Scalable Human-Robot Interactions in Active Sensor Networks

Decentralized sensor networks promise virtually unlimited scalability and can tolerate individual component failures. An experimental active sensor network that leverages environment-centric modes of human-robot interaction can keep up with a network's arbitrary growth.

Alexei A. Makarenko, Tobias Kaupp, and Hugh F. Durrant-Whyte
Australian Research Council Centre of Excellence for Autonomous Systems

Spatially distributed sensors provide better coverage, faster response to dynamically changing environments, better survivability, and robustness to failure. Taking an extra step to a decentralized system provides further benefits of scalability, modularity, and performance.

Our *active sensor network* is a collection of sensing platforms connected into a network. Some or all of the network components have actuators that we can control, making them, in this sense, active. A mobile robot with onboard sensors and a communication facility is an example of an active component. So is a video camera with adjustable pan, tilt, and zoom.

We investigate the scalability of an important aspect of an ASN: interaction with human operations. Specifically, we'd like to know the conditions under which human interaction with the ASN won't jeopardize the overall system's scalability.

Active sensor networks

We envision an ASN engaged in various information-gathering tasks: from traditional data fusion activities such as map building and target tracking to control-related exploration and search. So, an ASN as we've defined it must be built on two algorithms: data fusion and control. The individual nodes link to form a tree network. The *decentralized data fusion* network propagates all new information about the environment through local communication links. The acyclic network topology allows optimal fusion of information while avoiding double-counting. The DDF algorithm ensures that each network node contains the same consistent belief about the world's state (ignoring information propagation delays).

We can extend the DDF architecture to incorporate a decentralized control layer by defining a team reward function and by placing a local decision-making module at each platform (see the "Why Decentralized?" sidebar).[1] The control objective is typically to maximize the network's total information gain, subject to certain state and control constraints. We can incorporate other objectives—such as energy use, and safety—using utility theoretic methods. Modeling of platforms, sensors, and the environment as a set of continuous states, along with the use of information as payoff, lets us formulate the information acquisition problem as a standard optimal-control problem.

To fully leverage the decentralized approach, all aspects of the system must be decentralized. For an unmanned ASN, this applies to the data fusion and control algorithms. When interacting with human operators, decentralization must also extend to human-robot interactions.

Human-robot interactions

In any multirobot, multisensor system, human-robot interaction aims

1536-1268/03/$17.00 © 2003 IEEE ■ *Published by the IEEE CS and IEEE ComSoc*

Why Decentralized?

Compared to a centralized or a distributed system, a decentralized system has these constraints:[1]

- No single central information fusion or coordination center; no node should be central to the network's successful operation.
- No common communication facility; nodes can't broadcast results, and communication must be kept strictly node-to-node. Although, technically, a common communication facility violates a decentralized system's constraints, a broadcast medium is often a good model of real communication networks. See Figure A for an example of a communication network implementation.
- Sensor nodes don't have global knowledge of sensor network topology; nodes should know only about connections in their own immediate neighborhood.

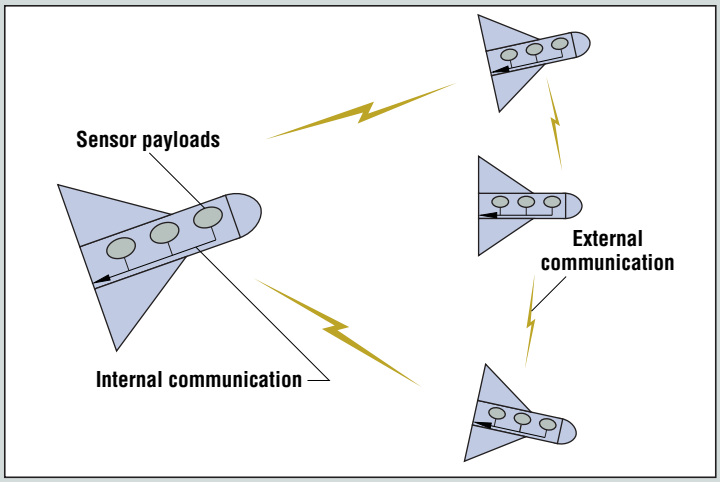


Figure A. A decentralized data fusion system implemented with a hybrid, broadcast, and point-to-point communication architecture.

By explicitly stating that no resource can be centralized, decentralized systems largely avoid problems associated with resource sharing. Conflicts do arise as a result of concurrent decision-making. Ben Grocholsky proposed one possible mechanism for resolving these conflicts.[2] It uses special negotiation channels maintained between decentralized controllers. By exchanging individual expected information gains, the controllers converge to a globally optimal action plan.

