Co-Fields: A Physically Inspired Approach to Motion Coordination

Orchestrating mobile autonomous agents can take inspiration from the laws of physics. Agents' movements could be driven by locally perceived computational force fields, or Co-Fields, generated by the agents themselves and propagated through an embedded infrastructure.

Marco Mamei, Franco Zambonelli, and Letizia Leonardi
Università di Modena e Reggio Emilia

As computing becomes increasingly pervasive, autonomous computers are going to be embedded in everyday objects in our physical environment. In such scenarios, mobility itself will be pervasive. Mobile users, mobile devices, computer-enabled vehicles, and mobile software components will define a dynamic, networked world in which a large set of autonomous components will interact with each other to orchestrate their activities.

Our work focuses on the problem of coordinating autonomous agents' movements in a distributed environment. The term *agent* can refer not only to software components but also to autonomous real-world entities with computing and networking capability, such as PDAs, robots, or modern cars. The goals of these coordinated movements can include letting the agents meet in a particular place, distribute themselves according to specific spatial patterns, or simply move in the environment without interfering with each other. This level of coordination requires some sort of context awareness on the part of the agents. An agent can coordinate with other agents only if it's somehow aware of its context. Given these considerations, our research focuses on the problem of dynamically providing agents with simple and effective contextual information to support their coordinated activities.

To achieve this goal, we take inspiration from the physical world—that is, from the way particles in our universe move and globally self-organize according to the contextual information represented by *fields*. In our approach, we express contextual information in the form of distributed computational fields called *Co-Fields*. Agents can generate specific fields that convey application-specific information about the local environment or about themselves; they can also perceive these fields and move accordingly, following the fields' gradient. The result can be globally coordinated and adaptive movement achieved with little agent effort.

Scenario and motivation

To illustrate the motion coordination problem and motivate the inadequacy of current approaches, let's focus on a representative case study. We focus on supporting tourists in planning their movements across a large and unfamiliar museum and in coordinating such movements with other, unknown tourists. Such activities might include scheduling attendance at specific exhibitions at specific times, having a group of students split up in the museum according to teacher-specific rules, helping a tourist avoid crowds, letting a group of tourists meet at a suitable location, and even helping tourists find emergency exits if necessary.

In our test cases, we assume that tourists have a software agent running on a wireless handheld device, such as a palm computer or cellular phone.

Published by the IEEE CS and IEEE ComSoc ■ 1536-1268/04/$20.00 © 2004 IEEE

The software agent gives the tourist suggestions on how and where to move. We also assume that the museum has an adequate computer network. Embedded in the museum walls is a network of computer hosts, each capable of communicating with each other and with the mobile devices located in its proximity via the use of a short-range wireless link.

The number of embedded hosts and the network topology will depend on the museum, but the basic requirement is that the embedded network topology mimics the topology of the museum plan (no network links between physical barriers such as walls). We also assume for the purpose of the case study that the devices have a localization mechanism to find out where they are actually located in the museum. This localization mechanism could be implemented by some kind of GPS-like device or with less expensive local hardware that relies on the properties of triangulating radio or acoustic signals.[1]

This scenario and the associated motion-coordination problems are of a very general nature, but the case study has many parallels to other scenarios, such as urban traffic management, forklifts activity in a warehouse (where equipped vehicles suggest to their drivers where to go), or software agents exploring the Web (where mobile software agents coordinate distributed researches by moving on various Web sites). The considerations based on our case study therefore have general applicability.

Current approaches

In the last few years, many researchers have proposed middleware and coordination models to address the problem of coordination and interaction in multiagent systems (see the "Related Work" sidebar for more information on additional projects). Systems such as Jini, Universal Plug and Play (UPnP), and Foundation for Intelligent Physical Agents (FIPA) specification-based agent systems[2] are examples of middleware infrastructures rooted in a direct-communication model.

That is, their components communicate with each other directly and explicitly. The problem with this approach is that it doesn't support context acquisition. Each agent must become context aware by discovering and explicitly interacting with the other entities in the environment. For instance, in the case study, an agent would have to explicitly retrieve and communicate with some services providing the museum map, discover which other tourists are currently populating the museum, and explicitly negotiate with them to agree on a specific motion coordination policy. From a software engineering perspective, this imposes a notable burden in the agent code.

This case study has many parallels to other scenarios, such as urban traffic management, forklifts activity in a warehouse, or software agents exploring the Web.

Models based on shared data spaces support interagent interactions through shared, localized data structures. These data structures can be hosted in some data space, as in JavaSpaces,[3] or they can be carried on by agents themselves and dynamically merged.[4] In these cases, agents no longer operate in a totally void space but live in an environment that can be modeled and described in terms of the information stored in the data spaces. Such information, typically referring to local conditions, can provide some sort of context awareness to agents without forcing them to communicate with each other directly.

For instance, in the case study, we can assume that each room and corridor has a local data space storing both local map information and messages left by the other agents about their presence and possibly about their intended next position. Still, to enforce a specific motion-coordination pattern, agents might have to access several data spaces to gather all the required information for building an internal representation of the current situation and then deciding the next movements. In other words, contextual information can express only raw local data; it's still up to the agents to understand and exploit it to achieve specific motion-coordination tasks.

Event-based models relying on publish-subscribe mechanisms make agents interact with each other by generating events and by reacting to events of interest without having them interact explicitly with each other. Typical infrastructures rooted in this model include Jini Distributed Events and UPnP General Event Notification Architecture. Without doubt, event-based models promote stronger context awareness because components can be considered embedded in an active environment capable of notifying them about what's happening.

