

Distributed Motion Coordination with Co-Fields: a Case Study in Urban Traffic Management

Marco Mamei^{1,2}, Franco Zambonelli², Letizia Leonardi¹

¹*Dipartimento di Ingegneria dell'Informazione – Università di Modena e Reggio Emilia
Via Vignolese 905 – Modena – ITALY*

²*Dipartimento di Ingegneria dell'Informazione – Università di Modena e Reggio Emilia
Via Allegri 13 – Reggio Emilia – ITALY*

{ mamei.marco, franco.zambonelli, letizia.leonardi }@unimo.it

Abstract

Coordinating the activities of distributed autonomous entities challenges traditional approaches to distributed coordination and calls for new paradigms and supporting middleware. This paper focuses on the problem of orchestrating the movements' of mobile autonomous agents in a large-scale distributed systems, and proposes an approach that takes inspiration from physics. Our idea is to have the movements of agents driven by force fields, generated by the agents themselves and propagated via some infrastructure. A globally coordinated and self-organized behavior in the agent's movements can then emerge due to the interrelated effects of agents following the shape of the fields and dynamic fields re-shaping. The approach is presented and its effectiveness described with regard to a concrete case study in the area of urban traffic coordination.

Keywords: *Autonomous Systems, Mobility, Coordination, Context Awareness, Traffic Management.*

1. Introduction

As computing is becoming pervasive, autonomous computer-based systems are going to be embedded in all our everyday objects and in our physical environment, and they are going to interact with each other in a globally connected network, possibly making use of wireless communication technologies [Est02]. In such a scenario, mobility too, in different forms, will be pervasive [4, 15]. Mobile users, mobile devices, computer-enabled vehicles, as well as mobile software components, define an open and dynamic networked world, in which large set of autonomous components

should be enabled to interact with each other and to orchestrate their activities.

In this paper, we specifically focus on the problem of coordinating the respective movements of a large set of autonomous “agents” in a distributed environment. Here, the term agent can generically refer to any autonomous real-world entity with computing and wireless-communication capability (e.g., a user carrying on a Wi-Fi PDA, a robot, or a modern car). The goals of their coordination can be various: letting agents meet somewhere [4], distribute themselves accordingly to specific spatial patterns [1], or simply move in the environment without interfering with each other and avoiding the emergence of traffic jams [8].

Traditional approaches to distributed coordination usually exploit context-unaware agents, i.e., agents that are blind with regard to what is around them [3, 9, 14]. Coordinating their respective motion in a complex environment thus requires complex built-in knowledge and algorithms, making agents unable to adaptively deal with dynamics or unpredictable environments, or complex and expensive communications to let them acquire the needed contextual knowledge. The core idea of the approach we propose in this paper is to provide agents – via an appropriate network infrastructure – with simple yet contextual information supporting and facilitating the required motion coordination activities. Context-aware agents, by then exploiting the available contextual information, can coordinate each other via very simple algorithms in a robust and adaptive way.

In particular, in our approach, context-related information is expressed in the form of “computational fields” (*Co-Fields*). We can imagine that each agent of the system can generate specific fields conveying some application-specific information about its local environment and/or about itself. Agents can perceive

these fields and can react accordingly, i.e., following the gradient downhill, uphill, or by following its equipotential lines. Therefore, agents' activities and movements are simply driven by these "force" fields, without any central controller. A case study in the area of traffic management is adopted through the paper to exemplify the main concepts and use of the Co-Fields approach and its suitability in effectively providing support for several motion coordination problems.

The following of the paper is organized as follows. Section 2 introduces the case study in the area of traffic management. Section 3 discusses the inadequacy of current coordination models to deal with traffic motion coordination. Section 4 describes the Co-Fields model. Section 5 details the use of Co-Fields model in the case study and presents the results obtained. Section 6 discusses related works in the area. Section 7 concludes the paper and discusses future works.

2. Case Study Scenario: Traffic Management

To fix the ideas on an application scenario and to clarify our model, we introduce a simple case study application consisting in a system to enable the coordination of the respective movements of the vehicles in a city.

