (16) we lump the viscous friction constant into $K_{Fv}(u(t))$. In particular, we would like to point out that generally this term will be different in the cases of power applied, and power not applied.

Now, we can state the speed dynamics as

$$\dot{v}(t) = \frac{1}{M}(K_{Fu}u(t) - K_{Fv}(u(t))v(t)) \tag{17}$$

where, M is the vehicle mass.

We can now move on to introducing steering dynamics. Steering on the vehicles is performed using a standard RC servo. Specifications from the manufacturer's manual make it evident that we can approximate the servo behavior as

$$\dot{\psi}(t) = K_{\psi}(\psi_r(t) - \psi(t)) \tag{18}$$

where K_ψ is the servo travel rate, and ψ_r is the reference steering angle to be tracked.

Using equations (17) and (18), the system differential equation becomes

$$\begin{aligned}
\dot{x_c}(t) &= \cos\theta(t)v(t) \\
\dot{y_c}(t) &= \sin\theta(t)v(t) \\
\dot{\theta}(t) &= \frac{1}{L}\tan\psi(t)v(t) \\
\dot{v}(t) &= \frac{1}{M}(K_{Fu}u(t) - K_{Fv}(u(t))v(t)) \\
\dot{\psi}(t) &= K_\psi(\psi_r(t) - \psi(t))
\end{aligned} \tag{19}$$

where $\psi_r(t), \psi(t) \in [-M_\psi, M_\psi]$.

Some parameters require special attention. K_{Fu} is the constant relating the force applied on the car by the motor to the effective input, $u(t) \in [-1, 1]$, under the assumption that the motor is locked. Force is related to the torque by the constant $K_{\omega \text{v}}$, which relates motor speed (rad/s) to vehicle speed (m/s). We have

$$K_{Fu} = T_{\max} K_{\omega \text{v}}$$

where $T_{\max}$ is the maximum locked motor torque.

$K_{Fv}(u(t))$ is a function of the input. If the input is applied $K_{Fv}(u(t))$ combines the effects of viscous friction due to moving parts of the vehicle and back emf generated by the motor. If the input is not applied, $K_{Fv}(u(t))$ only contains the effects of viscous friction due to moving parts of the vehicle. Thus, $K_{Fv}(u(t))$ can be expressed as

$$K_{Fv}(u) = \begin{cases} C_v, & u = 0, \\ \dfrac{T_{\max}}{s_{\max}} K_{ws}^2 + C_v, & u \neq 0. \end{cases}$$

where $s_{\max}$ is the speed that reduces the motor generated torque to zero, and C_v is the viscous friction coefficient due to moving parts of the vehicle.

We have included the estimated values of the vehicle parameters in Table 3.

Table 3: Vehicle parameters

Parameter	*Explanation*		*Value*	*Unit*
$T_{\max}$	maximum motor torque		0.15	N·m
M	mass of the vehicle		4.80	kg
C_v	viscous friction coefficient		0.85	N·s/m
$K_{\omega s}$	motor speed/vehicle speed	1st gear	396.0	rad/m
		2st gear	233.0	
K_{encoder}	encoder count/distance	1st gear	12978	1/m
		2st gear	7636.0	
K_ψ	servo steering rate		0.12	rad/s
M_ψ	maximum servo steering angle		0.35	rad
L_{axle}	distance between two axles		0.305	m

4.2 Hardware Components

4.2.1 Original vehicle

The original vehicle was an Emaxx 1/10 Scale Monster Truck RC-Car manufactured by Traxxas Inc. It is the vehicle of choice because it has relatively large size, low cost, and can be easily modified to provide a platform for the PC104 single board computer and accessories. The motors provided with the vehicle are powerful enough to accommodate additional weight. Figure 9 shows a top view of the vehicle chassis and components.

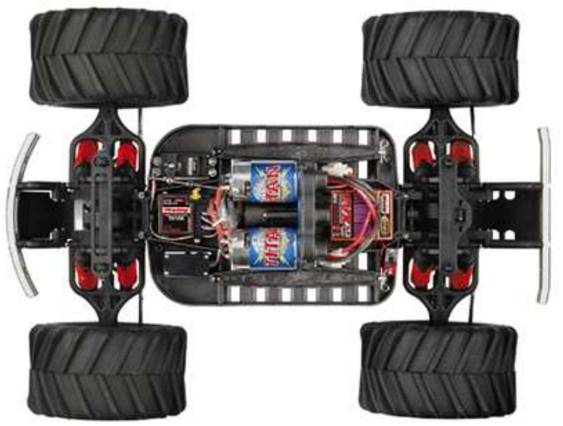


Figure 9: Top view of the vehicle chassis and components

4.2.2 Modifications

One of two motors on the vehicle was removed to reduce the top speed of the vehicle, and the gearing from the removed motor provides a convenient manner in which to include an encoder on the main drive shaft. The suspension of the vehicle can be modified to accommodate increased loads, determined primarily by the heavy batteries. The chassis of the vehicle provides sufficient space for a PC104 board, a DC/DC converter and a few other devices, while housing them safely in the event of a collision.

4.2.3 Single board computer

The single board computer used on these vehicles is the Kontron VNS-786 PC-104 board, shown in Figure 10. The VNS-786 offers a complete embedded PC platform with a footprint of 5.75 x 8 inches. The VNS-786 supports DiskonChip (DOC) solid-state hard drives. We selected a 64 MB DOC as it is sufficient to store the operating system and necessary programs. The board operates on 5V, has two enhanced parallel ports, and four RS-232 serial ports, as well as an integrated 10/100 Ethernet controller. The PC-104 platform is well-suited for this application due to its reliable and rugged stackable expansion bus, which we use for the PCMCIA expansion board to handle the wireless 802.11b Ethernet cards and our custom compass/encoder data acquisition board.

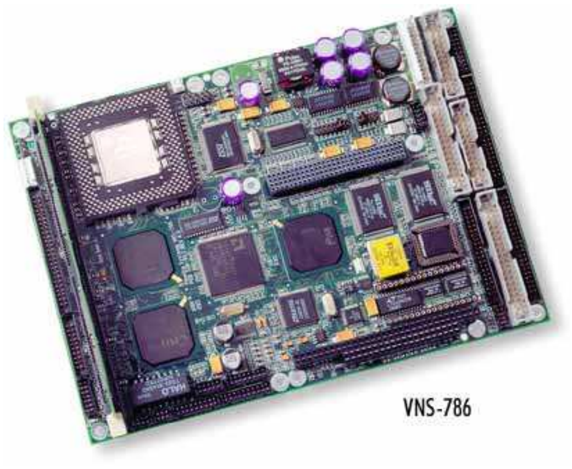


Figure 10: Kontron VNS-786 PC-104 Single-Board Computer

4.2.4 Servo controller board

The SV203 is a PIC16C73 based microcontroller based servo motor controller board that accepts RS-232 serial text based data from a host computer and outputs appropriate signals to control up to 8 RC (radio controlled) servo motors. There is also a 5-channel A/D port, built into the board for reading voltages in the range of 0-5V.

