Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses

Peter R. Wurman,* **Raffaello D'Andrea†** **& Mick Mountz**
Kiva Systems
Woburn, MA
{pwurman,rdandrea,mmountz}@kivasystems.com

Abstract

The Kiva warehouse management system creates a new paradigm for pick-pack-and-ship warehouses that significantly improves worker productivity. The Kiva system uses movable storage shelves that can be lifted by small, autonomous robots. By bringing the product to the worker, productivity is increased by a factor of two or more, while simultaneously improving accountability and flexibility. A Kiva installation for a large distribution center may require 500 or more vehicles. As such, the Kiva system represents the first commercially available, large-scale autonomous robot system. The first permanent installation of a Kiva system was deployed in the summer of 2006.

Introduction

Occasionally, mature industries are turned upside down by innovations. The years of research on robotics and multi-agent systems are coming together to provide just such a disruption to the material handling industry. While autonomous guided vehicles (AGVs) have been used to move material within warehouses since the 1950's, they have been used primarily to transport very large, very heavy objects like rolls of uncut paper or engine blocks. The confluence of inexpensive wireless communications, computational power, and robotic components are making autonomous vehicles cheaper, smaller and more capable.

In recent years, we have seen an increase in the use of autonomous vehicles in the field. Examples include teleoperated military devices like iRobot's Packbot and the pilotless Predator aircraft, both of which have seen service in Iraq and Afghanistan. The Mars rovers, Spirit and Opportunity, exemplify the use of autonomous robots in scientific exploration. Closer to home, the Aerosonde autonomous aircraft has been used to plumb weather systems, and recently flew in tropical storms that are unsafe for piloted aircraft. Commercially, autonomous vehicles are just hitting the market. ActivMedia's PatrolBot is a mobile monitoring system for buildings, and Aethon's Tug maneuvers supply carts around hospitals. Robots have even penetrated the home in an attempt to relieve homeowners of their most tiresome chores. iRobot sells the Roomba autonomous vacuum and the Scooba floor washer, and Friendly Robotics, among others, markets robotic lawn mowers.

Many more research projects are underway to build robots for search and rescue, mine exploration, land mine removal, and a wide variety of other intriguing tasks.

Recent advances in software engineering have laid the foundation for building large, complex systems of autonomous vehicles. In particular, the multi-agent programming paradigm has been shown to be an effective way to build and control complex systems (Jennings & Bussmann 2003). A great deal of research has focused on autonomous agents and multi-agent systems, with the expectation that, in the future, environments will be populated with hundreds or thousands of autonomous agents. In this paper, we will use the term *multi-agent system* (MAS) to refer to the general class of systems in which autonomous agents carry out actions and communicate with each other via messages, and the term multi-vehicle system (MVS) to refer to multi-agent systems with autonomous, robotic vehicles. Although systems with as many as 100 robots have been demonstrated, like the experimental CentiBot project (Konolige *et al.* 2004), the applications—disaster recovery or terrorist events—thankfully are not daily occurrences. Real, mundane applications with more than a few vehicles have been lacking.

Recently, Kiva Systems announced availability of an automated material handling system targeted at pick-pack-and-ship warehouses. The key innovation in the Kiva system is the application of inexpensive robots capable of lifting and carrying three foot square shelving units, called *inventory pods*. The robots, called *drive units*, transport the inventory pods from storage locations to stations where workers can pick items off the shelves and put them into shipping cartons. Throughout the day, the picker stays in her station while a continuous stream of robots presents pick-faces. By moving the inventory to the worker, rather than the other way around, we typically see worker productivity at least double. These results have been born out in pilot projects and at a permanent installation that went online in the sum-

*The first author is also an Associate Professor in the Computer Science Department at North Carolina State University.

†The second author is also an Associate Professor in the Sibley School of Mechanical and Aerospace Engineering at Cornell University.

mer of 2006.

Unlike the commercial robotic applications mentioned above, which consist of a handful of robots per deployment, a typical installation of a Kiva system in a large warehouse will involve hundreds of robots.