We assume that a city is provided with an adequate computer network embedded in its streets/corners, and that cars are equipped with a computer executing a navigator agent. Such hypothesis simply reflects the increasingly diffusion of intelligent traffic lights, computer-based traffic monitoring tools, and on-board computers. More in particular, we assume that each computer host of the network is capable of communicating with each other and, via wireless connections (e.g., IEEE 802.11b) with the vehicles located in its proximity. The number of the embedded hosts and the topology of the network depend on the city, but basically the requirement is that each host represents and manages a meaningful zone of the city (e.g. a traffic light or a roundabout), and that the embedded network topology mimics the topology of the city plan. Also, we require some sort of localization mechanism to be enforced by the network: although GPS may well serve that purpose, cheaper local mechanisms relying on the properties of wireless communications [7] may be enough to let a host determine which cars are in the proximities and where they are.

The above-described scenario may be of great help from both the local traffic administrators and the drivers' point of view. The local traffic administrators can trivially exploit it to monitor traffic and to implement clever traffic-light synchronization policies. Drivers can

trivially exploit it to retrieve location-dependent information (e.g., which gas stations or available parking slots are there in the proximities). Here we instead aim at exploiting the embedded network to globally coordinate traffic motion, depending on various needs. For instance, all drivers may wish to move in the city by avoiding traffic jams or queues, which at the same time may provides the advantage of an overall load balancing of the city traffic. As an additional example, there may be the need for a group of cars to meet with each other at the most suitable location or, similarly, there may be the need to evacuate in the most efficient way specific portion of the city. Also, there may be the need for specific group of vehicles to move in the city accordingly to specific formations (e.g., consider police cars in need to properly monitor an area of the city). These motion coordination patterns, as explained in the following sections, challenge current approaches to distributed coordination.

As an additional note, we emphasize that the outlined scenario and the associated motion coordination problems are isomorphic to other interesting scenarios, such tourists visiting a large museum, forklifts moving in a warehouse, software agents exploring the Web. Therefore, all our considerations are of a more general validity, besides the traffic management case study.

3. Limitations of current models and middleware

Any type of coordination, there included motion coordination, requires some sort of context awareness. In fact, an agent can coordinate with other agents only if it is somehow aware of "what is around" (i.e., its context). Unfortunately, agents cannot generally have enough information for their coordination tasks from scratch, because they are embedded in a possibly unknown, open and dynamic environment.

In the traffic management scenario, to let drivers make appropriate choices, they need information where their destination is and how to get there, and if they want to avoid traffic jams they must also know what the traffic conditions are. However, these data are not always available from scratch; for example, drivers familiar with the city map usually lack updated information on the traffic conditions; in addition, they can arrive in an unfamiliar city, or in a portion of a city that is undergoing major re-structuring and have no information about the city map and thus about where to go.

The core of any coordination activities must therefore be related to the gathering of contextual information and to its exploitation. In the last few years, several middleware and coordination models, addressing – among the others – the problem of coordination and

interaction in a multi-agent system (more generally, in a distributed multi-component system), have been proposed. These can fall into three main categories: (i) models based on direct communication (ii) models based on shared data-spaces (iii) and models based on event publish/subscribe. However, as detailed in the following, these approaches do not provide enough contextual information, or they provide it in an ineffective way.

In *direct communication models*, a distributed application is designed by means of a group of components that are in charge to communicate with each other in a direct and explicit way. Systems like Jini [9], UPnP [14] and FIPA-based agent systems [3] are examples of middleware infrastructures rooted on a direct communication model. The problem of this approach is that the model, per se, does not provide any contextual information. Agents have to “manually” become context aware by discovering the other entities in the environment. For instance, in the case study, a driver agent has to explicitly retrieve via some local service of the infrastructure the local city map, it has to discover which other cars are currently driving in its same area, and explicitly negotiate with them to agree on a specific traffic management policy. Therefore, the approach does not generally suit the coordination needs of distributed scenarios, in that it requires agents’ notable efforts (both of computation and communication) to acquire context-awareness and end up with ad-hoc solutions for a contingent coordination problem (decisions which are, consequently, brittle, not flexible, and not adaptable).

