Social potential fields: A distributed behavioral control for autonomous robots ★

John H. Reif *, Hongyan Wang [1]

Department of Computer Science, Box 90129, Duke University, Durham, NC 27708-0129, USA

Abstract

A Very Large Scale Robotic (VLSR) system may consist of from hundreds to perhaps tens of thousands or more autonomous robots. The costs of robots are going down, and the robots are getting more compact, more capable, and more flexible. Hence, in the near future, we expect to see many industrial and military applications of VLSR systems in tasks such as assembling, transporting, hazardous inspection, patrolling, guarding and attacking. In this paper, we propose a new approach for distributed autonomous control of VLSR systems. We define simple artificial force laws between pairs of robots or robot groups. The force laws are inverse-power force laws, incorporating both attraction and repulsion. The force laws can be distinct and to some degree they reflect the 'social relations' among robots. Therefore we call our method *social potential fields*. An individual robot's motion is controlled by the resultant artificial force imposed by other robots and other components of the system. The approach is distributed in that the force calculations and motion control can be done in an asynchronous and distributed manner. We also extend the social potential fields model to use spring laws as force laws. This paper presents the first and a preliminary study on applying potential fields to distributed autonomous multi-robot control. We describe the generic framework of our social potential fields method. We show with computer simulations that the method can yield interesting and useful behaviors among robots, and we give examples of possible industrial and military applications. We also identify theoretical problems for future studies.

Keywords: Artificial force laws; Potential fields; Autonomous robots; Distributed control; Distributed robotics

1. Introduction

1.1. Motivation: Very Large Scale Robotics (VLSR) systems

Much of the earlier work on robotic motion planning has considered the case of only a single or a few robots (see, e.g., the survey paper by Schwartz and Sharir [31], the book by Latombe [18], and the references therein). However, in the last decade, there has been a quickly growing literature on planning and cooperative control for systems of many robots. The main reason for this is probably the perceivable industrial and military applications of systems of many robots in the near future, as the costs of individual robots are going down, and as the robots are getting much more compact (utilizing the emerging miniaturized manufacturing capabilities, the robots are likely to be made

★ Supported by grants NSF CCR-9725021, NSF IRI-9619647, NSF/DARPA CCR-92725021, and ARO contract DAAH-04-96-1-0448.

* Corresponding author. E-mail: reif@cs.duke.edu

[1] E-mail: hongyan@cs.duke.edu

Preprint of paper appearing in Robotics and Autonomous Systems, Vol. 27, no.3, pp.171-194, (May 1999).

even at microscopic sizes for medical purposes), more capable, and more flexible, with sensing systems that will handle outside information faster and more accurately. Note that in the 1980s, electronic circuits went through a similar revolution in decreased physical size scale, cost and integration as the industry transitioned to VLSI.

We suggest the use of the term *Very Large Scale Robotic* (VLSR) systems for describing systems with hundreds to perhaps tens of thousands of autonomous robots. Possible industrial applications of VLSR systems include hazardous inspection, underwater or outer-space exploration, assembling, and transportation. Guarding, escorting, patrolling, and strategic and aggressive behaviors such as stalking an attacking are some examples of military applications.

In order to make effective use of the VLSR systems, we may wish the robots to exhibit some forms of cooperative behaviors. For example, we can use robots to clean, harvest or patrol a large area instead of human beings. The robots may need at times to cluster very tightly together, and at other times to scatter evenly in their working area so that the work can be partitioned evenly among them. Sometimes they need to spread out far away from their original position to explore, for example. In this paper, we propose a new distributed approach for the control of such VLSR systems.

1.2. Distributed control versus centralized control

A *centralized-control* paradigm assumes a single control which collects the information of the entire system (e.g. the states of all the robots), and then plans the motion for each individual robot. A centralized control may involve very high computational and communication complexity, may require very complicated planars, and lack of flexibility and robustness. (For complexity results, see [15,25]; for some works in the centralized approach, see [10,12,32].)

In a *distributed-control* paradigm, each robot determines its motion by observing the environment and by using some rules. This is done *locally* in the sense that a robot's plan is not based on the whole information of the system, but only on information accessible to itself. It does not know the plans of other robots either. However, with properly designed rules, the system may display desired global behaviors, or achieve predefined goals.

The theme of distributed control has long existed in robotics and in AI. In different contexts, an individual in the studied systems is sometimes called a *robot*, sometimes an *agent*, and sometimes a *particle*. The rules are sometimes called *social laws*, and sometimes called *basic behaviors*.

Moses and Tennenholtz [22,23] investigated several basic principles and formal aspects of distributed control with social laws. Shoham and Tennenholtz [33] studied the complexity of automatically deriving useful social laws to achieve certain goals. Parker [24] studied the problem of balancing global knowledge and local knowledge in designing cooperative robotic systems. He regards the information derivable from a robot's sensory and reflecting the environment near the robot as *local knowledge*, while *global knowledge* includes global goals, or information about the whole system that can only be obtained through communication channels other than sensory.

Interesting experimental results (mostly computer simulations) have been obtained in studies of using distributed control to achieve some specific goals. Sugihara and Suzuki [34] studied the problem of letting a group of robots form approximately geometric objects, such as circles and polygons. Their approach is to let the robots run an identical program asynchronously, where the program consists of instructions like "If $d > D$, then move towards the farthest robot", where d may be the distance between the robot and its farthest neighbor, and D is a fixed constant. Here a program can be considered as defining the rules. Mataric [20] uses the terminology *behavior-based systems*. In this system, each robot is programmed with a collection of simple behaviors such that the system will show some collective behaviors. For example, if each robot is programmed to compute the angle of the nearest two robots and to turn away from it, the system displays *dispersion* behavior. Gross and Deneubourg [14] studied how to use a group of robots to harvest by distributed control. Brooks, Maes, Mataric, and More [6] proposed to use many small autonomous robots to do lunar base construction. In their experimental system, the robots work without explicit coordination or communication, but nevertheless cooperate to achieve tasks such as digging out trenches.

Beni and Hackwood [3] studied what is called the *swarm intelligence*. A swarm, which is a collection of units, can converge to a wide range of distributions,

while no unit of the swarm is aware of the distribution it is working to realize. The swarm distributions can be determined by solving asynchronously systems of linear equations describing a difference equation with periodic boundary conditions.

For more studies in distributed control using simple rules, see [2,4,35] and references therein.

The distributed-control framework has also been applied to other fields such as Artificial Life (see [8,14] for some examples) and Computer Graphics. Reynolds [27] has successfully synthesized birds' behaviors such as collision avoidance, velocity matching and flock centering. A bird of his has limited vision and thus can only access information of nearby birds. To avoid collision with other birds and obstacles, a bird uses a *steer-to-avoid* rule.

The distributed-control paradigm seems to have strong justifications from studies in *sociobiology*. The group patterns by which animals search for food, attack enemies, and protect their own territories are surprisingly sophisticated (see [38] of Wilson), yet these patterns are accomplished by a distributed rather than a centralized control. For example, in the study of *grouping* behavior, observed data in [5,21] and computer simulations in [36] suggest that an individual is controlled by the sum of an *attraction* force (to stay with the group) and a *repulsion* force (to keep its own territory) to display the grouping behavior.

1.3. Previous work in the potential field method

Artificial potential methods have been previously used for obstacle-avoidance path planning. Work in this area was pioneered by Khatib [17], whose idea was simple and elegant. Imagine that in the configuration space of a robot (where the robot is a point), we assign negative electrical charges to the robot and the configuration obstacles, and positive charges to the goal configuration. Thus a potential field is formed with very high potentials close to the configuration obstacles and minimum potential at the goal configuration. The robot, guided by the potential fields, will go from high potential configurations to low potential configurations and hopefully will eventually get to the the goal configuration.

Potential field methods have been shown to be efficient and powerful in solving difficult motion planning problems such as one with high degrees of freedom (see [1,7,9,11,28,37] and references therein). However, because of the *local-minima problem* (i.e., a robot can be caught in a local minimum rather than the global minimum), except the work of Rimon and Koditschek [29,30], most of the potential-based methods are heuristics. Note that in [29], to avoid the local-minima problem, a combination of various force fields is used and the boundaries may not satisfy the definition of a potential field.

In the study of path planning using potential fields, the major concern is collision avoidance for a single robot. It can be regarded as a special case of a more general VLSR system described below. However, generating complex behavior by interaction via potential fields poses more significant challenges.

1.4. Social potential fields: A distributed control for VLSR systems

Within a distributed-control framework, we propose to use artificial potential fields as control laws. We allow the global controller to define (possibly distinct) pair-wise potential laws for ordered pairs of components (i.e., robots, obstacles or objectives) of a VLSR system. Each robot senses the resultant potential field from all other components (or as an approximation, at least the neighboring components) and acts under the resultant force. Once the force laws are defined, force calculations can be carried out by individual robots in a distributed manner, thus the control is completely distributed.

The force laws are inverse-power laws similar to those found in molecular dynamics. They are inverse-power laws of distances incorporating both attraction and repulsion. For example, we can define a force law where attraction dominates over long distances and repulsion dominates over short distances. A main reason for choosing inverse-power force laws as control laws is that studies have shown that molecules and plasma gases, while obeying simple inverse-power force laws, form interesting and complex patterns and exhibit a wide variety of dynamics. Our inverse-power laws are more general than the usual molecular force laws. In fact, the parameters of the force laws (i.e. the constants and the exponents) can be chosen arbitrarily. Indeed the parameters are chosen to reflect the relations of robots, e.g. whether they should stay close together or far apart. Using the force laws, the resulting system

displays "social" behaviors such as clustering, guarding, escorting, patrolling and so on. Thus we call our method *social potential fields*.

We also extend the social potential fields model to use spring laws as force laws. While the inverse-power laws are powerful in controlling the robots to form distributions like a more or less evenly distributed cluster, the spring laws can let the robots assemble into predefined exact structures.

In this paper, we describe the generic framework for social potential field control. We describe a hierarchical methodology for designing potential force laws to achieve given behaviors, which we feel is a practical approach. We illustrate this hierarchical methodology with a number of non-trivial examples (including computer simulations) which show the use of the proposed method to form interesting and useful behaviors among robots. We also point out possible industrial and military applications where our proposed method is useful. And also an important aim of this paper is to raise theoretical questions for future studies.

Nevertheless, we admit that this paper presents only a preliminary study of applying potential fields to distributed multi-robot control, since there is no known procedure for determining the potential laws to derive a given specified set of robotic behaviors. In Section 4, for some very restricted cases, we propose an iterative procedure to determine the parameters of a force law to achieve a given behavior. However the procedure is heuristic since we can not determine its convergence. In fact, the exact complexity of the simulation problem (determining the behavior induced by a set of potential laws) is not known. Reif and Tate [26] have shown that the complexity of determining the dynamics of an n-body potential system (e.g., the movement of n electrons with electrostatic potentials) is polynomial space hard, so the problem can not be decided efficiently by any known method. Therefore the problem of finding potential laws to achieve a given behavior may have no easy solution, except for the practical approach we suggest, i.e., our hierarchical methodology.

