Adaptive Multi Agent System for Guiding Groups of People in Urban Areas

Anais Garrell, Oscar Sandoval-Torres and Alberto Sanfeliu

Abstract This article presents a new approach for guiding a group of people using an adaptive multi agent system. For the simulations of the group of people we use social forces, with theses forces human motion is controlled depending on the dynamic environment. To get the group of people being guide we use a set of agents that work cooperatively and they adapt their behavior according to the situation where they are working and how people react. For that reason, we present a model that overcomes the limitations of existing approaches, which are either tailored to tightly bounded environments, or based on unrealistic human behaviors. In particular we define a Discrete-Time- Motion model, which from one side represents the environment by means of a potential field, and on the other hand the motion models for people and robots respond to realistic situations, and for instance human behaviors such as leaving the group are considered. Furthermore, we present an analysis of forces actuating among agents and humans throughout simulations of different situations of robot and human configurations and behaviors. Finally, a new model of multi-robot task allocation applied to people guidance in urban settings is presented. The developed architecture overcomes some of the limitations of existing approaches, such as emergent cooperation or resource sharing.

1 Introduction

The interest on developing social and cooperative agents has significantly increased throughout the recent years. In this work we present a new approach for guiding people in open areas of urban settings using multiple agents acting in a cooperative way. One of the agents is the *leader*, as a human tour-guide. It is placed at the front of the group and its role is to estimate the trajectory of both the people and the rest of agents. The other agents, called *shepherds*, are responsible for guiding the people, preventing any person leaving the group, and following the path given by the leader, considering in every instant people reactions using social forces [1].

Furthermore, in this research, we go one step ahead, presenting a method to optimize locally the tasks assignment to agents for doing their missions. Agents assignation are done

Anais Garrell · Oscar Sandoval-Torres · Alberto Sanfeliu
e-mail: {agarrell, osandoval, sanfeliu}@iri.upc.edu
Institut de Robòtica i Informàtica Industrial - Universitat Politècnica de Catalunya

by analyzing the minimum work required to do such task, where the function to minimize is based on one hand, by agents motion (which will be applied with robot on the future), and, on the other hand, by the impact of such motions on peoples displacement.

Moreover, an orientation where the main question is not about the division of tasks between agents is presented. In the developed approach the participation to solve a task is not limited to a single agent. Agents will try to participate in the tasks that give them more benefits, even when the task is already being done by someone else. In many cases the tasks can be performed by more than one agent. This feature has not been explored so far by other existing architectures.

This paper contents has been distributed as follows. We start presenting the representation model of the environment and people behavior. In section III the cost function for the rescuing people task is presented. In section IV the MRTA model we are presenting is applied to the task of people guidance. And last but not least, the results and conclusions are presented in sections V and VI respectively.

2 Modeling The Environment and People motion

For modeling the environment where the agents (future robots) work, we have developed a model called Discrete Time Motion model (DTM) which has two components: The Discrete Time component and the Discrete Motion component. The former estimates position, orientation and velocity of the robots and persons, and the position of the obstacles at a time instance k. It will be used to estimate the intersection of the people with the obstacles and detect if someone is leaving the group. The Discrete Motion component estimates the change of position, orientation and velocity of people and robots between two time instances k and $k+p$. It will be used to compute the robots' trajectory to reach the goal while preventing people leaving the group.

2.1 The Discrete Time Motion Model

The first task of the Discrete Time component is to estimate position, orientation and velocity of the robots and persons. This is done with a standard particle filter formulation [2].

Then, the Discrete Time component aims to represent the areas where the robots will be allowed to move, by means of potential fields. To this end, we define a set of functions that describe the tension produced by the obstacles, people and robots over the working area. These tensions are computed based on the area defined by a security region surrounding each one of the persons, robots and obstacles.

In order to decide the trajectories the robots will follow we will define a potential field over the working area, and perform path planning in it. In particular, the goal the robots try to reach will generate an attractive force pulling the robots towards it. On the other hand, the obstacles will generate a repulsive potential pushing a given robot away . We parameterized all these attractive and repulsive forces by Gaussian functions. For more detail of this model see [6].

$$T_p(\mu_p, \Sigma_p)(x) = \frac{1}{\left|\Sigma_p\right|^{1/2}(2\pi)^{n/2}} e^{-\frac{1}{2}(x-\mu_p)^T \Sigma_p^{-1}(x-\mu_p)} \quad (1)$$

2.2 Modeling People Motion

In order to model people's motion we will use the concepts introduced by the works of Helbing et al. [1], this research studies the dynamics of pedestrian crowds from the "social" point of view. More specifically, they describe the motion of pedestrians based on social forces which are the result of the internal motivations of the individuals to perform certain motions. For more information see [1].