Shared data-space models exploit shared localized data structures in order to let agents interoperate and coordinate with each other. Systems like Lime [12] and XMiddle [11] are implementations of this model. In these cases, agents are no longer placed in a void space but they live in an environment that can be modeled and described in terms of the information stored in the data spaces that, being accessible only from a locality, can provide some sort of contextual information to agents without forcing agents to directly communicate with each other. For instance in the case study, one can assume that the city infrastructure provides, at each street and corner, a local data-space, which stores local information as well as messages left by the other agents about their presence and eventually their intended next location. An agent accessing several data-spaces would have to build an internal representation of the actual traffic condition and then decide either by negotiating with other agents or accordingly to a predefined set of rules its next movement. Still, the problem of the approach is that contextual information usually expresses raw local data that can be difficult for agents to “understand” and exploit to achieve their coordination tasks. In other

words, coordination decisions have still to be taken directly by agents on the basis of the available data (thus requiring computational efforts), accordingly to some global policy that is either previously established (and thus is not flexible and adaptive) or it has to be acquired (thus requiring further communication efforts among agents).

In *event-based publish/subscribe models*, a distributed application is modeled by a set of components interacting with each other by generating events and by reacting to events of interest. Typical infrastructures rooted on this model are: Jedi [5], Jini Distributed Events [9] and UPnP General Event Notification Architecture (GENA) [14]. These models free agents from the need of explicitly querying the environment or the other agents (as in direct communication and data-space models), and thus leads to software systems that can be both computationally and communication efficient. Current software engineering practices are based on designing each agent’s program by specifying how the events update agents’ internal state (which encode agent’s context awareness) and how they trigger actions to let each agent behave accordingly to its perceived context. For instance, in the case study, a possible use of this approach would be to let each agent to notify its movements across the city, to update (accordingly to other agents’ notified movements) its internal representation of the traffic condition (i.e. context-awareness) and then move properly. The problem of this approach is that it is still too complicated: even if they are provided with all the information they need, agents have to apply a complex decisional algorithm to infer the right decision, about where to go, from their internal knowledge.

In the next section we are going to describe our proposal, stressing on how it improves current best practices, trying to solve the described problems.

4. The Co-Fields Approach

Although the models surveyed in the previous section are interaction models, prescribing only data exchange mechanisms, they typically induce coordination models on the application level. In other words this means that the tool used to communicate has a strong impact on what is communicated. In fact, the standard practice to exploit the above interaction models is to have contextual information represented by some kind of general purpose data, and to let agents acquire and process this data to take decisions relevant for their coordination task. This tends to keep the context representation and its usage by the agents strongly separated, typically forcing agents to execute complex algorithms to exploit the contextual information.

On the contrary, designing context-representation

together with the policy agents will adopt to take advantage of this information makes the process of exploiting contextual information automatic because the context has been represented knowing in advance what will be the agent's reaction to that information. This idea is at the basis of the Co-Fields model. Following this approach, agents achieve their goals not because of their capabilities as single individuals, but because they are part of an (auto)organized system that leads them to the goals achievement. The fact that the goals are accomplished is not a merit of the single agents, but of the system as a whole [15]. Such characteristics also imply that agents' activities are automatically adapted to the environmental dynamic, which is reflected in a changing view of the environment, without forcing agents to re-adapt themselves.

4.1. Co-Fields Overview

To apply the above concepts, the Co-Fields approach takes its inspirations physics and can be schematized in the following four points:

1. The environment is represented and abstracted by "computational fields", spread by agents and by the infrastructure. These fields convey some useful information for the agents' coordination tasks and provide agents with a local contextual perspective, tailored to specific coordination task, of the global situation of the system.
2. The coordination policy is realized by letting the agents move locally following the "waveform" of these fields, the same as a gravitational particle moves in accord to the locally perceived gravitational field.
3. Environment dynamics (through the infrastructure) and agents' movements induce changes in the fields' surface, inducing a feedback cycle that influences agents' movement (point 2).
4. This feedback cycle lets the system (agents, environment and infrastructure) to auto-organize, so that the coordination task is finally achieved.