Field conditions for MVSs can be characterized in a variety of dimensions. One is the degree to which the environment is known or unknown. The extreme example of an unknown environment is planetary exploration.[1] Search-and-rescue scenarios and land mine detection pose such challenging environments that robots to perform these tasks are still a long way from being cost effective. In contrast, the Kiva drive units operate in a controlled, known environment which greatly simplifies the design problem and makes the solution practical. The business case for installing a Kiva system usually projects a one to three year return on investment.

Another distinguishing attribute of multi-agent systems is the extent to which agents are *cooperative*—in the sense that they must coordinate activities to achieve a system goal—or are *self interested* and have independent, often conflicting, objectives. Although the overall system is cooperative, the Kiva robots are essentially independent. No robot depends upon any other robot to accomplish its task, although the system requires them all to succeed to complete a customer order. Even in this cooperative setting, the multi-agent paradigm helps break the system up into manageable components with clear interfaces, knowledge, and responsibilities (Lesser 1999).

Another common research topic is the resource-based nature of most multi-agent systems. In a purely computational setting, the resources may be memory and CPU time. In physical systems, the resources may include space and elements of the environment that are need to accomplish tasks. In the non-cooperative setting, the resource allocation is typically mediated, and therefore a great deal of the research is focused on economic metaphors (Boutilier, Shoham, & Wellman 1997; Wellman & Wurman 1998). With the exception of military scenario driven research (Butenko, Murphey, & Pardos 2003), many fewer papers address resource allocation when agents are cooperative, and among most of them, the common theme is task negotiation (Jennings 1996; Malone *et al.* 1988; Rosenschein & Zlotkin 1994). Although task negotiation is an approach that can be applied to the Kiva system, the issues addressed in the literature capture only a fraction of the complexity of the allocation problem. One goal of this paper is to present a fertile and interesting non-military cooperative resource allocation problem of practical importance.

The Distribution Center

Warehouses and Distribution Centers (DCs) play a critical role in the flow of goods from manufacturers to consumers. They serve as giant routing centers in which pallets of products from different manufacturers are split and the items are redirected into outgoing containers. Figure 1 illustrates a typical DC in which incoming pallets of products are first stored in the reserve inventory location. As needed, cases of product are moved into the forward location where they are opened and individual items accumulated into shipping cartons. The cartons may be plastic totes that are sent to a retailer for in-store restocking, or boxes sent directly to the consumer.

Although the tasks at the DC are necessary to the flow of products, it is generally considered a cost center, in that the work done at the facility adds no inherent value to the product. Thus, there is constant pressure to reduce operational costs and improve accuracy. However, surprisingly little has changed in the industry in the last twenty years. New automation systems have been introduced, such as carousels and high-speed sorters, but the majority of these have proven to be inflexible and costly. Thus, many warehouses still use manual processes or conveyors to move orders to the pick locations.

Consider, for example, what happens if you order two books from a highly sophisticated online book retailer. A large Internet bookstore may carry a million different book titles. Although they don't stock all of these products in their warehouses—some are shipped directly from the publisher—the company likely stocks hundreds of thousands of different products. The majority of the picking done in this type of warehouse involves what is called *open-case* picking; the retailer receives a case of books and ships them out as individuals. Only occasionally does someone order an entire case of books, and often that order is filled from a different part of the warehouse.

Naturally, when you order your books, it would be inefficient for the retailer to send a picker out into the warehouse to fetch the books for your order, and only your order. Instead, they may queue up several hours worth of orders and compute the best way to batch work in the warehouse. Often, multi-line orders are split across different pickers because the elements of the order are stored in distant locations. Each picker in a zone of the warehouse fills a tote with books that are part of several different customers' orders. The filled tote is conveyed to a mechanical sorting machine. A typical sorter is comprised of flat trays attached to a revolving track. An induction worker empties the tote and puts the items, barcode up, one-at-a-time, on empty trays as they circulate around the machine. The sorter has a scanner to identify the product on the tray, which allows it to dump the product off the tray at the appropriate moment and into a chute that serves the pack worker. To keep her busy, the pack worker is working on several orders at a time, and thus, on receiving products dumped off the tray, must *resort* them into the individual orders. When the pack worker has received all of the products you ordered, she puts them together into a box, inserts the packing slip, attaches the shipping label, and sends the completed box to the shipping dock.