For instance, in the case study, a possible use of this approach would be to let each agent transmit its movements across the museum, update its internal representation of other agents' movements, and then move properly by continuously adapting its plans according to newly received events. However, from a software engineering perspective, the information that events convey tends to

Related Work

Several projects in the last few years have worked to facilitate distributed-motion coordination. In robotics, the idea of potential fields driving robotic movement is not new.[1] For instance, one of the most recent manifestations of this idea, the Electric Field Approach,[2] was used to control a team of Sony Aibo legged robots in the RoboCup domain. Following the EFA approach, each Aibo robot builds a field-based representation of the environment from the images captured by its head-mounted camera and decides its movements by examining the fields' gradients of this representation.

Although close in spirit, EFA and Co-Fields differ from the implementation point of view. In Co-Fields, fields are distributed data structures actually spread in the environment; in EFA, fields are just an agent's internal representation of the environment and do not actually exist. Co-Fields require a supporting infrastructure to host field data structures, but they completely avoid the complex algorithms involved in field representation and construction.

Shifting from physical to virtual movements, the popular videogame "The Sims" (http://thesims.ea.com) exploits structures akin to computational fields called "happiness landscapes," which spread in the virtual city to drive the movements of nonplayer characters. For instance, if a character is hungry, it perceives and follows a happiness landscape whose peaks correspond to places where it can find food, such as a refrigerator. After eating, the character will follow a new landscape, depending on its needs. Although sharing the same inspiration, Sims' happiness fields are static and generated only by the environment. In Co-Fields, fields are dynamic and can change over time, and agents themselves can generate fields to promote a stronger self-organization perspective.

The Multilayered Multi Agent Situated System (MMASS) formal model for multiagent coordination[3] represents the environment as a multilayered graph in which agents can spread abstract fields representing different kinds of stimuli. The agents' behavior is then influenced by the stimuli they perceive in their location. In fact, agents can associate reactions to these stimuli, as in an event-based model, with the addition of the location dependency that is associated with events and reactions. The main difference between MMASS and Co-Fields is that, in Co-Fields, agents combine perceived fields and are constantly guided by the fields produced, while in MMASS, fields are considered independent from each other and are exploited only to trigger one-shot reactions. Also, the application domain of MMASS—simulation of artificial societies and social phenomena—is quite different from that of Co-Fields.

An area in which the problem of achieving effective context-awareness and adaptive coordination has been addressed via a field-based approach is amorphous computing.[4] The particles constituting an amorphous computer have the basic capabilities of propagating abstract computational fields in the network and of sensing and reacting to such fields. In particular, particles can transfer an activity state toward directions described by fields' gradients to make coordinated patterns of activities emerge in the system independently of the network's specific structure. This mechanism can be used to drive particles' movements and let the amorphous computer self-assemble in a specific shape.

Researchers have proposed a similar approach to self-assembly in the area of modular robots: having a robot consisting of multiple, flexibly connected components reshape itself dynamically to meet specific purposes.[5] Although serving totally different purposes, both approaches share with Co-Fields the idea of having a single, physically inspired mechanism to diffuse contextual information and organize adaptive motion-coordination patterns.

REFERENCES

1. O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *Int'l J. Robotics Research*, vol. 5, no. 1, 1986, pp. 90–98.
2. S. Johansson and A. Saffiotti, "Using the Electric Field Approach in the RoboCup Domain," *RoboCup 2001*, Springer-Verlag, 2001, pp. 399–404.
3. S. Bandini, S. Manzoni, and C. Simone, "Space Abstractions for Situated Multiagent Systems," *1st Int'l Joint Conf. Autonomous Agents and Multiagent Systems*, ACM Press, 2002, pp. 1183–1190.
4. R. Nagpal, "Programmable Self-Assembly Using Biologically-Inspired Multiagent Control," *1st Int'l Conf. Autonomous Agents and Multiagent Systems*, ACM Press, 2002, pp. 418–425.
5. W. Shen, B. Salemi, and P. Will, "Hormone-Inspired Adaptive Communication and Distributed Control for CONRO Self-Reconfigurable Robots," *IEEE Trans. Robotics and Automation*, vol. 18, no. 5, Oct. 2002, pp. 1–12.

be too low-level for agents to use it effectively, forcing them to catch and analyze a possibly large number of events to achieve a specific motion-coordination pattern. This process is of course a big burden in the agent code, making this kind of middleware unsuitable from a software engineering point of view.

Co-Fields approach

Our Co-Fields proposal aims to provide agents with abstract—simple yet effective—representations of context. Co-Fields delegate to the infrastructure the task of constructing and automatically updating an essential distributed view of the system situation—possibly tailored to application-specific motion-coordination problems—that "tells" agents what to do. The Co-Fields approach consists of a few key concepts:

- Contextual information is represented by computational fields, spread by agents or by the infrastructure, dif-

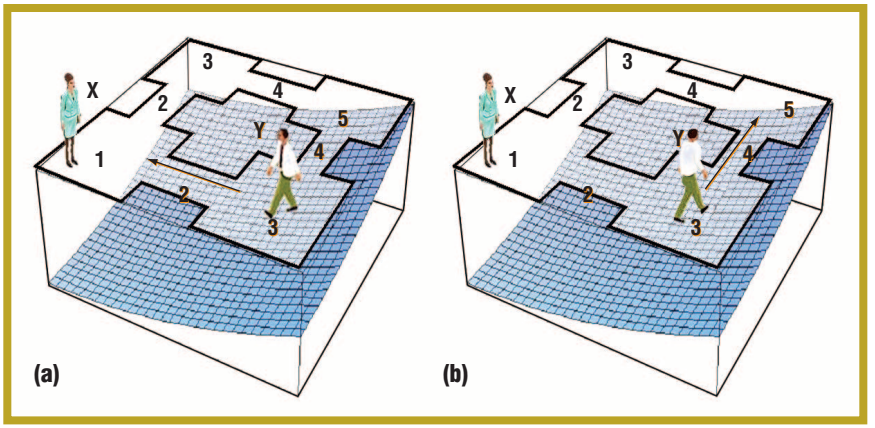


Figure 1. Agent X propagates a presence field whose value has a minimum where Agent X is located. Agent Y senses that presence field to (a) approach Agent X by following the gradient downhill and (b) move farther from Agent X by following the presence field's gradient uphill.