A field can be defined as a distributed data structure composed by a unique identifier, a value (representing the field magnitude in that particular point), and a propagation rule. Fields can be generated by the agents or by the environment, and are propagated through the space as specified by their propagation rule. To support fields' propagation a proper infrastructure or middleware is required. This middleware can be based on an external server in charge of storing fields' values, but it can also be embedded in agents themselves and rely on an epidemic communication schema. The final shape of the field surface will be determined both by the field's propagation rule and by the infrastructure topology.

Fields can be static or dynamic, basically a field is static if once propagated its magnitude does not change over time; it is dynamic if its magnitude does. A field can be dynamic because for example its source moves and the field, with some propagation delay, changes accordingly its magnitude. In a given environment, several different types of fields can exist and be propagated, accordingly to field-specific rules. Fields can derive from the environment itself or can be injected by application agents, to support application-specific problems. The achievement of an application-specific coordination task relies on the evaluation of an application-specific *coordination field*, as a combination (e.g., linear) of some of the perceived fields. The coordination field is a new field in itself, and it is 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 simply following (deterministically or with some probability) the shape of their *coordination field*, as if they were walking upon the *coordination field* associated surface. Basically their actions will be based on following downhill the decrease of the *coordination field*, on following its increase uphill, or on following one of its equipotential lines (see figure 1).

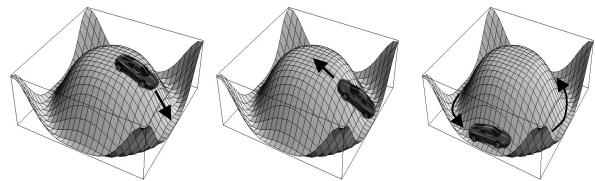


Figure 1. A car agent following the surface of its coordination field; (left) the agent follows the decrease of its coordination field; (center) the agent follows the increase of its coordination field; (right) the agent follows an equipotential line of its coordination field.

As an example related to traffic management, let us suppose that we need to have a group of agents move by maintaining a proper regular grid formation (e.g., with regard to traffic management, this could be the case of a group of police cars in need of monitoring the city), say a regular distance d from each other. In that case, each agent of the group can generate a computational field that propagates in the surroundings and reaches a minimum value at distance d from the agent itself (as the field depicted in figure 1). Then, if each agent in a given point of the space evaluates the local coordination field as the minimum of all the perceived fields, and moves by trying to reach the closest local minimum, the resulting movements of all the agents is in a regular grid formation

(see figure 2). We emphasize that, following the physical inspiration, a Co-Fields based system can be considered as a simple dynamical system. Agents are simply seen as balls rolling upon a surface whose shape is described by the coordination field. Complex movements are achieved not because of the agents' will, but because dynamic re-shaping of this surface.

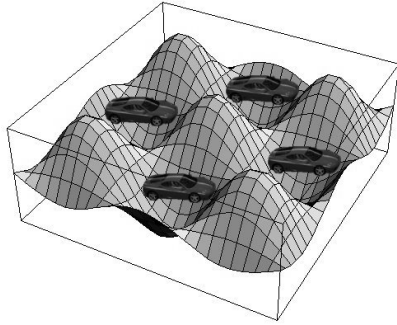


Figure 2. If all the agents generate the same field and try always to move in the local minimum of the resulting coordination field, the result is a regular mobile grid of agents.

4.2. Implementing Co-Fields

Co-Fields can potentially be implemented, as an overlay network, on any middleware providing basic support for data storing, communication and event-notification. In fact, what is required from the software infrastructure to implement Co-Fields is to provide simple storage mechanisms (to store field values), basic event-notification and subscription mechanisms (to notify agents about changes in field values and to enable agents to select those fields in which they are interested), and mobile-code services (to dynamically configure field-propagation algorithms and coordination fields composition rules). In addition, some sort of localization mechanisms must be enforced to discover where agents are [7].