Thus, the process of filling your order for two books involved four people: two pickers in different zones of the warehouse each picking batches of products, someone to split the batches back into individual items on the sorter, and

[1] It is an extraordinary engineering and scientific accomplishment that the Mars rovers are still in operation three solar years past their expected useful lives.

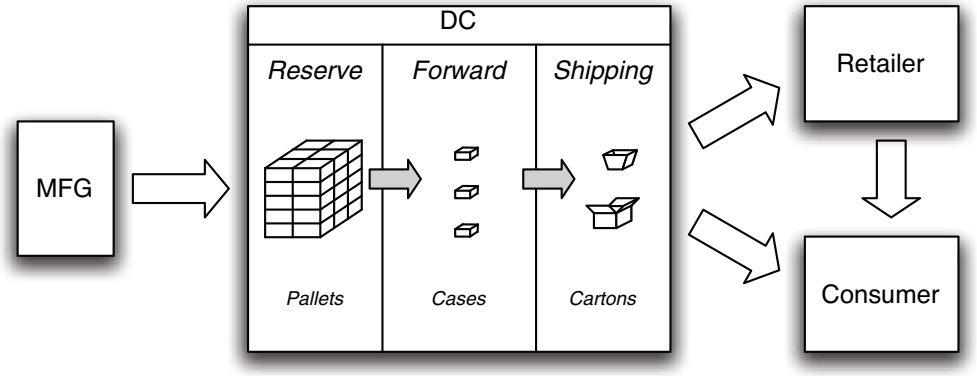

Figure 1: An abstraction of product flow from manufacturer to consumer through a distribution center (DC).

a fourth person to collect the products kicked off the sorter into customer orders. Moreover, it involved *miles* of expensive conveyor and sorting equipment (Gilmour 2003), which takes months to install and cannot be easily moved once set up.

Myriad automation solutions exist in the industry, including the aforementioned sorters and intelligent conveyors. Automated Storage and Retrieval Systems (AS/RSs) can achieve very high throughput for highly specialized and uniform products, like CDs. But among the automation solutions that are designed for a broad range of product types and sizes, the actual benefits have rarely lived up to the promises. For example, carousels—vertical shelves attached to a rotating track—appeared promising but have proven to be bottlenecks in high throughput situations. The very best of automation technologies claim to be able to achieve peak rates of two to four hundred lines per hour per picker, but most highly automated warehouses experience somewhat less than that. In fact, the high-profile online-grocer Webvan burned through several hundred million dollars in funding to open a few, highly automated warehouses (with carousels), but was unable to reduce costs to the point where they could make money on the orders.

Traditional automation approaches have several drawbacks:

- Costly: price tags for automated material handling systems typically run in the tens of millions of dollars.
- Long design cycles: most large DC projects takes 12 to 24 months to bring online, in part due to the difficulty of installing and tuning the automation.
- Inflexible: once installed, conveyors, sorters, carousels, and other systems are difficult and costly to move. They also tend to be inflexible to changes in the inventory mix.
- Not expandable: few automation systems can be incrementally expanded, which forces companies to buy enough capacity up front to handle several years worth of anticipated growth. This results in excessive capital expenditures for automation systems that run under capacity for several years.
- Batch processing: like in the bookseller example above, attempts to improve the efficiency of the picking task leads to aggregating orders. Once the products are picked in batches, the warehouse employs expensive automation (sorters) to undo the aggregation and break the batches back up into individual orders.
- Fixed locations: most of the automation systems rely on products being stored in fixed locations, and most warehouse management software assumes the pickable products are stored in a single location. Like in a retail store, a particular width of shelving must be designated to each product, which makes it difficult to adjust the shelf space to accommodate changes in the stocking level. Further, having fixed locations means that the replenishment worker must move the incoming product to that specific location in order to restock the product.
- Manual reslotting: because of the batch processing and fixed locations, warehouse managers are constantly evaluating inventory locations to keep workloads balanced and the most popular products in the choicest picking locations. Reslotting requires a lot of manual movement of product from one storage location to another.