1.5. Organization of the paper

Section 1 discusses the motivation for our work, introduces the basic idea of social potential fields, and surveys related works. We describe the generic framework of the social potential fields method in Section 2. Section 3 describes our hierarchical methodology for designing potential force laws to achieve given behaviors, and gives some examples (including computer simulations) of applying the proposed method in creating interesting and useful behaviors among robots, for example clustering, guarding, escorting, as well as demining (detection of mines by a group of autonomous robots). In Section 4, we propose a heuristic iterative procedure for determining the parameters of a force law for a given behavior. Section 5 describes an extension to the social potential fields model, namely, using spring laws as control laws. We draw conclusions in Section 6.

2. Our social potential fields model

2.1. Inverse-power force laws

In a VLSR system, the robots are considered as point particles named $1, 2, \ldots, n$ with positions at $X_1, X_2, \ldots, X_n$ in a fixed Euclidean space at a fixed time. Let $r_{ij} = \|X_i - X_j\|$ be the Euclidean distance between robots i and j. The force defined from robot j to robot i has the form

$$F_{i,j}(X_i, X_j) = \left(\sum_{k=1}^{L} \frac{c_{i,j}^{(k)}}{(r_{ij})^{\sigma_{i,j}^{(k)}}} \right) \left(\frac{X_j - X_i}{r_{ij}} \right).$$

Thus the force defined from j to i is a sum of L inverse-power laws where the kth inverse-power law is of the form

$$\left(\frac{c_{i,j}^{(k)}}{(r_{ij})^{\sigma_{i,j}^{(k)}}} \right) \left(\frac{X_j - X_i}{r_{ij}} \right).$$

Our force laws may differ from those in molecular systems in that we allow the global controller to arbitrarily define distinct laws for separate pairs of robots. The coefficient $c_{i,j}^{(k)}$ and the inverse power $\sigma_{i,j}^{(k)}$ are constants depending on the ordered pair $\langle i, j \rangle$ and k. We generally require $\sigma_{i,j}^{(k)}$ to be positive but the coefficient $c_{i,j}^{(k)}$ can be either positive or negative, depending on whether it is an attractive force or a repulsive one.

Note that if $\sigma_{i,j}^{(k)}$ is greater than $\sigma_{i,j}^{(k')}$ then the kth term of this sum dominates for smaller r_{ij} whereas the k'th term of this sum dominates for larger r_{ij}. Typically

(with the exception of pursuit applications), for fixed i and j, the kth term with the smallest $\sigma_{i,j}^{(k)}$ will have a negative (repulsive) coefficient $c_{i,j}^{(k)}$ to help insure that the robots do not collide. The force laws usually have the following property. As the distance between two robots becomes larger, the force between them gets smaller. When two robots get too close to each other, the force can get very strong to avoid collision between these two robots.

Also, note that force laws are not necessarily symmetric; force $F_{i,j}$ can be different from force $F_{j,i}$.

At a fixed time, the overall artificial force applied by the entire VLSR system upon a robot i is

$$F_i = \sum_{j \neq i} F_{i,j}(X_i, X_j).$$

2.2. Succinct definitions of force laws between groups of robots

Robots are assigned to various (not necessarily disjoint) groups, where a group defines some common behaviors for robots in the same group. For example, robots in a group called Cleaner are required to do cleaning and are required to form an even distribution pattern. Groups are not necessarily disjoint. A robot may belong to several groups and inherit all the behaviors from those groups.

We can define force laws between pairs of groups. Let $S_1, S_2, \ldots, S_m$ be fixed groups of robots $1, 2, \ldots, n$. Any robot in group S' imposes the same force on any robot (other than itself) in group S. Let $F_{S,S'}$ denote this force law. Suppose the robot in S' is at position X', the robot in S at position X, and the distance between them is r. Then the force is

$$F_{S,S'}(X, X') = \left(\sum_{k=1}^{L} \frac{c_{S,S'}^{(k)}}{r^{\sigma_{S,S'}^{(k)}}} \right) \left(\frac{X - X'}{r} \right).$$

The force law is again a sum of L inverse-power laws each of the form

$$\left(\frac{c_{S,S'}^{(k)}}{r^{\sigma_{S,S'}^{(k)}}} \right) \left(\frac{X - X'}{r} \right),$$

where the coefficient $c_{S,S'}^{(k)}$ and the inverse power $\sigma_{S,S'}^{(k)}$ are constants depending on ordered pair $\langle S, S' \rangle$ and k (again, we generally require $\sigma_{S,S'}^{(k)}$ to be positive). For $S = S'$, the force law defines the force for a pair of robots in the same group. $F_{S,S'}$ is zero for the pairs of groups between which there is no force relation.

The force from robot j to robot i, where $j \neq i$, is defined to be

$$F_{i,j}(X_i, X_j) = \sum_{\forall S,S' \text{ s.t. } i \in S \wedge j \in S'} F_{S,S'}(X_i, X_j),$$

which is simply the sum of the forces $F_{S,S'}(X_i, X_j)$ induced by pairs of groups $\langle S, S' \rangle$ where $i \in S$ and $j \in S'$. The overall artificial force applied by the entire VLSR system upon a robot i is again

$$F_i = \sum_{j \neq i} F_{i,j}(r_{ij}).$$

Note that even though a robot may belong to different groups among which some force laws are defined, the robot does not impose forces on itself.

We can see that once the robots are assigned to groups, the complexity of defining force laws is reduced. The number of force laws is reduced from $n(n-1)$ for all ordered pairs of individual robots to $m(m-1)$ for all ordered pairs of groups, where n is the number of robots and m the number of groups in the system. Usually, the number of robots is much larger than the number of groups in a VLSR system. Each robot can maintain a more succinct table of force laws defined for pairs of groups, which is easier to maintain and to modify.

2.3. Local autonomous control forces

We consider the resultant force calculated by each individual robot using force laws, i.e. F_i as defined in Section 2.1, as a *global control force*. It is global in that it coordinates the robots and determines the distribution of the robots in the system. Consider the scenario of a group of robots doing gold prospecting (or more realistically, garbage collecting). The robots should have an even distribution over the working area, neither crowding together nor getting too far away. This can be achieved by properly designed force laws. At the same time, for a robot to work properly, it should be able to move smoothly in an unknown and dynamic environment, avoiding obstacles. But most importantly it should be able to approach and collect gold. The control that allows a robot this kind of behavior is termed as a *local control force*. The local

control force may include force to avoid collision with obstacles and force to approach an object, etc. This kind of local control can be accomplished by programs like "Walk sideways if there is an obstacle in front at distance d".

There are various schemes in combining the global and the local control forces. One scheme is to assign different weights to different kinds of forces and to combine them by a weighted sum. In this scheme, the weights can also change depending on the situation. For example, in the gold-mining example given above, the weight on force to approach gold should be small at the beginning, since at the initial stage it is more important to let the robots disperse. Once a more or less even distribution has been achieved (and this can be observed by the global controller), the global controller can notify the robots to increase the weight on force to approach gold. In another scheme, a robot listens to only one of the two forces at a time, depending on which one has a larger magnitude.

Imagine the situation where there is a wall-like obstacle that lies in the middle of a working area and whose size is comparable with the size of the working area. The wall-like obstacle may effectively prevent the robots from spreading to the working area on the other side of the wall, and thus interfere with the design of the social potential laws. Also if one part of the working area is filled with big obstacles, the working robots may populate that part less densely than other parts of the working area. We say that an obstacle is *local*, if its size can be ignored when compared with the size of the working area. Otherwise, the obstacle is said to be *significant*. In most practical cases where we are concerned, the obstacles are local. In these cases, though the local control force is necessary for the robots to function properly, we believe that the force does not play a significant role in determining the behaviors of the robots. Behaviors different from the ones designed by the global control forces may actually help the global controller to identify significant obstacles whose sizes are comparable to the size of the working space, and inform the global controller to re-think its plan. For example, a river through an unknown working area may be discovered.

In the examples and simulations discussed in Section 3, we focus on how the global control force controls the behaviors of the systems. The local control force is not taken into account.

2.4. Sensing

Note that the absolute positions of robots are used in force calculations as described in previous subsections. A robot can maintain its absolute position if it knows its initial absolute position and it keeps updating its current position as it moves. From time to time, a robot may readjust this piece of information by consulting the global controller. The absolute position of a robot is accessible to other cooperative robots by communication via electromagnetic radiation, such as radio frequency, infrared, or ultrasonic.

In most cases, relative positions are sufficient in calculating forces. In this case, each robot carries its own coordinate system, and itself is a natural choice for the origin of the coordinate system. There are many techniques that have been developed for range finding [16]. Among them are the triangulation range finder, ultrasonic, and laser time-of-flight range finder. While all these techniques have some drawbacks (e.g., computational complexity, limited surface orientation or limited spatial resolution), our social potential technique is relatively robust and generally may not require exact measurements. Thus it seems likely that any of these sensing methods would suffice. Relatives positions have to be used in calculating forces from non-cooperative components, such as obstacles in natural settings and enemy robots and so on.

Note that in either sensing through sensors or through communication channels, the group(s) to which a robot belongs should be identified too in order to choose appropriate force laws.

2.5. Types of robots

We have identified three types of robots which are basic in building a VLSR system with complexity, namely, a *leading robot*, a *landmark robot* and an *ordinary robot*. Most robots in a VLSR system are ordinary robots, whose motions are controlled by the global control forces.

A leading robot is a robot whose motion is either programmed beforehand, or controlled directly by the global controller. A leading robot is not affected by the force laws from other robots, but does impose force laws on other robots. Usually there are only a few leading robots in a VLSR system, but they can be used to form or to change the behaviors of a system

effectively (for an example, see the use of leading robots in Section 5.4).

A landmark robot is used by the system as a landmark. Generally a landmark robot is static and it is not subject to force laws from other robots. But it imposes forces on other robots. For example, a landmark robot can be used to mark a place of interest to attract more ordinary robots. Or we can use a group of landmark robots to mark out, at least approximately, the boundary of a working area, so that ordinary robots, sensing repulsion forces from the landmark robots, will stay within the working area. In Section 3.2, landmark robots are used to mark out significant obstacles (i.e., obstacles whose sizes are comparable with the size of the working area) to guide the ordinary robots effectively.

A landmark robot can be a real robot (but static), in which case its existence can be detected by the ordinary robots through sensors or communication channels. It can also be implemented virtually, by letting the ordinary robots carrying a table of landmark robots and their absolute positions. Note that in this case, an ordinary robot needs to be aware of its own absolute positions and use the absolute position in calculating forces from the landmark robots.