fused in the infrastructure, and locally sensed by agents.
- A motion coordination policy lets agents move following the local shape of these fields, just as a physical mass moves in accordance with a locally sensed gravitational field.
- Both environment dynamics and agent movement might induce changes in the fields' surface, automatically propagated by the infrastructure, thus inducing a feedback cycle that we can exploit to achieve a global motion coordination pattern.

Of course, we could implement Co-Fields using existing coordination models, but this isn't the point. Rather, the point is how we can exploit middleware and coordination models to facilitate application-level management of complex motion-coordination policies.

Computational fields

A computational field is a distributed data structure characterized by a unique identifier, a location-dependent numeric value, and a propagation rule identifying how the field should distribute throughout the network and how its value should change during the distribution. Agents can access fields locally, providing them with a local perspective of the system's global situation.

For instance, the museum guide in our case study could spread in the museum network infrastructure a computational field—let's call it a *presence* field—whose value increases as it gets farther from the guide. Such a field implicitly lets any tourist, from any location in the museum, sense the guide's presence and distance. Also, by sensing the presence field's local gradient, the tourist could know in which direction to find the guide (Figure 1).

Either the agents or the environment can generate fields, and a proper distributed network infrastructure must support them. In the case study, we assume such a network to be part of the museum infrastructure. But in general, you can use any type of network, even ad hoc networks of mobile nodes, to support Co-Fields.[5]

When a specific source node injects a field into the network, it is propagated hop-by-hop through the network as specified by its propagation rule. This rule determines how the field's value should change during propagation. The field surface's final shape is determined by the field's propagation rule and by the network topology, which lets the same field adapt to and be suitable for different museum floor plans.

Motion coordination

In Co-Fields, the principle of enforcing motion coordination means having agents follow the local shape of specific fields. For instance, a tourist looking for a guide can follow the corresponding presence field downhill. Dynamic environment changes and agents' movements induce changes in the fields' surface, producing a feedback cycle that influences agent movement. For instance, if the guide moves around in the museum, the Co-Fields infrastructure automatically updates the corresponding presence field so that any tourist looking for a guide can adapt its movement accordingly. If multiple guides are in the museum, they could decide to sense each other's presence fields to stay as far as possible from each other, thus improving their accessibility to tourists.

In a Co-Fields-based system, agents are balls rolling on a surface; their movements are complex and adaptive not because of the agents' wills but because of the ability to dynamically reshape the surface.[6] Of course, such a physical inspiration and the strictly local perspective in which agents act promote a "greedy" approach in agents' movements: Agents act on the basis of their local viewpoint, disregarding whether a small sacrifice now—climbing a Co-Fields hill instead of descending it—could possibly lead to greater advantages in the future.

In a circular track, for example, a tourist looking for a guide that is moving clockwise, instead of following the presence field downhill (as the Co-Fields approach promotes), could decide to move uphill to meet the guide counterclockwise. However, this is a general drawback of distributed problem solving, where efficiency reasons often rule out the possibility of globally informed decisions by distributed agents.

Application-specific coordination

Achieving an application-specific coordination task rarely relies on evaluating an existing computational field, as in the case of a tourist looking for a guide and simply following its specific field. In most cases, an application-specific task relies on evaluating an application-specific

coordination field, a combination of some of the locally perceived fields.

The coordination field is a new field in itself, and it's built with the goal of encoding in its shape the agent's coordination task. Once a proper coordination field is computed, agents can achieve their coordination task by following their coordination field's shape uphill or downhill, as if they were walking on the field's associated surface. For instance, in the case study, for guides to stay as far as possible from each other, they can follow uphill a coordination field resulting from the combination of all the computational fields of each guide:

$$CF = \sum_{i=1}^{n} presence_i$$

We still haven't found a general methodology to help us identify, given a specific motion pattern to be enforced, which fields have to be defined, how they should be propagated, and how they should be combined in a coordination field. We are confident that some methodology to help in that direction will be found that would possibly make Co-Fields applicable to a wider class of distributed coordination problems, even beyond motion coordination.

Nevertheless, Co-Fields are immediately applicable because of the possibility of getting inspiration from a wide variety of motion patterns found in nature. Phenomena such as diffusion, birds flocking, ants foraging, bees dancing—to mention just a few—all can be easily modeled with Co-Fields, and all have practical application in mobile-computing scenarios.[7]

Phenomena such as birds flocking, ants foraging, and bees dancing can be easily modeled with Co-Fields, and all have practical application in mobile-computing scenarios.

Implementation issues

We can potentially implement Co-Fields on any distributed middleware that provides basic support for data storing, communication, and event notification. These considerations apply both to the case in which Co-Fields rely on a fixed network infrastructure—where a set of Co-Field servers can be made in charge of field propagation and to which mobile devices may connect—as well as to the case of mobile ad hoc networks, where each device acts as a Co-Fields server and propagates fields in an ad hoc way in the wireless media.[5] For both cases, some sort of localization mechanisms must be enforced for mobile devices.[1]

We implemented a prototype of Co-Fields on a fixed network infrastructure exploiting Mobile Agent Reactive Space programmable tuple spaces acting as Co-Fields servers.[8] MARS tuple spaces operate on IP nodes acting as 802.11 wireless access points. Users carrying a Linux IPAQ (pocket PC) with 802.11 wireless cards can access the MARS tuple spaces of the nodes to which they are connected. To enforce the strict locality in connections required by Co-Fields, each tuple space contains a tuple identifying its spatial location. Each IPAQ, in its turn, can dynamically determine its own location. Thus, an IPAQ can select which tuple space to interact with by comparing its own location with those of all the in-range tuple spaces.