The Kiva solution improves on all of these factors, and in some cases, eliminates the problem completely.

The Kiva Solution

Figure 2 shows the physical elements of the Kiva solution: an inventory pod and a drive unit. The drive units are small enough to fit under the inventory pod, and are outfitted with a mechanical lifting mechanism that allows them to lift pods off the ground. The pods consist of a stack of trays, each of which is subdivided into bins. A variety of tray sizes and bin sizes create the mixture of storage locations for the profile of products the warehouse stores.

Typically, a Kiva installation is arranged on a grid with storage zones in the middle and *inventory stations* spread around the perimeter. Figure 3 shows an exemplary layout, with storage cells in green, drive units in orange, and station queues on the left in purple and pink. The drive units are

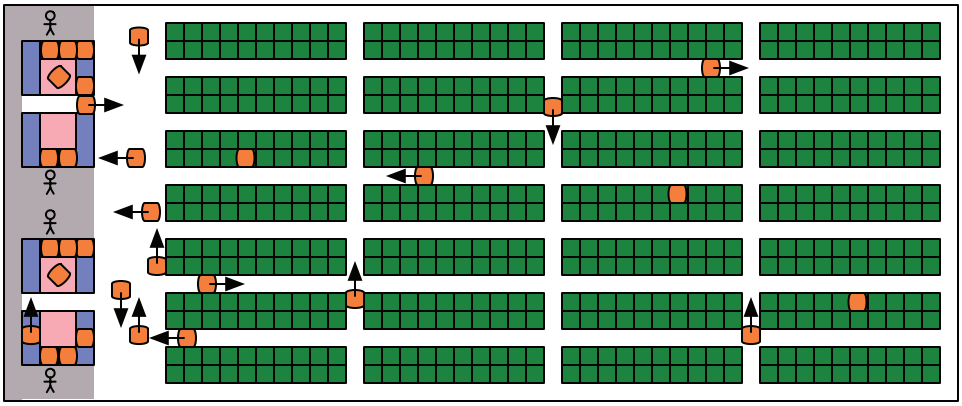

Figure 3: A small region of a Kiva layout. The green cells represent pod storage locations, the orange ovals the robots (with pods not pictured), and the purple and pink regions the queues around the inventory stations.

Figure 2: A Kiva drive unit and storage pod.

used to move the inventory pods with the correct bins from their storage locations to the inventory stations where a pick worker removes the desired products from the desired bin. Note that the pod has four faces, and the drive unit may need to rotate the pod in order to present the correct face. When a picker is done with a pod, the drive unit stores it in an empty storage location.

Each station is equipped with a desktop computer that controls pick lights, barcode scanners, and laser pointers that are used to identify the pick and put locations. Because every product is scanned in and out of the system, overall picking errors go down, which potentially eliminates the need for post-picking quality control. In general, every station is capable of being either a picking station or a replenishment station. In practice, pick stations will be located near outbound conveyors, and replenishment stations will be located near pallet drop off points.

The power of the Kiva solution comes from the fact that it allows every worker to have random access to any inventory in the warehouse. Moreover, inventory can be retrieved *in parallel*. When the picker is filling several boxes at the same time, the parallel, random access ensures that she is not waiting on pods to arrive. In fact, by keeping a small queue of work at the station, the Kiva system delivers a new pod face every six seconds, which sets a baseline picking rate of 600 lines per hour.[2] Peak rates can exceed 600 lines per hour when the operator can pick more than one item off a pod.[3]

For a large warehouse, the savings in personnel can be significant. Consider, for example, what a Kiva implementation of the book warehouse would involve. A busy bookseller may ship 100,000 boxes a day. With existing automation, this level of output would employ perhaps 75 workers

[2]This statistic is based on single unit picks and has been reproduced for extended periods in the Kiva test facility.