4.2.5 Speed controller

The EVX 3014 14.4V electronic Speed Controller comes packaged with the Emaxx truck. Specificaly, this speed control unit utilizes a PWM (Pulse Width Modulation) signal. In a simple way, PWM is a way of digitally encoding analog signal levels [8].

Through the use of counters, the duty cycle of a square wave is modulated to encode a specific analog signal level. At any given instant of time, the PWM signal level is either the full DC supply voltage or off. The voltage source is supplied to the analog load by means of a repeating series of on and off pulses. The on-time is the time during which the DC supply is applied to the load, and the off-time is the period during which that supply is switched off. Given a sufficient bandwidth, any analog value can be encoded with PWM [8].

Figure 11 shows three different PWM signals. Figure a shows a PWM output at a 10% duty cycle. That is, the signal is on for 10% of the period and off the other 90%. Figures 1b and 1c show PWM outputs at 50% and 90% duty cycles, respectively. These three PWM outputs encode three different analog signal values, at 10%, 50%, and 90% of the full strength. If, for example, the supply is 14.4V and the duty cycle is 10%, a 1.44V analog signal results.

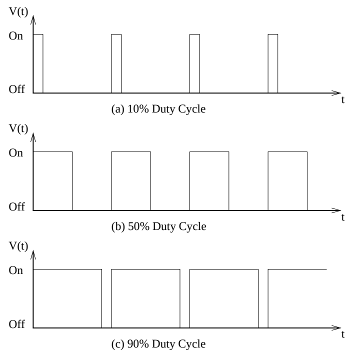


Figure 11: PWM signal samples

4.2.6 GPS sensor

The GPS sensor, Garmin GPS-16 LVS, was specifically developed for interfacing with mobile computing devices and wireless communications equipment, and specifically developed for outdoor, rugged use, see Figure 12. The GPS-16 provides a 1Hz data rate The NMEA serial interface is standard across all GPS units and was simple to interface with the PC-104 boards and the laptop base station. The unit provides an accuracy of 3-5 meters radius and less than 3 meters when using WAAS (Wide Area Augmentation System). WAAS consists of approximately 25 ground reference stations positioned across the United States that monitor GPS satellite data. Two master stations, located on either coast, collect data from the reference stations and create a GPS correction message. This correction accounts for GPS satellite orbit and clock drift plus signal delays caused by the atmosphere and ionosphere. The corrected differential message is then broadcast through one of two geostationary satellites, or satellites with a fixed position over the equator. The information is compatible with the basic GPS signal structure, which means any WAAS-enabled GPS receiver can read the signal.

Figure 12: Garmin GPS-16 LVS

4.2.7 CMPS03 digital compass

The CMPS03 was designed for use in robots as an aid to navigation. The compass uses two Philips KMZ51 magnetic field sensors mounted at right angles to each other to compute the direction of the horizontal component of the Earths magnetic field. The compass uses a PWM signal whose duty cycle represents the angle. The pulse width varies linearly from 1mS (0) to 36.99mS (359.9) and gives 1uS resolution at a rate of 50 Hz.

Figure 13: SM03 Digital Compass

4.2.8 Optical encoder

US Digital S1-50-B8 50 counts per revolution Optical Encoders are used as a solution to measure the local distance the vehicle travels in a given time interval. Optical encoders are typically very high resolution. For this application, high resolution is not necessary, in particular, considering the sampling intervals involved, and the backlash inherent to the four-wheel-drive system transaxle.

Figure 14: US Digital S1-50-B8 50CPR Optical Encoder

4.2.9 Wireless ethernet card

The Linksys WPC11 provides the wireless ethernet connection. The protocol used by the card is the IEEE802.11b standard, at 2.4 GHz, allowing a maximum transfer rate of 11Mbps. The card provides wireless communication up to a distance of 300'. The firmware has been updated for Ad-Hoc wireless setup.

4.3 Hardware Integration

The overall hardware integration diagram, where the components described in the previous section were integrated onto the vehicle, can be seen in figure 15. The PC-104 mainboard and DC/DC converter were mounted to a Lexan platform which was secured to the vehicle's chassis as can be seen in Figure 9. The PCMCIA adapter, used by the wireless ethernet card, and compass interface board were stacked on the PC104 expansion bus. The steering servo, speed control unit, motor batteries, and one motor were left as provided by the vehicle manufacturer, but the optical encoder was mounted in place of one of the motors. The optical encoder and digital compass were connected to the interface board via ribbon cables, carefully designed in such a way that prevents hardware damage if the cables are connected improperly. A Ni-MH Computer Battery is connected to the mainboard through the DC/DC power converter that provides steady 5V for the mainboard logic.

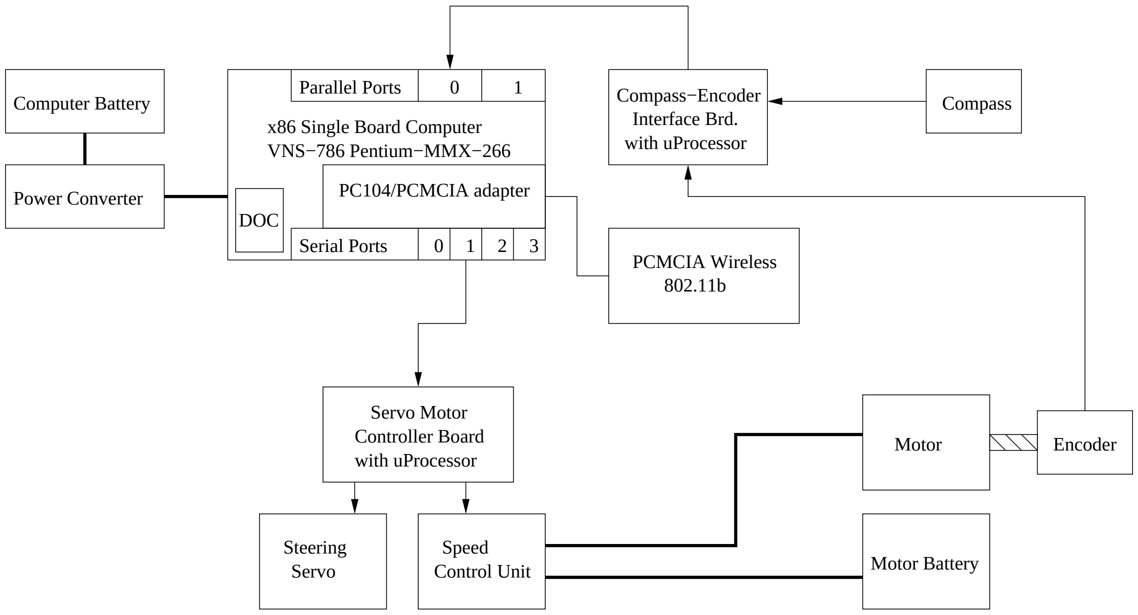


Figure 15: Hardware component integration diagram

A custom interface board was designed to interface the mainboard with the some of the vehicle's sensors via the enhanced parallel port. These sensors include the optical encoder and the digital compass. The board performs its functions via 16F877A microcontroller. The microcontroller tracks the optical encoder via an interrupt line, and tracks the digital compass via a PWM signal sent from the CMPS03. The microcontroller then relays the sensor readings to the interface software that runs as a thread. The board is mechanically connected to the PC104 expansion bus.