The type of a robot can be changed by the global controller. But such cases should be rare in order to avoid heavy communication to and from the global controller.

2.6. Distributed calculation of forces by autonomous robots

Even though the force laws are defined globally by a global controller, the actual control is carried out in a distributed manner. Each robot is equipped with sensors and also has a table of force laws, thus force calculations can be done simultaneously by individual robots. Each robot carries out a cycle of operations including sensing, calculating the global control force and the local control force, and realizing the control to motion. The robots usually act asynchronously.

Let F_i be the combined force (of the global and the local control forces) calculated by robot i. There are several ways that robot i's motion can be controlled by the force F_i. For example, robot i can gain an acceleration proportional to F_i. Or it can move in the direction of F_i, either for a length proportional to the magnitude of F_i, or for a fixed length.

2.7. Convergence and stability of the system

A *state* of a VLSR system is a snapshot of the system which describes the robots' positions, their velocities, and other relevant information. A *behavior* of the system can be *static*, in which case the behavior can be described by a state, or it can be *dynamic* (and should be periodic in this case). A dynamic behavior can be described by a function mapping $[0, T]$ to a state space, where T is the length of a period. We say that a VLSR system has *converged* to a behavior $\mathcal{B}$ if the system is displaying $\mathcal{B}$ (possibly within a predefined error term) over a long enough period of time. The converged behavior is called the *equilibrium behavior*.

A *density function* is a function which for a position in the space gives the density of robots at that position. Note that a density function describes a distribution, but not a state of a VLSR system. Even though a density function conveys less information than a state does, it suffices to describe certain behaviors. For example, a behavior of an evenly distributed cluster of robots covering a designated area can be sufficiently described by a density function. In fact, in cases like this, a density function describes the whole picture of a system better than a state (with the exact positions of all robots) does. Let $D(x)$, where x is a position in a given space, be the desired density function. We say that a system has converged to $D(x)$ after time τ, if

$$\int (D'_t(x) - D(x))^2 < \epsilon, \quad \text{for } t \geqslant \tau,$$

where $D'_t(x)$ is the density function of the system at time t, and ϵ is a predefined error term. Note that even though the distribution has converged, it does not necessarily mean that each robot stays close to a fixed position. They may still move around.

We can extend the density function to be a function not only of position, but also of time. Thus periodic distributions can be described by the extended density functions, and whether a system has converged to a periodic distribution can also be judged accordingly.

We can help to stabilize the system by adding a damping factor. For example, we can set a threshold such that a global control force smaller than the

threshold is regarded as zero. Or we can always reduce the global control force by a small fraction. The functionality of a damping factor is similar to a friction term that dampens a robot's movement by reducing energy change from potential energy to kinetic energy and helps to stabilize the system. In particular, the addition of a damping factor insures that the system will tend toward zero kinetic energy during periods when there is no other input of energy (such as from local autonomous control forces) into the system. Note: local autonomous control forces, if limited and random, may be analogous to certain quantum effect in molecular dynamics which can continuously and randomly perturb the system, keeping the kinetic energy to non zero.

Also note that in particular, the damping factor may help to stop a robot from oscillating, under both inverse-power law control and spring-law control.

2.8. Changing system behaviors dynamically

The force laws are not necessarily fixed over time. Instead, they can be changed by the global controller in order to change the behaviors of the system dynamically. Changing the behaviors dynamically is necessary in some cases. For example, at working time, we may want the robots to spread out over the work field evenly. Once the work is done, we may want the robots to gather together tightly so that they may be stored compactly. And again when they are needed for work, they should display a sparsely and evenly distributed pattern. When to change the behaviors is judged by the global controller, who can collect global information once in a while through sensors or communication with robots. Another example of changing the behaviors dynamically is given in Section 3.5.

Also note that the global controller can also change the behaviors of the system dynamically by controlling the leading robots. This technique is especially used in spring-law control; see Section 5.4. What's more, the type of a robot or the group(s) to which it belongs can also be changed by the global controller. This may also change the behaviors of the system.

Note that this paper attempts to describe (in most part qualitatively) a generic framework for applying potential fields to distributed multi-robot control. Detailed implementations such as how a robot's motion is controlled by a resultant force, how a local control force is combined with a global one, how frequent sensing and force calculations should be performed, and how some damping factors are added to stabilize the system, should depend on specific applications.

3. Examples of design of potential force laws

3.1. Our hierarchical methodology for designing potential force laws

In general, the problem of determining a set of potential force laws to achieve a given behavior appears to be a very difficult open problem, and most likely undecidable. As mentioned in the introduction, Reif and Tate [26] have shown that the complexity of determining the dynamics of an n-body potential system (e.g., the movement of n electrons with electrostatic potentials) is polynomial space hard, so the problem can not be decided efficiently by any known method. Therefore, the problem of synthesizing potential laws may have no effective solution.

However, we propose a practical approach to this problem. We have developed a hierarchical methodology for designing potential force laws to achieve given behaviors.

In our methodology, we proceed as follows:

- We first require a specification of the *required behavior*. In particular, we need to specify the various groups of robots and their interactions. In our examples, all components (e.g., robots, obstacles and other objects) are represented by point robots. An object with a complex shape can be approximated by putting many point landmark robots along the boundary of the object. For example, wall-like obstacles are approximated by a group of landmark robots in this way in Section 3.3.
- Then, we *design the potential laws* in stages. Applying our hierarchical methodology for designing potential force laws,
  - we first define *intra-group social force laws* among individuals of each given robot group, and then
  - define *inter-group social force laws* between individuals of distinct groups.
- Furthermore, in a number of examples, we define a dynamic series of potential laws, that change over time at discrete intervals.

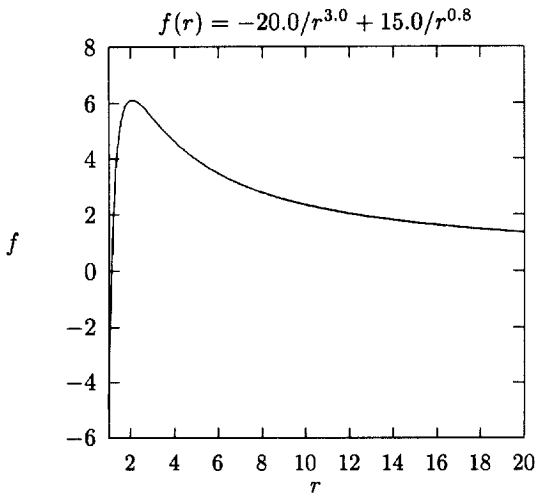


Fig. 1. The magnitude of a force law as a function of distance.

We give some examples (including computer simulations) of applying our proposed method in creating interesting and useful behaviors among robots, for example clustering, guarding, escorting, and demining (detection of mines by a group of autonomous robots).

An equation for potential laws. In the following examples, the magnitude of the force laws usually has the same form as below

$$f(r) = -\frac{c_1}{r^{\sigma_1}} + \frac{c_2}{r^{\sigma_2}},$$
$$c_1, c_2 \geqslant 0, \quad \sigma_1 > \sigma_2 > 0, \tag{1}$$

but with different parameters c_1, c_2, σ_1, and σ_2. The curve of the magnitude of the force law with a set of specified parameters is shown in Fig. 1.

These laws have two components:

- *Attraction*, indicated by the term c_2/r^{σ_2}. This will dominate when two robots sufficiently separate.
- *Repulsion*, indicated by the term $-c_1/r^{\sigma_1}$. Since $\sigma_1 > \sigma_2 > 0$, the repulsion will dominate when two robots get close to each other. Collision avoidance is guaranteed since the force goes to infinity as two robots get closer and closer.

A force law whose magnitude can be expressed as in Eq. (1) with $c_1, c_2 > 0$, i.e., with both non-zero attraction and repulsion, is called a *clustering force law*. A group of robots with an identical clustering force law defined among them are likely to display a clustering behavior. The attraction component of the clustering force law keeps the robots together, while the repulsion component prevents the robots from colliding with each other. Also the combination of attraction and repulsion determines how tightly the cluster forms.

Tuning the clustering. The *equilibrium distance* is defined as $d = (c_1/c_2)^{1/(\sigma_1-\sigma_2)}$. If the force law defines bilateral effect between two robots, then they reach equilibrium state if they are distance d apart. If the force law defines mutual effect among a group of robots, the group can either densely congregate if d is small or sparsely distribute if d is large. Even though in this case, the inter-robot distance must differ from d, d is still a good indicator as to how tightly the cluster will form. A big σ_1 implies that the repulsion will be strong at short range but decays rapidly with distance. A small σ_2, on the other hand, means that the attraction will have long range effect. Again, our method is called social potential fields method, because the force laws reflect and enforce social relationships.

Our computer simulations. For simplicity, we use a very straightforward method to compute the forces, which takes $\Theta(n^2)$ time if we have n robots in the system. We are experimenting with about hundreds of robots and the efficiency is bearable. But keep in mind, there have been efficient algorithms for computing forces (developed for molecular simulations) with high precisions (see, for example, the pioneer work by Greengard [13]). Those efficient algorithms enable us to experiment with tens of thousands of robots in large systems.

In our computer simulations, in each iteration, a robot moves a fixed length to the direction pointed by the resultant force.

In Figs. 2–5, the pictures are dumped window images. The windows give visual presentation of how the systems evolve. In the windows, the robots are represented by arrows pointing to their directions of motions.

In the following description of examples, we name separate groups and their members. The name of a group and the name of its members are the same except that the name of the group starts with a capital letter.

3.2. A single cluster of many robots

Required behavior. Our goal is a single evenly distributed cluster of robots.

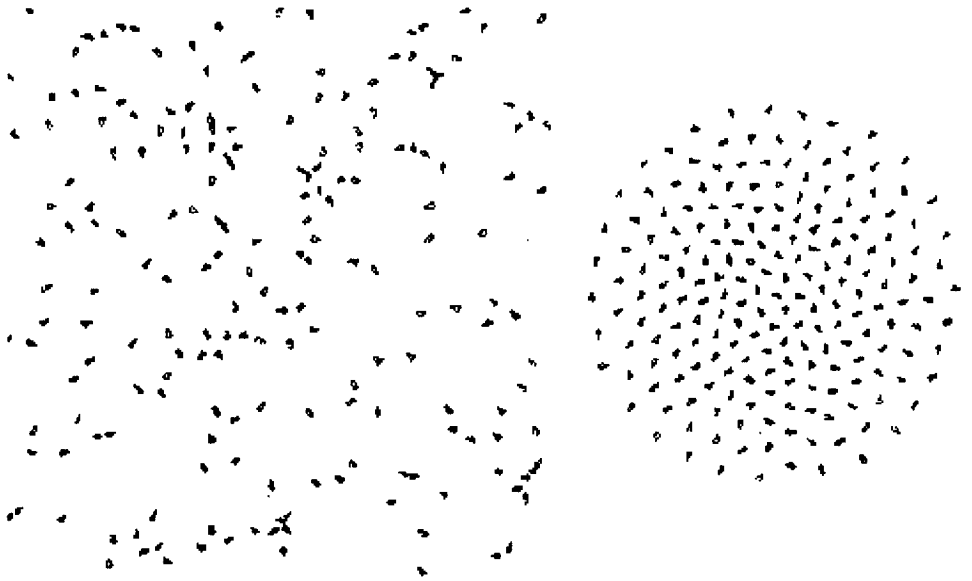


Fig. 2. An evenly distributed cluster of robots formed with an identical force law. The first picture shows the distribution after 3 iterations. In the second picture, the robots are shown to have converged to a disc after 225 iterations.