[3]This statistic was verified when a small Kiva demonstration system was brought to a drugstore distribution center where operators picked at nearly 700 lines per hour.

picking, sorting, and packing orders over two 8-hour shifts. Now consider a Kiva solution. At a conservative 500 lines per worker per hour, and two lines per outgoing box, the day's work would require 400 hours of picking. Maintaining the 16 hours of picking a day, the warehouse would need 25 Kiva inventory stations to generate the 200,000 outgoing boxes. It would take about 200 drive units to serve the 25 pickers at the stations, delivering and storing inventory pods.[4] However, the warehouse can be run with 50 fewer employees, at $10/hour for 16 hours a day, which means the book warehouse is saving $8,000/day. With 250 workdays, the net savings is $2m/year.

In addition to the productivity gains, the book seller would experience several other benefits by using the Kiva system:

- Greater accountability: each order is filled completely by a single individual, improving accuracy and accountability by reducing the number of "touches" on the product.
- No downstream dependencies: no one worker's productivity depends on the performance of workers earlier in a sequential process. Instead, each worker's station is complete and self-contained.
- No batch processing: in a Kiva warehouse, everything is done in real time. An order can literally be filled within minutes of being received.
- Location-free replenishment: because any station can be used to put product away, the replenishment process is greatly simplified.
- Adaptive slotting: because the resolution of the allocation of storage is bins rather than the faces of static shelves, the system much more easily adapts to changes in stocking policies. Every product is automatically given just enough pick faces.
- No single point of failure: unlike a conveyor, if a drive unit fails, it does not stop the whole floor. The rest of the system continues to operate, and most likely there is no noticeable impact on productivity.
- Rapid deployment: because there is no fixed infrastructure, a 50 station warehouse can be brought online in a matter of weeks rather than months. Kiva has set up small, two-station systems in a single day.
- Spatial flexibility: Kiva systems can accommodate poles, flow into multiple rooms, and handle other oddities of the environment. By incorporating automated lifts, a Kiva installation can use mezzanines to fill the vertical space.
- Expandability: if a warehouse needs more capacity, they simply add more pods, drives and stations.

The last three bullets combine in possibly game-changing ways. The ease with which warehouses can be brought online and expanded means that managers do not need to purchase automation with the capacity to handle the volume forecast for five years out. Instead, they need only a big enough building, and they can buy the Kiva components to handle the growth as it occurs. Furthermore, companies can plan and deploy warehouses in months rather than years, enabling a faster, more accurate response to changing market conditions.

[4]The actual ratio of drive units to workers varies depending on attributes of the products and order profiles.

AI Techniques Used

The Kiva system is highly influenced by AI techniques, though many of the techniques used are textbook implementations of well-known algorithms. The software architecture reflects the fact that the Kiva system is, by its very nature, a multi-agent system. Each drive unit and each station is a computational device that can receive requests and act on them. At the same time, the system embodies a massive, real-time, resource allocation problem. The resources in question include shelf space at the station, drive units, storage pods, inventory, and physical space.

Multi-agent System

Each robot is represented in the system by a *drive unit agent* (DUA), and each station by an *inventory station agent* (ISA). System-wide resource allocation is centralized in the *Job Manager* (JM), which also communicates with the warehouse management system.[5] Agents communicate with each other via XML messages. Well over 100 message types are sent among the agents.

The JM receives customer orders that need to be filled, and assigns drives, pods, and stations to carry out the tasks. Figure 4 illustrates the multi-agent nature of the architecture. The combination of the resource allocation (in the JM) with task planning, path planning, and motion planning (in the DUA) is the control stack with abstraction layers typically seen in the literature (Simmons *et al.* 2002). The ISA on the station computer manages the GUI and the picking lights, and communicates with the other agents to receive, request, and report accomplishment of its tasks.

We found the following rules useful in deciding how to partition the system into agents:

- Physical correspondence: in most situations, it makes sense for there to be one agent for each physically distinct object.
- Information encapsulation: each agent should know just enough information to do its job. This cuts down on the amount of information that needs to be communicated or accessed through the database.
- Single-agent ownership: all of the important data elements are owned by only one agent. Although multiple agents may need the information, only one can permanently change it and write it to the database.
- Separation by job: when there is a resource allocation that needs to be done whose antecedents and effects can be separated from other tasks, it is a candidate for encapsulating as a separate agent.