4.3.1 Docking Cart and Power Supply Flexibility

The docking cart developed for the lab allows indoor testing by providing a stand, an intelligent Ni-MH battery charger, and an ATX power supply for each vehicle, up to a maximum of six vehicles. A deep-cycle 12V battery is mounted on the stand and provides two important features: the ability to run the vehicles on external 12V rather than their on-board Ni-MH batteries for increased autonomy, and the ability to operate the Ni-MH battery chargers outside the lab if necessary. The vehicles, equipped with four male banana-jacks, facing downwards, on the bottom of their chassis', can be placed on the stands so the banana jacks fit into female receptacles. The four jacks serve to connect the battery charger to the vehicle, and 12V from the deep cycle battery. The docking cart includes a monitor, keyboard and mouse that may be connected to a PC-104 board for diagnostic purposes.

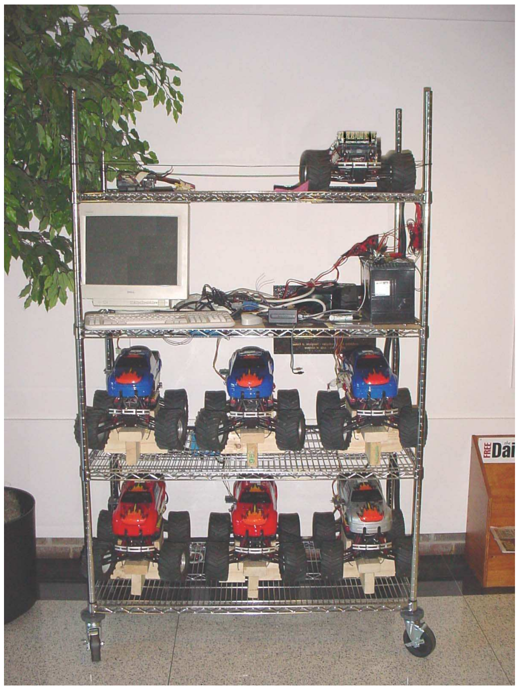


Figure 16: Front view of testing platform

The docking cart, providing two external sources of power and a charger, provides a great degree of flexibility, for a schematic see 17. When running hardware-in-the-loop testing, we can operate continuously without a need for powering down the CPU by running off the ATX power supply. Furthermore, when performing tests outdoors, we can run the vehicles off the deep-cycle battery until we need to deploy them, and can seamelessly remove them from their stands and run them off their Ni-MH batteries until test completion and replace them on their stands for recharging. The deep cycle battery can be charged daily as needed, or can be assisted by a 110V charger when power is available.

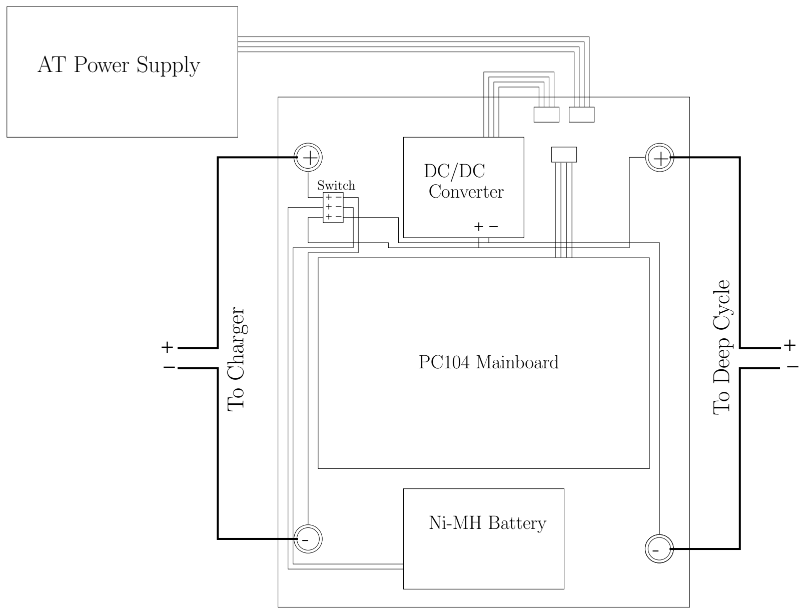


Figure 17: Chassis power system layout

4.4 Ground Station Sofware

A graphical interface was developed for the ground station to allow a user to visualize the positions of the vehicles in a region of interest, or the playing field. The interface can perform many other tasks including sending a variety of commands to the vehicles, sending initial conditions to the vehicles, as well as performing logging operations and interfacing with external devices such as the joystick and a GPS. The interface is a multithreaded application developed entirely in Perl with Tk graphical interface modules. Perl v5.8 was required for its multithreading capabilities, not available in previous versions. External executables developed in C interface with the joystick and GPS and relay the data to the interface through FIFO's, a standard way to communicate between unrelated processes.

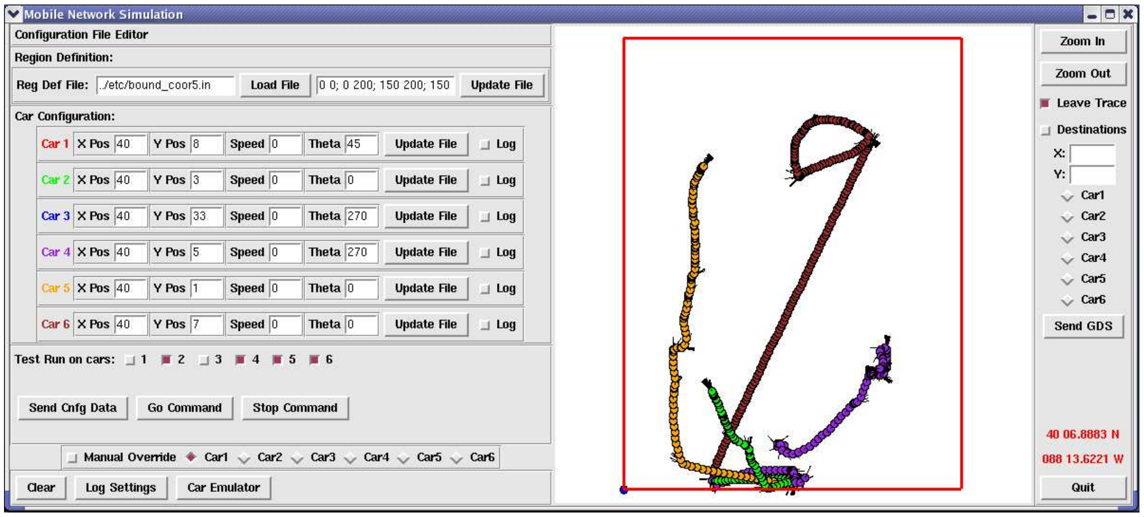
The graphical interface has the following specification:

(i) Receive all data from all vehicles

- Performed by a Listen thread implemented in Perl.