Motivation. A more or less evenly distributed cluster of robots is a useful pattern for jobs like cleaning, patrolling, or exploring an unknown area. Inter-robot distance is useful not only for robot collision avoidance, but also for getting work partitioned and distributed. For example, in the future we may send hundreds or thousands of robots to explore Mars before we send any human beings on it. To explore the planet efficiently, we want the robots to cover the surface of the planet evenly instead of crowding together. A centralized control mechanism is not applicable especially because of the communication complexity involved.

Design of an intra-group potential law. All the robots are identical ordinary robots, and there is an identical intra-group force law defined among them. The force law is a clustering force law, whose magnitude can be expressed by Eq. (1), with $c_1, c_2 > 0$.

Computer simulation. For this simulation, we used Eq. (1) with the parameters set to the following: $c_1 = 60.0$, $c_2 = 1.0$, $\sigma_1 = 2.0$ and $\sigma_2 = 1.0$. The equilibrium distance is $d = 60.0$. (The distances are measured in pixels.) The number of the robots is chosen to be 200, and initially, the 200 robots are randomly distributed within a square area of size 400×400. At the beginning, for a robot at the periphery, since there are many robots distant away (farther than 60.0), the robot feels an attraction toward the center of the cluster and the cluster starts to shrink. After a number of iterations, the cluster forms a disc with radius of about 280.0. At this time, each robot shows a periodic movement, while the density function of the cluster has stabilized. At this stage, the robots show a more or less even distribution within the disc. The initial and the equilibrium distributions are shown in Fig. 2.

Variations of single clusters. Note that the even distribution can be changed once some places of interest are identified by some robots. These robots can be turned into landmark robots by the global controller and impose a stronger attraction force on other robots. In this way, interesting areas will be more densely occupied by robots and can be explored more efficiently.

In the above computer simulations, the cluster usually forms discs. We can let the cluster display an arbitrary shape by using landmark robots as follows. For a given shape, we can use landmark robots to approximate the boundary of the shape. The landmark robots impose repulsion forces on the cluster of ordinary robots, so that the ordinary robots are confined within the boundary marked out by the landmark robots. By setting appropriate force laws, e.g.,

by choosing a large enough equilibrium distance for the ordinary robots so that they expand close to the boundary but are not "pushed" hard enough to go beyond the boundary, we can let the robots distribute more or less evenly within an area of the given shape. Note that in this case, the landmark robots may be virtual robots who do not exist physically but whose positions are remembered by the ordinary robots.

3.3. A moving robot cluster with collision avoidance

Required behavior. As pointed out in Section 1.3, potential fields have been applied in collision-free motion planning for a single robot in previous studies. In our example which we will call *bypassing the walls*, a group of robots are supposed to move together through an area filled with wall-like obstacles to reach a goal region. In this case, we not only want the robots to reach the goal with collision avoidance, but also want the robots to move more or less in a group without colliding with each other. An industrial application of this behavior might be, for example, using robots to do transportation through an area filled with static as well as dynamic obstacles (including other robot vehicles).

There are three groups of robots:

- The group Goal consists of only one landmark robot marking out the goal region.
- The group Transporter consists of ordinary robots who are supposed to move to the goal region.
- The group Wall consists of a group of landmark robots who are static robots (see Section 2.5) used to mark out the significant obstacles. In this case, the obstacles are wall-like, which can be abstracted as line segments.

Design of potential laws. Applying our hierarchical methodology for designing potential force laws, we define an intra-group force laws among the transporters themselves as well as inter-group social force laws from both the Wall and the Goal to the Transporter.

An intra-group potential law. There is a clustering force law combining attraction and repulsion defined among the transporters so that they tend to stay together as they move around without colliding with each other.

Inter-group potential laws.

- There is a force law from the Goal to the Transporter containing only an attractive force (i.e. $c_1 = 0$ in Eq. (1)). We set σ_2 to be small so that this attraction has a long-range effect.
- We let the Wall impose a repulsive-only force over the Transporter (i.e. $c_2 = 0$) and set σ_1 to be large so that the repulsion is strong in short range.

3.4. Guarding and escorting behaviors

Required behavior. In our computer simulation called *guarding a castle*, we wish to simulate how a group of robots form guarding patterns and how they react to invaders. There are three groups of robots. The group Guard consists of a number of identical ordinary robots, while the group Invader consists of only one ordinary robot. The castle is simulated by the group Castle which consists of only one landmark robot.

Design of potential laws. Applying our hierarchical methodology for designing potential force laws, we first define an intra-group social force law among the guards themselves, and then give inter-group social force laws from the Castle to the Guard, from the Castle to the Invader, and also between the Guard and the Invader.

An intra-group potential law. There is a clustering force law combining attraction and repulsion defined among the guards. The intention of this force law is to let the guards form a guarding cluster surrounding the castle.

Inter-group potential laws.

- The force defined from the Castle to the Invader consists of only an attraction.
- The force defined from the Castle to the Guard combines both attraction and repulsion.
  - The combination of the attraction and the repulsion makes the guards tend to stay around the castle (not too close and not too far away).
  - The attraction component of the force law also keeps the guards from going too far away while chasing off the invader.
- The Invader imposes both attraction and repulsion upon the Guard so the guards tend to move to where the invader is but to avoid contact with the invader.

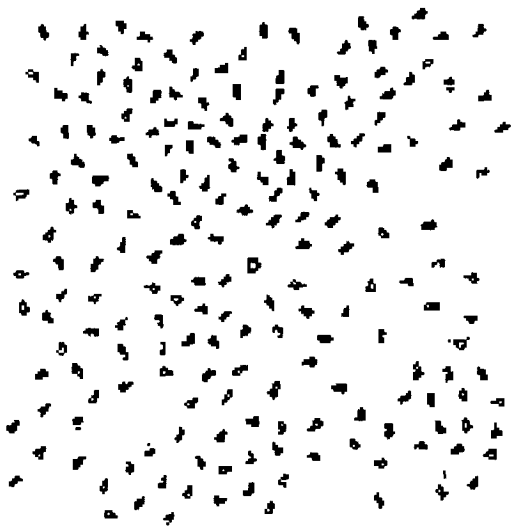


Fig. 3. A dynamic guarding behavior. Part 1: Initial distribution of guards around the castle, which is represented by a rectangle.

- The Guard imposes only a repulsion force upon the Invader.

Computer simulation. There are three stages in this simulation.

1. In the first stage, the invader robot does not appear. Initially the guards are scattered in a square area around the castle. Under the force law imposed by the castle and the force law defined among the guards, the guards converge gradually to form a ring surrounding the castle.
2. At the beginning of the second stage, the invader appears at a place far away from the castle. Under the attraction force from the castle, the invader approaches the castle. The guards will approach where the invader is. They move from the castle to the invader but stop chasing when the invader is a certain distance away from the castle. The invader, which is both attracted to the castle and repulsed by the guards at the same time, instead of going away from the castle, reacts by moving toward the weaker point of the guarding where there are fewer guards.
3. Finally the system converges to a dynamic behavior where the guards are chasing the invader and the invader is moving around the castle to find weaker points in guarding. See Figs. 3, 4, and 5 for the results.

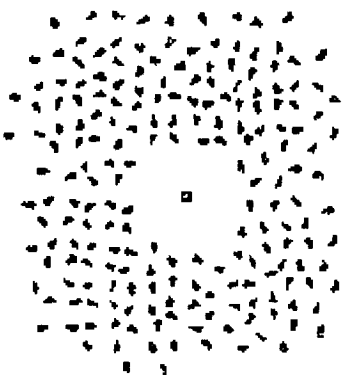


Fig. 4. A dynamic guarding behavior. Part 2: The guards have converged to a ring surrounding the castle.

From the above simulation, we can see the advantages of the social potential fields method. We avoided defining complicated rules as to where a particular robot guard should stay, which guards should chase the invader when the invader is at a certain position, and how far the guards should chase the invader. By simply defining force laws, the above questions are answered.

Variations of the guarding behavior. More complex and interesting behaviors can be simulated by

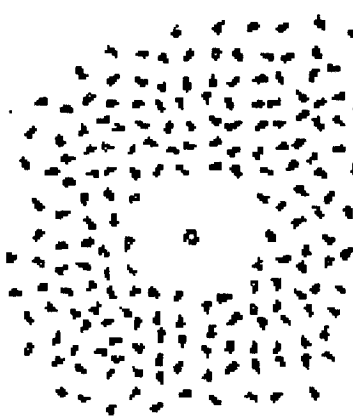


Fig. 5. A dynamic guarding behavior. Part 3: A dynamic behavior where the invader is trying to find weak points in the ring while the guards are chasing the invader.

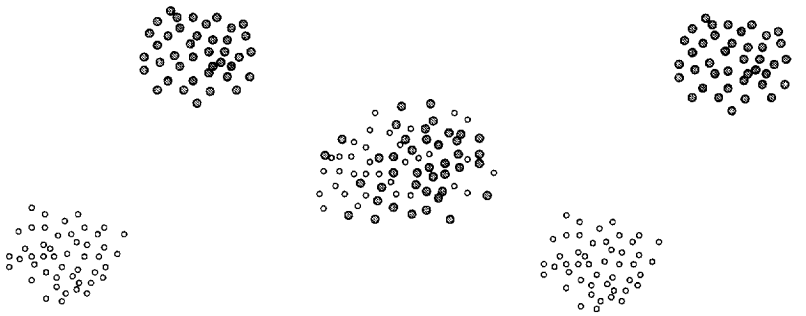


Fig. 6. Reorganizing after bivouacking.

modifying the above system. The castle can have an arbitrary shape instead of just being a point. Its shape can be marked out by a group of landmark robots and the shape will impact the guarding pattern. There can be more than one invaders, in which case the guards should partition to groups to handle each invader. If the static castle is replaced by some dynamic object, which can be simulated by some leading robots whose motions are pre-programmed, an escorting behavior can be simulated.

In practice, we might want robots to guard places like important military bases, high priority labs, factories and so on. We might also want to use robots to escort other robots. We can also transport a large number of robots from place to place by simply defining a leading robot to go to the designated place and letting the other robots follow it as if they are escorting the leader.