In a preliminary set of implemented experiments, such servers have been implemented upon MARS tuple spaces [4]. MARS spaces have been allocated by IEEE 802.11 access points, have been programmed so as to store fields and to notify agents about local field changes, and have been complemented with a set of support agents in charge of propagating fields to neighbor access points. Of course, the choice of implementing a Co-Fields based coordination system as an additional layer over an existing middleware infrastructure, although providing for generality and portability, is not the most efficient solution. The abstraction mismatches between the two layers can introduce computational and communication inefficiency. For these reasons, we are currently

completing the definition and the implementation of a micro-kernel system explicitly conceived as a middleware support for Co-Fields, matching the Co-Fields abstractions, and light enough to be installed also on resource-constrained devices.

5. Traffic Management with Co-Fields

Applying the Co-Fields' model to the case study scenario described in section 2 is straightforward: we assume the presence of an embedded host in each of the city's street and corners. These hosts will be used to store and propagate different types of fields, representing different aspects of the environment. Agents access the infrastructure by connecting to their closest host. Once connected, an agent can access only to the host's stored fields and to the fields stored in the host's closest neighbors (we will consider one-hop-radius neighborhood). In this way a strong locality scope for agent perception and interaction is enforced. Accessing at least a small hosts' neighborhood is required, because agents' movements are based on the locally perceived gradient of a field (see figure 1) and that gradient can only be determined by evaluating the difference between fields' magnitude in different hosts.

We show here how to achieve two coordination tasks: load balancing and meeting. To this end, the three simple fields described in the next sub-sections are required, to be composed in application-specific way to achieve a given coordination task.

5.1. Street/Corner Field

This field, *SCF*, is a field generated by every city street/corner. It simply has value 1 in the street/corner that generates it and its value increases as the distance from the source (measured in terms of hops number) increases. In particular, we can simply imagine the field value is increased by 1 at every hop. Figure 3 left shows a simple city map, with corner A field's values reported. Because the propagation rule follows a breadth first algorithm, problems related to multiple paths are avoided. The above fields are static and they do not change over time.

5.2. Presence and Traffic Fields

The *vehicle presence* field, *PRES*, is generated by each vehicle. It simply has value 1 where the vehicle is located and its value increases monotonically as the distance from the source increases. Thus, the value of the fields is dynamic and varies both in space and time, depending on the current location of a vehicle. The *traffic-field TRF* can be derived from the *PRES* field: it measures the amount of traffic in a street/corner, and it is evaluated by summing the total number of *PRES* field with value 1,

determining the cars present in that place (i.e. connected to a certain server) and normalizing that number to the dimensions of the street/corner (see figure 3 right). Again, the traffic field is dynamic and adjusts its values over time, depending on agents' movements.

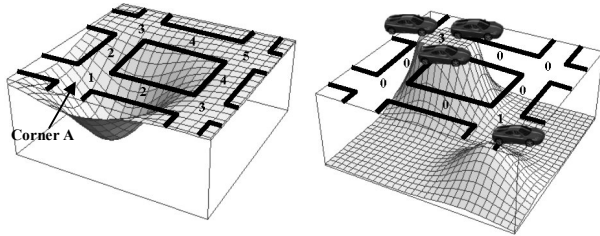


Figure 3. (left) Street/Corner Fields: 'Corner A' generated field, (right) Traffic Field.

5.3. Traffic Load Balancing

The aim of this service is to help a driver to avoid traffic or queues while visiting the city. As explained in the previous subsections, we assume that each street/corner in the city propagates the corresponding *street/corner field* and that the infrastructure computes the *traffic field* as described above. The model implementation is then quite straightforward: basically each agent evaluates its *coordination field (CF)* as the sum between a minimum combination of the *street/corner fields (SCF)* in its visit schedule (fields are combined by taking in each point the minimum one) and the traffic field (*TRF*).