The infrastructure's nodes are virtually connected with each other according to a topology resembling the museum floor plan. You could create different topologies to map different floor plans simply by providing each node with only the IP addresses of its virtual neighbors in the intended topology, and have the communication between nodes restricted to those virtual neighbors. To implement field propagation in the virtual topology in a hop-by-hop way, we exploited the fact that we could program MARS tuple spaces to react to any access event with arbitrary computations.

We can program each MARS tuple space to react to the arrival of a tuple such as `(TYPE="Presence", NAME=any, VAL=any)` with a computation that accesses neighbor tuple spaces and inserts the same tuple with the `VAL` field increased by one. The overall result of this process is a field-like distributed data structure spread across the virtual network and having its `VAL` entry reflecting the hop distance from the source. Similar processes apply for other types of fields as well as for dynamically updating the distributed shape of a field upon agents' movements.

In addition to the prototype implementation, we've developed a simulation environment for Co-Fields to test the effectiveness of our approach in large-scale scenarios. We developed the simulation environment using the Swarm toolkit (www.swarm.org) for multiagent system simulations. The Co-Fields simulation let us model the specific museum map, the presence in this environment of Co-Fields servers connected to the museum plan, and the presence of agents each with a specific plan of visit to the museum.

As far as the practical scalability of Co-Fields implementations, there are two considerations. Because each server can store local field values in terms of simple tuples of a few bytes, Co-Fields do not introduce storage scalability problems. We could store a large number of fields locally and

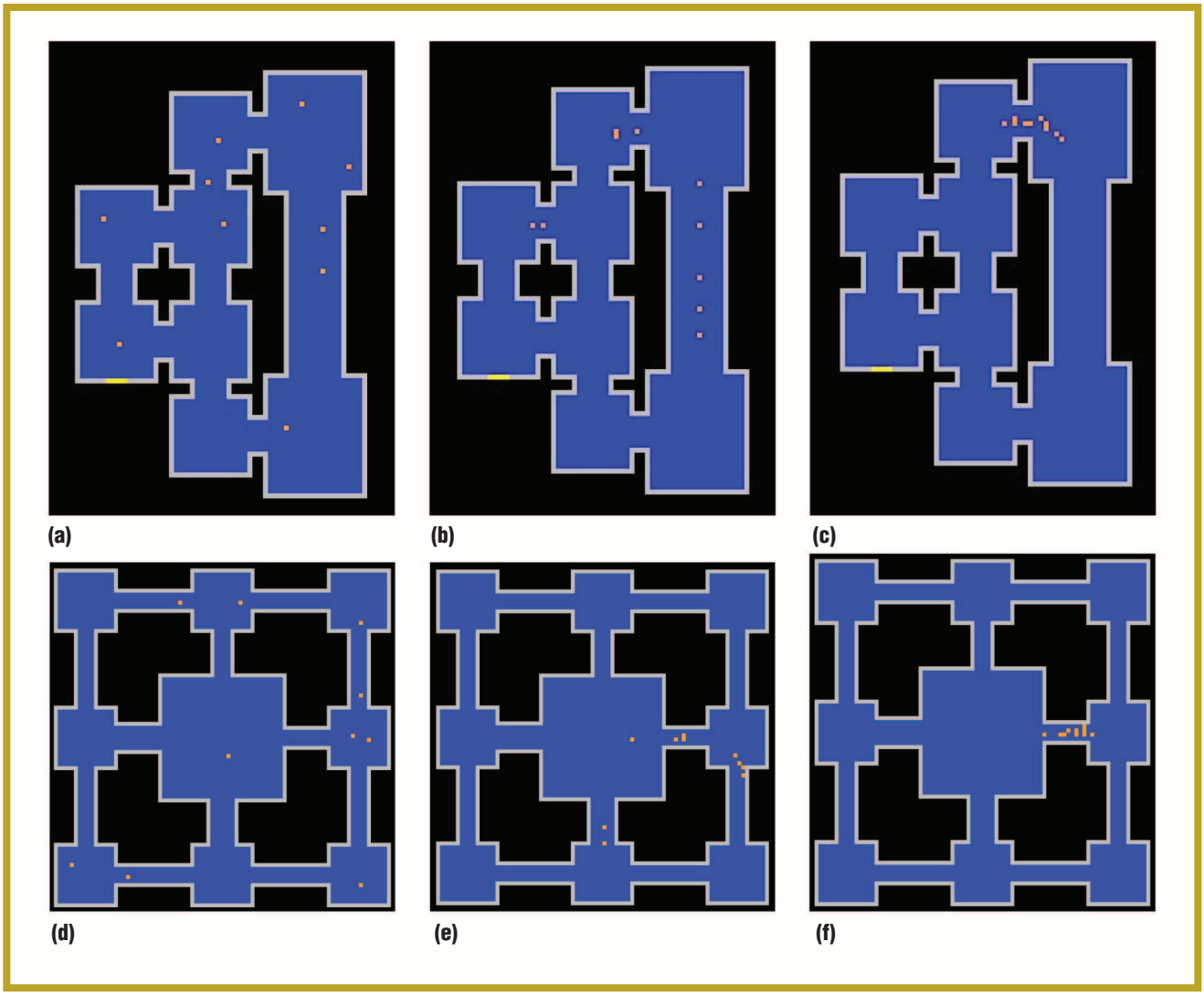


Figure 2. From left to right, three phases (a–c and d–f) of the meeting process in two different simulated museums. Only the tourists in the meeting group are shown.

even on network nodes with low storage capabilities. On the other hand, the speed at which agents move—for example, tourists in the museum—is extremely low compared to the speed at which fields in a network can propagate. Thus, the inevitable delay at which field propagation occurs doesn't affect agents' behavior significantly, even in large systems.