3.5. Dynamic force laws to achieve hierarchical spatial organization of robots

Required behavior. Here we give an example of adapting force laws by the global controller to achieve hierarchical spatial organization of robots. The example is called *reorganizing after bivouacking* and imitates the following military behavior. Normally during maneuvers, an army will be spatially organized in a highly hierarchical manner. An army is usually divided into units which are further divided into sub-units and so on, with different units occupying different areas. While in camp, to efficiently utilize resources such as army canteens, the army may become disorganized, with different units clustering together in the same area. However, going back to maneuvers, the entire army must automatically organize itself spatially as before. Fig. 6 depicts a simplified version of the example where there are only two units (one represented by the empty circles and the other the shaded circles). The organized army (shown in the leftmost picture) bivouacs (shown in the middle picture) and then reorganizes (shown in the rightmost picture).

Design of potential laws. In order to accomplish this *reorganizing after bivouacking* behavior, the global controller has to change the force laws dynamically. In the following, we show the design of force laws at different stages, taking the two-unit army as an example.

Dynamic potential laws.

- *The disorganizing stage.* At this stage, there should be an identical attractive force to the goal (camping area) imposed on robot soldiers of both units. Also there should be an identical force law combining attraction and repulsion between individuals in both units to achieve a clustering behavior.
- *The bivouacking stage.* After both of the units reach the camping area, the attractive force to the camping area can be inactivated by the global controller.
- *The reorganizing stage.* To reorganize, each unit should be attracted to its own area respectively. There should be (possibly different) clustering force laws defined among robots for each unit. Also there should be a repulsive force defined between two robots of different units. This force is to prevent two robots from different units from colliding with each other (this force should have the property that it diminishes fast as the distance gets larger, so that it

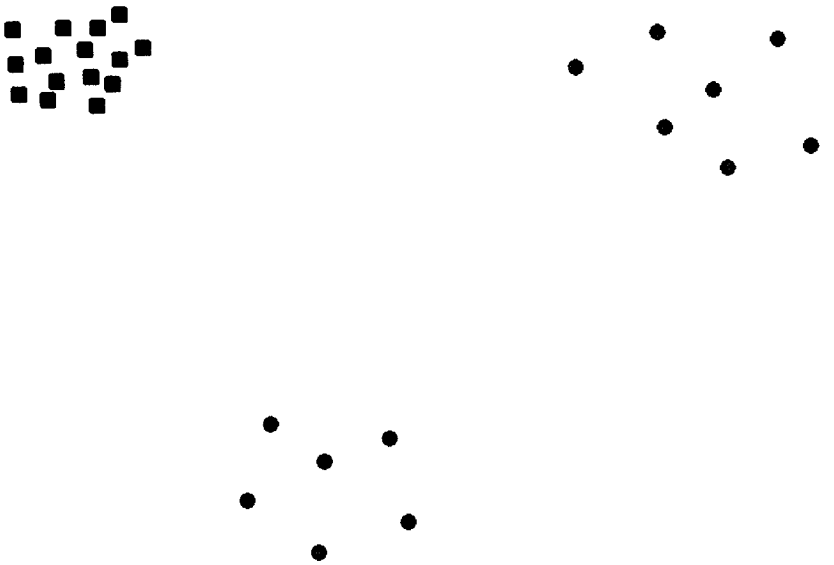


Fig. 7. Initial distribution of deminers.

does not have a big effect on the goal-approaching behaviors and the clustering behaviors of the two units).

- *The organized stage.* After each of the two units reaches their respective destinations, the attractive force laws to goals and the repulsive force between two robots of different units can be deactivated. Therefore, at this stage, there is no force law between the two units. The distribution of each unit is formed by (possibly different) clustering force laws among robots within each unit.

The spatial organization of robots occurs in industrial applications too. For example, at work in a factory, different groups of robots are dispatched to different areas. Within each group, the robots are further divided into subgroups for different tasks such as cleaning, transporting or assembling. At work, we want the groups of robots to be spatially distributed and there should be no interactions among them. At other times (say cleaning the factory), we want all the groups to crowd together.

3.6. Demining using robot clusters

Required behavior. We next discuss the application called *demining*, which shows detection and deactivation of mines by a group of autonomous robots. A cluster of demining robots (deminers) is searching for bombs in a mine-field. A deminer can detect a bomb only when it is within a certain range from it. When a deminer detects a bomb, it will move closer to it and then deactivate it when the distance to the bomb is less than the so-called *deactivating distance*. It will take a certain period of time for a bomb to be deactivated. In this period, the region close to the bomb is considered to be dangerous and thus deminers should avoid getting too close to the bomb.

Design of potential laws. Applying our hierarchical methodology for designing potential force laws, we first define an intra-group social force law among the deminers themselves and then inter-group social force laws from a bomb to a deminer.

An intra-group potential law. The force law among the deminers is a clustering force law which keeps the deminers on the mine field and also distributes them evenly.

Inter-group potential laws.

- When a deminer detects a bomb, it feels an attraction to the bomb until it gets into the deactivating range with the bomb (i.e., its distance to the bomb is less than the deactivating distance).

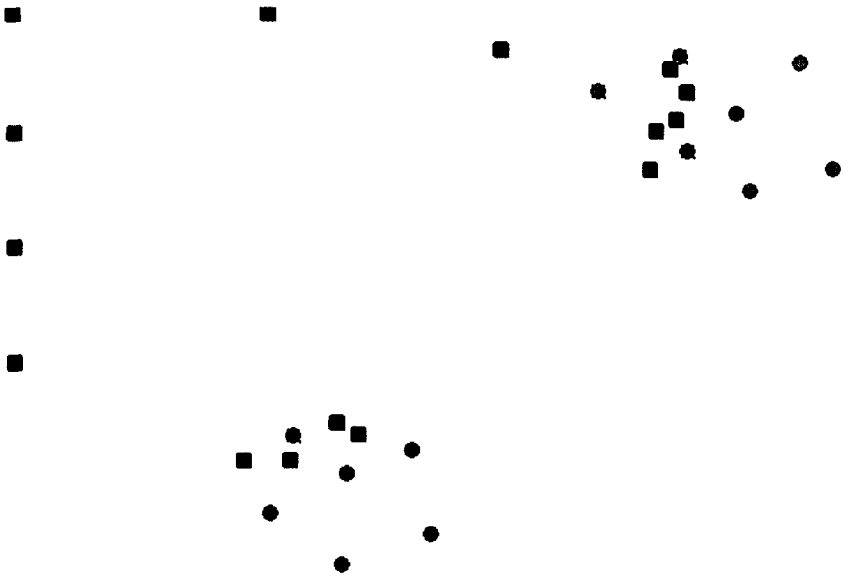


Fig. 8. Discovery of mines and their initial deactivation by deminers.

- When a bomb is being deactivated, it imposes a repulsion on the rest of the deminers to guide them move away from itself.

Computer simulation. The results of a computer simulation are shown in Figs. 7, 8 and 9. (An online demo of this simulation due to Zheng Sun is available at "http://www.cs.duke.edu/~sunz/java/demine/Animation.html"). In the figures, dark squares represent deminers; gray circles represent undiscovered bombs; dark circles represent bombs being deactivated; and gray circles with a cross represent deactivated (safe) bombs.

4. An heuristic for synthesizing force laws for VLSR systems

In Section 3, we described some computer simulations of interesting behaviors of robotic systems. In those simulations, the force laws were derived from our hierarchical methodology for designing potential force laws to achieve given behaviors. Two interesting problems are:

- to predict the equilibrium behavior (i.e., a behavior to which the system converges) of a system from predefined force laws, or
- vise versa, to define force laws such that a desired equilibrium behavior can be achieved.

These problems are hard in general for dynamic systems with iterative processes. Here we propose a heuristic iterative method to solve the above two problems for a very restricted case. In this case, *the robots are identical robots with a single force law defined among them.* (In a variation of this, there may be a few other robots and different force laws.) *The equilibrium behavior is static and can be described by a density function. Also, in the equilibrium behavior, the robots are at their equilibrium states, i.e., the overall force on any ordinary robot in the system should be zero.*

The problem of computing equilibrium density functions for given force laws is discussed in Section 4.1, and the problem of finding force laws to achieve given density functions is discussed in Section 4.2.

4.1. Computing the density function for given force laws

4.1.1. The general idea

The density function is computed in an iterative manner. The iterative equation is derived based on that in an equilibrium state (i.e., a static equilibrium

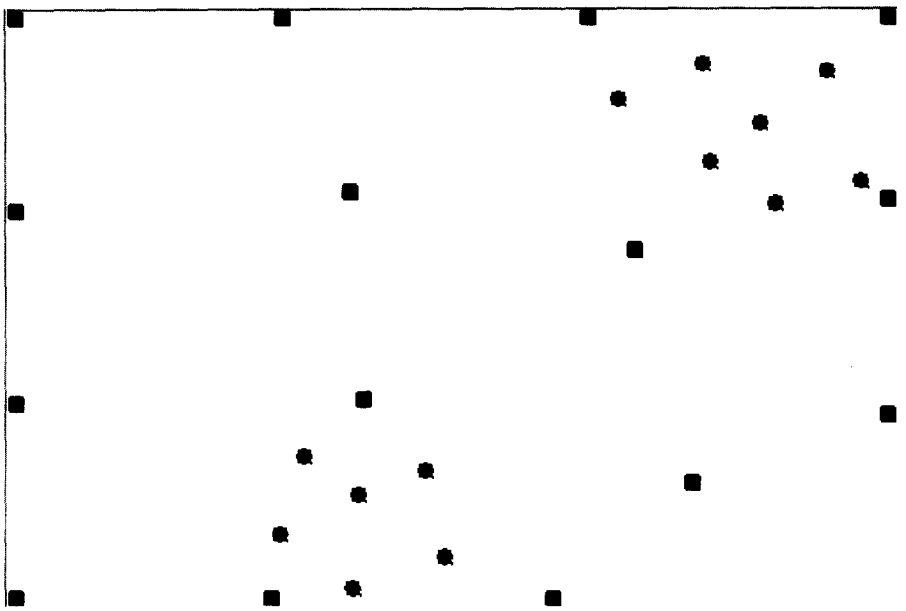


Fig. 9. Completed deactivation of mines by deminers.

behavior), the overall force on any ordinary robot in the system should equal to zero. For a robot at position x within domain of interest, let $F_{\text{nearby}}(x)$ denote the total force from nearby robots. Robots are nearby robots if they are within a distance μ, where μ is a parameter. Let $F_{\text{faraway}}(x)$ denote the total force from faraway robots. Thus in an equilibrium state, we have

$$F_{\text{nearby}}(x) + F_{\text{faraway}}(x) = 0.$$

Recall that the force functions usually have the property that they have large magnitudes and derivatives over short distances and small magnitudes and derivatives over long distances. Therefore we have decided to calculate $F_{\text{nearby}}(x)$ by summing up the forces from nearby robots discretely, since an integration may cause large errors, and to calculate $F_{\text{faraway}}(x)$ by integrating the forces from faraway robots, since an integration is a good enough approximation and it reduces the time to compute $F_{\text{faraway}}(x)$ from proportional to the number of robots in the system to that of doing an integration which depends on the domain of interest and the precision required.