To a large extent, decentralization's principles and benefits have been demonstrated experimentally. Stewart Grime and Hugh Durrant-Whyte demonstrated the original ideas of channel filters on a model process control plant comprising over 150 distributed sensors.[3] More recently, Salah Sukkarieh and colleagues experimentally validated an application for tracking point targets on the ground using four custom-built Unmanned Aerial Vehicles.[4] They flew up to three of these aircraft simultaneously during a three-year flight program, with up to six nodes operating concurrently and in real time.

REFERENCES

1. J. Manyika and H.F. Durrant-Whyte, *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*, Ellis Horwood, 1994.
2. B. Grocholsky. *Information-Theoretic Control of Multiple Sensor Platforms,* doctoral dissertation, School of Aeronautical, Mechanical, and Mechatronic Eng., Univ. of Sydney, 2002.
3. S. Grime and H. Durrant-Whyte, "Communication in Decentralized Systems," *IFAC Control Eng. Practice*, vol. 2, no. 2, Feb. 1994, pp. 849–863.
4. S. Sukkarieh et al., "The ANSER Project: Data Fusion across Multiple Uninhabited Air, *Int'l J. Robotics Research*, vol. 22, nos. 7–8, 2001, pp. 505–540.

- To present the user with information about the world that the system collects
- To let human operators influence the actions of many information-gathering robots during their operation

To date, little research has covered the instance of many operators interacting with many robotic platforms. Most work has focused on applying techniques developed for one-to-one human-robot interaction with small teams of humans and robots. Consequently, scalability has remained a secondary issue. (See the "Related Research" sidebar for details.)

We aim to build a system that scales to large numbers, even if it forces us to sacrifice some of the richness of human-robot interactions. Later, we'd like to bring some of the richness back into the picture.

Operator's role in DDF

When designing an ASN, presenting the world picture built by a decentralized system to a human operator poses a challenge. You must do it without making the operator's station a communication or processing bottleneck.

In broad terms, you can query a network for two types of information: about the environment and about network components. We'd like to emphasize that the former doesn't increase in size as the network grows and the latter does.

This leads to our definition of *scalable*

Related Research

We place active sensor networks at the intersection of the converging fields of multirobot systems, sensor networks, and human-robot interactions. Figure B shows this overlap graphically.

Research in interactions between human and multirobot systems (Area II in Figure B) has mostly been limited to communication of one or several humans with a small number of robots. One of the most active research areas is *adjustable autonomy*, which aims to span the gap between teleoperation and full autonomy. Michael Goodrich and his colleagues surveyed adjustable autonomy and mixed-initiative methods for a two-robot system.[1] Terrence Fong and his colleagues developed a system called *collaborative control*, which calls for bidirectional communications through a human-robot dialogue.[2] This system is a form of teleoperation, which compensates for conventional approaches' limitations by integrating human expertise into the control loop. With respect to scalability, this approach suffers from increased communication traffic and the human operator's physical limits to offer assistance to several robots simultaneously.

Kevin Dixon and his colleagues describe a software framework that provides a real and virtual environment for running and managing multiple heterogeneous mobile robot systems called RAVE (*r*eal *a*nd *v*irtual *e*nvironment).[3] They use three GUIs with different authority levels to interact with the system and describe experiments with up to eight robots.

Khaled Ali and his colleagues propose a component-centric method that is scalable.[4] They address the multiagent system's global behavior—either by adding a behavior or by changing the robots' behavioral settings—in the context of reactive robot control. Goodrich and colleagues use a similar approach, by letting the user add goal and threat icons that the system treats as new behaviors.[1]

The work of Ashley Tews and his colleagues is the only one we've seen that directly addresses scalability in a system composed of hundreds of entities.[5] Here, scalability refers to the communication bandwidth between humans and robots that must be kept low for large numbers of entities in the system. They propose that the amount of communication increases with decreasing robot autonomy, tighter human-robot coupling, and the number of robots. They suggest that a large-scale interaction mechanism must allow for many-to-many, one-to-many, and one-to-one interactions. Additionally, it must allow heterogeneous robot teams and different levels of human-robot coupling.

Usually, a human operator is not considered in the context of passive sensor networks (Hairong Qi and her colleagues give a good review[6]). Wolfgang Rencken and his colleagues describe a decentralized data fusion system of four cameras tracking moving targets.[7] This work explicitly considers a human operator, so it falls into Area III. The system provides the human operator with a global view of the network state. An adaptive task allocation algorithm queues targets that the network or operator must track or classify. Decentralized control doesn't factor into this work.