(ii) Start and Stop vehicles (possible feature)

- ssh into vehicles and start program


Mobile Network Simulation

Configuration File Editor

Region Definition:

Reg Def File: ../etc/bound_coor5.in | 0 0; 0 200; 150 200; 150

Car Configuration:

Car 1	X Pos 40	Y Pos 8	Speed 0	Theta 45	Update File	☐ Log
Car 2	X Pos 40	Y Pos 3	Speed 0	Theta 0	Update File	☐ Log
Car 3	X Pos 40	Y Pos 33	Speed 0	Theta 270	Update File	☐ Log
Car 4	X Pos 40	Y Pos 5	Speed 0	Theta 270	Update File	☐ Log
Car 5	X Pos 40	Y Pos 1	Speed 0	Theta 0	Update File	☐ Log
Car 6	X Pos 40	Y Pos 7	Speed 0	Theta 0	Update File	☐ Log

Test Run on cars: ☐ 1 ■ 2 ☐ 3 ■ 4 ■ 5 ■ 6

☐ Manual Override [✓ Car1] Car2 Car3 Car4 Car5 Car6

■ Leave Trace

☐ Destinations

X: Y:

40 06.8883 N

088 13.6221 W

Figure 18: Snapshot of station software output

- Send instructions to vehicles

(iii) Data Logging

- Writes data to a file with appropriate timestamp every time data is received.

(iv) Visualize environment map with cars superimposed

- Displays boundary contour
- Scales everything to fit on 500x500 pixel display
- Maintains 1:1 aspect ratio
- Updates every time new data is received
- Displays color-coded vehicles with vectors indicating direction

The program is divided into the following threads:

- create_window
- socket_udp_listen
- gps_listen
- speak_thread

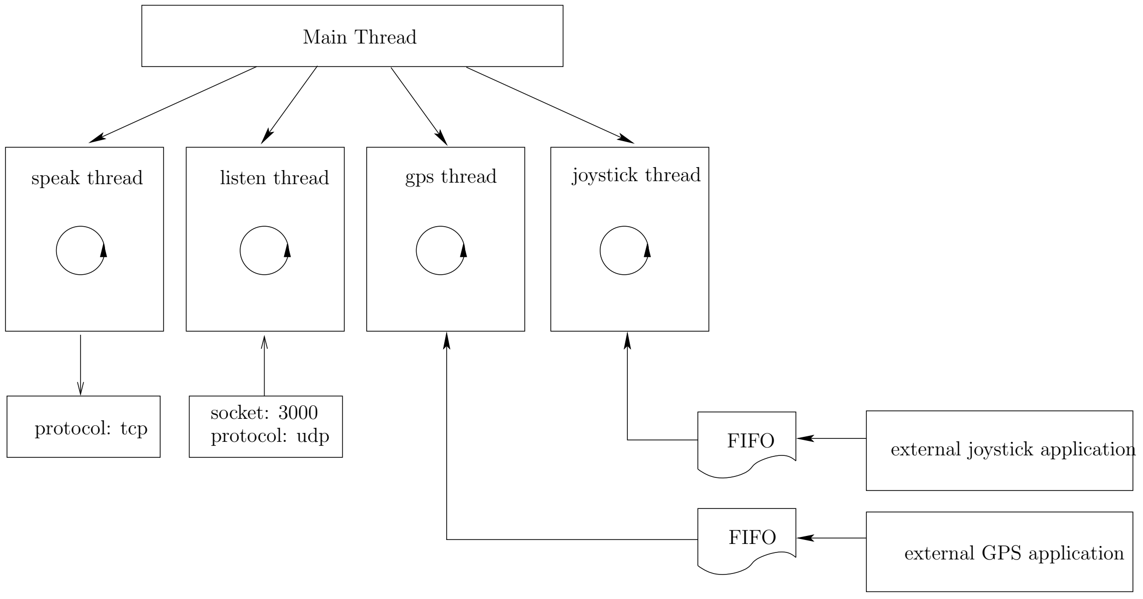


Figure 19: Station software diagram

4.4.1 create_window

create_window is responsible for creating the main graphical display window with all its components, and dealing with all the Tk graphical interface issues such as refreshing the vehicles on the canvas, scaling the canvas and refreshing any data on the interface that is updated in real time. The components used in the graphical display are buttons, labels, text-entry fields, radio buttons, check buttons, and a canvas. The objects in the canvas are removed and regenerated automatically every 100ms by a timer that calls a refresh function. The refresh function removes all objects and redraws them. A "leave trace" option is available if the user would not like to remove the objects every 100ms, but rather redraw the objects on the screen, leaving a trace on the canvas of the vehicle's past positions. To reduce the memory usage of the interface program, a new object is not created if the position of a vehicle has not changed. Nonetheless, for extended tests, leaving traces does cause significant memory usage, causing the system to lag.

4.4.2 socket_udp_listen

socket_udp_listen serves the purpose of listening for UDP broadcasts of vehicle positions, on the base station's established UDP communication port: port 3000. It populates global variables that the refresh routine then uses to draw the vehicles on the canvas.

4.5 gps_listen

gps_listen is a thread that communicates with the GPS. A self-standing executable was written to read the serial port, and communicates with the interface application via a FIFO, a common way of communicating between unrelated processes. gps_listen creates a FIFO, if not already present, and reads data from the socket and populates global variables to store the GPS position. The graphical interface refresh routine then uses this data to broadcast the ground station's position to the vehicles.

4.5.1 speak_thread

speak_thread simply broadcasts the ground station's position in (x,y) coordinates (in the form of an EDT string) and the ground station's GPS position (in the form of an RMC string) so that the vehicles can determine their frame of reference; the ground station determines the location of the playing field.

4.6 Vehicle Software

The operating system we have used on the vehicle computers are an embedded distribution of Linux, referred as LEAF. It has been designed for embedded firewalls, which makes it easy to modify for the scope of our work. The code running on the vehicles is multi-threaded and the threads are divided into two categories real time threads, running with high priority, and non-real time threads running with low priority. The real time threads are compass thread, GPS thread, estimator thread and control thread. The non-real time threads are trajectory thread, command_listen thread, listen thread, speak thread, and main thread. For a visual layout see Figure 20.