To compute $F_{\text{nearby}}(x)$, we need to compute the force from each of the nearby robots. To do this, we need to know the relative positions of the nearby robots with respect to the robot at position x. The density function does not give us enough information here, and we need to assume that *the robots also form certain structures, namely, they form hexagons in 2D and hexahedrons in 3D*. We believe that this assumption is reasonable because these structures provide the highest packing capacities among all structures. $F_{\text{faraway}}(x)$ can be computed as an integration

$$\int_{\|x'-x\| \geq \mu, D(x')>\theta} F(x, x')D(x'),$$

where $F(x, x')$ is the force law from a robot at position x' to the robot at position x, μ is a threshold of distance to distinguish nearby robots from faraway robots, and θ is a threshold of density. Note that we have to fix a single force law to do the integration and that is why our method only works for the case where there is only a single force law defined among a group of identical robots.

Let $F_1(x)$ denote the force from one of the nearest neighbors and $F_{\text{nearby}-1}(x)$ the force from the rest of the nearby robots. It holds that

$$F_1(x) = -(F_{\text{nearby}-1}(x) + F_{\text{faraway}}(x)). \quad (2)$$

The density function appears on both sides of the above equation. In this way, we can get an iterative equation for the density function. Section 4.2 shows an example of deriving an iterative equation of a density function for a specific form of force laws.

We have sketched our the general idea to compute the density functions by an iterative process. For this iterative process to work, we have to assume that *the forms of the density functions are fixed and known, and only the parameters of the density functions are to be determined.* We generally assume that a density function can be expressed as a polynomial of certain degrees, and only the coefficients of the polynomial are to be determined.

(Note: Convergence of this iterative method is not likely to be quickly determined in the general case, since Reif and Tate [26] have shown that the complexity of determining the dynamics of an n-body potential system (e.g., the movement of n electrons with electrostatic potentials) is polynomial space hard.)

4.1.2. An example of a single cluster of working robots

In this example the robots are point particles moving on a 2D plane. The position of a robot is given by $z = (x, y)$. Assume that the density function is a polynomial of variables x and y, and is of degree k, as follows,

$$D(z) = D(x, y) = \sum_{i+j=0}^{i+j=k} c_{ij} x^i y^j.$$

To compute the density function is to determine the coefficients $c_{00}, c_{10}, c_{01}, c_{20}, c_{11}, c_{02}, \ldots, c_{0k}$, where c_{ij} is the coefficient of the term $x^i y^j$. Let C denote the vector $(c_{00}, c_{10}, c_{01}, c_{20}, c_{11}, c_{02}, \ldots, c_{0k})$. The number of coefficients to be determined is $l = (k+2)(k+1)/2$.

We assume that the single force law defined among the robots has the same form as in Eq. (1) with $\sigma_1 = 3.0$, $\sigma_2 = 2.0$, $c_1 = \alpha_1$ and $c_2 = \alpha_2$.

For a robot at position $z = (x, y)$, since we have assumed that the robots form hexagonal structures, then one of its closest neighbors is at position $(x + 1/D(x, y), y)$, where $D(x, y)$ gives the density at position (x, y). Let F_1 in Eq. (2) denote the force from robot at $(x + 1/D(x, y), y)$ to robot at (x, y). Let F^x denote the projection of force F along x-axis. Therefore $F_1^x(z) = -\alpha_1/(1/D(z))^3 + \alpha_2/(1/D(z))^2$ and Eq. (2) is transformed to

$$\frac{-\alpha_1}{(1/D(z))^3} + \frac{\alpha_2}{(1/D(z))^2} = -(F^x_{\text{nearby}-1}(z) + F^x_{\text{faraway}}(z)).$$

$F^x_{\text{nearby}-1}(z)$ is computed by approximating the positions of the nearby robots using the density function and the hexagonal structure. $F^x_{\text{faraway}}(z)$ is approximated by computing an integration, also using the density function. Let the right hand side value be denoted by $\delta(z)$. Rearrange the equation, we get the following

$$-\alpha_1 D(z) + \alpha_2 = \delta(z)/D(z)^2,$$
$$D(z) = \left(\frac{\delta(z)}{D(z)^2} - \alpha_2\right) \Big/ (-\alpha_1).$$

We consider the left hand side $D(z)$ as $D^{t+1}(z)$ and the $D(z)$ in right hand side $D^t(z)$, where t is the number of iterations. Therefore we get an iterative equation:

$$D^{t+1}(z) = \left(\frac{\delta(z)}{(D^t(z))^2} - \alpha_2\right) \Big/ (-\alpha_1).$$

Suppose we have computed $D^t(z)$. To compute $D^{t+1}(z)$ is to compute the coefficient vector C from the above equation, and it can be done in the following way. Take m sample positions $z_1, z_2, \ldots, z_m$. For each position z_i, we can compute

$$b_i = \left(\frac{\delta(z_i)}{(D^t(z_i))^2} - \alpha_2\right) \Big/ (-\alpha_1)$$

numerically. Let $B = (b_1, b_2, \ldots, b_m)$ be a vector of m so computed values. Then we have the following linear system

$$MC = B,$$

where M is a matrix of $m \times l$, with the ith row of M being a vector

$$(1, x_i^1 y_i^0, x_i^0 y_i^1, x_i^2 y_i^0, x_i^1 y_i^1, x_i^0 y_i^2, \ldots, x_i^0 y_i^k),$$

where $(x_i, y_i) = z_i$. C can be solved using the Least Square approximation. Thus we can compute $D^{t+1}(z)$ from $D^t(z)$. Iterating this procedure, we can find the density function.

4.1.3. A single cluster of working robots with several control robots

In this scenario, the system consists of two groups of robots, the group Worker and the group Controller. A robot in the Worker group is an ordinary robot, while a robot in Controller is a landmark robot. The controllers are static and their positions are fixed by the global controller in order to control the distributions of the workers. While there is an identical force law among the workers, there is also a force law from the group Controller to the group Worker. In this case, the total force on a worker robot at position x can be approximated as

$$F_{\text{nearby}}(x) + F_{\text{faraway}}(x) + F_{\text{control}}(x),$$

where as before $F_{\text{nearby}}(x)$ is the total force from worker robots close by and can be computed discretely, and $F_{\text{faraway}}(x)$ is the total force from worker robots faraway and can be approximated by an integration. $F_{\text{control}}(x)$ is the total force from all the controller robots, and can be computed discretely by summing up the forces from all controllers, since usually the number of controllers should be small.

By rearranging, we get the following equation

$$F_1(x) = -(F_{\text{nearby}-1}(x) + F_{\text{faraway}}(x) + F_{\text{control}}(x)),$$

which is quite similar to Eq. (2). From here on, given specific force laws, we can derive in a way similar to that described in Section 4.1 and get an iterative equation in terms of the density function.

Again note that in approximating the density function, we do not need to make any assumption about the shape of the cluster distribution. The shape should be implied by the density function. However, we do need to assume that the structure of the distribution is hexagonal in 2D, even with control robots. This is a reasonable assumption when the number of working robots is huge. The structure of the distribution far away from the control robots may well be hexagonal, while the structure close to a control robot may be deformed.

4.2. Defining force laws for given density functions

Let $D^*(x)$ be the given density function, where x is a position in a d-dimensional Euclidean space. Let $D_F(x)$ be the density function resulted from a set F of force laws; $D_F(x)$ can be approximated using the method discussed in Section 4.1. The problem of finding force laws to achieve the given density function is to find the set F such that

$$\min_F \int (D_F(x) - D^*(x))^2$$

is achieved.

We make the assumption that *the forms of the force laws are fixed and known, and only the parameters (coefficients and exponents) are to be determined.* For example, we can assume that all the force laws have the same form as in Eq. (1), and only c_1, c_2, σ_1, and σ_2 are to be determined. Let A denote the vector of the parameters and $D_A(x)$ the density function resulted from the force laws with parameters described by A. Let $H = \int (D_A(x) - D^*(x))^2$. Minimizing H is equivalent to solving the equation $\partial H/\partial A = 0$. Since H is not an explicit function of A, we can use Quasi Newton Method to solve it as follows.

Given $D^*(x)$, we start with A^0, an initial guess of the parameters, and iterate to find better guesses using the following equation:

$$A^{n+1} = A^n - \frac{(\partial H/\partial A)(A^n)}{(\partial^2 H/\partial A^2)(A^n)}.$$

Note that since H is not an explicit function of A, we can approximate $(\partial H/\partial A)(A^n)$ by

$$\frac{H_{(A^n)'} - H_{A^n}}{(A^n)' - A^n},$$

where $(A^n)'$ has a small enough displacement from A^n. For $n > 1$, $(A^n)'$ can be A^{n-1}. $(\partial^2 H/\partial A^2)(A^n)$ can be approximated in a similar way.

At each step, we need to compute H_{A^n}. This comes down to computing $D_{A^n}(x)$, whose calculation has been described in Section 4.2.

5. An extension to social potential fields: Spring laws

5.1. Motivation: Forming exact structures

In this section, we extend the social potential fields method to use spring laws as force laws. Besides the different forms of the force laws, there are two significant differences between the inverse-power law model

and the spring law model. In the spring law model, the force law defined from robot i to robot j is the same as that from robot j to robot i, while the force laws are not necessarily symmetric in the inverse-power law model. In the spring law model, a force law defined between two robots is only one and not a summation of several spring laws as a force law may be a summation of several inverse-power laws in the inverse-power law model. Because of the above two differences, force law relations among robots in the spring law model can be abstracted to an undirected graph, where each vertex represents a robot and each undirected edge a spring (thus a spring law) between two robots. If this graph has certain properties, we can control the robots to automatically form and maintain predefined structures. Note that the inverse-power law model can only let the robots form certain distributions (which may be dynamic), but does not have the power to guide the robots to form exact structures. This is why we are interested in choosing spring laws as another type of control laws. We will see that by designing proper spring law relations among the robots, we not only can let the robots form predefined structures, but also can let the robots change from one structure to another dynamically by controlling explicitly only a few robots.

There are situations where we want a group of robots to form a predefined structure, to maintain the structure while moving around, and to change from one structure to another dynamically. For example, in military maneuvers or warfare, we often see that troops, battle flights or submarines, when attacking, withdrawing or moving, keep certain formations in order to protect or to attack more efficiently. These defensive or offensive phalanxes change from one form to another under different circumstances. When we want the robots to do some industrial jobs such as harvesting, cleaning, and fishing, we want the robots to keep certain assemblies. For example, we may want the robots to form a harvesting line which sweeps from one end of a field to the other end. In this case, the robots should be equally spaced along the line so the work load is balanced. When a robot is confronted with an obstacle, it should be allowed the flexibility to move out of its way to get around the obstacle. After that, it should go back to its proper place.