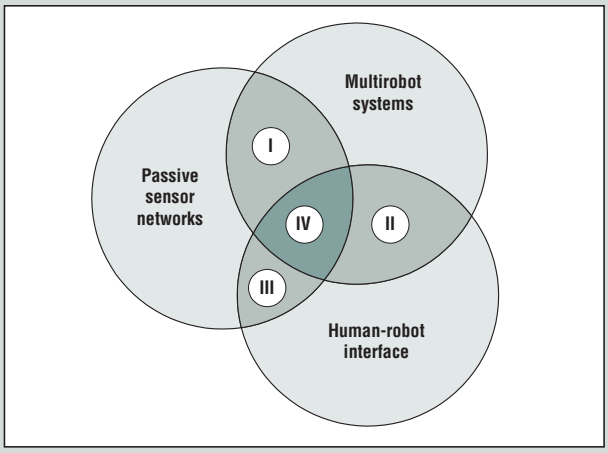


Figure B. Convergence of the traditional field of multirobot systems with the fields of passive sensor networks and human-robot interactions. At the intersection lie (I) active sensor networks and human interface with (II) multirobot systems, (III) sensor networks, and (IV) ASN.

Intelligent spaces is another active research area that often falls into Area IV. Joo-Ho Lee and Hideki Hashimoto describe goals and scenarios very similar to ours but don't address scalability and the precise mechanism of data fusion.[8]

REFERENCES

1. M.A. Goodrich et al., "Experiments in Adjustable Autonomy," *Proc. Int'l Joint Conf. Artificial Intelligence Workshop Autonomy, Delegation, and Control* (IJCAI 01), AAAI Press, 2001.
2. T. Fong and C. Thorpe, "Robot as Partner: Vehicle Teleoperation with Collaborative Control," *Multi-Robot Systems: From Swarms to Intelligent Automata*, Kluwer, 2002, pp. 195–202.
3. K. Dixon et al., "Rave: A Real and Virtual Environment for Multiple Mobile Robot Systems," *Proc. IEE/RJS Int'l Conf. Robotics and Systems* (IROS 99), 1999.
4. K.S. Ali and R.C. Arkin, "Multiagent Teleautonomous Behavioral Control," *Machine Intelligence and Robotic Control*, vol. 1, no. 2, 2000, pp. 3–10.
5. A.D. Tews, M.J. Mataric, and G.S. Sukhatme, "A Scalable Approach to Human-Robot Interaction," *Proc. IEEE Int'l Conf. Robotics and Automation* (ICRA 03), IEEE Press, 2003, pp. 1665–1670.
6. H. Qi, S.S. Iyengar, and K. Chakrabarty, "Distributed Sensor Fusion: A Review of Recent Research," *J. Franklin Inst.*, vol. 338, no. 6, Sept. 2001, pp. 655–668.
7. W.D. Rencken and H. Durrant-Whyte, "A Quantitative Model for Adaptive Task Allocation in Human-Computer Interfaces," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 23, no. 4, Aug. 1993, pp. 1072–1090.
8. J.-H. Lee and H. Hashimoto, "Intelligent Space: Its Concept and Contents, *Adv. Robotics J.*, vol. 16, no. 3, 2002, pp. 265–280.

as applied to a sensor network. We call an architecture scalable with the network's size if the computation and communication requirements don't increase as you add new components to the network. So, if we can map an environment with 10 platforms, the algorithms will let us do it with 1,000—the extra platforms improving the speed and robustness.

There's no denying that the amount of information depends on the environment's size and type. If we want to observe and map a large, complex, and dynamically changing world, we must have facilities that can deal with the information flow. Even though the ASN architecture doesn't address these issues directly, we can take some steps to reduce the burden. For example, we could use data compression, decrease spatial resolution, or reduce the internodal update rate.

Environment information

Any ASN's primary goal is to collect information about the environment, fuse it into a consistent belief, and take actions on the basis of this belief. The graphical user-interface software presents this belief to the human operator. For instance, in a feature-tracking application, the GUI would display the location of known features with their corresponding uncertainty. In a Certainty Grid representation, the GUI would draw the spatial distribution of the variable of interest—for example, the terrain's traversability.

The underlying DDF architecture makes this problem easy to solve. The DDF algorithm ensures that each node in a DDF network has the same complete knowledge of the world. Consequently, a GUI in a DDF network needs to query only a single DDF node to obtain the entire network's knowledge.

The communication and computational load of any node is independent of the number of network components. We can say the same about the GUI interface, which obtains its information from a single DDF node. We easily gain this important property because of the DDF architecture's special structure.

Component information

The amount of information about the nodes—such as their physical location in space—grows proportionally to the number of nodes in the network. So, a GUI that collects and relies on state information from network components can't scale to arbitrary size.