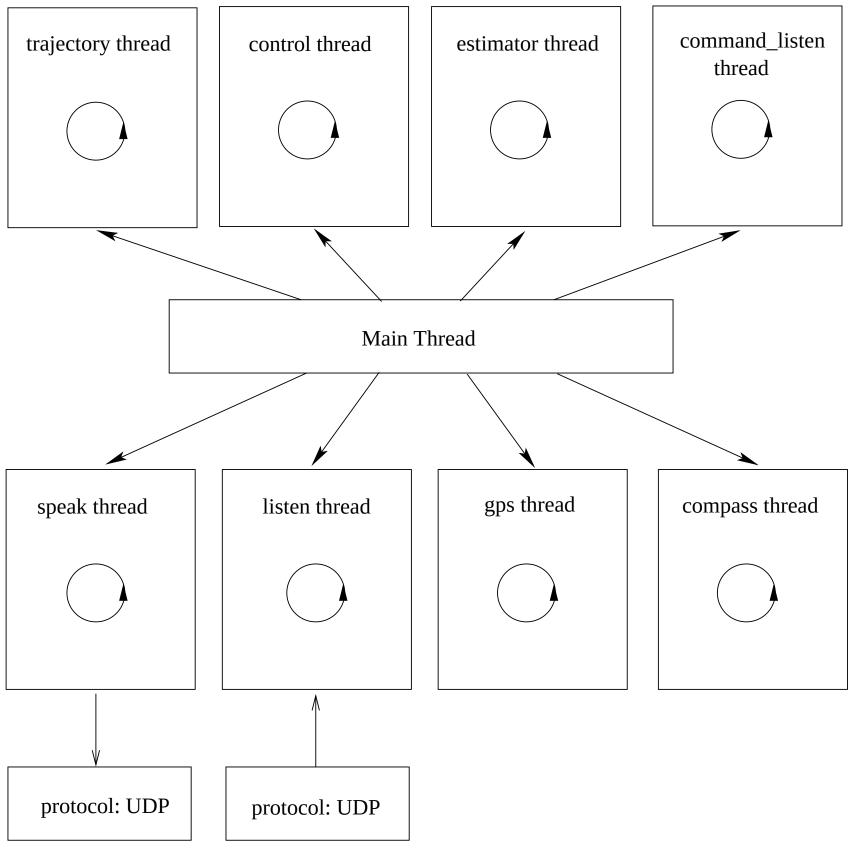


Figure 20: Software Diagram

4.6.1 Compass thread

The Compass thread is responsible for communicating with compass-encoder board interface board over the parallel port. The initialization sequence is followed by a loop that reads requests and reads compass data from the micro-controller, pauses for the amount of time required to make the loop execute at 10Hz. The data is filtered using a simple median filter to remove stray readings that occur. The compass thread posts a semaphore to inform the estimator thread that new data is present so he may compute a new position estimate. The flowchart of this thread can be seen in Figure 21.

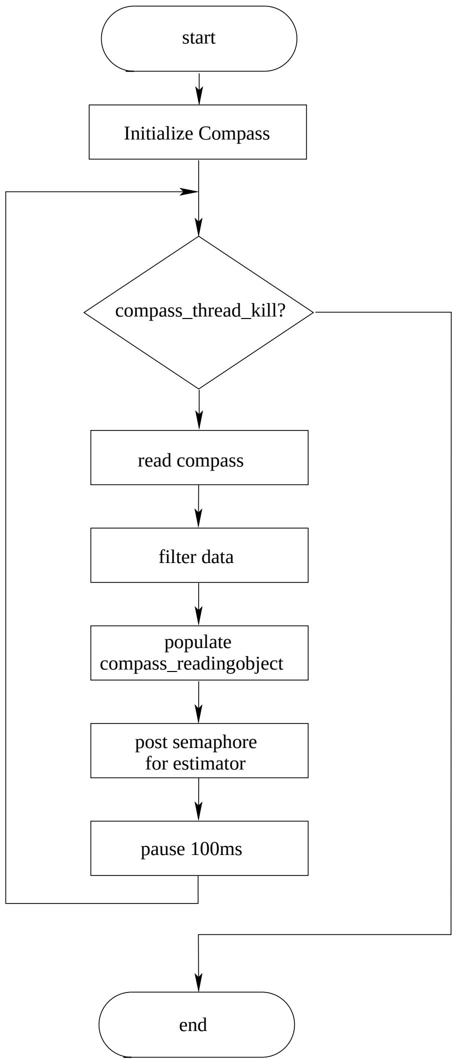


Figure 21: Flow chart for Compass Thread

The communication to other threads are done through the data structure, designated by CDT, declared as,

```
typedef struct{
  long int D_count;
  float angle;
  float *virtual_angle;
  char valid;
  pthread_mutex_t mutex;
}CDT;
```

where, `D_count` contains encoder ticks between samples, and `angle` contains compass heading. `valid` and `mutex` are common for all threads and they carry informations of validity and availability.

4.6.2 GPS thread

GPS Thread opens a serial port connection to the GPS, located on ttyS0 and sets up the port settings consistent with those of the NMEA protocol used by all GPS units. The GPS unit transmits RMC strings, which contain position information, as well as time, date, and other useful data. Following an RMC string, the GPS will always transmit an RME string, which contains estimated error values on the position readings obtained in the previous RMC. Further information on NMEA strings see, [1]. This data will be used by the Kalman filter in better estimating the position of the vehicle. The GPS unit transmits RMC strings once every second. When a RMC string is received, the thread parses out all data. If an RME is received, the valid bit from the previous RMC is checked, and if valid, then the thread will parse the relevant data from the string. In the case of valid data, whether RME or RMC, the thread will then populate the data structures RMC, and RME declared as,

```
typedef struct {
  TIME time;
  POSITION latitude;
  POSITION longitude;
  char valid;
  float groundspeed;
  float course;
  DATE date;
  float magneticvariation;
  char mvdirection;
  char modeindicator;
  pthread_mutex_t mutex;
}RMC;
```

and,

```
typedef struct {
  float horizontal_position_error;
  float vertical_position_error;
  float position_error;
  char   valid;
  pthread_mutex_t mutex;
}RME;
```

The flowchart of this thread is given in Figure 22.

4.6.3 Estimator thread

Estimator thread is responsible for running a Kalman filter on the data grabbed by compass-encoder thread, GPS thread and the inputs. The result is best estimate of vehicle state. The best estimate is represented by the data structure EDT, declared as,

```
typedef struct EDT_struct {
  COORDINATE coordinate;
  float angle;
  float speed;
  float psi;
  char valid;
  pthread_mutex_t mutex;
```

