Polymorphic Control and Trajectory Optimization of an Autonomous Ground Vehicle over Wireless Mobile Networks

Corey Ippolito[1]
NASA Ames Research Center, Moffett Field, CA 94035

Khalid Al-Ali[2]
Carnegie Mellon University at Silicon Valley, Moffett Field, CA, 94035

John Dolan[3]
Carnegie Mellon University, Pittsburg, PA

Adapting existing autonomous vehicle platforms to operate in new environments with additional capabilities through redesign and hardware augmentation can be expensive and risky; an alternative approach is to cooperatively utilize remote resources available from remote entities concurrently operating in the same environment. Polymorphic Control Systems research investigates highly dynamic control structures that can automatically reconfigure across vehicles to share remote sensors, actuators, and other resources available throughout the system of vehicles. In this paper, we investigate extending this framework to a smart space environment, allowing a ground vehicle without sufficient instrumentation or available onboard resources to navigate a complex indoor environment by coordinating with concurrently operating building control system agents. We develop a complex simulation of the environment, develop polymorphic control laws that restructure the control system over multiple entities during operation, and present a trajectory generation approach concurrently addresses topological optimization and dynamic control optimization of the collaborative control structure. Our results demonstrate viability of solving the two-part polymorphic control optimization problem utilizing pseudo-optimizing solvers that trade optimality for feasibility and real-time performance, and show that control polymorphism provides robust resource sharing between agents in a smart building infrastructure.

I. Introduction

Adapting autonomous vehicles for new environments can be a risky and expensive process, especially when the vehicle has been specifically optimize for a different environment. Recently, conceptual long duration autonomous experiments were proposed for NASA's Mobile Autonomous eXplorer (MAX) 5A unmanned ground vehicle (UGV) requiring precise payload-directed navigation of a rugged outdoor environment along with robust navigation inside an office building. The experiment requires a suite of scientific instruments consuming the vehicle's full payload capacity. Although the MAX 5A vehicles were specifically designed to support experiments requiring high performance outdoor navigation in rugged environments, these vehicles do not have the instrumentation required to perform robust indoor navigation. For instance, the vehicle's navigation system relies on data from a Global Positioning System (GPS) receiver which will not function in an office building environment. One proposed solution was to augment the UGV with additional sensors and hardware specifically for the indoor navigation task, but this would require a substantial design and development effort to increase the platform's payload and power capacity, impairs the vehicle's outdoor performance, and adds significant project and schedule risk. The projected cost and resource estimates are beyond the scope and budget of the project.

An attractive alternative approach utilizes distributed sensors that are external to the vehicle for localization, allowing the unmodified vehicle to meet requirements for robust indoor navigation. Mobile navigation using environment sensors has been investigated in the literature in the context of solving specific parts of the technical

[1] Research Scientist, Intelligent Systems Division, Mail Stop 269-3, AIAA Member.
[2] Senior Fellow, Carnegie Mellon University at Silicon Valley.
[3] Senior Systems Scientist, Robotics Institute, Carnegie Mellon University.

problems involved, such as tracking, pose-estimation, or path tracking with obstacle avoidance. There are many benefits to this approach. First, the cost and complexity of a vehicle system is kept minimal. Second, multiple remote sensors can give more complete and accurate information than utilizing onboard sensors. Third, offloading the sensing and processing functions greatly reduces the load on the UGV's onboard systems. Finally, the infrastructure for sensor communication supports further collaborations with other systems. Although several approaches for low level control and sensing have been shown to be technically feasible and robust, experiments have generally involved specific controlled conditions, such as fixed pre-programmed information structural topologies with all conditions known in advance and preprogrammed into the system, or fixed reprogrammed trajectories. In-situ resources such as sensors in an intelligent space will likely be utilized by other systems with varying access priority, and resource sharing and competition remains an open problem. Uncertainty in the environment may require agents to dynamically adapt and form unexpected control loops, requiring for instance the UGV to search the smart environment for suitable sensors, make plans based on sensor position and availability, and form temporary control loops around these sensors while navigating through the intelligent space. Resource constraints may preclude formation of particular control loops, for instance if vision processing is too expensive to perform on the onboard computer, or if bandwidth limitations preclude sending raw camera video in real time. Complex constraints may also exist, such as sensor occlusion by environment obstacles, which must be accounted for in the trajectory algorithm. Certain sensors may not be available, may only be temporarily available, or may be under control of non-collaborating system. For instance, the pan and tilt angle of security camera may be under strict control of the security system, so planning must be performed to account for a time-varying sensor swath.

Control system polymorphism may provide a method for solving these issues, providing outer-loop guidance and navigation solutions, and planning dynamic trajectory and paths to meet performance objectives of multiple agents in the environment under multiple constraints and objectives. The Polymorphic Control Systems (PCS) project envisions dynamic control systems that can automatically reconfigure and share data across distributed intelligent autonomous agents to optimize performance based on the current situation. This flexibility allows distributed intelligent agents to autonomously interact and collaborate through these collaborative control structures, performing tasks with greater efficiency, expanding the capabilities of individual agents, and enabling agents to pursue goals that are unachievable independent of cooperation. Collaboration would allow agents to dynamically share sensors, actuators, control laws, and resources. Structural configurations may only be temporary; for instance, a mobile sensing agent may plan a movement trajectory through a specific location for a short period of time when sensing that location would be beneficial to another agent. Restructuring occurs through coordinated changes of the topology of the controller graphs, changing the information structure that represents data flows through the distributed system. Restructuring can occur in response to resource constraint changes or other environmental conditions, in response to faults such as loss of sensors or actuators, or when reconfiguration provides more optimal performance. PCS provides the mathematical foundation to describe these constantly evolving distributed control structures, enabling systems to optimize shared performance objectives through wireless data communication, and enabling systems to establish dynamic control configurations that provide more global optimal performance. The general PCS problem formulation results in a two-part optimization problem; PCS solutions must optimize the structural topology of the controller graphs for each system, and the solution must optimize the control laws that operate over a given structural topology. Further, the global systems are often non-linear in nature, and must be optimized over non-convex performance objective functions and poorly characterized constraints.

The goal of this investigation is to develop control laws that implement a robust polymorphic solution to the problem of indoor navigation in an realistic scenario, develop a moderately high fidelity simulation infrastructure to test this goal, demonstrate the feasibility of utilizing polymorphism to solve this problem, and develop a solution to the two-part optimization problem that meets the requirements of this problem. In this paper, we explore polymorphic control of an autonomous unmanned ground vehicle navigating in an indoor intelligent space environment. In this scenario, the UGV is not instrumented to independently localize and navigate indoors, so it must search the environment for available sensors (security cameras) to utilize, and coordinate with the security agents who are controlling the cameras who have disparate objectives. The UGV must localize, navigate, and avoid obstacles in the indoor environment utilizing and controlling distributed cameras that have pan and tilt functionality. The UGV must collaborate with various security system agents with competing objectives to monitor the area with the cameras. The mobile robotic vehicle must choose its path to ensure it lies in the sensor swath of the cameras while minimizing shared global objectives. The onboard computation platform cannot process the raw vision data stream onboard because of constraints in both communication and limited onboard processing capability, so vision processing algorithms must be dynamically offloaded to computers embedded in the environment. The UGV must plan the appropriate path to take, schedule uploading of vision processing algorithms to the environment, and schedule control structure changes with the security camera systems that it relies on for localization. We address the

fundamental difficulty of the two-part topological and control optimization problem through a novel trajectory based search approach, inspired by the rapidly-exploring random tree approach introduced in [1] and expanded in [2]. This approach maps trajectories to structural configurations and imparts constraints on the trajectories to ensure feasible controller configurations. The trajectories are then optimized through the random search approach under these constraints. We present simulation results of this experiment under varying conditions and optimization criteria, evaluate these results, and describe future extensions to this research.

II. Related Work

A. Pervasive Computing and Polymorphic Control

Concepts in control polymorphism build on research in mobility and pervasive computing. Weiser's seminal paper [3] lays out a vision for computing that is ubiquitous and calm, where computational agents merge with their environments and become indistinguishable and invisible. The concepts and challenges for distributed, mobile, and pervasive computing were further developed from this vision by Satyanarayanan in [4]. Intelligent spaces – often referred to as active environments or smart spaces [4], are environments imbued with embedded sensing and computation, and accessible to agents operating in the environment. Agents in this context are defined as any goal-oriented entity in the environment capable of communicating, sensing, thinking, and acting. Agents must be able to take part in interactions in the environment with resources and other agents, have the ability to receive input from onboard or remote sensors, make decisions based on their own self-interests, and take actions that leads the agent closer to their goals. Agents may include humans, mobile devices, software agents, and autonomous machines. Intelligent spaces allow agents to interact and communicate seamlessly throughout the environment, enabling humans and machines to operate more efficiently and effectively [5]. As the number of agents increase, resource sharing and scarcity can be a difficult issue to address. For instance, Gajos [6] presents design considerations for intelligent spaces where agents are expected to collaborate and resource competition is expected to occur.

Mobile devices face fundamental limits in processing power, energy, and information communication, and adaptation strategies are used to address this limitation. Research into adaptation for mobility often seeks strategies to provide reduce functionality or degraded performance to allow continued operation [4][7], such as bandwidth adaptation in file systems [8], transcoding by proxies [9], adaptive resource management [10]. Polymorphic control strategies extend these concepts to adaptation in control systems when encountering control system constraints in a distributed system of autonomous agents.