Information about network components is always useful and often invaluable for debugging and status monitoring. For example, system maintainers might want to know a platform's current speed or fuel level, or a sensor's status. They might also want the ability to change the components' configuration during runtime. However, relying on this information makes the GUI a computational or communication bottleneck.

In addition to increasing bandwidth requirements, keeping track of component information also necessitates either using a broadcast medium or a direct link between every component and every GUI station. However, both options go against the philosophy of decentralization.

Other examples of component-specific information include current or future actions (plans), component models, individual sensor observations, and decision policies.

Operator's role in decentralized control

Others have suggested several classification systems for the roles that a human operator could play while interacting with robots. For example, Jean Scholtz identifies *mechanic*, *supervisory*, and *peer-to-peer* levels of human-robot interaction.[2] Alternatively, an operator could tell a robot or a group of robots

- What to do
- What operating mode to switch to
- What outcomes to favor
- What additional information about the world to use

Technically, we can implement all of these options in a scalable fashion provided that we use attribute-based naming and have the proper communication infrastructure.[3] A tree network that the DDF algorithm creates and maintains also works for transmitting such command messages.

As before, we group the four options into two broad categories: component- and environment-centric. Consider, for example, the following scenario: a fleet of Unmanned Aerial Vehicles on an information-gathering mission. An operator receives a phone call from an observer on the ground reporting the location of a thunderstorm cloud. The operator's goal is to modify the UAVs' actions to reduce the risk of losing aircraft because of the storm.

Component-centric control

With component-centric control, we have various types of control options. Under *direct* control, the operator sends an action or a sequence of actions through the network addressed to components that possess the specified attributes. These commands execute directly and don't require autonomy on the part of the platform beyond the ability to execute the tasks. This approach directly extends the teleoperation or mechanic mode of operation. It can be invaluable in certain situations, but we can't consider it a primary control tool in large decentralized sensor networks.

In the thunderstorm example, an operator queries all UAVs for their locations, assumes direct control of the aircraft in most immediate danger, and flies it back to the base. Drawbacks to this approach include both the amount of state infor-

mation to be transmitted and the human operator's physical limitations in performing teleoperation. Both points violate a scalable system's objectives.

Under *mode switch* control, the network's components operate autonomously most of the time but are periodically commanded to switch from one preprogrammed action policy to another. This control method falls under the supervisory mode.

In our example, the operator sends a command to change the action policy from "observe" to "go home" mode. The operator can make this command unaddressed or attribute-addressed, for example, "if close to a certain location."

Under *payoff change* control, the operator sends a change in the utility function to a subset of the network components addressed by appropriate attributes. This is also a supervisory control mode.

In the example, the operator sends out a broadcast or attribute-addressed message to decrease the reward for information gain and increase the payoff function's risk aversion property. This, of course, implies that the UAVs can recognize a dangerous situation (a thunderstorm cloud).

As we've seen, for any component-centric method, the operator workload will increase with the network's size. We can mitigate this effect somewhat with appropriate use of attribute-based naming. Its use is limited, however, to the supervisory level of interaction, and we particularly can't apply it to teleoperation.

Environment-centric control

Instead of component-centric control, we suggest using what we call *environment-centric control*. In this mode, a human operator enters information about the environment that has become available from outside of the network. It's then fused into the decentralized belief about the world's state and will affect network components' actions. In this peer-to-peer mode, the operators and network components act on the same level of authority.

You can enter the information into the network either as a raw observation that a node will use to update the affected states or, more directly, as a posterior of a particular state. In the first case, the operator acts as a sensor submitting information to a node. In the second, the operator plays the role of another node sending feature updates through the DDF link.

In practice, a subtle difference exists between these forms of information input. Typically, you treat a single observation from a sensor with more caution and perform sanity checks—such as waiting for repeated observations, if practical. It's more difficult to verify an update coming from another node because the sources of information at this point are completely anonymous.

In the UAV example, the operator can enter that a thunderstorm has developed at the specified location with a certain probability. The operator could also enter the raw observation of a cloud and let the network fuse this information with its current weather estimate. This, of course, requires even more autonomy, such as a weather model, from the platform.

Influencing the network components' actions by providing them with additional information is a natural extension of the underlying ASN architecture. You need only to transmit the information the operator enters to a single node, and the network propagates it through the rest of the system, potentially leading to actions by far-removed components that can react to the change.

Human networks

As long as human operators play the roles of decentralized sensors or nodes, there's no restriction on the number of operators simultaneously connected to the network. However, the rules used to construct the DDF network, intended to prevent information double-counting, also apply to human operators.

In sensor networks, we see encounter double-counting most readily if a loop in the information flow is formed. In decision theory, this effect is called, not surprisingly, *rumor propagation*. A way to avoid this situation is by adhering to a tree network topology.