Examples of Co-Fields use

To clarify how to use Co-Fields, we outline some specific motion-coordination problems that museum tourists might face. In these examples, we assume that the museum network's topology mimics the museum map, with an infrastructure node in each room and corridor, and that the infrastructure's nodes can locate the rooms containing tourists. We've already shown how to exploit the presence field to detect a person's location in the museum, but we can also use it to enforce a variety of other interesting motion patterns.

Meeting and surrounding

For the first example, let's consider a meeting service to help a group of tourists or museum guides dynamically meet with each other. This service could be of great use in other scenarios, such as emergency evacuation situations and distributed gaming. Although we could enact various policies to let a group of tourists meet somewhere—a specific point or by a specific tourist—here we concentrate on having a group of tourists collaboratively walk toward each other and eventually meet in some dynamically determined intermediate point (see Figure 2). If each member i of the group generates a *presence*$_i$ field, then each tourist can evaluate its coordination field by taking the maximum presence field of all the other tourists:

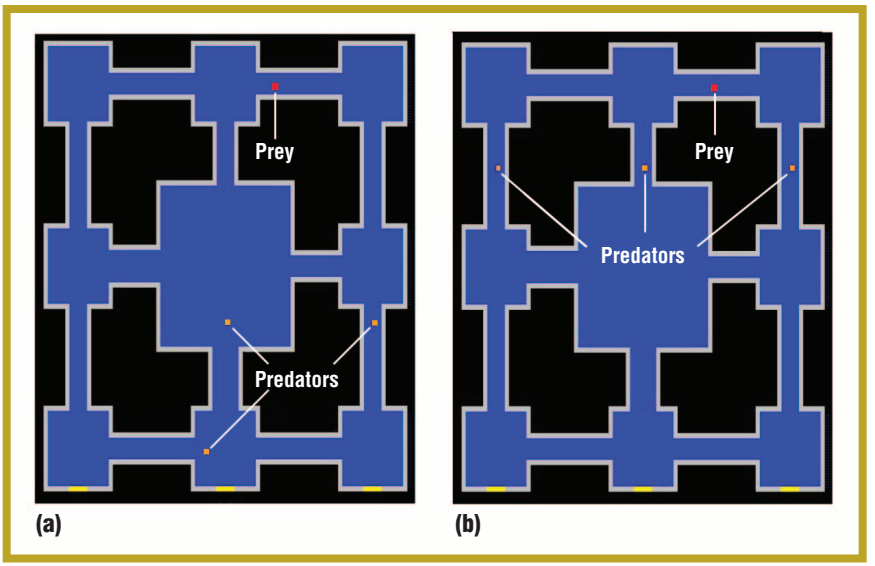


Figure 3. From various starting positions (a) in a simulated museum, three predators move (b) to surround a prey.

$$CF_i = \max(presence_x)$$
$$\text{where } i = 1, \ \ldots \ , n$$

and then following such coordination fields downhill. Although the coordination fields continuously change because of the concurrent movements of all the members of the group, each following its own coordination fields, the members gradually do approach each other until they collapse into a single point. In other words, because agents actually attract each other, the system naturally converges to the situation in which all the agents are at the same point.

Figure 2 shows the result of the Co-Fields meeting process for two different simulations (performed with the already mentioned Swarm environment) on different museum maps. The key point to emphasize is that the meeting process—and Co-Fields in general—is adaptive. It even works well independently of the environment's characteristics, without having to change the agent code or the computational fields' structure or propagation.

As another example of exploiting the presence fields, consider a group of security guards in charge of searching, surrounding, and catching a child that got lost and is wandering in a museum. This is a peculiar instance of the more general problem of a group of predator agents in an environment trying to surround and capture a prey agent—a problem that has several applications in other scenarios, including military actions and collaborative mobile robots.

By getting inspiration from the behavior of wolves, which succeed in collaboratively surrounding prey using the simple strategy of approaching the prey while trying to stay as far as possible from each other, we can define the coordination field of a generic predator i in this way:

$$CF_i^{pred} = presence^{prey} - \sum_{j=1, j\neq i}^{n} presence_j^{pred}$$

The equation expresses the fact that the predators follow the prey's presence field downhill and, at the same time, try to stay far from all other predators. The result (see Figure 3) is that predators, rather than simply approaching the prey, can effectively surround it. Of course, should the prey move much more quickly than the predators, the process might never converge.

Avoiding crowds

Any tourist moving in the museum will want to avoid crowds. Co-Fields could help here. Let's assume that each room in a museum generates a room field whose value increases with the hop-distance from the generating room (see Figure 4a). Room fields are static, do not change over time, and depend only on the room location. Clearly, an agent following a room field downhill will inevitably reach the source room. So for an agent to visit a specific set of rooms in the most efficient way, it simply has to follow the coordination field as a minimum combination *mComb* of all the room fields (where room fields are combined to take, in each room, the one with the minimum value) in its visit schedule:

$$CF = (1-w)mComb(room_1, room_2, \ \ldots \ , room_n)$$

If an agent follows the coordination field downhill, it enters the closest room of its visit schedule. Then, as it completes visiting that room, it can remove the corresponding room field from *mComb* and continue to visit the other rooms. All of this still takes place without considering crowd conditions.