In the following, we first introduce the concept of rigidity in Section 5.2. We will see that rigidity is the desired property of the graphs representing the spring law relations among robots. Then in Section 5.3, we discuss the relation between graph connectivity and rigidity. The relationship provides us a way to design rigid graphs. Finally Section 5.4 is about applying spring law model to robotic control.

5.2. Rigidity of spring graphs

Let G be a finite undirected graph with a set of vertices $1, \ldots, n$ and with a non empty set E of undirected edges. Each element in E is designated as an ideal spring. Let $L = \{\langle l_{ij}, k_{ij}\rangle\}$ defines springs between vertex i and vertex j for all $(i, j) \in E$, where l_{ij} is the length without compression or extension, and $k_{ij} > 0$ is the force constant, of the spring between i and j. A graph G with a spring relation L is called a *spring graph*, and is denoted by G_L. If two vertices i and j connected by a spring $\langle l_{ij}, k_{ij}\rangle$ are at distance r_{ij} apart, according to Hooke's Law, the magnitude of the force between the two vertices is:

$$f_{ij} = k_{ij}|r_{ij} - l_{ij}|,$$

where the potential energy stored is given by

$$P_{ij} = \tfrac{1}{2}k_{ij}(r_{ij} - l_{ij})^2.$$

An *embedding* of G is an assignment of the vertices into a d-dimensional Euclidean space R^d. Let $p = (p_1, \ldots, p_n)$ be an embedding of G, where p_i is the position of vertex i in R^d. The potential energy of a particular embedding p of G_L is

$$E_L(p) = \frac{1}{2}\sum_{ij} k_{ij}(\|p_i - p_j\| - l_{ij})^2.$$

The minimum of $E_L(p)$ is achieved when $\nabla E = 0$. This happens when all the vertices are at their equilibrium states, i.e. when

$$F_i = \sum_{(i,j)\in E} F_{ij} = \sum_{(i,j)\in E} k_{ij}(r_{ij} - l_{ij})\frac{p_i - p_j}{r_{ij}} = 0, \tag{3}$$

for all vertices, where $r_{ij} = \|p_i - p_j\|$.

Imagine that the vertices of a spring graph are robots, and the edges represent spring force laws defined among them. As in the inverse-power law model, we let the robots calculate the resultant forces

from all other robots and move in a way to reduce the resultant forces. The robots, with some damping factors, will thus converge to a minimum-energy embedding. If we can design the spring graph such that the minimum-energy embedding is the same as a predefined structure, then we have found a method in controlling the robots to form a predefined structure automatically and distributedly.

However, if the spring graph has more than one minimum-energy embeddings, then we do not know to which embedding the system will converge. Thus it is desirable to design spring graphs that have unique minimum-energy embeddings. A spring graph G_L, i.e. a graph G with a spring relation defined by L, is *rigid* in a d-dimensional Euclidean space if there is a unique embedding (up to translation and rotation) of G into the space which carries the minimum potential energy. Note that with rigid spring graphs, we do not have to worry about local-minima problem.

The relation between rigidity and graph connectivity discussed in Section 5.3 provides us a way to design rigid spring graphs.

5.3. Graph connectivity and rigidity

A graph G is *k-connected* if there is no subset of k vertices, which if deleted, disconnects G. [19] studies the relation between graph connectivity and rigidity. One result points out that if a graph G is $(d+1)$-connected, then for almost any spring relation L (that is all except for certain singular spring relations), the spring graph G_L is rigid, i.e. it has a unique minimum-energy embedding. (Note that the minimum-energy embeddings may differ and depend on specific spring relations for the same graph.) Therefore graph connectivity provides a highly accurate and simple way in designing rigid spring graphs.

Fix an arbitrary subset $C \subset V$ of $d+1$ vertices and fix an arbitrary embedding of the vertices of C in $\mathbb{R}^d$. A *C-embedding* of G is an embedding in $\mathbb{R}^d$ consistent with the already fixed embedding of C. Let G be a $(d+1)$-connected graph and G_L a rigid spring graph. Let C be an arbitrary subset of $d+1$ vertices and let C be embedded. It is further pointed out in [19] that the C-embedding that carries the minimum potential energy among all C-embeddings of G is also unique. We call this C-embedding with the minimum energy the C-minimum-energy embedding. It is easy to see that the C-minimum-energy embeddings depend not only on the spring graphs but also on the choices and embeddings of C. By changing the embeddings of the subset C, we can get different unique C-minimum-energy embeddings.

As a special case, if a graph G is 3-connected (4-connected respectively), then almost all of its spring graphs are rigid in 2D (3D respectively). If we choose a subset C of 3 (4 respectively) vertices and fix an embedding of C, then we have uniquely determined a C-minimum-energy embedding in 2D (3D respectively).

Notice that in a C-minimum-energy embedding, Eq. (3) holds for all vertices not in C. Let G be a $(d+1)$-connected graph and let one of its rigid spring graph G_L represent the spring laws defined among robots (vertices of the graph). We can choose a subset C of $d+1$ robots as leading robots and let the rest be ordinary robots. The leading robots are controlled by the global controller to a certain embedding. The rest of the robots, whose motions are controlled by the spring laws, will converge to the unique C-minimum-energy embedding. This sketches the approach of using spring law control to form predefined structures, and the details can be found in Section 5.4.

5.4. Applying spring laws to distributed control

Given a predefined structure in a d-dimensional Euclidean space, applying spring law control to let the robots form and maintain the structure goes as follows. First, we need to design a $(d+1)$-connected graph with a proper spring relation, to choose a subset C of $d+1$ vertices, and to embed C into the d-dimensional Euclidean space, such that the C-minimum-energy embedding is the same as the given structure.

The vertices represent robots, with vertices in C as leading robots and the rest of the vertices as ordinary robots. A leading robot is controlled explicitly by the global controller, while an ordinary robot's motion is controlled by the spring forces from other robots. An ordinary robot i stores a table of $\langle l_{ij}, k_{ij}\rangle$, for $(i,j) \in E$. The robot calculates a force using the following expression:

$$F_i = \sum_{(i,j)\in E} F_{ij} = \sum_{(i,j)\in E} k_{ij}(r_{ij} - l_{ij})\frac{p_i - p_j}{r_{ij}},$$

where p_i and p_j are the positions of robots i and j respectively, and $r_{ij} = \|p_i - p_j\|$. Then as in the inverse-power law model, robot i moves in a way to reduce F_i. (Note that there are no physical springs between robots so the motions of the robots are not restricted in any way by springs.) With all ordinary robots operate simultaneously in this way and with the leading robots at the appropriate positions, the system, with some damping factors, is expected to converge to the given structure, which is the same as the C-minimum-energy embedding.

Once a structure is formed, to control the robots to move around while maintaining the structure can be done as follows. Let the global controller control the leading robots to move while maintaining their relative positions. When the leading robots move, the structure is deformed and no longer carries the minimum potential energy. The ordinary robots are no longer at their equilibrium states and they will move in order to restore the minimum-energy embedding, which is the predefined structure. When the minimum-energy embedding is restored again, it looks as if the whole robot system moves while maintaining the predefined structure. Note that how fast the system moves can also be controlled by controlling the speed of the leading robots. The structure can also be changed by changing the relative positions of the leading robots. A simple case is to scale the structure by scaling the relative distances between the leading robots. An ordinary robot is allowed the flexibility to overcome local obstacles and to perform tasks. After a deformation, a robot will tend to go back to its correct position in the structure.

In the description given so far, an important step is left out, which is to design the spring graph and to embed a subset of vertices to realize the predefined structure. Here we provide a simple method (not the only or the best way) of doing this. All the springs will have the same force constant, say a real number $k > 0$. Suppose that the distance between robot i and robot j in the given structure is r_{ij}, then we can add an edge between i and j designating a spring whose length without compression or extension is r_{ij}. We keep on adding edges in this manner until the graph is $(d + 1)$-connected. Note that for a graph to be $(d + 1)$-connected, each vertex of the graph has to have at least $d + 2$ edges. We can choose arbitrarily $d + 1$ vertices as the subset, and let their relative positions be the same as in the predefined structure. It is easy to see that the minimum-energy embedding is the same as the predefined structure, and in fact the potential energy in this embedding is zero.

Usually the regularity of a structure provides some guidelines in how to add edges. For example, if the structure is a lattice in 2D with more than 3 rows and more than 3 columns, we add edges between a robot and its four closest neighbors in the lattice. It can be shown that the resulted graph is 3-connected, and the spring graph is rigid.

If an application does not desire an entirely rigid structure, but flexible ensembles of rigid subcomponents, this can be accomplished by combining spring law control and inverse-power law control. Each rigid sub-component can be accomplished by spring law control, where the coordinations of the sub-components can be accomplished by inverse-power law control applied to the leading robots of the sub-components.

An advantage of this spring law approach is that we have an efficiently computable simulation method to guide the design, since a minimum-energy embedding can be computed in polynomial time by solving a linear system of size n by n, where n is the number of vertices in the graph. We can use computer simulations to guide in designing the springs between pairs of robots to let them display desired structures, lest to change the structures by changing the positions of a few leading robots.

6. Conclusions

In this paper, we have proposed a distributed method for autonomous multi-robot control, namely, the social potential fields method. It is a very simple and generic method. The force laws have simple forms, yet our simulations have shown that by designing proper force laws, we can let a system of robots display interesting and useful behaviors which may soon have practical applications in industry, military and other areas.

This paper presents the first study, and also a preliminary one, in applying potential fields to distributed robotic control. In drawing conclusions, we raise the following problems for future studies.

6.1. Convergence and local-minima problem

The convergence problem is to determine, for a given initial state and a set of force laws, whether a system of robots will converge and to what behavior it will converge. This is a hard problem in general for dynamic systems with iterative processes. The system may not converge or it may converge to one of many local minima.

We did the computer simulation *guarding the castle* twice, with exactly the same force laws but with different initial distributions. In one simulation, the robots were initially distributed around the castle. The system converged to a ring around the castle which is the desired behavior. In the second simulation, the robots were initially distributed in a square area far away from the castle. The robots converged to a disc on their way approaching the castle, but not surrounding the castle.

All known potential field methods can introduce many local minima. The local-minima problem has not been solved even in cases much simpler than ours (see [30]). A partial and practical solution to the local-minima problem resorts to the global controller. Once the global controller detects a local minimum situation, it can change force laws to help the system escape from the local minimum.

For a given problem, it will be interesting to do a systematic study and to categorize the conditions for convergence, by varying the initial state and the parameters of the force laws.