When faced with a team of human operators working in close contact with each other, these rules could prove harder to implement and enforce.

Our long-term project objective is to cooperatively build and maintain a multiattribute map of a dynamic environment.

Network implementation

To demonstrate the decentralized approach's capabilities for sensor networks, we wanted to design a sufficiently general and flexible system to let us build various systems with minimal modifications. The architecture had to support an ASN's main aspects: data fusion, control, and human interaction. (More details are available elsewhere.[4])

We identified the four fundamental types of decentralized objects:

- *Frame:* A physical object placed in the physical world. It is the *only* physical object, so to have a location in the world, all other objects must be attached to a frame.
- *Node:* An abstract object that processes information and communicates with other nodes to reach a consistent

Figure 1. Elements of a practical decentralized ASN: (a) an indoor robot Pioneer 2 with a laser range finder mounted on top; (b) a stationary sensor unit also with a laser range finder; (c) a mobile GUI access point running on an iPAQ handheld computer; and (d) a stationary GUI access point on a Linux workstation.

belief about the world.

- *Sensor:* An abstract object that gathers information from the environment.
- *Controller:* An abstract object that maximizes a certain reward function on the basis of the current state and the known model of the world.

You can connect a sensor to a frame, which means that it's physically attached to it with a known offset. You can connect a node to a frame, meaning that the processor located on the frame executes its computations. A sensor can be connected to a node, which means it sends its observations to that node. When a node is connected to another node, they establish and maintain a DDF link, which synchronizes their belief about the world's state. Finally, a controller can connect to a frame, node, and sensor. So, based on the frame's current pose, the sensor and actuator models, and the current world state supplied by the node, the controller will issue commands to the actuator that will maximize a certain reward function.[1]

You can attach any number of nodes, sensors, and actuators to a single frame, but each node, sensor, and controller is assigned to one and only one frame. Likewise, an arbitrary number of sensors may contribute their observations to a particular node, but any particular sensor sends its observations only to a single node. The current implementation is limited to a maximum of four DDF links per node. This number suffices for creating an arbitrary tree network and prevents the forming of bottlenecks at highly connected nodes.

We use the term *platform* to describe the combination of a frame and all other components attached to it. A platform is characterized by stable offsets of sensors (relative to the frame's motion) and low communication costs (relative to the interframe communication's cost).

Our object definitions are by no means unique, but we've found the level of abstraction and granularity sufficient for the type of networks we've set out to build. The modular design allows plug-and-play operation, which speeds up design and setup. Once fully implemented, this architecture will allow, for example, plug-and-play sensor operation. A sensor component designed to convert laser scans into point features will continue working unmodified when the physical sensor is moved from an indoor robot to an outdoor all-terrain vehicle, or even a UAV. The localization service, provided by the frame component, abstracts away localization and locomotion details. Furthermore, the control module designed to guide an indoor robot with a laser sensor will also control a UAV with a video camera if the world representation is the same. A set of standard interfaces between software objects hides the underlying differences.

Figures 1a–d show several physical components that constitute the indoor ASN. In experiments, we've used up to three indoor Pioneer 2 robots and one stationary sensor unit. So far, we've exclusively used SICK laser range finders as sensors. It gives a planar range scan with an angle of 180 degrees and a range artificially limited to 2 meters. We're developing a vision sensor unit.

We installed the ASN in an office space, approximately 20 meters on each side. Our long-term project objective is to cooperatively build and maintain a multiattribute map of a dynamic environment. We'll be mapping distributed properties of the environment—for example, traversability, ambient light, and network connection quality—as well as discrete environmental features, both stationary and moving.

Presently, several poles placed around the office serve as stationary features.