These rules also guide feature implementation, because the information boundaries force some types of solutions. The following example illustrates some of the above points.

[5]Most of Kiva's customers have existing warehouse management software that pass orders into the Kiva system.

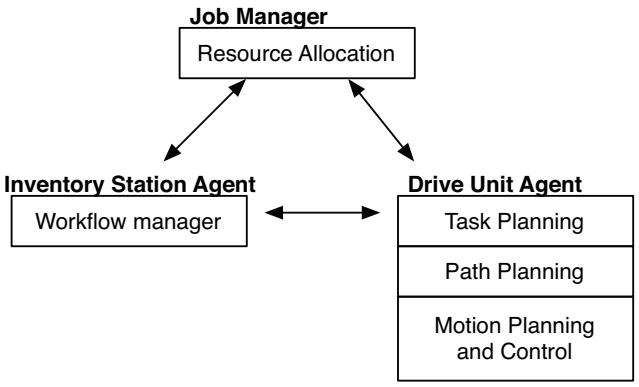

Figure 4: The multi-agent control system.

Suppose the JM decides that drive X should deliver pod Y to station Z so that the worker can pick a product off the A face of pod Y. The JM tells the station to ask drive X to fetch pod Y. The station does not need to know where pod Y is located; if the drive unit doesn't already have pod Y, it will look up its location and go fetch the pod. Similarly, when the station asks drive X to present the A face of pod Y, it does not tell the drive what it intends to do with pod Y, only that it needs to see face A. Once the pick tasks are complete, the station releases the pod, and the DUA, if it has no other tasks for the pod, requests a storage location.

The benefits of the multi-agent architecture can be divided into two classes: computational and organizational. The computational benefits include a natural decomposition of the computation that can be spread across as many servers as necessary. In addition, the multi-agent design makes it clear where to focus effort when making the system robust to failures. Every agent must be able to handle not just error responses, but the outright failure of the agents it communicates with. The organizational benefits include code compartmentalization, which makes it easier to know where to put certain functionality. The multi-agent design also establishes clear boundaries of ownership among the software developers.

Path Planning

The grid constitutes a 2-D graph of paths which may be given weights at design time. The drive units use a standard implementation of A* to plan paths to storage locations and inventory stations. The DUAs also maintain a list of high level goals, and are responsible to prioritizing the goals and accomplishing them as efficiently as possible. For instance, more than one station may ask for the same pod, and a station may ask to see more than one face of a pod. The DUA decides which station to visit first, and in what sequence to show the faces to minimize travel time. The DUA uses a simple AI-style planner to make these decisions.

Resource Allocation

Our overarching design goal is to keep the pickers and replenishment workers as busy as possible with the least amount of hardware, warehouse space, and inventory on hand. One can imagine keeping everyone busy by having 100 robots and a complete copy of all of the products per worker, but it would be unnecessarily expensive. Clearly, good resource allocation algorithms are critical to getting the work done with a cost-effective amount of hardware.

Although one could describe the resource allocation task as one large, global optimization problem, it is impractical to do so for a variety of reasons. First, the resource allocation decisions must be made in real time—there is no window in which to do the offline computation because most customers are receiving orders throughout the day for same day shipping, and are constantly re-prioritizing jobs to match truck schedules. Second, the problem descriptions are quite large, and may include tens of thousands of orders and hundreds of thousands of bin choices. Third, the optimal solution also depends on the actual paths and interactions of the vehicles, which is dynamic. Fourth, the system includes constant human interactions, with their variable, and unpredictable, response times. The human variability and the dynamic nature of the rest of the system means that any optimized solution is likely to be fragile.