$$CF = \min(SCF_1, SCF_2, \dots, SCF_n) + \lambda \cdot TRF$$

The first term of the coordination field, expresses a field surface having its minimum points in correspondence of the street/corners the agent has to visit. So, because each agent follows downhill the coordination field, this term guides the agent to visit the street/corners in its schedule. In order not to get stuck in a minimum, when the driver completed the visit of a place, the corresponding field is removed from the combination. The place is thus removed and so it does not represent a minimum anymore. The second term of the coordination field takes into consideration the traffic management. In fact the term $\lambda \cdot TRF$ with $\lambda \geq 0$ is a field that has its maximum points where the traffic is particularly intense. When this term is added to the minimum combination, it changes the steepness of the coordination field in the crowded zones. In particular a crowded zone tends to be a peak in the coordination field and thus it tends to repulse the income of other agents. It is easy in fact to understand that the agent will follow the "greed path" - the one indicated by $\min(SCF_1, SCF_2, \dots, SCF_n)$ - only if CF decreases towards the same direction indicated by $\min(SCF_1, SCF_2, \dots, SCF_n)$. For this reason λ can be

regarded as a term specifying the relevance of the traffic field. If λ is too high the agent will suggest the driver to follow alternative (possibly longer) uncrowded paths towards the destination whenever the "greed" path will be a bit crowded. If λ is too low the agent will accept always to remain in the "greed" path disregarding the traffic conditions.

To evaluate the performance of the Co-Fields mode, we experimented with a set of simulations in which a group of agents roam the city independently visiting the street/corners in the city according to their schedule. We compared the case in which the agents are interested to the traffic field, and thus the traffic management applies, to the case in which agents are not interested in the traffic field and no traffic management applies (figures 4). The results confirm the effectiveness of the approach and its suitability in managing large multi agent system consisting of more than one hundred agents.

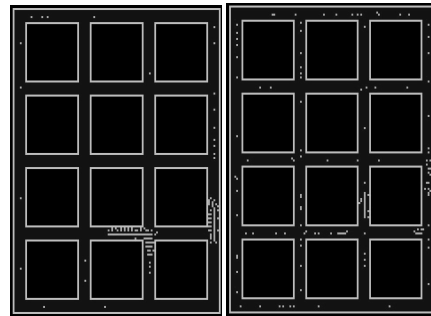


Figure 4. Without traffic management (left) large traffic jams appears. These traffic jams are avoided when the traffic management is active (right).

5.4. Meetings

Let's turn our attention to a "meeting" service whose aim is to help a group of vehicles to dynamically find and move towards the most suitable street for a meeting. The definition of this coordination policy can rely on the previously introduced fields by changing the coordination fields computed by vehicles. Several different policies can be thought related to how a group of vehicles should meet:

1. The group wants to meet in a particular street x . This is the simplest case and each of the user has to compute the coordination field $CF = SCF_x + \mu \cdot TRF$. In this way every vehicle is directed to the minimum of SCF_x that leads to the meeting street. The *TRF field* term enforces the load balancing policy also in this case.

2. The group wants to meet in the street where vehicle x is located. This is very simple as well: each vehicle has to compute the coordination field $CF = PRES_x + \mu \cdot TRF$. Where $PRES_x$ is the presence field generated by vehicle x . In this way every vehicle is directed to the minimum of $PRES_x$ that leads to the meeting street (where vehicle x is located). It is interesting to notice that this approach works even if vehicle x moves after the meeting has been scheduled. The meeting will be automatically rescheduled in the new minimum of $PRES_x$.
3. The group wants to meet in the street that is between them (their “barycenter”). To this purpose each vehicle i can compose its coordination field by combining the fields of all the other vehicles: $CF_i = \sum_{x \neq i} PRES_x + \mu \cdot TRF$. In this way all vehicles “fall” towards each other, and they meet in the barycenter street. It is interesting to notice, that this street is evaluated dynamically and the process takes into consideration the traffic. So if a street is overcrowded it will not be chosen, even if it is the barycenter one. Similarly, if some vehicles encounter traffic in their path to the meeting street, the meeting street is automatically changed to one closer to these unlucky vehicles.