Another property of a polymorphic control approach is the ability to offload processing, sensing, and actuation to other agents. This approach builds on the concept of cyber foraging that enhances performance of mobile clients, which are constrained by finite processing power, communication, bandwidth, and energy. Cyber foraging uses opportunistically discovered servers embedded in an enabled environment to offload processing jobs from the mobile device to the environment seamlessly and invisibly. Balan applies this concept to distribute file system access in [11], and presents an adaptive runtime environment that support developing cyber foraging applications in [12]. Kristensen presents a framework for enabling cyber foraging on small mobile devices [13]. Sivavakeesar presents strategies for service discovery in smart environments that enable cyber foraging on devices [14]. Cyber foraging for use in polymorphic control reconfiguration requires the ability to perform adaptation independent of human operators, be supported by a common framework across all agents in the environment, and be easy for the developers to use.

The concept of autonomous cooperation between intelligent agents has become important in many diverse fields and applications, including information technology [15], complex problem solving [16], urban traffic optimization [17], distributed manufacturing scheduling [18], search and rescue robotics [19], and robotics for planetary exploration [20][21] [22]. Autonomous cooperative agents are an effective method for implementing large-scale decentralized control systems [23]. Mobile computational agents have been studied in the context of pervasive computing [24] and intelligent spaces [25]. Mobile robots exploiting active environments were used for precise estimation through coordinated mobilization of onboard sensors [26][27] and providing assistance to the disabled and elderly [28]. Competition for resources in these environments causes issues of resource sharing and scarcity, which must be addressed [6].

B. Indoor Navigation through Smart Environments

Indoor localization and navigation is a challenging open problem for autonomous robotics, and one approach exploits intelligent spaces to accomplish this goal. Several approaches have been investigated that utilize ad-hoc wireless communication to access in-situ sensing resources for the navigation problem [29][30][31][32][33]. Mobile navigation using environment sensors has been investigated in the literature in the context of tracking [34],

pose-estimation [35], and path tracking with obstacle avoidance [36]. However, these approaches rely on fixed pre-programmed information structural topologies. Navigation for a car-like wheeled robot was accomplished using a distributed active-vision network-space system subject to fuzzy variable-structure decentralized control in [37]. The vision tracking scheme was robust to complex visual patterns, non-uniform illumination, and strong reflections while precisely estimating the dynamic pose of the mobile robot. Similarly, tracking and obstacle avoidance in a car-like mobile robot through remote sensors in the environment was investigated using a decentralized control approach based on a mixed H_2/H_∞ control law [38]. Tracking occurred on piecewise-linear segments which showed reduced energy consumption, bounded tracking error, attenuation of output disturbance, and improvement of control performance. A similar approach utilized an RFID system embedded in the environment for efficient localization of a differential drive mobile robot [39]. Lee explored resource constraint aspects of the robotic localization in a distributed sensor network [40]. This approach embedded the intelligence needed to localize and control the rovers in the sensor network devices rather than on the robot, and sensor devices would direct the robot to its destination. The approach utilizing polymorphic approach to solve this problem builds on these related works. In this paper, however, we focus mainly on framework, planning and navigation problems. This experiment utilizes a simple waypoint controller based on cascaded PID control, but is compatible with the related methods utilizing optimal and robust control. Similarly, this experiment utilizes simple methods for vision estimation, and is compatible with the ongoing research in this field as possible future extensions to this project.

C. Polymorphic Control Systems

The Polymorphic Control Systems [41][41] approach applies control theoretic analysis and synthesis techniques to a mathematical construct that describes distributed component-based systems and decentralized control problems. The requirements for this approach can be conceptually divided into three main components: a *physical layer*, a *mathematical topological construct*, and an *analysis/synthesis approach*. The physical layer refers to the actual implementation of the distributed embedded plug-and-play environment that allows for immediate on-the-fly reconfiguration over local and global vehicle systems. The physical layer has been implemented in the Reflection Architecture [42] as a prototype software architecture for PCS experimentation. The topological construct is a hyper-graph description of component-based control systems similar in form to controller graphs. This topological construct is capable of describing a large class of control configurations, and can accommodate modeling latency, bandwidth limitations, line errors, and uncertainty in the data signals. The PCS formulation also provides structural/topological definitions and operations for manipulation of the controller graph. These constructs model component-based intercommunications largely through a signal-routing architecture model. Finally, the analysis/synthesis approach defines the analytical and synthesis engines required to solve PCS problems posed in the context of the topological construct, which in turn generates control algorithms and procedures to implement these results. Subsequently, these solutions are implemented on the physical layer. PCS has been successfully applied to simulation problems involving coordinated vehicle traverse [41][41]. The formulation was used to autonomously land an autonomous aerial vehicle encountering sensor failure [43], where the onboard control system was restructured around sensors from a ground based rover.

For a general class of PCS problems, the synthesis engine must be able to solve two simultaneous optimizations: topological optimization, and trajectory optimization. The topological optimization must optimize the topology of the control system given the resource constraints and the shared set of objective functions. The trajectory optimization can be considered to be constrained by a particular topology: given a particular topology, synthesize the most efficient control inputs that maximize expected performance of the objective functions. The required mathematical preliminaries are detailed in [41][41], see this reference for more detail.

III. Polymorphic Control Approach

A. Problem Statement

The Exploration Aerial Vehicles (EAV) laboratory at NASA Ames Research Center operates a set of MAX 5A UGV platforms. The MAX 5A specifications are shown in Figure 1 (right). These ground vehicles were custom designed specifically for precision outdoor GPS-based navigation. The goal for these vehicles is to support science missions where multiple UGV's autonomously navigate and record data measurements over very long durations (several weeks or months in length) over large areas (such as the NASA Ames Research Center campus) without direct surveillance of a human operator, with similar mission design as [44]. To enable this goal, these autonomous UGV's must navigate autonomously through indoor environments and recharge.

Figure 1. Senseta Corporation Max 5A UGV, Specifications (right) and Simulation Model (left)

The Max 5A UGV's were custom designed for outdoor navigation and operation, with a MEMS-based inertial measurement unit (IMU) for orientation, angular velocity, and linear acceleration measurements, magnetometers for orientation measurements, and differential Global Positioning System (GPS) receivers for sub-meter positioning. Unfortunately, GPS signals within most buildings are too weak for effective localization, and the IMU dead reckoned solutions without absolute position corrections will not remain accurate long enough to allow the rover to navigate within the building. Further, additional the science payloads for these ground vehicles utilize the entire allowed payload in terms of size, weight, and power [44]. Outfitting these vehicles with additional indoor navigation equipment would require an extensive redesign and development effort beyond the scope of our research.

Indoor navigation for insufficiently instruments vehicles is a challenging problem. The hypothesis for this investigation is that PCS provides a compelling alternative solution to the indoor navigation problem, utilizing existing sensing equipment already embedded in the NASA campus environment to localize the vehicle. Through a PCS infrastructure, the vehicles can polymorphically restructure control systems around environment sensors in real-time as needed, allowing indoor navigation without the need for hardware modification.

The goals of the experiment are as follows.

1. Develop the control laws and high level control algorithms that implement a robust polymorphic solution to the problem of indoor navigation in a realistic scenario involving multiple competing constraints with large structural uncertainty.
2. Develop a moderately high fidelity simulation infrastructure to test UGV operation in a realistic indoor/outdoor environment in order to evaluate the PCS alternative.
3. Demonstrate the feasibility of utilizing polymorphism to solve this problem.
4. Develop a solution to the two-part optimization problem that meets the requirements of this problem.

B. Experiment Configuration

In order to investigate the feasibility of using polymorphic control to enable indoor navigation for the MAX 5A UGV in a smart space environment, a concrete simulation scenario was developed around the conceptual mission's indoor segment. This scenario was then implemented in a moderate fidelity simulation, which was needed to explore systems level issues with this proposed approach. In this scenario, the UGV is performing an autonomous data gathering experiment outdoors in the NASA ARC / CMU-SV campus. The onboard power levels have just passed a low value threshold that triggers the control system to switch into a low power recharging mode. In this mode, the UGV is required to perform the following activities:

1. Terminate the current experiment. This involves closing data files, archiving data, documenting the end of the experiment.
2. Switch to low-power operations mode. Power down non-critical systems, such as the science payloads and other systems that aren't required for the remaining tasks.
3. Perform a search for nearby accessible charging locations.
4. Identify the most appropriate charging location, and construct a plan to navigate to this location. The plan should identify specific performance objectives to be considered, which may include maximizing reserve power levels, maximizing safety, maximizing success probability, and minimizing the time to accomplish the task.
5. Autonomously navigate to the selected location satisfying the required performance objectives.

The indoor mission segment requires timely completion and deterministic power consumption. The vehicle is required to detect and avoid unexpected obstacles in the occupied buildings, and localize and navigate in dynamic indoor environments. Completion success rate is a key performance objective for the indoor navigation task.

The simulation is initialized with the rover at a random outdoor position. A fictitious library building was added to the virtual campus for this simulation, the layout is shown in Figure 2. This building is PCS enabled and outfitted with four security cameras mounted on pan and tilt actuators throughout the building. Each camera is under control of its own autonomous security system agent. These security agents are responsible for surveillance of the library, and command the camera actuators to provide acceptable coverage. The simulation environment provides monitoring of the environment, including the camera systems and UGV, as shown in Figure 3. (Figure 2, left)

Figure 2. Virtual Smart-Building in Campus

Smart building location (left), UGV navigating towards building (middle), building layout (right)

Figure 3. Virtual Smart-Building Environment.

Rover's view and security camera views are shown along the bottom of the screen. The rover's plan and current position is shown in the map view on the left.