Instead of attempting global optimization, we take the approach of making decisions on the fly using utility-based heuristics, where the utility is measured as the cost to the warehouse owner. The decisions that the system makes using utility metrics include:

- Job Assignment: this is the task of assigning orders to workers at stations. The system costs go down when we can pick more than one item off a pod, which is more likely to happen when the orders at the station share common products. Thus, the heuristic looks at the similarity of this job to other jobs already assigned to the station. The assignment problem is sometimes further constrained by that fact that some stations or workers may have specialized capabilities, like gift wrapping, that are required by the order.
- Pick-task assignment: once a job is assigned to a station, we pick the pods and drive units that will be used to bring over the products that go into the carton. The heuristic combines distance and the number of needed products that are available on the candidate pods when making this selection decision.
- Replenishment-task assignment: when the time comes to replenish a case of product, the system decides where to store it. There are obviously bin packing issues involved in selecting appropriate bins based on the dimensions of the case, but there are other features to consider, like the current location of the pod.
- Pod storage: when a station is done with a pod, the system selects an open parking spot for the pod. The location of the selected storage cell affects not only the time to free up the current drive unit, but also the time it will take to deliver that pod the next time it is needed. The heuristic balances these costs and keeps the pods that are more likely to be used closer to the stations, and the pods with slow products in the back of the room.

We believe that the utility-based approach gives an unprecedented amount of flexibility and adaptability to the system. The system runs well through a wide range of configurations and operational challenges. At one point during a deployment in a customer's warehouse, an electrician needed to drive a large cherry-picker onto the floor to do some electrical work in the rafters. Using Kiva's software tools, we were able to block off a large area of the floor and give the cherry picker access to the pole. The system routed the robot traffic around the keep-out zone and was able to continue to supply the workers with inventory so that they were able to continue filling orders.

Fielded Systems

In the preceding sections, we used a hypothetical warehouse to illustrate some of the drawbacks of existing automation and contrast them with a Kiva implementation. We now present some results from actual Kiva deployments, to the extent permitted by the customers. In the spring of 2005, Kiva set up two small installations and ran each for three months. Since both systems were pilots, we did only minimal integration with the existing warehouse management systems. More recently, Kiva installed a permanent system in an warehouse in Pennsylvania for a major office supply retailer.

The first pilot was conducted in a candy warehouse that supplied samples to salespeople. In the candy warehouse before the Kiva pilot, the picker drove around the warehouse with a forklift fetching the needed products from pallet racks and returned them to a sorting and packing station. With the Kiva system, the worker remained at the station while the candy samples came to her. The picker did five to six times more orders with Kiva than with the pre-existing, manual system. By the end of the pilot, the Kiva robots had expanded through doorways into three different rooms of the warehouse. The project was very successful, and the warehouse manager at that facility did not want the system taken out when the pilot contract expired.

The second pilot took place in a distribution center for a Fortune 500 office supply company. This particular business unit filled delivery orders for office supplies. After a successful 3-month pilot that proved both the effectiveness and robustness of the Kiva system, the retailer purchased a permanent installation for one of its warehouses in Pennsylvania. This second system came online during the summer of 2006 with 30 robots and five stations. Over the next three months, the office supply company purchased more Kiva equipment, until the system more than quadrupled in size, and now routinely accounts for half of the facility's output each day. At well over 120 robots working 24 hours day, we believe that this warehouse represents the largest MVS in existence in a single facility. Most importantly to our customer, pick workers on the Kiva side fill orders at more than twice the rate of workers using the old conveyorized system.

In addition to these installations, Kiva has a permanent demonstration facility in Woburn, Massachusetts to showcase the many features of the picking system. The demonstration warehouse has several different station configurations, 250 inventory pods and 60 drive units. The facility also includes a vertical lift to demonstrate using the Kiva system in conjunction with mezzanines. Figure 5 shows the demonstration facility.

Design and Development

The Kiva *Material Handling System* (MHS) was written in Java, including all of the agents mentioned above and several tools which are used by the warehouse staff to control the order flow and to manage the hardware. A MySQL database is used to persist the data, though the software can use any SQL database. The vast majority of the software was designed and written by a team of 13, led by two PhDs with AI backgrounds and two more with expertise in control systems. The hardware was designed by a similarly sized team of mechanical and electrical engineers.

In addition to the MHS, we have developed an elaborate simulation of the robotic fulfillment system using a discrete event programming language. We use this tool to design customer installations and to explore algorithms and analyze their systemic impacts. The simulation includes detailed modeling of the drive unit motion, pod geometry, and the human operations, and can be fed simulated order data, or actual customer data. We have simulated installations with as many as 150 picking stations, 1500 drive units, and 20,000 storage pods. Such a facility would have the capacity of the largest customer site we know of.