To take into account crowd conditions, the infrastructure nodes could locally generate a *crowd* value to measure the amount of crowding in the museum rooms. You could calculate this amount, for each room, as the total number of local presence fields with value 1, normalized to the room's dimensions to better represent how much a particular room is crowded. The result-

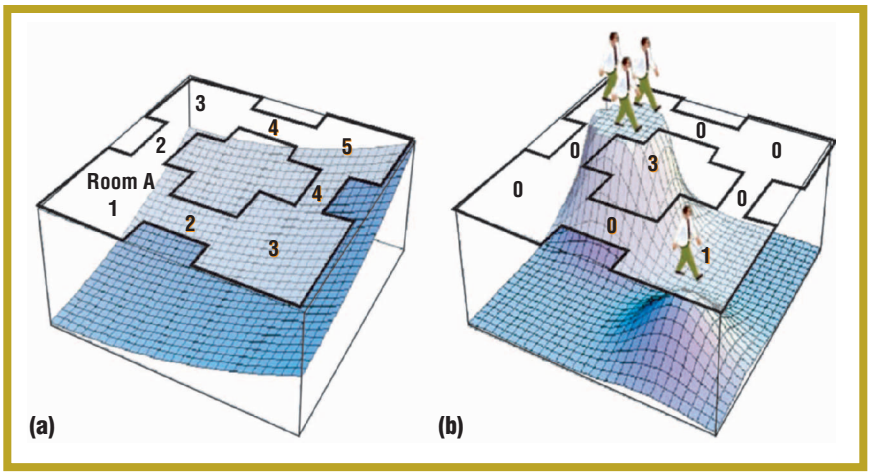


Figure 4. (a) The room field of room A has a minimum in that room and monotonically increases with the distance from the source. (b) The crowd field has peaks where the crowd is most dense.

ing global crowd field's landscape (see Figure 4b) dynamically varies according to agents' movements and expresses the global crowd conditions in the museum. Clearly, if all agents always follow the crowd field downhill, the overall result is a global convergence toward a balance of the crowding conditions in the museum rooms.

If an agent wants to simultaneously visit the next room in its schedule and avoid crowd conditions, it can use the following coordination field:

$$CF = (1-w)mComb(room_1, room_2, \ldots, room_n) + w \times crowd$$

The first term of the coordination field tends to attract an agent toward the rooms in its visit schedule, while the second tends to repel it from crowded areas. We can use the weight w to specify the crowd field's relevance; we assume a tourist can specify it via the help of some user interface. For $w = 0$, the load-balancing mechanism is turned off and agents proceed toward their closest destinations in the greed path, disregarding crowded areas. If w is low, the agent will proceed toward its destination rooms, following its greed most of the time and being diverted to alternative paths only in the case of very crowded conditions.

As w gets higher, the agents would see a greater number of alternative paths, particularly whenever the greed path is a bit crowded. For $w = 1$, agents ignore their visit schedule and simply move to balance the load in a diffusion process. Clearly, coordination fields could be used for load balancing in several other scenarios, such as when a set of agents must move according to their own plans in a crowded environment. Such scenarios could include a forklift in a warehouse or cars in a city.

Testing the validity

To test the validity of our approach, we developed a set of simulations in which we set up agents with a random visit schedule and an average of half the rooms of the museum to visit. Agents move to rooms they want to visit, stay there for a specific amount of time, then walk to the next room in the schedule according to the coordination field. Agents increase their stay in a room and decrease their walking speed proportionally to the local crowd conditions.

Figure 5 shows the results of the experiments on two different museum maps. For both maps, the Co-Fields method for load balancing prevents the emergence of highly crowded zones, which are likely to appear when tourists disregard the Co-Fields. These results show an improvement of the average visit time for tourists. On one hand, for very small values of w, agents give little importance to crowd conditions and aren't able to take full advantage of the Co-Fields. For very high values of w, agents overestimate the importance of crowds and tend to select longer paths to avoid even rooms that aren't very crowded, with an overall worsening of the average visit time.

Comparing the results for the two different maps (5a, 5b, and 5c versus 5d, 5e, and 5f in Figure 5) confirms that the Co-Fields approach is adaptive. However, the actual effectiveness of our approach might depend on the characteristics of the environment itself. Co-Fields don't work for museums in which few alternative paths are available to tourists. The performance improvement in the map of Figure 5a is lower than in the more connected map of Figure 5d. In general, a highly constrained environment necessarily constrains the extent of applicability of any motion-coordination policy.

We can apply the same load-balancing approach even to other motion-coordination problems. For instance, in the meeting problem, we can consider a coordination field that, in addition to evaluating the presence field, also takes into account the crowd field:

$$CF_i(x, y, t) = (1-w)\max(presence_x) + w \times crowd$$

In this way, the agents as they move toward each will also consider the possibility of avoiding crowded rooms.