The Reflection Architecture [42] is a component-based plug and play environment for simulation and control of embedded vehicle systems. Simulations and embedded systems can be created by loading instances of modules into the architecture at run-time. When loaded, modules register interfaces through a dynamic run-time type identification and binding system, and establishing data routes between interfaces of the modules. The Reflection architecture was previously extended to support the requirements for a Polymorphic Control System transport layer, capable of functionality such as transferring compiled executable libraries over wireless links during run-time, run-time rerouting of information and data flow, run-time construction and deletion of modules, run-time discovery and

reflection of data variables, and communication synchronization of multiple distributed Reflection kernels through a transparent reflective communication layer. The onboard MAX 5A UGV system, ground control station, and simulations are implemented in this architecture.

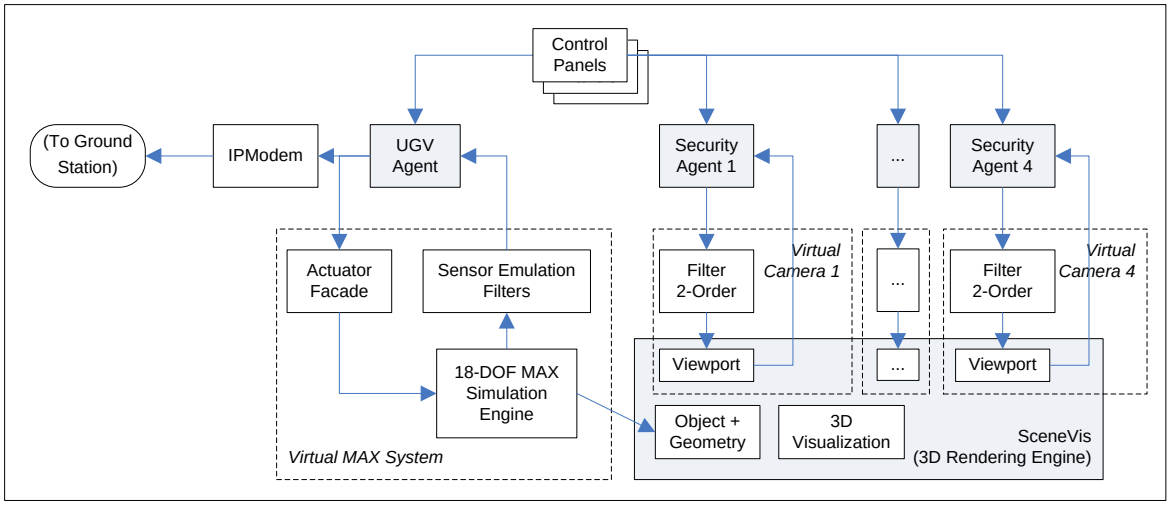


Figure 4. Reflection Architecture Simulation Configuration.
Not all components are shown. Composite components are shaded. Edges represent static data routes. Dashed boxes represent emulation of real-world hardware.

A simplified representation of the Reflection configuration for this simulation is shown in Figure 4. Boxes represent Reflection component instances. Shaded components represent multiple objects for the purpose of this drawing. For instance, the UGV Agent is composed of multiple components in Reflection as described below. The MAX 5A simulation model was created previously, implementing an 18 degree of freedom mathematical model that can simulate various nonlinear friction models over complex virtual terrains. The new modules created for this simulation are the various agents.

C. Polymorphic Reconfiguration Objective

The solution approach for this problem is shown conceptually in Figure 5. In this scenario, the rover must utilizing onboard actuation to traverse the indoor environment through areas that allow the rover to be observed by the smart security system cameras. The video images from the smart system cameras can be used to identify, track, and then localize the UGV. The data must be received by the UGV's control system in real-time, with sufficient update frequency and latency to allow for real-time autonomous navigation through its feedback control system. The camera systems are controlled individually by Security Agents, components of vision input, processing and control components. The UGV Agent is responsible for controlling the MAX 5A UGV system, and is also compose of sensors, processing, and actuation.

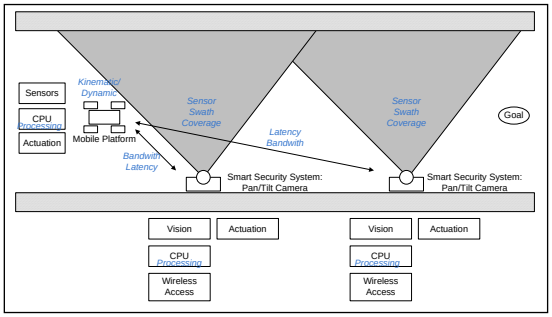


Figure 5. Problem Configuration: UGV Traversing the Smart Building Environment

The initial control configurations for the UGV and security camera systems are shown in Figure 6. Under these information structure topologies, control is decentralized and no communication occurs between systems. Each system is an autonomous agent that acts according to their objective functions. The rover's objective function is

based on optimizing the path to the power port with sufficient battery power. Battery power draw is a function of processing load: each block that is hosted on board the UGV will increase the power draw of the system. The UGV's motor system also draws power from the main battery; a trajectory that minimizes the power draw will be similar to the trajectory that minimizes distance. The power constraint is added to the final state term in the cost function (L), and it must be above a pre-specified threshold.

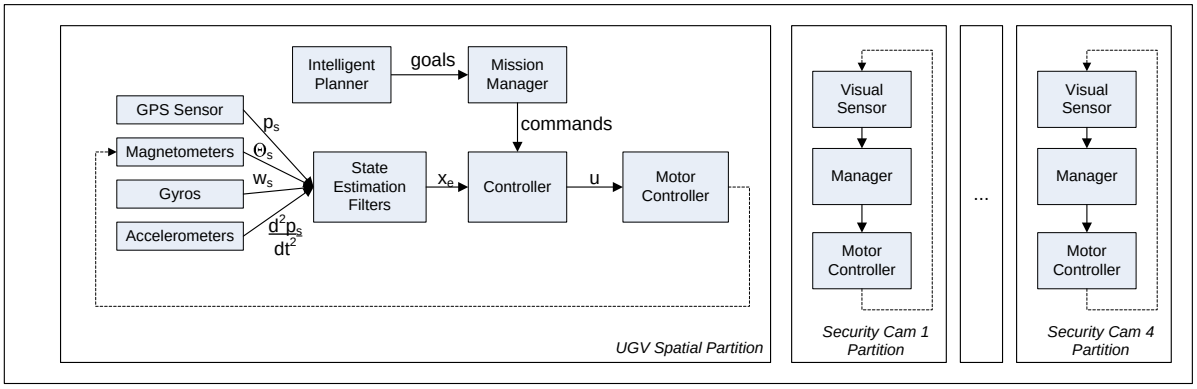


Figure 6. Initial Disjoint Control Structure for the UGV and Security Camera Systems
Initial configuration for UGV and security cameras systems feature completely disjoint control structures.

In order for the UGV to traverse the environment shown in Figure 5, the UGV Agent must be provided a set of candidate topologies that achieve the navigation goal by utilizing embedded resources in the smart environment. In addition to the initial control topology in Figure 6, four additional candidate control configurations were designed. Each candidate configuration identically closes the loop between the UGV and one of the security camera systems, as shown in Figure 7.

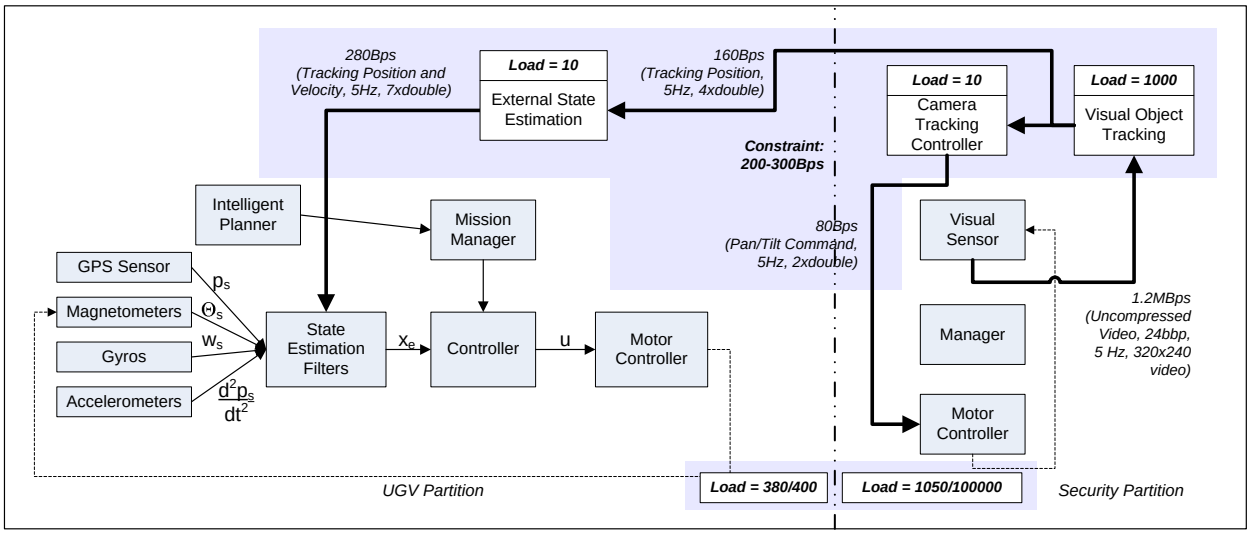


Figure 7. Candidate Reconfigured Control Structure (one of four possibilities)

Each candidate topology – except for the initial decentralized topology – results in a control structure that crosses over a single boundary between the UGV partition and a security camera partition. The PCS engine is required to determine a partition of the control structure graph that will assign a particular component to a particular processor. The partition for the reconfigured control structure is shown in Figure 7, where each component is assigned a processing load estimate, and each edge is assigned a bandwidth estimate. The processing load estimates for each component are simple estimates, while the bandwith estimates on each edge were calculated based on the amount of data passed from one module to the other, in terms of bytes per second. For instance, the vision system produces an uncompressed 320x240 image at 24 bits per pixel, and will provide images at 5Hz, which results in around 1.2 MB of data per second. Note that this number does not include the overhead bandwidth. The load

constraint on the UGV partition, set arbitrarily to 400 units, is much more restrictive than the load balance of the security system, set to an arbitrary large number. The constraint for communication between the UGV and security system was set to either 200Bps or 300Bps.