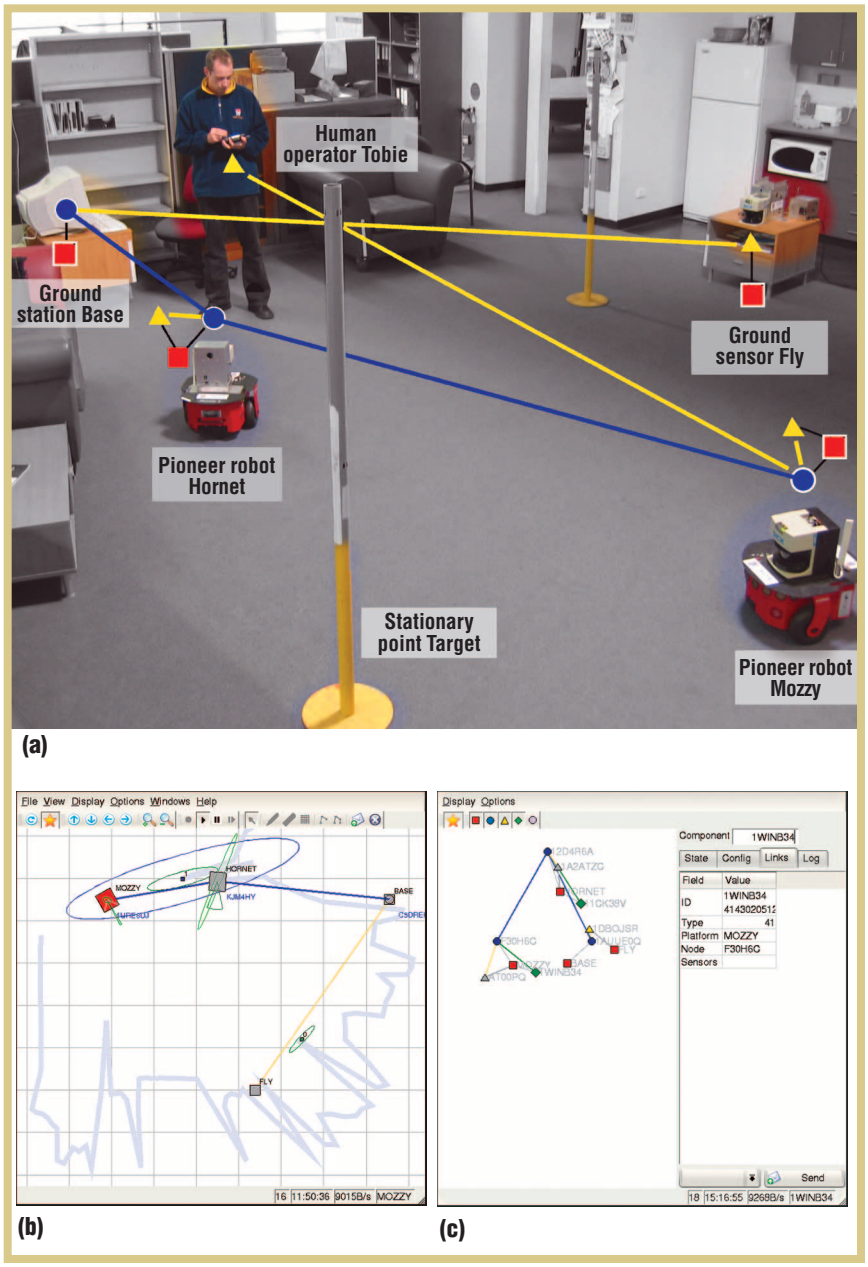


Figure 2. Three views of the same active sensor network: (a) physical components in the office environment, information links are shown as lines; (b) world view in the GUI; and (c) network topology in the GUI. The network view shows several ASN components: frames (□), nodes (○), sensors (Δ), and controllers (◊), with corresponding links. The world view shows mobile platforms connected by decentralized data fusion links. The views also show point features known to the network with corresponding covariance ellipses.

The laser range finder can easily identify them as point features. The network aims to find and localize the point features. We implemented an *information-surfing* controller on the Pioneer robots.[1] The controller tries to minimize the total amount of uncertainty in all features of the environment. It uses a zero look-ahead control law that moves the platform in the direction of the steepest ascent in the information space.

All software components run as separate applications and communicate using a common communication library developed in our laboratory. We support communication over different mechanisms: on the same processor and networked, both wireless and connected by a bus. The components use a simple dynamic-configuration protocol to join the network on start-up. We can perform component configuration either at start-up or at any time during the operation through the GUI.

The GUI in DDF

Our GUI's first goal is to provide a human controller with a view of the environment as the ASN perceives it. We'd also like, to the extent possible, to show the network's state.

Figure 2 compares the view of the actual physical space with two views available in the experimental GUI: the world and network views. We implemented two types of network access points for the operator: a stationary GUI on a desktop computer and a mobile one running on an iPaq handheld computer. All screen shots come from the desktop version. The two GUIs' functionality is similar, but the information's presentation differs because of the iPaq's limited screen size and the peculiarities of pen-based input.