By consider the third of the above possibilities, the simulation works well: vehicles approach each other and eventually meet in the street representing the barycenter (see figure 5).

As a final note, we emphasize that the Co-Fields approach is adaptive: once the shape of the required fields have been identified, automatically adapt to any environment. For instance, with regard to the meeting problem, figure 6 shows that the meeting works well even if we change the structure of the streets, and without having to change a bit in the code of agents/vehicles or in the structure and propagation of computational fields

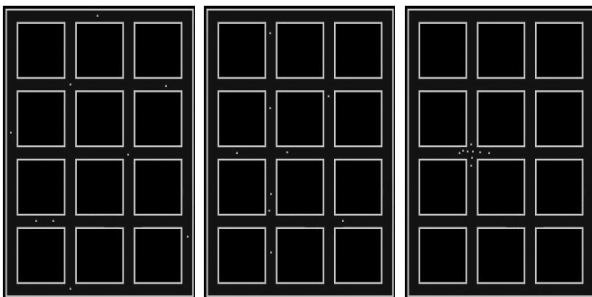


Figure 5. From left to right, different phases of the meeting process.

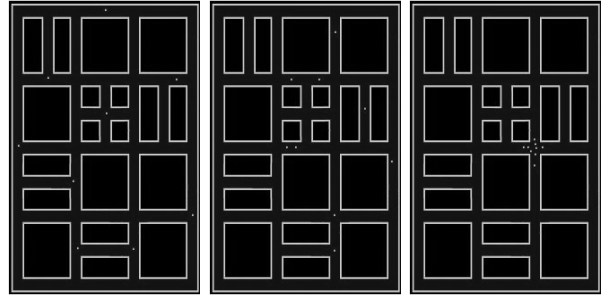


Figure 6. From left to right, different phases of the meeting process in a modified city map.

6. Related Works

Several proposals in the last years are challenging the traditional ideas and methodologies of software engineering to the best of our knowledge recent approaches to modular robot configuration [10], the way in which Non Player Characters are controlled in the videogame “The Sims” [Sims], and the MMass [2] multi agent systems model, are those that best match the Co-Fields approach.

Modular robots are made up of autonomous components, flexibly connected with each other, and globally enabling a reconfigurable shape [10]. The shape of the robot may need to be adapted to meet specific needs (e.g., a robot may need to change its shape to move in an constrained environment, the same as a snake change its shape): there is the need of globally orchestrating the movement of the different components. To this end: each component of the robot can generate a sort of computational field, to be propagated along the different component of the robot; components of the robot try to minimize the potential of a specific field, by changing their relative positions; these relative positions change determines the required dynamic re-shaping of the robot. Such an approach, while exploiting computational fields in a way very similar to Co-Fields, is less general. In fact, unlike in Co-Fields, a single computational field is usually generated in a robot, and it cannot compose with other fields to achieve more complex motion activities.

Another closely related approach is the one exploited to control the Non Player Character in the videogame “The Sims” [13]. The Sims are characters, living in a virtual world, whose behavior is directed by a “happiness landscape”: the Sims traverse a spatial landscape of happiness values trying to increase their happiness. Characters behave by climbing gradients of happiness, if they are hungry, they perceive a happiness landscape where things providing food will have higher peaks. So if they happen to be on the slope of a fridge, they will start climbing that slope until getting to the fridge. After

eating, all of a sudden the peak will collapse and a new landscape will appear to represent character happiness new requirements. The main difference between this and our approach is that “Sims’ happiness fields” tend to be static and generated only by the environment. On the contrary in our approach fields are dynamic and can change over time and agents themselves are able to generate fields and thus a stronger (auto)organizational perspective is enforced.