Let us now explain mathematically. People usually take the shortest path, which may be formally represented as the shape of a open polygon with edges $\mathbf{r}_\alpha^1 \ldots \mathbf{r}_\alpha^n := \mathbf{r}_\alpha^0$, where α refers to a given person and $\mathbf{r}_\alpha^0$ the destination he/she wants to reach.

The desired motion direction $\mathbf{e}_\alpha(t)$ of a pedestrian α will then be: $\mathbf{e}_\alpha(t) := \frac{\mathbf{r}_\alpha^k - \mathbf{r}_\alpha(t)}{\left\|\mathbf{r}_\alpha^k - \mathbf{r}_\alpha(t)\right\|}$ where $\mathbf{r}_\alpha(t)$ is the *current position* and $\mathbf{r}_\alpha^k$ is the subsequent edge of the polygon that will be reached. A deviation of the desired speed, v_α^0, from the current velocity, $\mathbf{v}_\alpha^0(t) := v_\alpha^0 \mathbf{e}_\alpha(t)$, may also exist due to deceleration or obstacle avoidance processes:

$$\mathbf{F}_\alpha^0(\mathbf{v}_\alpha, v_\alpha^0 \mathbf{e}_\alpha) := \frac{1}{\tau_\alpha}(v_\alpha^0 \mathbf{e}_\alpha - \mathbf{v}_\alpha) \tag{2}$$

where τ_α is a relaxation term. In practice we set the term τ to 0.5 for all the pedestrians [1].

3 Adaptation Model for Rescuing People

One of the biggest issues when guiding a group of people using multi agents, it is the possibility that a person or some people escape from the formation, in this case the agents have to adapt to the new situation to solve the new task. To this end, we will speak in term of robot instead of agent because all this theory will be applied to robots. The cost function, described below, speaks in Work terms, and it can be divided into two blocks: *(i)* Robot work motion, and *(ii)* Human work motion. In order to know what robots' tasks are, we have considered the following situations: *(i)* One robot has to look for the person (or people) that can potentially escape from the crowd formation and push him (or them) to regroup him (or them) into group, *(ii)* one robot has to go behind the people in order to push them in case that the crowd formation is broken down while the Leader guides the formation.

Firstly, the leader robot computes a path planning and moves to the next point. We also assume that there exists a *drag force* that will attract people behind the robot. Here, the robot has only to move from the present position to the next one of the guiding path. The *Pushing task* occurs when the robot pushes a person that has gone away in order to reach the crowd formation. This task can be also applied when a robot pushes a person (or people) who is (are) going behind the crowd formation in order to regroup people when the formation is broken down. Finally, *Crowd traversing task*, where the robot has to move through the formation to achieve the estimated position of the person that goes away from the crowd formation. In order to compute the dragging, pushing and crowd traversing forces, we use the equations defined in previous works on human behavior with other individuals [1]. Working with autonomous mobile robots, the robot *i* work motion is expressed by:

$$f_i^{mot} = m_i a_i; \quad W_i^{mot} = f_i^{mot} \Delta s_i \tag{3}$$

where m_i is the mass of the i-th robot, a_i its acceleration and Δx_i the space traversed by the robot to achieve its goal.

In this problem it is necessary to consider the *dragging*, *pushing* and *crowd intrusion forces* that robot's motion produces and that can affect to people. This component is called *Human Work Motion*, and it is the expense of people's movements as a result of robot's motions. As it has been mentioned several times in this paper, the group follows the robot guide/leader, and there is a set of robots that help to achieve their goal.

The dragging force is necessary when the leader robot guides the group of people from one place to another. It acts as an attractive force, hence the force applied by robot leader i to each person j is:

$$f_{ij}^{drag}(t) = -C_{ij}\mathbf{n_{ij}}(t) = -C_{ij}\frac{x_i(t)-x_j(t)}{d_{ij}(t)}; \quad d_{ij}(t) = \left\|x_i(t)-x_j(t)\right\| \tag{4}$$

where $d_{ij}(t)$ is the normalizated vector pointing from person j to robot i at instant t. C_{ij} reflects the attraction coefficient over the individual j, and it depends on the distance between the robot leader and person j. Thus, the dragging work that robot leader applied to each individual is defined by:

$$W_{drag} = \sum_{\forall \text{ person j}} f_{ij}^{drag}\Delta s_j \tag{5}$$

Where Δs_j is the distance traveled by the person j.