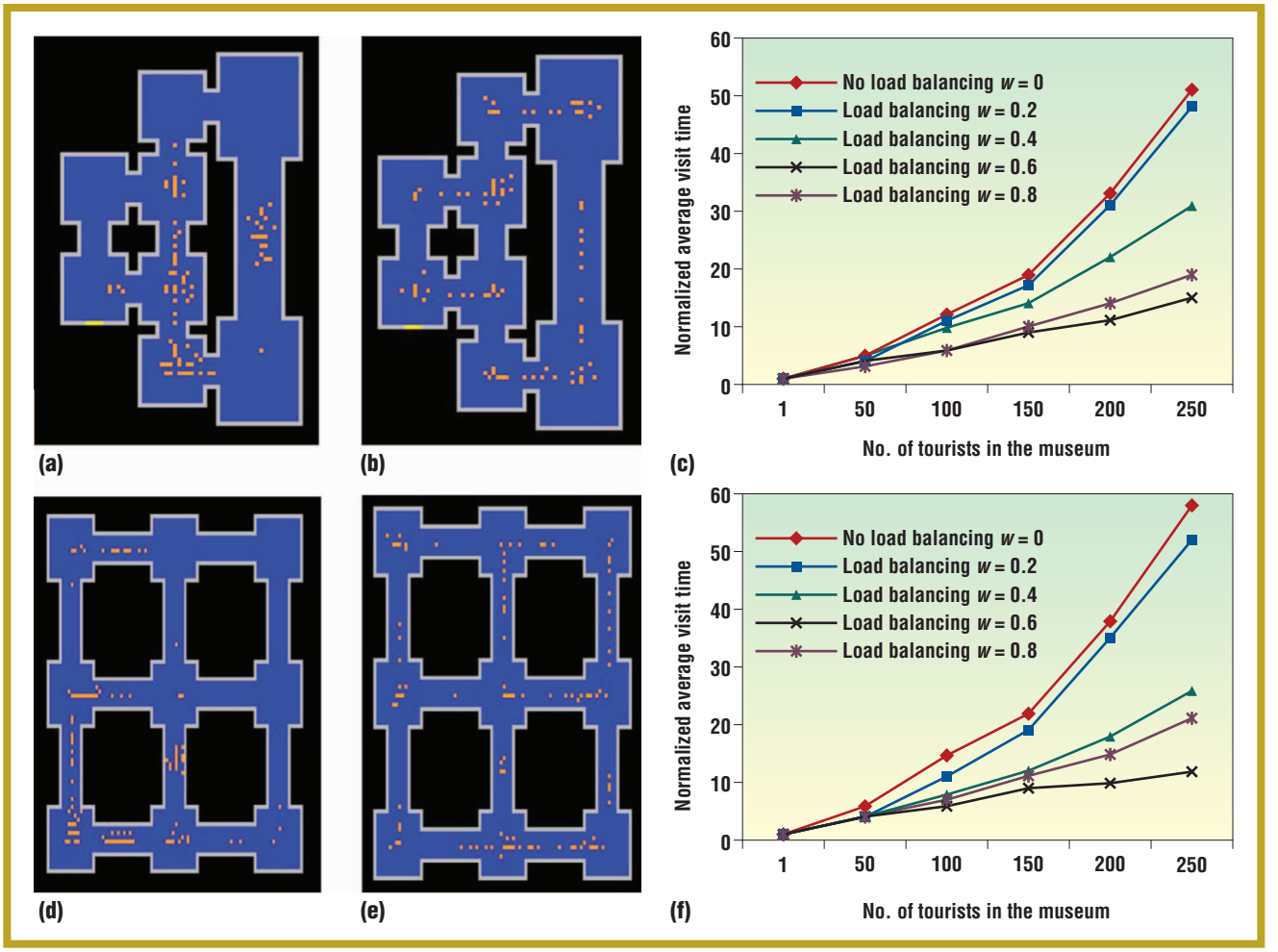


Figure 5. Load balancing in two simulated museums: (a, d) When agents don't exploit Co-Fields load balancing, crowded zones appear. (b, e) Crowded zones are mostly avoided when agents follow a coordination field that takes into account the load-balancing factor. (c, f) Overall, tourists' average visit time (normalized to the average visit time of a single tourist in an empty museum) decreases the more they take into account the load-balancing factor ($w > 0$), until this weight becomes excessive (for example, $w = 0.8$).

A potential problem that might affect load balancing and meeting strategies relates to the possible emergence of spurious local minima not associated with the agents' goals in the coordination fields. In general, whenever agents combine two fields expressing an attracting part (the room fields) and a repelling one (the crowd field) in a coordination field, such spurious local minima might arise. However, the fact that the agent computes the coordination field internally enables any agent to understand whether a minimum is spurious by simply analyzing the single components of the coordination field.

In load balancing, an agent can easily understand whether a minimum it has reached corresponds to a room it wished to visit or is a false minimum created by a peculiar combination of the room and crowd fields (see Figure 6a). In the latter case, the agent, depending on its internal policy, can decide either to ignore the crowd field and proceed (see Figure 6b) or find alternate paths (Figure 6c).

While both a preliminary prototype implementation and the outcomes of our simulation show the feasibility of the approach, we must pursue several research directions to improve its generality and its practical applicability. In addition to the need for general methodologies to help design specific Co-Fields coordination patterns, we plan to explore the potential of Co-Fields to encode general distributed coordination patterns, possibly not related to motion.

ACKNOWLEDGMENTS

This work was supported by Italy's Ministry for Higher Education, Training, and Research and its National Research Council in the "Progetto Strategico IS-MANET: Infrastructures for Mobile Ad-Hoc Networks" project and by Nokia Research Center Boston.