6.2. Robustness and efficiency

We believe that the social potential fields method is robust in that the method can tolerate errors in sensors and actuators. As a future work, we can test this claim by computer simulations. We can simulate the errors in sensing by adding random perturbations to the positional numbers used in force calculations. The errors in actuators can be simulated by perturbing randomly in the length and direction in which a robot moves. Then we can compare the results of a simulation with errors and one without to study the robustness of this method.

The effect of communication latency on equilibrium behaviors can also be studied by computer simulations. For example, previous instead of current positions of robots can be used in calculating the forces; in this case, each robot has to keep not only one but a series of positions.

We have yet to find appropriate criteria to use in judging the efficiency of our social potential fields method.

6.3. Loss of information

In the distributed control paradigm, each individual applies *local rules* to the current state of the system. The description of the state can be very complicated (e.g. Reynolds' simulation of birds) and thus the local rules can be very complicated. In our VLSR system, the state of the system is simplified to only the distances. Thus information about the system is lost. Due to this, some behaviors can not be accomplished by social potential methods.

The force laws should be kept simple so they can be manipulated but they should be complex enough to generate behaviors we need.

6.4. Lack of powerful tools for defining force laws

As we have mentioned at the beginning of Section 4, our method for defining force laws is quite restrictive. For example, the method can not compute the density function given a cluster of robots with different force laws defined among them.

Furthermore, our iterative method only deal with static equilibrium situations. This is because the iterative equation is derived from the fact that the resultant force on a single robot is equal to zero in the equilibrium state. But for practical purpose, many equilibrium states are dynamic. For example, in the simulation *guarding a castle*, the invader and the guards form a dynamic chase-and-run behavior. How to define dynamic equilibrium behaviors and how to define force laws to achieve them need further study.

Acknowledgements

We wish to thank Zheng Sun for his generous help in editing figures and for the description of the demining simulation.

References

[1] J. Barraquand, J.C. Latombe, Robot motion planning: A distributed representation approach, Technical Report, Dept. of Computer Science, Stanford University, 1989.

[2] G. Beni, The concept of Cellular Robot, in: IEEE Proceedings Symposium on Intelligent Control, 1988, pp. 57–61.

[3] G. Beni, S. Hackwood, Stationary waves in cyclic swarms, in: Proceedings IEEE International Symposium on Intelligent Control, 1992, pp. 234–242.

[4] G. Beni, J. Wang, Theoretical problems for the realization of Distributed Robotic Systems, in: Proceedings IEEE International Conference on Robotics and Automation, Sacramento, CA, 1991, pp. 1914–1920.

[5] C.M. Breder, Equations descriptive of fish schools and other animal aggregations, Ecology 35 (1954) 361–370.

[6] R.A. Brooks, P. Maes, M.J. Mataric, G. More, Lunar base construction robots, in: Proceedings IEEE International Workshop on Intelligent Robots and Systems, 1990, pp. 389–392.

[7] J.F. Canny, M.C. Lin, An opportunistic global path planner, in: Proceedings IEEE International Conference on Robotics and Automation, Cincinnati, OH, 1990, pp. 1554–1559.

[8] A. Colorni, M. Dorigo, V. Maniezzo, Distributed optimization by ant colonies, in: Proceedings First European Conference on Artificial Life, 1992, pp. 134–142.

[9] C.I. Connolly, J.B. Burns, R. Weiss, Path planning using Laplace's equation, in: Proceedings IEEE International Conference on Robotics and Automation, Cincinnati, OH, 1990, pp. 2102–2106.

[10] M. Erdmann, T. Lozano-Pérez, On Multiple moving objects, Technical Report, Artificial Intelligence Laboratory, MIT, Cambridge, MA, 1986.

[11] B. Faverjon, P. Tournassound, A practical approach to motion planning for manipulators with many degrees of freedom, in: Proceedings Fifth International Symposium on Robotics Research, 1989, pp. 65–73.

[12] S. Fortune, G. Wilfong, C. Yap, Coordinated motion of two robot arms, in: Proceedings IEEE Conference on Robotics and Automation, San Francisco, CA, 1986, pp. 1216–1223.

[13] L.F. Greengard, The Rapid Evaluation of Potential Fields in Particle Systems, MIT Press, Cambridge, MA, 1987.

[14] S. Gross, J.L. Deneubourg, Harvesting by a group of robots, in: Proceedings First European Conference on Artificial Life, 1992, pp. 195–204.

[15] J.E. Hopcroft, J.T. Schwartz, M. Sharir, On the complexity of motion planning for multiple independent objects: PSPACE hardness of the "warehouseman's problem". International Journal of Robotics Research 3 (4) (1984) 76–88.

[16] R.A. Jarvis, A perspective on range finding techniques for computer vision, IEEE Transactions on Pattern Analysis and Machine Intelligence 5 (2) (1983) 122–139.

[17] O. Khatib, Real time obstacle avoidance for manipulators and mobile robots, International Journal of Robotics Research 5 (1) (1986) 90–99.

[18] J.C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, Norwell, MA, 1991.

[19] N. Linial, L. Lovasz, A. Wigderson, Rubber bands, convex embeddings and graph connectivity, Combinatorica 8 (1) (1988) 91–102.

[20] M.J. Mataric, Distributed approaches to behavior control, in: Proceedings SPIE, vol. 1828, Sensor Fusion V, 1992, pp. 373–382.

[21] R.S. Miller, W.J.D. Stephen, Spatial relationships in flocks of sandhill cranes (Grus canadensis), Ecology 47 (1996) 323–327.

[22] Y. Moses, M. Tennenholtz, Artificial social systems. Part I: Basic principles, Technical Report CS90-12, Weizmann Institute, Rehovot, Israel, 1990.

[23] Y. Moses, M. Tennenholtz, On formal aspects of artificial social systems, Technical Report CS91-01, Weizmann Institute, Rehovot, Israel, 1991.

[24] L.E. Parker, Designing control laws for cooperative agent teams, in: Proceedings IEEE International Conference on Robotics and Automation, San Diego, CA, 1994, pp. 582–587.

[25] J. Reif, Complexity of the generalized movers problem, in: J. Hopcroft, J. Schwartz, M. Sharir (Eds.), Planning, Geometry and Complexity of Robot Motion, Ablex, Norwood, NJ, 1987, pp. 267–281.

[26] J. Reif, S.R. Tate, The complexity of N-body simulation, in: Proceedings Twenth Annual Colloquium on Automata, Languages and Programming (ICALP'93), 1993, pp. 162–176.

[27] C.W. Reynolds, Flocks, herd, and schools: A distributed behavioral model, Computer Graphics 21 (4) (1987) 25–34.

[28] E. Rimon, J.F. Canny, Construction of C-space roadmaps from local sensory data: What should the sensors look for? in: Proceedings IEEE International Conference on Robotics and Automation, San Diego, CA, 1994, pp. 117–123.

[29] E. Rimon, D.E. Koditschek, The construction of analytic diffeomorphisms for exact robot navigation on star worlds, Transactions of the American Mathematical Society 327 (1) (1991) 71–116.

[30] E. Rimon, D.E. Koditschek, Exact robot navigation using artificial potential functions, IEEE Transactions on Robotics and Automation 8 (1991) 501–518.

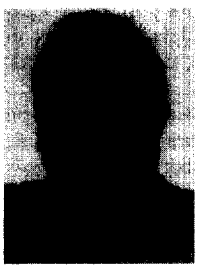
[31] J.T. Schwartz, M. Sharir, Algorithmic motion planning in robotics, in: J. van Leeuwen (Ed.), Algorithms and Complexity, Handbook of Theoretical Computer Science, vol. A, Elsevier, Amsterdam, 1990, Chapter 8, pp. 391–430.

[32] M. Sharir, S. Sifrony, Coordinated motion planning for two independent robots, in: Proceedings Fourth Annual ACM Symposium on Computational Geometry, 1988, pp. 319–328.

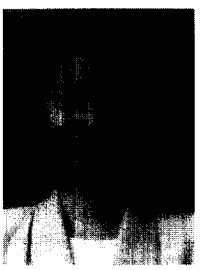
[33] Y. Shoham, M. Tennenholtz, On the synthesis of useful social laws for artificial agent societies, in: Proceedings Tenth National Conference on Artificial Intelligence (AAAI-92), San Jose, CA, 1992, pp. 276–281.

[34] K. Sugihara, I. Suzuki, Distributed motion coordination of multiple mobile robots, in: Proceedings Fifth IEEE

International Symposium on Intelligent Control, 1990, pp. 138–143.
[35] T. Ueyama, T. Fukuda, Self-organization of Cellular Robots using random walk with simple rules, in: Proceedings IEEE International Conference on Robotics and Automation, Atlanta, GA, 1993, pp. 595–600.
[36] K. Warburton, J. Lazarus, Tendency-distance models of social cohesion in animal groups, J. Theoretical Biology 150 (4) (1990) 473–488.
[37] C.W. Warren, Global path planning using artificial potential fields, in: Proceedings IEEE International Conference on Robotics and Automation, Scottsdale, AZ, 1989, pp. 316–321.
[38] E.O. Wilson, Sociobiology, the new synthesis, The Belknap Press of Harvard University Press, Cambridge, MA, 1985.



John H. Reif was born in 1951 in Madison, Wisconsin. He received in 1973 a magna cum laude B.S. in Applied Math and CS from Tufts University. He received his M.S. (1975) and Ph.D. (1977) in Applied Mathematics from Harvard University. From 1977–1978 he was Research Associate, University of Rochester. He was Assistant (1978–1979) and Associate Professor (1979–1996) at Harvard University. From 1996 he has been Professor in the Department of Computer Science at Duke University. He has been visiting Professor at CMU, MIT, and Berkeley. He has been awarded Fellow of ACM (1996), Fellow of IEEE (1993), and Fellow of Institute of Combinatorics (1991). His research combines theory and practice. Although primarily a theoretical computer scientist, he also has made a number of contributions to practical areas of computer science including parallel computation, robotics, data compression, molecular simulations, and optical computing. He has worked for many years on the development and analysis of parallel and randomized algorithms for various fundamental problems including solutions of large sparse systems, sorting, and graph problems. He has done a number of implementations of sophisticated parallel algorithms such as parallel nested dissection on massively parallel machines as well as implementations of parallel data compression algorithms into special purpose chips. His research is now focused particularly in emerging new areas such as nano-robotics and biomolecular computing.



Hongyan Wang was born in 1969, in Shanxi, China. She obtained her B.S. in Computer Science from Peking University, Beijing, China, in June 1991. She then continued her graduate study in the Department of Computer Science at Duke University, North Carolina, USA, with Professor Dr. John Reif. Her research interest is in the areas of algorithmic motion planning, computational geometry, and general algorithmic design. She received her Ph.D. in December 1996. She worked as a software engineer at Parametric Technology Corporation, Massachusetts, USA, from June 1996 to February 1998. Now she lives in Hiroshima, Japan with her family.