The PCS system is responsible for assigning components to each partition that balance the load on each processor along with the amount of communication required between partitions. At the minimum, the PCS system needs to establish feasible a feasible partition that meets the resource constraints. In general, load balancing is an NP hard problem, and many different approaches have been discussed in the literature [45]. For this problem, the load balancing algorithm must solve the following problem: Find the partition which maximizes the estimated load margin on all partitions while maximizing the bandwidth margins across all partitions. The exact equation for calculating the graph partitioning cost function to be minimized is shown in equation (1). Here, $/p = [p_{ij}]$ is the partition matrix, $a = [a_{ik}]$ is the partition adjacency matrix, L_j is the processing cost estimate for component j, and B_{ik} is the sum of the edge bandwidth costs for all edges that cross from partition i to partition k. The values L_{m_i} and B_{m_i} represent the maximum allowable processing load and total outgoing bandwidth from component i, which represent constraints on the solution. The value α is a normalization factor, where $\alpha = 10$ was used in this simulation.

$$J(a,p) = \sum_i \left(\left(L_{m_i} - \sum_j p_{ij} L_j \right)^2 + \alpha \left(B_{m_i} - \sum_k a_{ik} B_{ik} \right)^2 \right) \quad (1)$$

The partition matrix p is a binary matrix of size |**P**| by |**C**|, where |**P**| is number of partitions in the system, and |**C**| is the number of components. The elements of p are defined by $p_{ij} \equiv 1$ if component j is assigned to partition i, and $p_{ij} \equiv 0$ otherwise. Similarly, the partition adjacency matrix a is a |**P**| by |**P**| binary matrix defined by $a_{ik} \equiv 1$ if partitions j and k are adjacent, otherwise $a_{ik} \equiv 0$. See [41][41] for more details.

The current algorithm performs a simple brute force search across all possible partitions to minimize, which is possible for this configuration given the small number of components. The "External State Estimation" component in Figure 7 will be assigned to different partitions depending on the bandwidth constraint selected. However, since the simulation is performed on a single machine, no performance difference was observed.

D. Control System Architecture of the UGV Agent

The UGV Agent is responsible for controlling the virtual UGV vehicle system, controlling interaction with the environment, and controlling interaction with other agents. The control architecture for the UGV Agent is a hierarchal control structure, shown in the graph in Figure 8. Higher-level (superior) components have responsibility over their direct subordinate modules, as indicated by solid arrows in the drawing. These responsibilities include assigning tasks, goals, control commands, and information requests to their subordinate systems. The dashed edges represent static data flow routes from one component to another

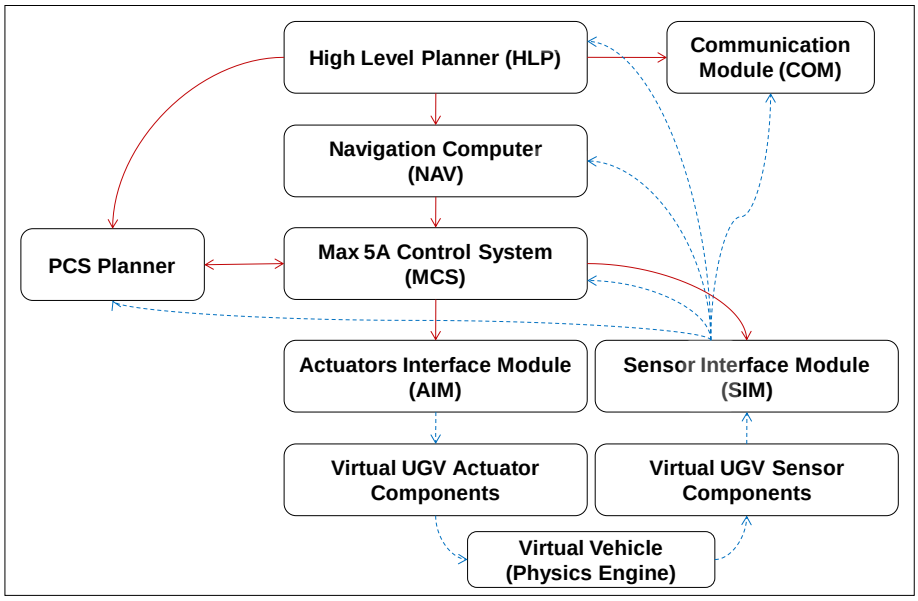


Figure 8. Intelligent Control System Architecture for the UGV Agent

E. High Level Planner

The High Level Planner (HLP) module has the highest level of responsibility in the UGV control architecture, and is responsible for instigating all actions. The HLP module produces sequences of commands (such as "Direct To" and "Track To") for the lower level Control System. Mission planning is outside the scope of this paper and experiment, so a sufficient algorithm was created to control execution through the experiment. This algorithm is recursive in nature, and shown in Figure 9.

Function: bool **LPMSA** (RL, Ppa)

Description: Low Power Mode Search Algorithm

Given:
RL : Recursion Level
P_{pa} : Position in the parent's axis system

Return bool: TRUE if successful, FALSE otherwise.

1. Establish recursion level
2. Enable appropriate COM hardware
3. Establish communication with provider
4. Download map
5. Establish location within map: $P_{la} = M_{pa2la} * P_{pa}$
6. If supported, query for set of possible valid targets.
7. Disable COM.
8. Process Map
8.1. Decompose map into search primitives
8.2. Search for possible valid targets.
9. Perform PCS Analysis
9.1. Generate mode/trajectory solution
9.2. Select optimal solution
10. Load solution into executive.
11. Execute solution.
12. If not at lowest recursion level, then
12.1 Recurse, ***LPMSA (RL-1, Pla)***
13. Return TRUE

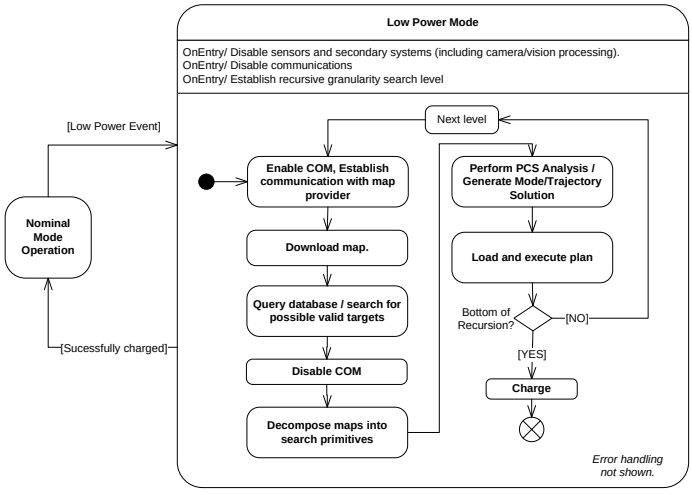


Figure 9. Higher Level Planner's "Low Power Mode" Algorithm

At each recursion level, the planner communicates with a central authority for that level. This authority provides the planner with the objective and constraints for that level, which is the navigation target location and the constraints. The HLP translates these goals and constraints to the PCS planner. In this experiment, two levels are implemented, campus level and building level. The authority provides the HLP with bitmap shown in Figure 10, which correlates to the maps from Figure 2 (a) and (b), where navigation targets are shown in red. The PCS planner utilizes direct pixel lookup to determine navigable locations in the map. As the UGV transitions to the building level, the authority also provides location and possible sensor locations of the cameras, and shown in Figure 10 (c-f).

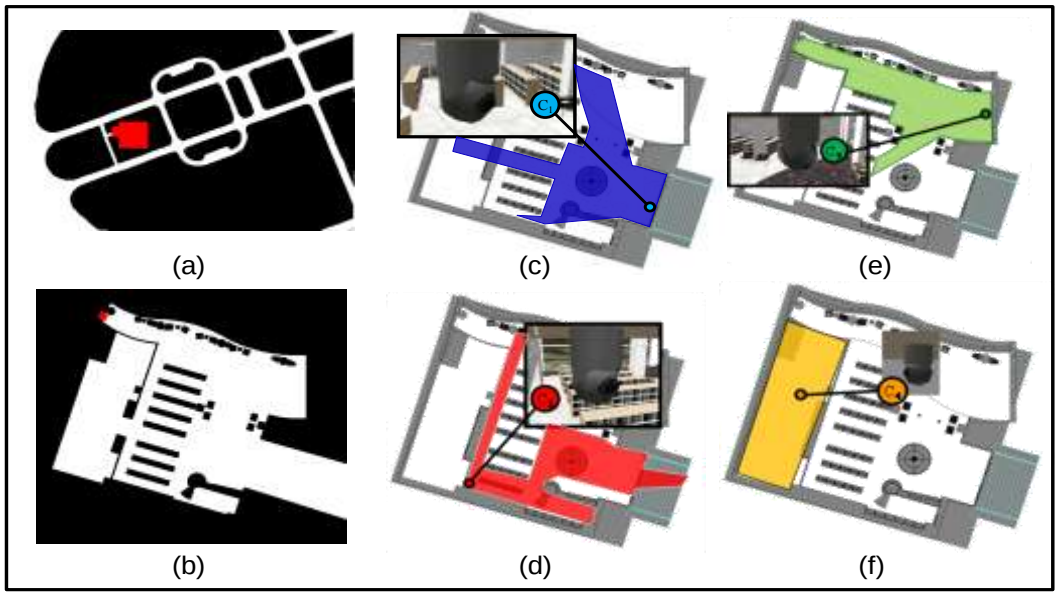


Figure 10. Navigation Objectives and Constraint Bitmaps

The allowable navigation areas for the campus (a) and the building (b). The camera locations and the sensor coverage areas for each are shown in (c)-(f), monochromatic bitmap are shown overlaid on building map.