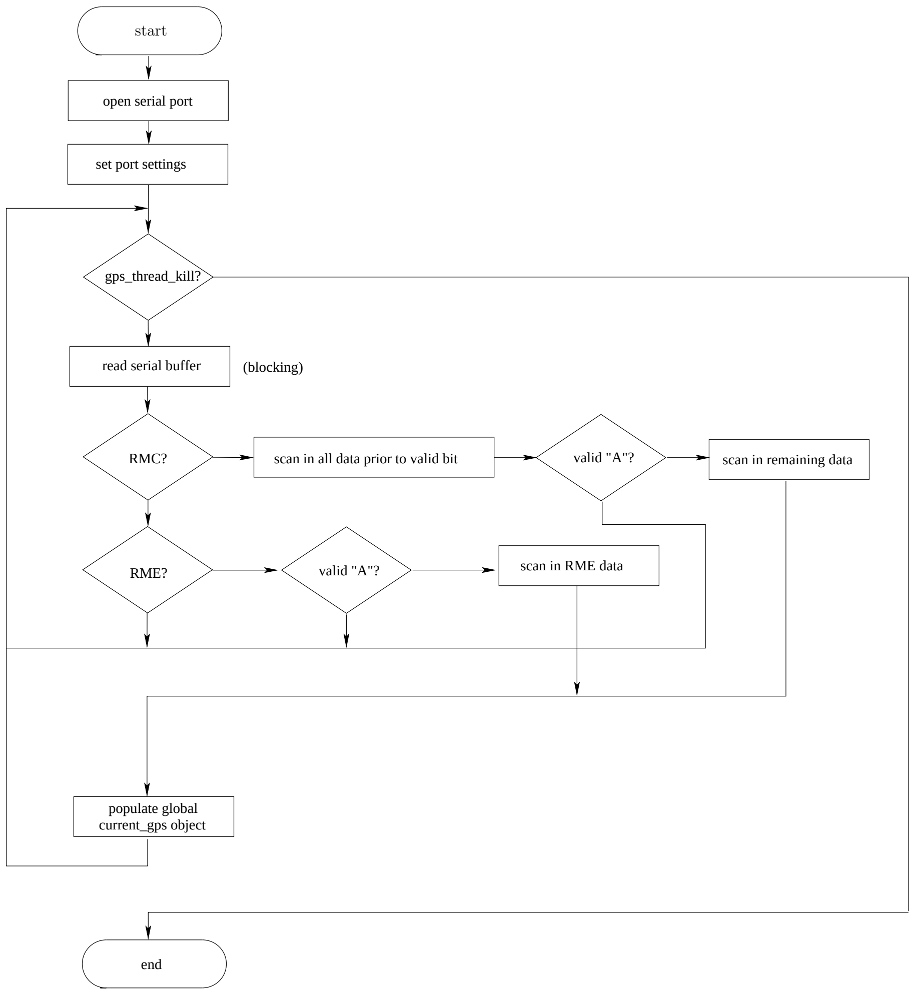


Figure 22: GPS Thread

```
}EDT;
```

4.6.4 Control thread

Comparing the best estimate of current state with given trajectory, control thread computes appropriate actuator signal and sends it to servo control board via RS-232 serial port. It also accepts signals from command_listen thread such as “pause”, which stops vehicle from moving.

4.6.5 Trajectory thread

Trajectory thread computes a feasible trajectory given initial state, goal state and vehicle constraints. Currently we assume that there are no obstacles in the region the vehicles are moving.

4.6.6 Speak thread

The Speak Thread broadcasts the position of the vehicle on the broadcast port (port 300) once every second. Timing is not extremely critical, so this thread is not timed with any other threads through semaphores or any other such synchronization method. If terminated by means of setting the client_thread_kill flag, the speak thread sets client_thread_kill equal to 5, in order to acknowledge.

4.6.7 Listen thread

The Listen thread opens a socket on the listening port (port 3000) and listens to broadcasts made by the speak thread of other vehicles. It parses out the relevant data and populate the data structures that keep track of the other vehicles' positions. A flowchart of this thread is given in Figure 23

4.6.8 Command_listen thread

Command Listen creates a TCP socket on the command port (port 4000) and listens for commands from the ground station. Commands may be one of the following:

Stop Commands	(STC)	Stops the vehicle unconditionally
Go Commands	(GOC)	Initiates a test sequence
Go Destination command	(GDS)	Tells a vehicle to go to a specific (x,y) position in the playing field
Manual Override command	(MOR)	Sends the vehicle steering and throttle Commands

Once a command is received, the loop returns to waiting for the next command. If the buffer received did not contain any of the commands listed above, the connection was closed. The flowchart of this thread can be seen in Figure 24

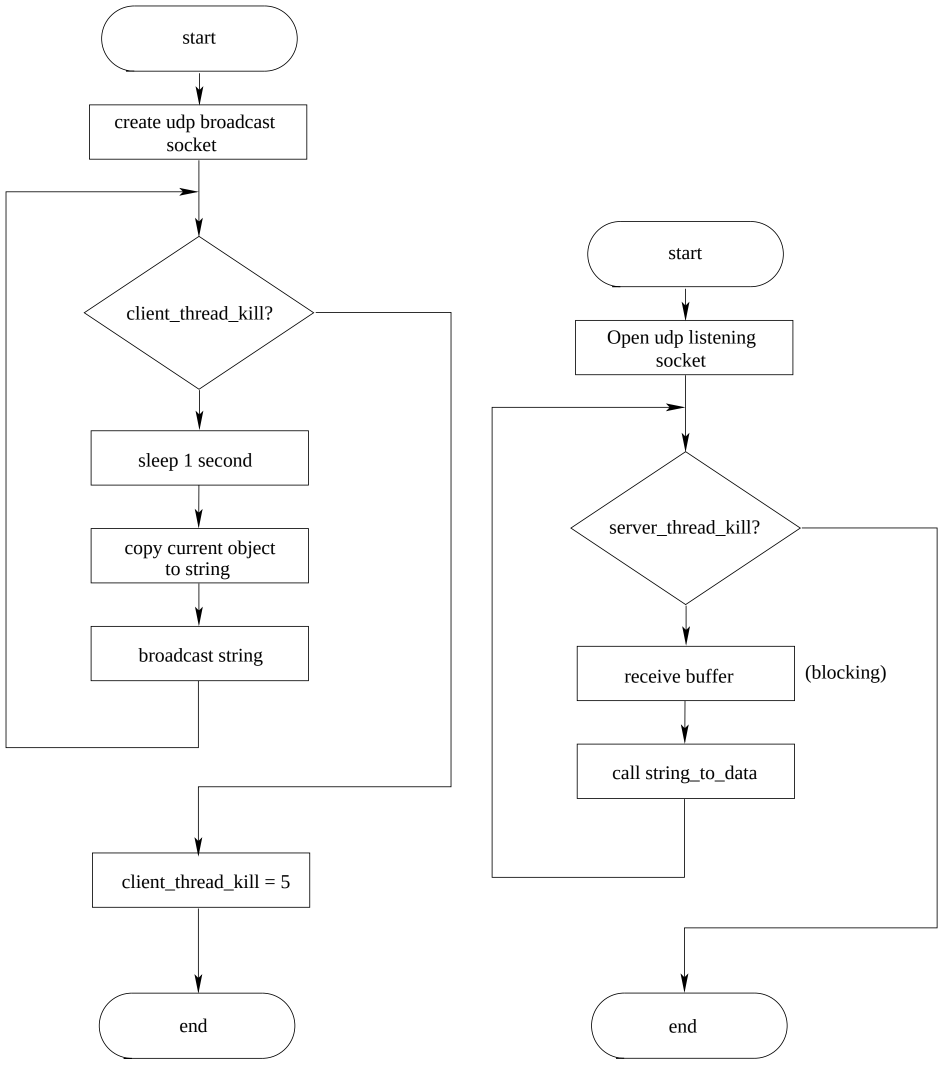


Figure 23: Flow charts for Speak Thread (left) and for Listen Thread (right)