The Mmass formal model for multi-agent coordination, described in [2], represents the environment as a multi-layered graph in which agents can spread abstract fields representing, different kinds of stimuli, through the nodes of this graph. The agents’ behavior is then influenced by the stimuli they perceive in their location. In fact agents can associate reactions to these stimuli, like in an event-based model, but because of the explicit representation of the environment these events and reactions are location dependent. The main difference between this and our approach is that in our approach agents combine perceived fields and are constantly guided by the field produced. In their approach fields tend to be considered separately and they trigger one-shot reactions instead of guiding agents behaviors. Moreover also the application domain is quite different, while we are using this approach for the coordination of a multi agent system in an embedded computing scenario, they are mainly focused on an agent approach to simulation, using a MAS to simulate artificial societies and social phenomena.

7. Conclusions and Future Works

In this paper we presented Co-Fields, a new model to coordinate the movements of a large number of agents. The main advantage of Co-Fields is that:

- In Co-Fields, agents are provided with simple yet effective contextual information, enabling them to coordinate their movements in an implicit and natural way, thus requiring a tiny computational effort.

The main disadvantage is that:

- A general engineered methodology to map a coordination policy into the fields’ shape is still missing. This is not a specific problem of our approach and all the described related works suffer from the same problem.

We think that such an engineered methodology, at least for a specific set of problems, could be found in the future, allowing to exploit Co-Fields, well beyond the examples described in this paper.

Acknowledgement

Work partially supported by Nokia Research Center Boston and by MIUR Project “MUSIQUE”.

References

- [1] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman and R. Weiss, “Amorphous Computing”, *Communications of the ACM*, 43(5), May 2000.
- [2] S. Bandini, S. Manzoni, C. Simone, “Space Abstractions for Situated Multiagent Systems”, 1st International Joint Conference of Autonomous Agents and Multiagent Systems, Bologna (I), ACM Press, July 2002.
- [3] F. Bellifemine, A. Poggi, G. Rimassa, “JADE - A FIPA2000 Compliant Agent Development Environment”, 5th International Conference on Autonomous Agents (Agents 2001), pp. 216-217, Montreal, Canada, May 2001.
- [4] G. Cabri, L. Leonardi, F. Zambonelli, “Engineering Mobile Agent Applications via Context-Dependent Coordination”, *IEEE Transactions on Software Engineering*, 28(11):1040:1058, Nov. 2002.
- [5] G. Cugola, A. Fuggetta, E. De Nitto, “The JEDI Event-based Infrastructure and its Application to the Development of the OPSS WFMS”, *IEEE Transactions on Software Engineering*, 27(9): 827-850, Sept. 2001.
- [6] D. Estrin, D. Culler, K. Pister, G. Sukhatme, “Connecting the Physical World with Pervasive Networks”, *IEEE Pervasive Computing*, 1(1), Jan. 2002.
- [7] J. Hightower, G. Borriello, “Location Systems for Ubiquitous Computing”, *IEEE Computer*, 34(8): 57-66, Aug. 2001.
- [8] A. Howard, M. Mataric, G. Sukhatme, “An Incremental Self-Deployment Algorithm for Mobile Sensor Networks”, *Autonomous Robots*, 13(2):113-126, Sept. 2002.
- [9] Jini, <http://www.jini.org>, Sun Microsystems
- [10] J. Kubica, A. Casal, T. Hogg, “Agent-based Control for Object Manipulation with Modular Self-Reconfigurable Robots”, *International Joint Conference on Artificial Intelligence*, Seattle (WA), Aug. 2001, pp. 1344-1352.
- [11] C. Mascolo, L. Capra, W. Emmerich, “An XML based Middleware for Peer-to-Peer Computing”, In *Proc. of the IEEE International Conference of Peer-to-Peer Computing (P2P2001)*, Linkoping, Sweden, Aug. 2001.
- [12] G. P. Picco, A. L. Murphy, G. C. Roman, “LIME: a Middleware for Logical and Physical Mobility”, *International Conference on Distributed Computing Systems*, IEEE CS Press, July 2001.
- [13] The Sims, <http://thesims.ea.com>
- [14] Universal Plug and Play, <http://www.upnp.org>
- [15] F. Zambonelli, V. Parunak, “From Design to Intentions: Sign of a Revolution”, 1st International Joint Conference on Autonomous Agents and Multi-agent Systems, Bologna (I), ACM Press, July 2002.