F. MAX Control System (MCS)

The Max Control System (MCS) (Figure 8) is responsible for engaging the specified mode commands from the NAV component, and implementing the feedback control laws that produce output to the vehicle actuators. The MCS implements a simple controller graph shown in Figure 11. Here, cross-track error (XTE) is used for lateral guidance through a simple PID control feedback law to compute the desired heading, and heading error is used to control the steering error through a PID inner loop controller. The along-track error (ATE) is ignored, and the speed controller (not shown) is a PID law from the wheel speed error to the throttle.

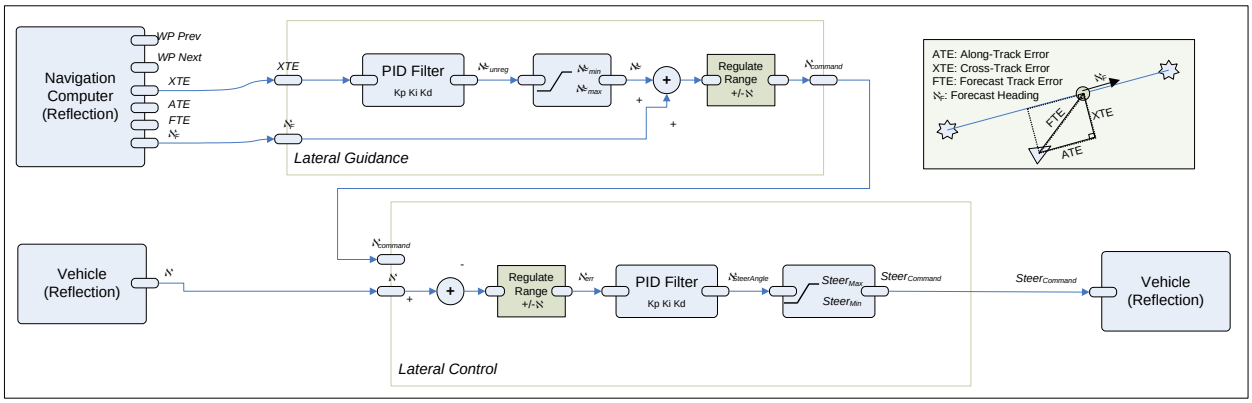


Figure 11. Max Control System (MCS) Controller Graph

G. Trajectory Planning through Suboptimal Search

The PCS planning module in Figure 8 must plan a trajectory to the target locations in Figure 10 while avoiding obstacles in (a) and (b), and navigating through observable locations in (c)-(f). The PCS Planner utilizes a random and pseudo-optimal trajectory optimization algorithm to solve the trajectory optimization function, adapted to include probabilities on the output. The algorithm is a generic search tree algorithm placed into the context of an optimal control formulation, and modified for trajectory pseudo-optimization over objective functions. This algorithm is a random tree search, similar in most respects to the rapidly exploring random trees algorithms (see [2] for a more complete treatment). It differs in that a heuristic distance estimate is not used to connect random locations back to closest points in the tree. The system relies on branch generation system through feedback control

design that solves a subspace of the global problem. The branch generation system does not necessarily correlate to the vehicle systems and controllers used in implementation. The optimization does not necessarily correspond to the vehicle model space or the actual input space, and can represent the parameters over which any decision or control can be accomplished. In this problem, the input space is augmented with a discrete variable representing the decision to switch to a new camera.

Consider any general non-linear vehicle model of the following form, where $x \in X$ represents state, $y \in Y$ represents output, $u \in U$ represents costs, and $t \in \tau$ represents the planning time interval. Generally, $X \subseteq \mathbb{R}^n$, $Y \subseteq \mathbb{R}^l$, $U \in \mathbb{R}^m$, $\mathcal{T} \in [t_0, t_f]$, $t_0, t_f \in \mathcal{R}$, $t_0 < t_f$.

$$\begin{aligned} \dot{x} &= f_C(x, u, t) \\ y &= h(x, u, t) \end{aligned} \tag{2}$$

The dynamic model f_C is assumed to be in a form augmented to enforce general equality or inequality constraints applied to the system is placed in the set $C = \{C_E, C_I\}$, where $C_E(x, \dot{x}, u, t) = 0$ and $C_I(x, \dot{x}, u, t) \leq 0$. Let C_0 be inequality constraint for obstacles which will be handled explicitly by the tree search algorithm, $C_O(x, t) \leq 0$. Define the goal space $\tilde{X}$ to be a subset of the search space $\hat{X}$ which is a subset of the state space X, such that $\tilde{X} \subseteq \hat{X} \subseteq X$. Define a branch generation control system defined as $G_b: (\mathcal{T} \times \mathrm{X} \times \tilde{\mathrm{X}} \times \mathbb{R}) \rightarrow (X \times U \times \mathcal{T})$, which generates a trajectory given a time $t_0 \in \mathcal{T}$, state $x(t_0) \in X$, and destination goal $P_g \in \tilde{X}$.

$$\left(x'_{[t_0,t_1]}, u'_{[t_0,t_1]}, t_1, J'\right) = G_b(t_0, x(t_0), P_g) \tag{3}$$

Define final state cost $L(x_f, t_f)$, and define model sensors and objectives as a set $\Phi = \{\phi_i(x, u, t)\}$ of integral cost functions. The trajectory problem can be stated as follows. Find control input $u^*(t)_{[t_0,t_1]}$ and associated trajectory $x^*(t)_{[t_0,t_1]}$ leading from a given $x(t_0)$ to a given x_{goal} that minimizes J subject to constraints C.

$$\begin{aligned} u^* &= \arg\min_{\mathrm{u}} \left(\sum_i J_i(x, u, t) \right) \\ &subject\ to\ C_O, where \\ J_i(x, u, t) &= L_i(x_f, t_f) + \sum_{\Phi} \left(\int_{t_0}^{t_f} \int \phi_i(x, u, t) dt \right) \end{aligned} \tag{4}$$

Our extension to the previous algorithm presented in [2] involves random tree branch node selection that is biased under a probability distribution applied to the tree nodes based on their cost function J, encouraging lower cost nodes to be expanded in the search. The probability function γ is given by the following, where σ and κ are constant factors for shaping the distribution. This function can be modified to include time, to encourage tree node selection down branches further in the tree.

$$\gamma(t) = \frac{1}{1 + \mathrm{e}^{\left(\frac{(t-\kappa)}{\sigma}\right)}} \tag{5}$$

The modified algorithm is shown in Figure 12. Branches are expanded by random selection over a probability mass function distribution. A priority queue is established in parallel with the tree, supporting linear time insertion and selection over a probability distribution.

FUNCTION GenerateBranch ($P_{goal} \in \tilde{x}$, Tree T, Priority Queue Q, Constraints C_O)

1. Select a random point $(x_0, t_0) \in (X, \mathcal{T})$ on the tree T under the probability density function γ.
2. Generate a random goal point $P_{goal} \in \tilde{X}$ in the goal space.
3. Use G_b to generate a candidate trajectory branch $b' = \left(x'_{[t_0,t_1]}, u'_{[t_0,t_1]}, t_1, J'\right) = G_b(t_0, x(t_0), P_g)$ from x_0 to P_{goal} based on augmented plant.
4. If b' violates obstacles in C_O, trim the trajectory.
5. $T \leftarrow T + b'$ (add b' to tree T)
6. Q.Enqueue (b', J')

Figure 12. Trajectory Algorithm

1. *Vehicle Control System and Branch Generation*

A custom branch generation system (BGS) facilitates rapid generation of branches and costs. The BGS must be efficient, as it will be repeatedly called during generation of the tree. This is conceptually shown in Figure 13. The branch generation system used for this experiment is a simplified closed-loop plant model, with sensor objectives model that calculates costs as shown below. An environment model was not required for this experiment.

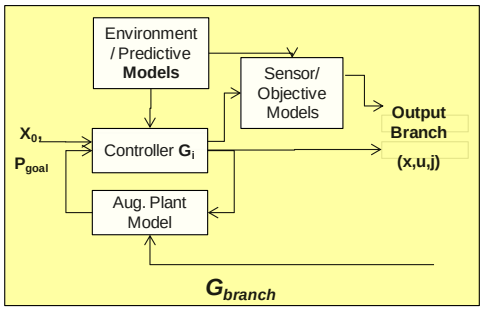
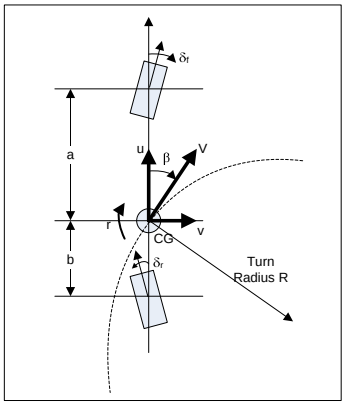
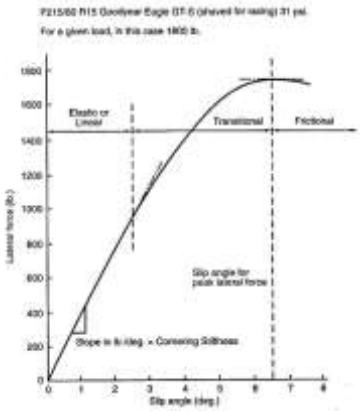


Figure 13. Branch Generation System.