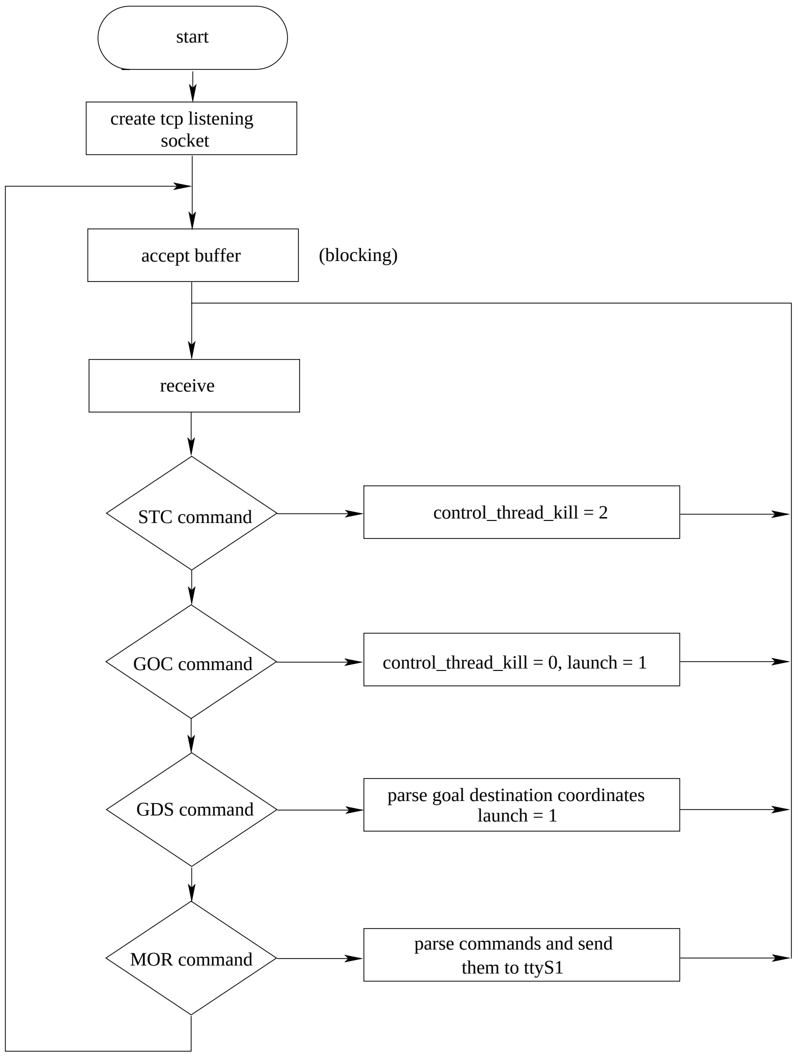


Figure 24: Command Listen Thread

5 Conclusion and Future Work

This thesis research has two objectives. First, we will complete our study of control and coordination algorithms for groups of vehicles. The focus of this objective is on coverage control of autonomous vehicle networks performing distributed sensing tasks. Second, we will develop a testbed composed of a group of autonomous, networked robotic vehicles. By utilizing this testbed, we will test and illustrate the distributed control methodologies for multiple vehicle coordination, formation and stabilization that we have introduced in Section 3. The research steps taken to date and future directions on the two stages are summarized below.

Algorithm Development: We have started our work by studying a locational optimization problem. We have studied different aspects of Voronoi partitioning, along with centroidal Voronoi partitions. By a novel approach, we have shown that centroidal Voronoi partitions correspond to local minima of the locational optimization problem. We also have studied classical Lloyd Descent algorithm and introduced a novel family of discrete and continuous time Lloyd algorithms that guarantee convergence to a local minima. Then, based on [68], we have proposed an asynchronous and distributed procedure to calculate and update the Voronoi cells and control inputs that guarantee convergence under some technical conditions. Additionally, we have shown that our coverage control algorithm can solve some of formation control problems via the design of appropriate distribution density functions. The next step will be to extend the set of Lloyd Descent algorithms to 4 wheeled mobile robots, by means of an appropriate combination of feedforward and feedback control design. We also plan to address the issue of collision avoidance of vehicles with each other, as well as with the objects in the environment.

Testbed Development: First we have selected the minimum set of required hardware components that will result in a functional testbed. We have modified the structure of the vehicles and integrated all the electronic and the mechanical components. We have defined the minimum set of threads and data structures that will allow the agents to perform coordinated tasks. Further we have completed the development of the "skeleton code" and tested the code with success. We still have to design, and implement a path planner, a trajectory tracking controller, and an estimator. The contribution of this stage will be integrating of all the components to a functional platform, and illustrating the effectiveness of the proposed algorithms.

References

[1] *GARMIN GPS 16 GPS receiver/antenna quick start guide.*

[2] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999.

[3] R. C. Arkin. *Behavior-Based Robotics.* Cambridge University Press, New York, NY, 1998.

[4] J. P. Farrell B. Nichols, D. Buttlar. *Pthreads Programming.* O'Reilly & Associates, Inc., 1996.

[5] R. Bachmayer and N. E. Leonard. Vehicle networks for gradient descent in a sampled environment. In *IEEE Conf. on Decision and Control*, pages 112–117, Las Vegas, NV, December 2002.

[6] T. Balch and R. Arkin. Behavior-based formation control for multirobot systems. *IEEE Transactions on Robotics and Automation*, 14(6):926–39, 1998.

[7] T. Balch and L. E. Parker, editors. *Robot Teams: From Diversity to Polymorphism.* A K Peters Ltd., Natick, MA, 2002.

[8] M Barr. "Introduction to pulse width modulation". In *Embedded Systems Programming*, volume 14. CMP Media LLC, 9 2001.

[9] J. Barraquand and J-C. Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10(6):628–649, 1991.

[10] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Athena Scientific, Belmont, MA, 1997.

[11] J. T. Betts. Survey of numerical methods for trajectory optimization. *AIAA Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[12] R. A. Brooks. A robust layered control-system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[13] M. Cao and C. Hadjicostis. Distributed algorithms for Voronoi diagrams and application in ad-hoc networks. Preprint, October 2002.

[14] J. Catsoulis. *Designing Embedded Hardware.* O'Reilly & Associates, Inc., 2002.

[15] C. Cattani and A. Paoluzzi. Boundary integration over linear polyhedra. *Computer-Aided Design*, 22(2):130–5, 1990.

[16] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks: variations on a theme. Lisbon, Portugal, July 2002. Electronic proceedings.

[17] L. Cremean, W. B. Dunbar, D. van Gogh, J. Hickey, E. Klavins, J. Meltzer, and R. M. Murray. The Caltech multi-vehicle wireless testbed. In *IEEE Conf. on Decision and Control*, pages 86–88, Las Vegas, NV, December 2002.

[18] T. B. Curtin, J. G. Bellingham, J. Catipovic, and D. Webb. Autonomous oceanographic sampling networks. *Oceanography*, 6(3):86–94, 1993.

[19] R. D'Andrea and M. Babish. The RoboFlag testbed. In *IEEE American Control Conference*, pages 656–660, Denver, CO, June 2003.

[20] F. E. Daum. *"New exact nonlinear filters," Bayesian Analysis of Time Series and Dynamic Models.* Marcel Dekker, Inc., NY, 1988.

[21] J. P. Desai, J. P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, 2001.

[22] Z. Drezner, editor. *Facility Location: A Survey of Applications and Methods.* Springer Series in Operations Research. Springer Verlag, New York, NY, 1995.

[23] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[24] P. J. Enright and B. A. Conway. Discrete approximations to optimal trajectories using direct transcription and nonlinear programming. *AIAA Journal of Guidance, Control, and Dynamics*, 15(4):994–1002, 1992.