The scenario involves four platforms: two Pioneer robots, Mozzy and Hornet; a stationary sensor module, Fly; and a ground station, Base (see Figure 2a). We equipped each of the two Pioneers with a laser range finder. Each runs four ASN software components: a Pioneer frame (□),Gaussian point node (○), laser-to-point sensor (Δ), and an information-surfing controller (◊). The stationary sensor unit has the same laser sensor, but it can't move so it doesn't need a controller and its frame module is much simpler. To demonstrate the network's flexibility, we didn't run the software on Fly, presumably because its

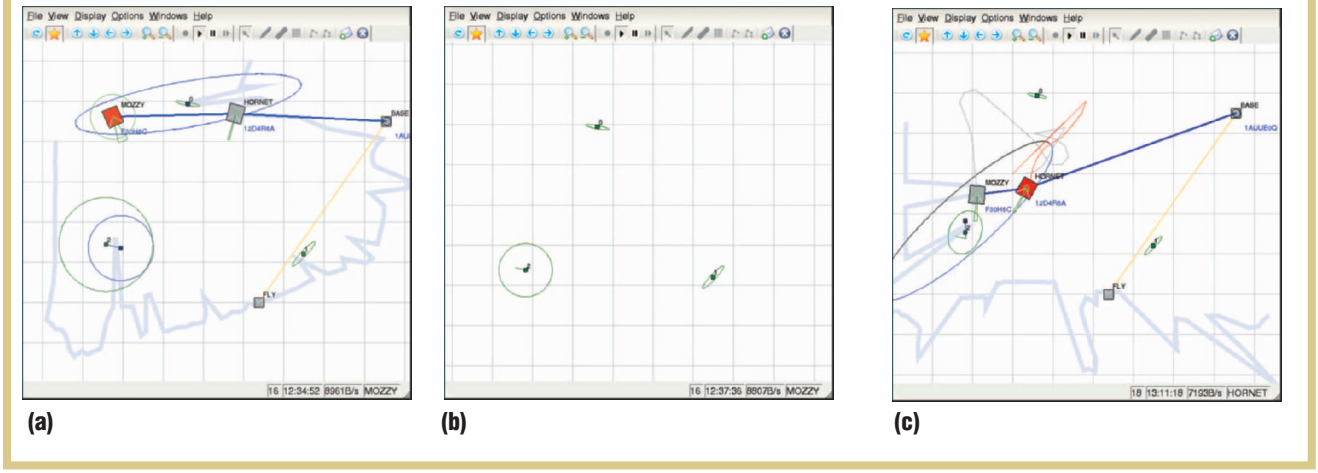


Figure 3. The human decentralized data fusion node in action. The operator enters a feature unknown to the sensor network: (a) the platforms are superimposed on top of the world map; (b) only environment information is shown (this GUI mode is truly scalable); (c) the two active nodes respond to the insertion of the new feature. They converge onto the feature and make observations.

processing power was insufficient. The sensor module on Fly failed to find a local node and connected to a remote one. The remote node happened to be running on the ground station, which has no local sensors of its own.

In this example, a set of point features represents the environment. The world view (see Figure 2b) shows the two known features and their position uncertainty with relatively small ellipses. The larger ellipse centered on Feature 1 represents the uncertainty of the recent observation made by a sensor mounted on Mozzy. The difference in the size of the ellipses is due to the information accumulation inherent in the data fusion process.

In one respect, a GUI module differs from other network components in that it can accumulate nonlocal information about the network, such as the global topology shown in Figure 2c. We use this information purely for visualization and debugging purposes, so it doesn't undermine the overall decentralized approach. If the communication bandwidth limit is reached due to the network's size, we can configure the network components to stop sending their state updates to the GUI. This doesn't affect the information about the environment's state.

Human DDF node

The GUI's second purpose is to let a human operator influence the outcome of actions of active components within the ASN. In this demonstration, we concentrated on the environment-centric control approaches.

Figure 3 shows manual input of feature information through the GUI. The initial configuration is the same as in Figure 2. Figures 3a and 3b shows the complete world view and the world view without component information. In the second case, the state information from the components is neither transmitted to nor stored by the GUI. This makes the approach much more scalable at the cost of decreased information at the operator's disposal.

The ASN has information about two features (numbered 0 and 1) displayed with their uncertainty ellipses. The human operator identifies the likely location of a feature currently unknown to the network, which lies outside the sensor range of all platforms (Feature 2).

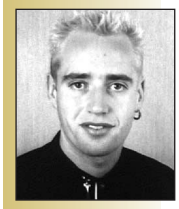
The operator draws an uncertainty circle centered on the new feature's most probable location, and the software interprets the circle's radius to be equal to three standard deviations of its position uncertainty. This circle's properties are sent to the GUI node as a sensor observation message, which is indistinguishable from messages submitted by regular (robotic) sensors. We can send this observation to any one (but only one) node. In our implementation, the operator can select the platform to whose node the observation will be sent. After initializing the new feature, **the node connected to the GUI** informs all other nodes about its existence.

In this example, the operator sends two consecutive observation messages. The second is more certain (the circle is tighter), and the mean lies to the original observation's right (see Figure 3a). The node associates the second observation with an existing feature and fuses them. This behavior is identical to what happens when observations arrive from multiple sensors. The fact that they're entered by an operator (or several operators) makes no difference.