The components of the BGS are a dynamic model, a control system, and the objective functions. This system was implemented in Matlab Simulink, and autogenerated into C. The objective functions were described earlier for the rover and the security cameras. A simplified dynamic model of a rover was constructed for this control system. The model is shown in Figure 14.





$$\begin{bmatrix}\dot{x}_w\\ \dot{y}_w\end{bmatrix}=\begin{bmatrix}V\cos(\psi+\beta)\\ V\sin(\psi+\beta)\end{bmatrix}$$

$$\begin{bmatrix}\frac{(C_F+C_R)}{mV} & \frac{(aC_F-bC_R)}{mV}-1 & 0\\ \frac{(aC_F-bC_R)}{I_z} & \frac{(a^2C_F+b^2C_R)}{VI_z} & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\beta\\ r\\ \psi\end{bmatrix}+\begin{bmatrix}-\frac{C_F}{mV} & \frac{C_R}{mV}\\ -\frac{aC_F}{I_z} & -\frac{bC_R}{I_z}\\ 0 & 0\end{bmatrix}\begin{bmatrix}\delta_F\\ \delta_R\end{bmatrix}$$

Figure 14. Planning Model in the Branch Generation System.
System plant is based on a two-wheel steering bicycle model.

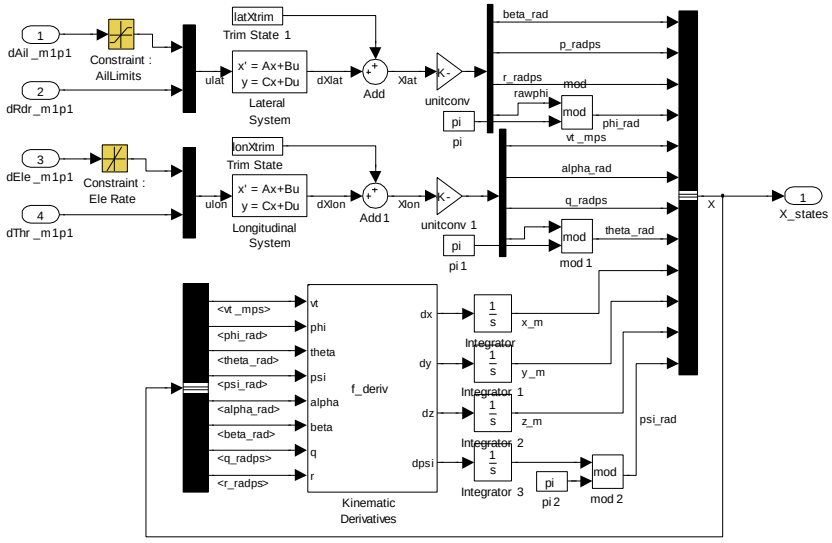


Figure 15. Implementation of the BGS plant in Matlab Simulink.

A track-to heading controller was designed for the BGS, taken from the UGV's embedded control system.

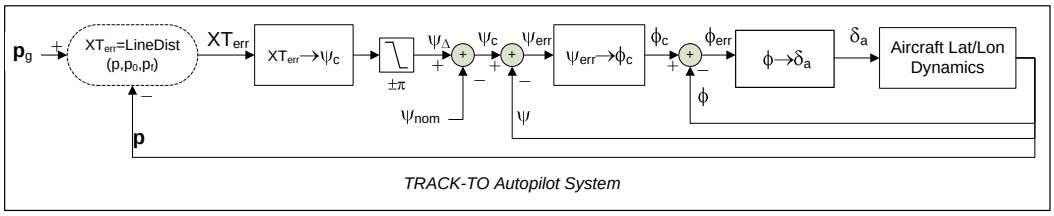


Figure 16. Track-To Control System

2. Multi Objective Cost Functions

Five cost functionals J_1 through J_5 were utilized in (5) to provide optimization objective for the trajectory planner, each with a gain factor that requires tuning and is the subject of sensitivity analysis for the results section. J1 cost function minimizes total time to complete the mission, J2 maintains a specified distance to the current camera, J3 minimizes the amount of time spent in each camera, J4 minimizes the number of times the camera switches, and J5 minimizes the total distance traveled. The cost functions are shown in (6) below.

$$
\begin{aligned}
J_1(x,u,t) &= K_1 t \\
J_2(x,u,t) &= K_2\, MIN\left(1, \frac{(\|p_{cam}(t,x) - p(x)\| - R_{des})^2}{R_{des}^2}\right) \\
J_3(x,u,t) &= K_3 \mathcal{T}_{cc} \\
J_4(x,u,t) &= K_4 N_{switch} \\
J_5(x,u,t) &= K_5 \int_{t_0}^{t} v(x)dt
\end{aligned}
\tag{6}
$$

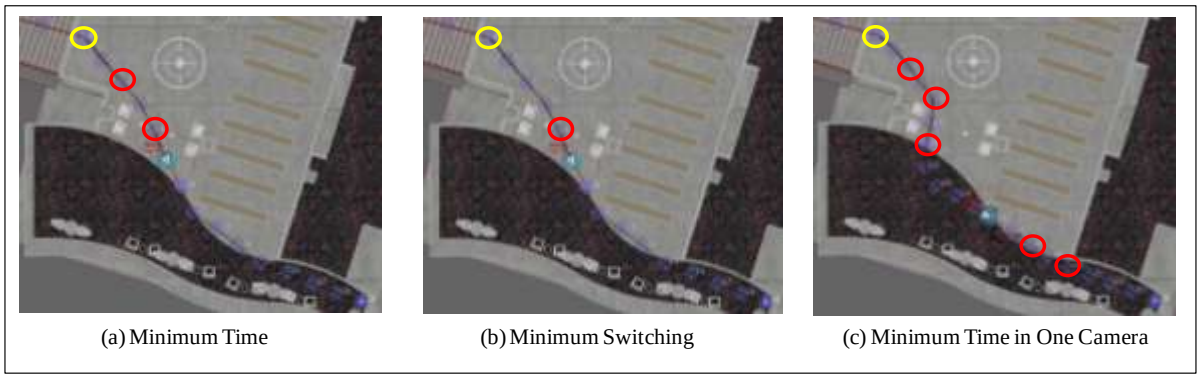
Here $\mathcal{T}_{cc}$ is the amount of time spent in the current camera, N_{switch} is the number of topological switches that have occurred (number of times the UGV has moved from one camera to another), p(x) and v(x) are the position and velocity vectors respectively, $p_{cam}(t,x)$ is the position of the camera that would be utilized, R_{des} is the desired distance to maintain from the camera to the rover, and MIN(•,•) is a function that returns the minimum of two values.

The objectives in (6) are competing and nonlinear, with a complex topology that is illustrated in the simulation results. For this reason, we utilize a random tree-search algorithm.

IV. Simulation Results

The simulation tests were implemented in Reflection in the C++ programming language. Matlab Simulink graphs were converted into C++ through MathWorks Real-Time Workshop. Tests were run on an Intel Core 2 Duo X9650, 3 GHZ, 3GB RAM, Nvidia Quadro GPU. The system allowed the UGV planning times of 10 seconds, sharing processor with simulation. Search algorithm averaged 48,202 nodes (states) added to the tree, with 3435 branches, taking 15,002,030,502 CPU clock cycles. To simplify the rendering of the search tree results, results include up to 200 full length branches considering complete path distance and branch cost.

Trajectory generation may be able to utilize various trajectory optimization objectives to generate preferable trajectories and contingencies. To study this, the solutions were studied under varying gain proportions to represent differing agent priorities. The resulting trajectories generated by optimizing over various priorities are shown in Figure 17. In (a), only time was considered. The resulting path switches control topology around three cameras in the order (C_1,C_3,C_2), and the resulting trajectory is a direct path to the destination. In (b), the number of topology reconfigurations were minimized as well as time, and the resulting path is deflected towards the camera C2 to allow only two cameras (C_1,C_2) to be utilized (resulting in a single topology reconfiguration). In (c), the amount of time spent in a single camera was minimized along with overall time. The resulting trajectory requires five switches, and the path diverges from previous solutions to allow for viewability from the different cameras.



(a) Minimum Time (b) Minimum Switching (c) Minimum Time in One Camera

Figure 17. Results of Optimizing Specific Objectives.
Screenshots of moving maps display showed. Starting position for the building level navigation is shown in yellow. The red circles show location for control switching.