The *Pushing force* is given by the repulsive effect developed by shepherding robot on the group of people, for regrouping a person (or the broken crowd) in the main crowd formation. This repulsive force is due by the intrusion of the robot in the people's living space, which is five feet around humans. The territorial effect may be described as a repulsive social force:

$$f_{ij}^{push} = A_i exp^{(r_{ij}-d_{ij})/B_i}\mathbf{n_{ij}}\left(\lambda_i + (1+\lambda_i)\frac{1+cos(\varphi_{ij})}{2}\right) \tag{6}$$

Where A_i is the interaction strength, $r_{ij} = r_i + r_j$ the sum of the radiis of robot i and person j, usually people has radii of one meter, and robots 1.5 m, B_i parameter of repulsive interaction, $d_{ij}(t) = \left\|x_i(t)-x_j(t)\right\|$ is the distance of the mass center of robot i and person j. Finally, with the choice $\lambda < 1$, the parameter reflects the situation in front of a pedestrian has a larger impact on his behavior than things happening behind. The angle $\varphi_{ij}(t)$ denotes the angle between the direction $\mathbf{e}_i(t)$ of motion and the direction $-\mathbf{n_{ij}}(t)$ of the object exerting the repulsive force. We can write pushing work by:

$$W_{push} = \sum_{\forall \text{ person in } \Omega_i} f_{ij}^{push}(t)\Delta s_j \tag{7}$$

Where Ω_i is the set of people in which one of the helper robots have reached the living space.

And last but not least, the *Traversing force* is determined by the forces applied by the robot when is traversing the crowd. For security reasons, we have considered in this research that the value of this force is infinity, so we will ensure that a robot will not cross the crowd in order to avoid any damage.

The cost function for agent (robot) i, given a specific task, is the following one:

$$W_i = \delta_{mot}W_i^{mot} + \delta_{drag}W_i^{drag} + + \delta_{push}W_i^{push} + \delta_{trav}W_i^{trav} \tag{8}$$

$$\text{where } \delta_k = \begin{cases} 1 \text{ if this task is assigned} \\ 0 \text{ if this task is not assigned} \end{cases}$$

Where *k* could be *pushing, dragging, traversing* or *motion*. For each period of time, the leader and shepherded agents (robots) will be given a task in the guiding mission, which will imply one or several robot motion works and human robot works.

Finally, the task assignment for the agents (robots) will be the one which minimizes the minimum assigned work cost required to do the global task. It is computed by the following way:

$$C = argmin\{W_{total}(c)\},\ \forall \text{ configuration c} \tag{9}$$

where the *Configurations* mean how the tasks are distributed among the agents, for each configuration *c* agents compute W_{total} which is the addition of all W_i for all agents *i* that are working cooperatively.

4 Multi Agent Coordination

In the previous section, we have described a const function that it would be used when one person or more people escape from the formation. In the present section we will define a new approach of Multi Robot Task Allocation (MRTA). Our proposal addresses the challenge of people guidance in a way no previously explored, using MRTA. In agents and robotics systems there are many approaches to task allocation, but almost all solve tasks that does not involve human interaction. Other shortcoming in current architectures is that most of them assign one task to one robot, limiting the capacities of robot teams to work cooperatively. A good review and analysis of current multi-robot task allocation architectures is [3] of Gerkey and Matarić.

In general, existing proposals attempt to allocate one task to one robot. Only a few consider the case for cooperating to solve a task, and with the exception of ASyMTRe [5] in which cooperation is somehow preset on the system definition, cooperation only occurs when there are special situations (i.e. errors) during execution.

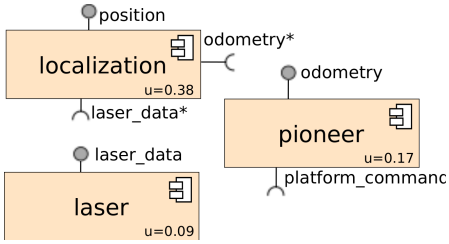


Fig. 1 A component includes uncertainty *u* and required ports (marked with an asterisk).

Our architecture, Selfish Task Allocation (STA), follows the Component Based Development, meaning that all the software to be used inside STA must be constructed as a component (see figure 1, therefore it must have a name and include input and/or output ports. Furthermore we add uncertainty to each component and the property *required* to each port.

Each component $C_i = (n^{C_i}, P^{C_i}, u^{C_i})$, has a name n^{C_i}, a list of ports $P^{C_i} = \{P_1^{C_i}, P_2^{C_i}, P_3^{C_i}, ..., P_m^{C_i}\}$, and an uncertainty value u^{C_i}. Each port $P_j^{C_i} = (t^{P_j^{C_i}}, w^{P_j^{C_i}}, r^{P_j^{C_i}})$ is described by an *information*

type $t^{P_j^{C_i}}$ that represents the kind of information transported in the port, $w^{P_j^{C_i}}$ is a binary value that stores the *direction* of the port (input or output), and $r^{P_j^{C_i}}$ that indicates if a port is *required* or not to accomplish that component.

As each piece of software inside a robot should be designed as a component, a robot can be conceived as a collection of components $R = \{C_1, C_2, C_3, ..., C_n\}$.