Figure 3c displays the two mobile platforms' response to the human-entered feature information. Guided by an information-surfing controller, the platforms converge onto it. Note that the feature's location has moved and its uncertainty has decreased due to the actual observations made by sensors mounted on the robots.

the AUTHORS

Alexei A. Makarenko is a PhD student in robotics at the University of Sydney, Australia. His research focuses on decentralized architectures for active sensor networks. He received an MS in aeronautics and astronautics from the Massachusetts Institute of Technology. He is a student member of the IEEE. Contact him at the Australian Ctr. for Field Robotics, Bldg. J04, Univ. of Sydney, Sydney, NSW 2006, Australia; alexei@cas.edu.au



Tobias Kaupp is a MSc student at the University of Applied Sciences, Weingarten, Germany. He's doing research at the Australian Centre for Field Robotics in human-robot interaction to complete his master's degree. Contact him at the Australian Ctr. for Field Robotics, Bldg. J04, Univ. of Sydney, Sydney, NSW 2006, Australia; t.kaupp@acfr.usyd.edu.au.

Hugh F. Durrant-Whyte is a professor of mechatronic engineering with the Department of Mechanical and Mechatronic Engineering at the University of Sydney, Australia, where he leads the ARC Centre of Excellence in Autonomous Systems. His research focuses on automation in cargo handling, surface and underground mining, defense, unmanned flight vehicles, and autonomous subsea vehicles. He has a PhD in systems engineering from the University of Pennsylvania. He is a member of the IEEE. Contact him at the Australian Ctr. for Field Robotics, Bldg. J04, Univ. of Sydney, Sydney, NSW 2006, Australia; hugh@cas.edu.au.

To truly demonstrate the approach's scalability, we'd like to increase the hardware network's size. Our short-term goal is to build a robust and flexible indoor sensor network, consisting of up to 50 components similar to the ones described in this article. Current development efforts focus on implementing a robust reconfiguration algorithm that will allow long-term operation of the network—well beyond a single computer's lifetime.

Once we've built a network of certain size, we plan to look into scenarios that involve several human operators simultaneously. An office environment with constant human interaction is a natural setting for such experiments.

The architecture's flexibility and modularity will also let us use it in other environments—such as outdoors—and apply it to other robotic platforms, such as all-terrain vehicles, aircraft, and underwater vehicles. Part of the challenge will be to consistently and efficiently include human input.

Finally, it would be interesting to examine the extent to which we can scale the concepts of adjustable autonomy. P

ACKNOWLEDGMENTS

The ARC Centre of Excellence program, funded by the Australian Research Council and the New South Wales State Government, supported this work.

REFERENCES

1. B. Grocholsky et al., "Scalable Control of Decentralised Sensor Platforms," *Proc. 2nd Int'l Workshop Information Processing in Sensor Networks* (IPSN 03), LNCS 2634, Springer-Verlag, 2003, pp. 96–112.
2. J.C. Scholtz, "Creating Synergistic Cyber Forces," *Multi-Robot Systems: From Swarms to Intelligent Automata*, Kluwer, 2002, pp. 177–184.
3. D. Estrin et al., "Next Century Challenges: Scalable Coordination in Sensor Networks," *Proc. 5th Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking*, ACM Press, 1999, pp. 263–270.
4. A. Makarenko et al., "Building a Decentralized Active Sensor Network," *Proc. IEEE Int'l Conf. Advanced Robotics* (ICAR 03), IEEE Press, 2003, pp. 332–337.

How to Reach Us

Writers
For detailed information on submitting articles, write for our Editorial Guidelines (pervasive@computer.org) or access http://computer.org/pervasive/author.htm.

Letters to the Editor
Send letters to

Shani Murray, Associate Lead Editor
IEEE Pervasive Computing
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
pervasive@computer.org

Please provide an email address or daytime phone number with your letter.

On the Web
Access http://computer.org/pervasive or http://dsonline.computer.org for information about *IEEE Pervasive Computing*.

Subscription Change of Address
Send change-of-address requests for magazine subscriptions to address.change@ieee.org. Be sure to specify *IEEE Pervasive Computing*.

Membership Change of Address
Send change-of-address requests for the membership directory to directory.updates@computer.org.

Missing or Damaged Copies
If you are missing an issue or you received a damaged copy, contact membership@computer.org.

Reprints of Articles
For price information or to order reprints, send email to pervasive@computer.org or fax +1 714 821 4010.

Reprint Permission
To obtain permission to reprint an article, contact William Hagen, IEEE Copyrights and Trademarks Manager, at whagen@ieee.org.

IEEE pervasive COMPUTING
MOBILE AND UBIQUITOUS SYSTEMS