Although Kiva has built a working and cost effective system for everyday use in warehouses, there is plenty of room for improvement. To make it easy for researchers to study MVS problems like those found in the Kiva system, we have created an open-source, stylized warehouse environment called *AlphabetSoup* (Hazard, Wurman, & D'Andrea 2006). The AlphabetSoup warehouse must build words out of letter tiles. The letter tiles are stored in buckets, and moved around the warehouse by *bucketBots*. The project details and source code can be found at http://research.csc.ncsu.edu/alphabetsoup/.

Conclusion

So far, the Kiva system has been well received by the marketplace. As the Kiva approach spreads through the industry, there are likely to be many interesting new computational and organizational problems that need to be solved. In this paper, we describe some of the interesting aspects of the system that clearly draw on an AI heritage, while highlighting some of the very challenging allocation and design problems. In addition to the issues outlined here, there are a large number of robotic research topics that would benefit systems like Kiva's.

We have found the opportunity to work on systems involving hundreds of robots to be very energizing, and we hope it will inspire others to work on related issues. In the very near future, we look forward to the opportunity to stand on an observation platform in a Kiva deployment and watch several hundred mobile robots busily getting work done.

Figure 5: The Kiva demonstration facility.

Acknowledgments

Building a working MVS requires a core set of great mechanical, electrical, and software engineers. It is yet another thing to turn it into a commercial product and manage the manufacture, assembly, and deployment of these systems. We thank the world-class Kiva employees who have breathed life into this vision.

References

Boutilier, C.; Shoham, Y.; and Wellman, M. P. 1997. Economic principles of multi-agent systems. *Artificial Intelligence* 94(1):1–6.

Butenko, S.; Murphey, R.; and Pardos, P. M., eds. 2003. *Cooperative Control: Models, Applications and Algorithms*. Springer.

Gilmour, K. 2003. Amazon warehouse, amazon adventure. *Internet Magazine*.

Hazard, C. J.; Wurman, P. R.; and D'Andrea, R. 2006. Alphabet soup: A testbed for studying resource allocation in multi-vehicle systems. In *Proceedings of the 2006 AAAI Workshop on Auction Mechanisms for Robot Coordination*, 23–30.

Jennings, N. R., and Bussmann, S. 2003. Agent-based control systems: Why are they suited to engineering complex systems? *IEEE Control Systems Magazine* 61–73.

Jennings, N. R. 1996. Coordination techniques for distributed artificial intelligence. In O'Hare, G. M. P., and Jennings, N. R., eds., *Foundations of Distributed Artificial Intelligence*. Wiley. 187–210.

Konolige, K.; Fox, D.; Ortiz, C.; Agno, A.; Eriksen, M.; Limketkai, B.; Ko, J.; Morisset, B.; Schulz, D.; Stewart, B.; and Vincent, R. 2004. Centibots: Very large scale distributed robotic teams. In *Proceedings of the International Symposium on Experimental Robotics*.

Lesser, V. R. 1999. Cooperative multiagent systems: A personal view of the state of the art. *IEEE Transactions on Knowledge and Data Engineering* 11(1):133–142.

Malone, T. W.; Fikes, R. E.; Grant, K. R.; and Howard, M. T. 1988. Enterprise: A market-like task scheduler for distributed computing environments. In Huberman, B. A., ed., *The Ecology of Computation*. North Holland.

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter*. Cambridge: The MIT Press.

Simmons, R.; Smith, T.; Dias, M. B.; Goldberg, D.; Hershberger, D.; Stentz, A.; and Zlot, R. 2002. A layered architecture for coordination of mobile robots. In Schultz, A., and Parker, L., eds., *Multi-Robot Systems: From Swarms to Intelligent Automata*. Kluwer.

Wellman, M. P., and Wurman, P. R. 1998. Market-aware agents for a multiagent world. *Robotics and Autonomous Systems* 24:115–25.