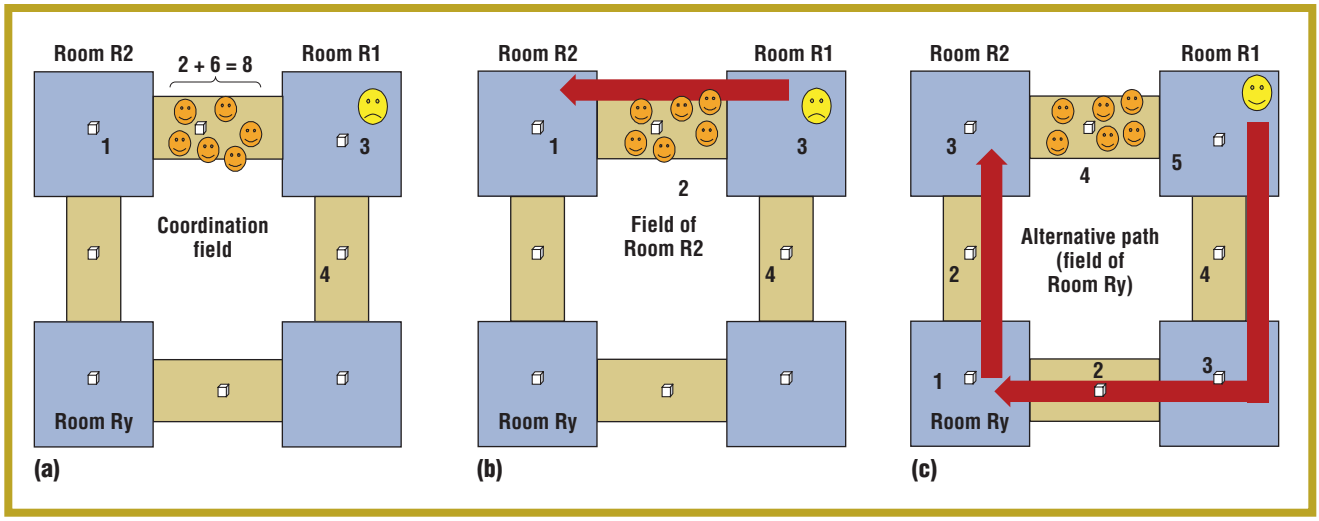


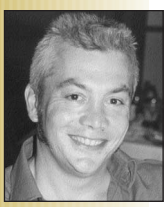
Figure 6. (a) An agent willing to visit Room R2 senses a coordination field having value 3 (R2 *room* field = 3 + *crowd* = 0) in its current location, a value 8 (R2 *room* = 2 + *crowd* = 6) in the corridor to its left, and a value 4 (R2 *room* = 4 + *crowd* = 0) in the corridor leading down. It's trapped in a local minimum. (b) The agents can analyze the components of its coordination field (R2 *room* field only), discover the presence of the local minimum, and decide to proceed, ignoring the *crowd* field, toward R2. (c) Or, it can evaluate alternative paths by looking for *room* fields decreasing both toward R2 and toward another direction (such as the field of Room Ry). If such a field exists, the agent can follow the Ry field and, once Room Ry's minimum has been reached, switch back to the original coordination field.

REFERENCES

1. J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *Computer*, vol. 34, no. 8, Aug. 2001, pp. 57–66.
2. F. Bellifemine, A. Poggi, and G. Rimassa, "JADE: A FIPA2000 Compliant Agent Development Environment," *Proc. 5th Int'l Conf. Autonomous Agents*, ACM Press, 2001, pp. 216–217.
3. E. Freeman, S. Hupfer, and K. Arnold, *JavaSpaces Principles, Patterns, and Practice*, Addison-Wesley, 1999.
4. G.P. Picco, A.L. Murphy, and G.C. Roman, "LIME: A Middleware for Logical and Physical Mobility," *Proc. 21st Int'l Conf. Distributed Computing Systems*, IEEE CS Press, 2001, pp. 524–536.
5. M. Mamei and F. Zambonelli, "Programming Pervasive and Mobile Computing Applications with the TOTA Middleware," *2nd Int'l Conf. Pervasive Computing and Communication*, IEEE CS Press, 2004, pp. 263–277.
6. M. Mamei, F. Zambonelli, and L. Leonardi, *Co-Fields: An Adaptive Approach to Motion Coordination*, tech. report 5-2002, Univ. of Modena and Reggio Emilia, 2002; www.agentgroup.unimo.it.
7. E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence*, Oxford Univ. Press, 1999.
8. G. Cabri, L. Leonardi, and F. Zambonelli, "Engineering Mobile Agent Applications via Context-Dependent Coordination," *IEEE Trans. Software Eng.*, vol. 28, no. 11, Nov. 2002, pp. 1040–1058.

the AUTHORS

Marco Mamei is a doctoral student in computer science at the University of Modena and Reggio Emilia, where he received the Laurea degree in computer science. His current research interests include distributed and pervasive computing, complex and adaptive systems, and multiagent systems. He is a member of the IEEE, AIIA (Associazione Italiana per l'Intelligenza Artificiale), and TABOO (Associazione Italiana Tecnologie Avanzate Basate su concetti Orientati ad Oggetti). Contact him at Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia, Via Allegri 13, Reggio Emilia, Italy; mamei.marco@unimo.it.



Franco Zambonelli is a professor of computer science at the University of Modena and Reggio Emilia. His research interests include distributed and pervasive computing, multiagent systems, and agent-oriented software engineering. He received his PhD in computer science from the University of Bologna. He is a member of the IEEE, the ACM, AIIA, and TABOO. Contact him at Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia, Via Allegri 13, Reggio Emilia, Italy; franco.zambonelli@unimo.it.

Letizia Leonardi is a full professor in the Department Of Information Engineering at the University of Modena and Reggio Emilia. Her research interests include the design and implementation of mobile-agent systems. She received her PhD in computer science from the University of Bologna. She is a member of TABOO and Associazione Italiana per l'Informatica e il Calcolo Automatico. Contact her at Dipartimento di Ingegneria dell'Informazione Università di Modena e Reggio Emilia, Via Vignolese 905, 41100 Modena, Italy; letizia.leonardi@unimo.it.