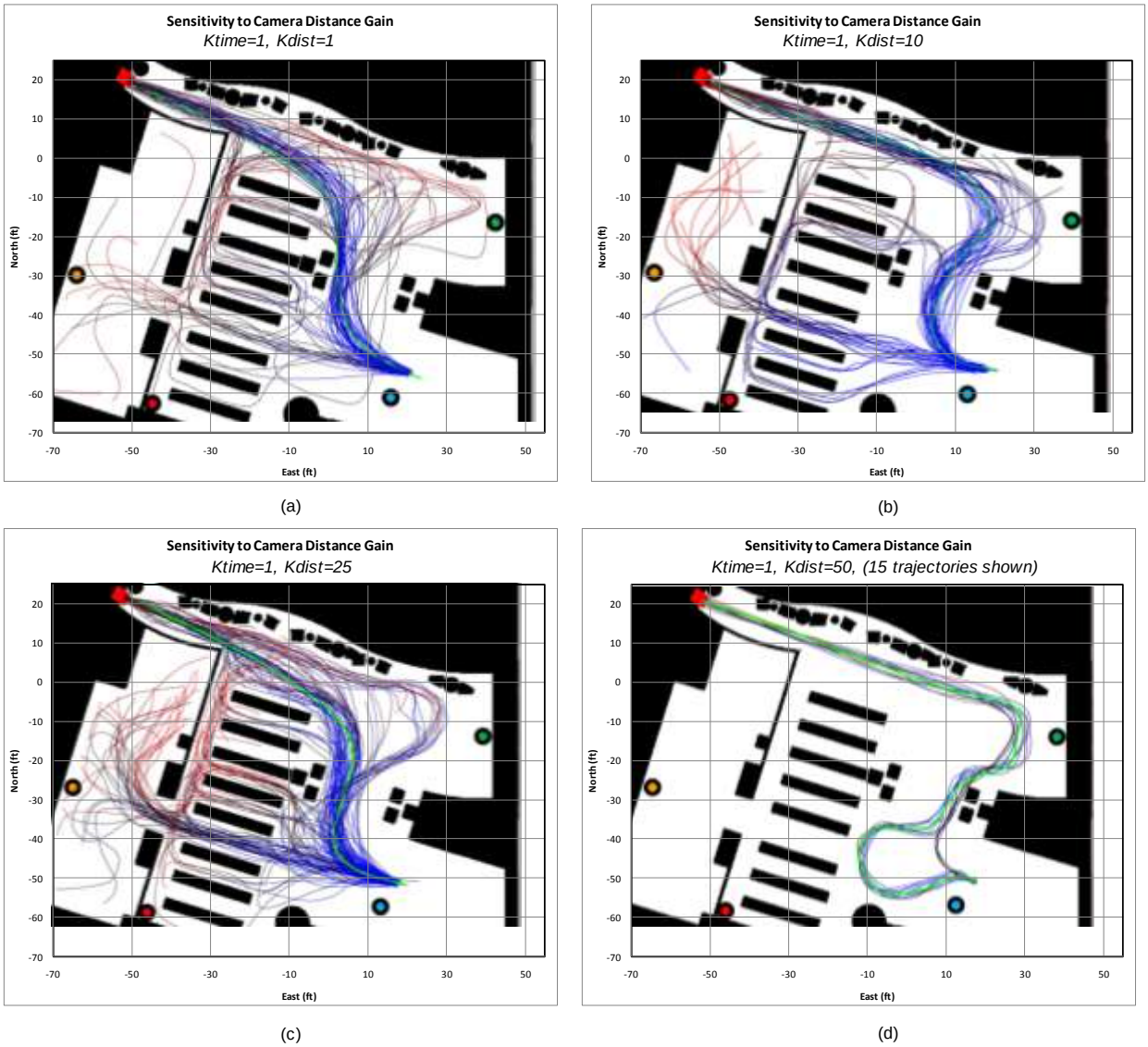


Figure 18. Sensitivity of Trajectory Search Trees to Time in Camera Gain

Optimizing over the time the UGV spends in a single camera results in the trajectory being diverted towards the distance from specific cameras. The sensitivity to this parameter was studied by varying the gain from K_{dist}=(1,10,25,50), and portions of the resulting trajectory trees are shown in Figure 18 (a)-(d). The path color indicates the cost of realizing that particular position, where blue represents lower costs, and red represents the highest state cost rendered in the plot.

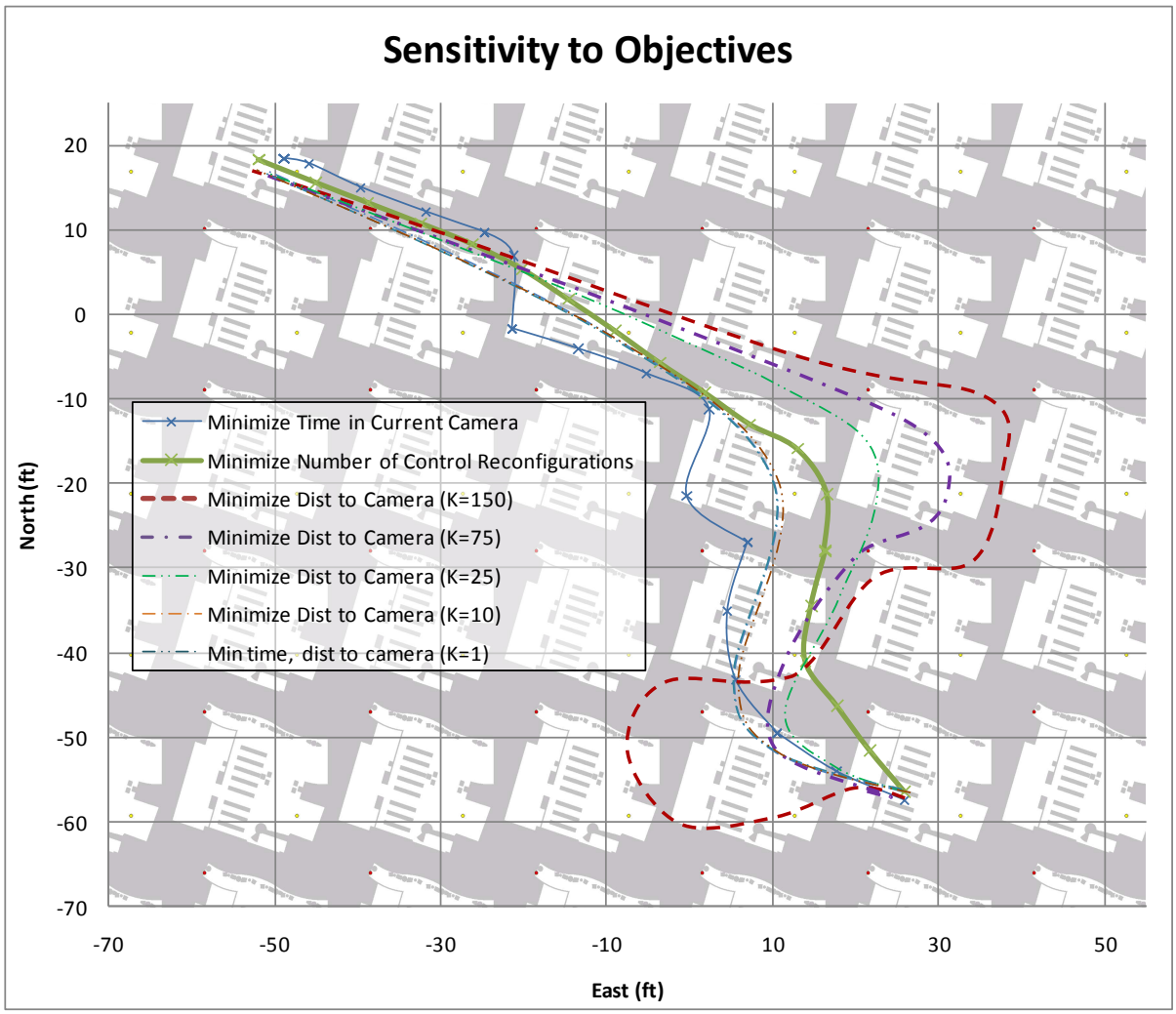


Figure 19. Comparison of Solution Objective Sensitivity

The final solution trajectories for various optimization objectives are summarized in Figure 19. Increasing or decreasing various objectives results in adjusted trajectories as shown, but the results are still viable.

V. Conclusions

The goals of this investigation were to determine if the framework for Polymorphic Control Systems was applicable to the domain of intelligent spaces for smart building control, and demonstrate the relative advantages of this approach through a detailed and realistic implementation exercise. The simulation experiment reported on in this paper was successful in meeting this goal and very promising in supporting the hypothesis of this research. The UGV agent interacts with a smart environment in a robust and resource efficient manner, with polymorphic control laws in a realistic scenario involving multiple competing constraints with large structural uncertainty. Performance of the control laws in a detailed simulation environment supports the hypothesis that PCS is applicable and powerful approach to enabling agent interactions under multiple competing objectives. The optimization provides feasible, though suboptimal, solutions to the complex non-linear optimization problem. These results demonstrate viability

of solving the two-part polymorphic control optimization problem utilizing pseudo-optimizing solvers that trade optimality for feasibility and real-time performance.

Overall the results of this experiment supports the hypothesis that control polymorphism provides robust resource sharing between agents in a smart building infrastructure for a particular indoor task. This simulation shows a more complete trajectory and guidance layer solution than has been explored in the literature to date. However, this simulation exercise does not show the complete end to end system; for instance, the path planner would have to manage all mission segments, but in this exercise, it only focuses on controlling the rover during the indoor navigation segment. The trajectory algorithm, while fast, is suboptimal, but sub-optimality of the solution was not investigated. The sensitivity study for the solutions was established, but margins for convergence were not addressed. Optimal approaches to the trajectory generation problem, requiring simplified dynamics and constraints, can also be investigated. These questions represent some of the open questions that can be investigated.

VI. Acknowledgements

This investigation was started as a class project at Carnegie Mellon University in Silicon Valley under Pei Zhang. The authors would also like to thank Kalmanje Krishnakumar, member of the Adaptive Control and Evolvable Systems group at NASA Ames Research Center, and fellow CMU PhD students for continued input into research on Polymorphic Control Systems. We would also like to acknowledge Ritchie Lee and Yoo-Hsiu Yeh from Carnegie Mellon University for their tireless efforts in support of these projects.