[25] N. Faiz, S. K. Agrawal, and R. M. Murray. Trajectory planning of differentially flat systems with dynamics and inequalities. *AIAA Journal of Guidance, Control, and Dynamics*, 24(2):219–227, 2001.

[26] M. S. Fontan and M. J. Mataric. Territorial multi-robot task division. *IEEE Transactions on Robotics and Automation*, 14(5):815–822, 1998.

[27] E. Frazzoli, M. A. Dahleh, and E. Feron. A hybrid control architecture for aggressive maneuvering of autonomous helicopters. In *IEEE Conf. on Decision and Control*, pages 2471–6, Phoenix, AZ, December 1999.

[28] E. Frazzoli, M. A. Daleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[29] Bill O. Gallmeister. *POSIX.4:Programming for the Real World*. O'Reilly & Associates, Inc., 1995.

[30] J. Gao, L. J. Guibas, J. Hershberger, Li Zhang, and An Zhu. Geometric spanner for routing in mobile networks. pages 45–55, Long Beach, CA, October 2001.

[31] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, April 1993.

[32] R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998. Commemorative Issue 1948-1998.

[33] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *AIAA Journal of Guidance, Control, and Dynamics*, 10(4):338–342, 1987.

[34] C. Hollabaugh. *Embedded Linux Hardware, Software and Interfacing.* Addison-Wesley, 2002.

[35] A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed scalable solution to the area coverage problem.

In *International Conference on Distributed Autonomous Robotic Systems (DARS02)*, pages 299–308, Fukuoka, Japan, June 2002.

[36] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[37] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering on Automatic Control*, 82(D):34–45, 1960.

[38] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional space. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[39] Kayton and Fried. *Avionics Navigation Systems*. John Wiley & Sons, 2 edition, 1997.

[40] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, Englewood Cliffs, NJ, second edition, 1995.

[41] E. Klavins. Communication complexity of multi-robot systems. In J.-D. Boissonnat, J. W. Burdick, K. Goldberg, and S. Hutchinson, editors, *Algorithmic Foundations of Robotics V*, volume 7 of *STAR, Springer Tracts in Advanced Robotics*, Berlin Heidelberg, 2003. Springer Verlag.

[42] R. Klein. *Concrete and abstract Voronoi diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer Verlag, New York, NY, 1989.

[43] E. Krotkov and J. Blitch. The Defense Advanced Research Projects Agency (DARPA) tactical mobile robotics program. *International Journal of Robotics Research*, 18(7):769–76, 1999.

[44] H. J. Kushner. Dynamical equations for optimum non-linear filtering. *Journal of Differential Equations*, 3:179–190, 1967.

[45] J.-C. Latombe. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research*, 18(11):1119–1128, 1999.

[46] X.-Y. Li and P.-J. Wan. Constructing minimum energy mobile wireless networks. *ACM Journal of Mobile Computing and Communication Survey*, 5(4):283–286, 2001.

[47] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. Presented as Bell Laboratory Technical Memorandum at a 1957 Institute for Mathematical Statistics meeting.

[48] M. Micheli and M. I. Jordan. Random sampling of a continuous-time stochastic dynamical system. In *Proceedings of 15th International Symposium on the Mathematical Theory of Networks and Systems*, South Bend, IN, Aug 2002. University of Notre Dame.

[49] R. M. Murray, Z. X. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, Boca Raton, FL, 1994.

[50] K Ogata. *Modern Control Engineering.* Prentice-Hall, inc., Englewood Cliffs, NJ, 2nd edition, 1990.

[51] A. Okabe, B. Boots, and K. Sugihara. Nearest neighbourhood operations with generalized Voronoi diagrams: a review. *International Journal of Geographical Information Systems*, 8(1):43–71, 1994.

[52] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* Wiley Series in Probability and Statistics. John Wiley & Sons, New York, NY, second edition, 2000.

[53] A. Okabe and A. Suzuki. Locational optimization problems solved through Voronoi diagrams. *European Journal of Operational Research*, 98(3):445–56, 1997.

[54] R. Olfati-Saber and R. M. Murray. Agreement problems in networks with directed graphs and switching topology. In *IEEE Conf. on Decision and Control*, 2003. Submitted.

[55] A. Richards, T. Schouwenaars, J. How, and E. Feron. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *AIAA Journal of Guidance, Control, and Dynamics*, 25(4):755–765, 2002.

[56] R. Rogers. *Applied Mathematics in Integrated Navigation Systems.* AIAA Education Series. AIAA, 2000.

[57] P. E. Rybski, N. P. Papanikolopoulos, S. A. Stoeter, D. G. Krantz, K. B. Yesin, M. Gini, R. Voyles, D. F. Hougen, B. Nelson, and M. D. Erickson. Enlisting rangers and scouts for reconnaissance and surveillance. *IEEE Robotics & Automation Magazine*, 7(4):14–24, 2000.

[58] A. C. Schultz and L. E. Parker, editors. *Multi-Robot Systems: From Swarms to Intelligent Automata*. Kluwer Academic Publishers, 2002. Proceedings from the 2002 NRL Workshop on Multi-Robot Systems.

[59] H. W. Sorenson. *Kalman Filtering: Theory and Application.* IEEE Press, NY, 1985.

[60] H. W. Sorenson and A. R.Stubberud. Non-linear filtering by approximation of the a posteriori density. *Iinternational Journal of Control*, 8(1):33–51, 1968.

[61] R. Stengel. *Optimal Control and Estimation.* Dover, 1994.

[62] W. R. Stevens. *Interprocess Communication.* Number 2 in UNIX Network Programming. Prentice Hall, 2nd edition, 1999.

[63] A. Stubbs, V. Vladimerou, A. Vaughn, and G. E. Dullerud. Development of a vehicle network control testbed. In *IEEE American Control Conference*, pages 3028– 3033, Anchorage, AK, May 2002.

[64] K. Sugihara and I. Suzuki. Distributed algorithms for formation of geometric patterns with many mobile robots. *Journal of Robotic Systems*, 13(3):127–39, 1996.

[65] A. Suzuki and Z. Drezner. The p-center location problem in an area. *Location Science*, 4(1/2):69–82, 1996.

[66] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. 28(4):1347–1363, 1999.

[67] C. Tomlin, G. J. Pappas, and S. S. Sastry. Conflict resolution for air traffic management: a study in multiagent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–21, 1998.

[68] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–12, 1986.

[69] S. ulier, J. Uhlmann, and H. F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477 – 482, March 2000.

[70] O. von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1-4):357–73, 1992.

[71] C. R. Weisbin, J. Blitch, D. Lavery, E. Krotkov, C. Shoemaker, L. Matthies, and G. Rodriguez. Miniature robots for space and military missions. *IEEE Robotics & Automation Magazine*, 6(3):9–18, 1999.

[72] K. Yaghmour. *Building Embedded Linux Systems.* O'Reilly & Associates, Inc., 2003.

[73] H. Yamaguchi and T. Arai. Distributed and autonomous control method for generating shape of multiple mobile robot group. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 800–807, Munich, Germany, September 1994.