VII. References

[1] S.M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept, Iowa State University, Tech. Rep. TR: 98–11, 1998.

[2] C. Ippolito, Y Yeh, and C Campbell, "A Trajectory Generation Approach for Payload Directed Flight," in *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, Orlando, Florida, 2009.

[3] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 94-104, 1991.

[4] M. Satyanarayanan, "Pervasive computing: vision and challenges," *IEEE Personal Communications*, vol. 8, no. 4, pp. 10-17, 2001.

[5] J.H. and Hashimoto, H. Lee, "Intelligent Space concept and contents," *Advanced Robotics*, vol. 16, no. 3, pp. 265-280, 2002.

[6] K. Gajos, L. Weisman, and H. Shrobe, "Design Principles for Resource Management Systems for Intelligent Spaces," in *Self-Adaptive Software: Applications*. Berlin / Heidelberg: Springer, 2003, pp. 143-158.

[7] RH Katz, "Adaptation and mobility in wireless information system," *IEEE Personal Communications*, vol. 1, no. 1, pp. 6-17, 1994.

[8] L.B. Mummert, M.R. Ebling, and M. Satyanarayanan, "Exploiting Weak Connectivity for Mobile File Access.," in *In Proceedings of the 15th ACM Symposium on Operating Systems Principles*, Copper Mountain Resort, CO, December, 1995.

[9] A. Fox, S.D. Gribble, E.A. Brewer, and E. Amir, "Adapting to Network and Client Variability via On-Demand Dynamic Distillation.," in *In Proceedings of the Seventh International ACM Conference on Architectural Support for Programming Languages and Operating Systems.*, Cambridge, MA, October, 1996.

[10] B.D. Noble et al., "Agile application-aware adaptation for mobility," *ACM SIGOPS Operating Systems Review*, vol. 31, no. 5, pp. 276-287, 1997.

[11] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.I. Yang, "The case for cyber foraging," *Proceedings of the 10th workshop on ACM SIGOPS European workshop*, p. 92, 2002.

[12] R.K. Balan, D. Gergle, M. Satyanarayanan, and J. Herbsleb, "Simplifying cyber foraging for mobile devices," in *Proceedings of the 5th international conference on Mobile systems, applications and services*, 2007, p. 285.

[13] M.D. Kristensen, "Enabling Cyber Foraging for Mobile Devices," in *Proceedings of the 5th MiNEMA Workshop*, 2007, pp. 32-36.

[14] S. Sivavakeesar, O.F. Gonzalez, and G. Pavlou, "Service discovery strategies in ubiquitous communication environments," *IEEE Communications Magazine*, vol. 9, no. 44, pp. 106-113, 2006.

[15] L Kerschberg, "Knowledge rovers: Cooperative intelligent agent support for enterprise information architectures," in *Cooperative Information Agents*, 1202nd ed. Berlin / Heidelberg: Springer, 1997.

[16] D. Capera, J George, M Gleizes, and P. Glize, "The AMAS theory for complex problem solving based on self-organizing cooperative agents," in *Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Linz, Austria, 2003.

[17] E. Bitting and A. A. Ghorbani, "Cooperative Multiagent Systems for the Optimization of Urban Traffic," in *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04)*, Beijing, China, 2004.

[18] W Shen, "Distributed Manufacturing Scheduling Using Intelligent Agents," *IEEE Intelligent Systems*, vol. 17, no. 1, pp. 88-94, 2002.

[19] A. Birk, S. Schwertfeger, and K. Pathak, "A Networking Framework for Teleoperation in Safety, Security, and Rescue Robotics (SSRR)," *IEEE Wireless Communications, Special Issue on Wireless Communications in Networked Robotics*, vol. 6, no. 13, pp. 6-13, 2009.

[20] A. Birk et al., "Intelligent Autonomous Functions for Planetary Exploration," *IEEE Robotics and Automation Magazine*, vol. 16, no. 3, 2009.

[21] PS Schenker, "Advances in rover technology for space exploration," in *2006 IEEE Aerospace Conference*, 2006, p. 23.

[22] A. Seeni, B. Schafer, B. Rebele, and N. Tolyarenko, "Robot Mobility Concepts for Extraterrestrial Surface Exploration," in *2008 IEEE Aerospace Conference*, 2008, pp. 1-14.

[23] S.J. Chung and J.J.E. Slotine, "Cooperative robot control and concurrent synchronization of Lagrangian systems," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 686-700, 2009.

[24] J. Lu, Y. Zhu, B. Du, and R. Li, "Application of Mobile Agent in Wide Area Pervasive Computing System," in *IEEE World Congress on Software Engineering*, Xiamen, China, 2009, pp. 476-479.

[25] L. Yang and F.Y. Wang, "Driving into intelligent spaces with pervasive communications," *IEEE Intelligent Systems*, vol. 22, no. 1, pp. 12-15, 2007.

[26] D. Brščić and H. Hashimoto, "Tracking of Humans Inside Intelligent Space using Static and Mobile Sensors," in *The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON 2007)*, Taipei, Taiwan, 2007.

[27] A. Winfield, "Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots," *Distributed Autonomous Robotic Systems*, vol. 4, pp. 273-282, 2000.

[28] F. Capezio, D. Femia, F. Mastrogiovanni, A. Sgorbissa, and R. Zaccaria, "A Distributed Simulator for Intelligent Spaces and Robots," in *Smart homes and beyond: ICOST 2006: 4th International Conference on Smart Homes and Health Telematics*. Fairfax, VA: IOS Press, Inc., 2006.

[29] I. Fernandez et al., "Guidance of a mobile robot using an array of static cameras located in the environment," *Autonomous Robots*, vol. 23, no. 4, pp. 305-324, 2007.

[30] K. Morioka, J.H. Lee, and H. Hashimoto, "Human-Following Mobile Robot in a Distributed," *IEEE Transactions on Industrial Electronics*, vol. 51, no. 1, 2004.

[31] B.J. Lee, H.G. Lee, J.H. Lee, and G.T. Park, "New architecture for mobile robots in home network environment using Jini," in *IEEE International Conference on Robotics and Automation*, 2001, pp. 471-476.

[32] M. Broxvall, M. Gritti, A. Saffiotti, B.S. Seo, and Y.J. Cho, "PEIS ecology: Integrating robots into smart environments," in *Proc of the IEEE Int Conf on Robotics and Automation*, 2006, pp. 212-218.

[33] M Kim and N Y Chong, "Direction Sensing RFID Reader for Mobile Robot Navigation," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 1, pp. 44-54, 2009.

[34] D. Pizarro, M. Mazo, E. Santiso, and H. Hashimoto, "Localisation and Reconstruction of Mobile Robots in Intelligent Spaces. A single camera solution," in *IEEE International Symposium on Industrial Electronics*, 2007, pp. 2185-2190.

[35] J H Lee, K Morioka, and H Hashimoto, "Intelligent Space and Mobile Robots," in *The Industrial Information Technology Handbook.*, 2005.

[36] C Hwang, L Chang, and S Han, "Path Tracking and Obstacle Avoidance of Car-Like Mobile Robots in an Intelligent Space Using Mixed H2/Hinf Decentralized Control," in *2006 IEEE International Conference on Systems, Man, and Cybernetics*, 2006.

[37] C Hwang and C Shih, "A Distributed Active-Vision Network-Space Approach for the Navigation of a Car-Like Wheeled Robot," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 3, 2009.

[38] C. K. Hwang and L. J. Chang, "Trajectory Tracking and Obstacle Avoidance of Car-Like Mobile Robots in an Intelligent Space Using Mixed H2/H∞ Decentralized Control," *IEEE/ASME Trans. Mechatronics*, vol. 12, no. 3, pp. 345-352, 2007.

[39] S. Han, H. S. Lim, and J. M. Lee, "An efficient localization scheme for a differential-driving mobile robot based on RFID system," *IEEE Transactions on Industrial Mechatronics*, vol. 54, no. 6, pp. 3362-3369, 2007.

[40] J.H. Lee and H. Hashimoto, "Controlling mobile robots in distributed intelligent sensor network," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 5, pp. 890-902, 2003.

[41] C. Ippolito and K. Al-Ali, "Topological Constructs for Automatic Reconfiguration of Polymorphic Control Systems," in *AIAA Infotech@Aerospace Conference and Exhibit, AIAA-2007-2832*, 2007.

[42] C. Ippolito, G. Pisanich, and K. Al-Ali, "Component-Based Plug-and-Play Methodologies for Rapid Embedded Technology Development," in *AIAA Infotech@Aerospoace Conference*, Arlington, VA, 2005.

[43] C. Ippolito, S. Joo, K. Al-Ali, and Y.H. Yeh, "Flight Testing Polymorphic Control Reconfiguration in an Autonomous UAV with UGV Collaboration," in *IEEE Aerospace Conference*, Big Sky, Montana, USA, 2008.

[44] R Lee, C Ippolito, Y Yeh, J. Spritzer, and G. Phelps, "Payload- Directed Control of Geophysical Magnetic Surveys," in *AIAA Infotech@Aerospace*, Atlanta, GA, 2010.

[45] B Hendrickson and K Devine, "Dynamic Load Balancing in Computational Mechanics," *Computational Methods in Applied Mechanics and Engineering*, vol. 184, no. 2-4, pp. 485-500, 2000.

[46] C. Ippolito, G. Pisanich, and K. Al-Ali, "Component-Based Plug-and-Play Methodologies for Rapid Embedded Technology Development," in *AIAA Infotech@Aerospace 2007 Conference and Exhibit*, Rohnert Park, California, 2007.