Since our approach addresses the problem as a task allocation system, we need to formalize the list of tasks $T = \{T_1, T_2, T_3, ..., T_o\}$ and the description of each of that tasks $T_k = (n^{T_k}, g^{T_k}, s^{T_k})$.

where n^{T_k} represents the name of the task, g^{T_k} the geographic information, if any (for example in people guidance the position of the group to be guided and the goal of the group), and s^{T_k} the status of the task (unsolved, attempted or solved).

We decided to tackle the problem of multi-robot task allocation with a different point of view, instead of use a complex group algorithm to distribute tasks, our approach uses the single robot task selection algorithm, allowing each robot to select the task that better fits its capabilities. When all the robots selects their best task, a task management algorithm coordinates the actions of the robots on the same task and task allocation just emerge. This section explores the Single Robot Task Selection (SRTS) algorithm we develop for our architecture. We understand SRTS as the algorithm used to define which is the best task for each robot.

Our approach for single-robot task selection is inspired by the proposal of Tang and Matatić, ASyMTRe [5], which in turn is inspired in information invariants. Like previous proposals (information invariants and ASyMTRe), our proposal called Selfish Task Allocation (STA) *"allows robots to reason about how to solve a task based upon the fundamental information needed to accomplish the task"* [5].

The Single-Robot Task Selection (SRTS) algorithm, part of STA, begins when it receives the list of tasks T, then $\forall T_k \in T$, the SRTS algorithm searches if $\exists C_i$ s.t. $n^{C_i} = n^{T_k}$, once C_i is found the algorithm tries to *activate* that component satisfying the following rules:

1. C_i can be activated iff $\forall P_j^{C_i} \in P^{C_i}$ where $r^{P_j^{C_i}} = true$ can be connected to a *compatible port* (see rule 2). At the same time components providing those ports must be activated.
2. Two ports $P_a^{C_i}$ and $P_b^{C_i}$ are compatibles iff they have the same information type $t^{P_a^{C_i}} = t^{P_b^{C_i}}$ and opposite direction $w^{P_a^{C_i}} \neq w^{P_b^{C_i}}$.

As result of the search some solutions could be obtained, we represent each solution as the list of the components involved in the solution of the task $S_l = \{C_x, C_y, ...\} \subset R$. And as can be inferred, if there exists more that one algorithm to solve the same problem (particle filter- and odometry-based localization) or if two or more components provide the same information type, could exist many ways to satisfy the required port of a component, and therefore many alternative solutions for each task $S = \{S_1, S_2, ..., S_l\}$ between all these alternatives, to select the task to be performed we use the solution with less uncertainty in connections u_{T_x} and displacement $u_D(pos, g_{T_k})$.

$$T_s = T_x, \text{ s.t. } u_{T_x} = \min_{\forall T_k \in T} u_{T_k} \tag{10}$$

with, $u_{T_k} = u_D(pos, g^{T_k}) + u_{S_m}$, s.t. $u_{S_m} = \min_{\forall S_l \in S} u_{S_l}$, where, u_{S_m} represents the solution S_l with less uncertainty, $u_D(pos, g^{T_k})$ the displacement uncertainty, uncertainty of reach the position of the task g_{T_k} from the robot position *pos* (only for mobile robots), u_{T_k} the uncer-

tainty of T_k when the solution with less uncertainty is selected, and T_s the finally selected task.

Once the robot selects the task that it can perform with less uncertainty (T_s), it tries to tackle that task, and probably other robot is already working on it. We propose an algorithm to face up the possible issues generated for the multi-robot task tackling.

We distinguish two main challenges when more than one robot acts over one single task, the first related to task-specific behavior (i.e. the position of each robot in cooperative box pushing) and the second related to detect when robots become hindering each other.

Our approximation to the task-specific coordination challenge lies on the assignment of an *id* to each robot that joints to solve the same task, the id assignment and the number of robots participating in the task is managed in a distributed manner and is only valid inside that task.

The id and the number of robots into each task must be used by the component designer to define the robot specific behavior for each task.

To face up the challenge of detect when there are too many robots tackling the same task, we propose the use of performance functions. While acting, each robot continuously calculates the performance of the team in the specific task

$$p_t^{T_k} = P^{T_k}(V_{t-1}^{T_k}, V_t^{T_k}) \tag{11}$$

Where, $V_t^{T_k}$ represents the value of environment variables involved in T_k at moment t. $P^{T_k}()$ The specific performance function for T_k. And $p_t^{T_k}$ the performance of T_k at time t.

As mentioned before, in our proposal of task selection each robot chooses the task that better fits their capabilities. But this selection mechanism means that many times the robots select the same task. Here it is when performance metrics make sense. When a robot tries to participate in a task where other robots are already working, it is accepted in test mode, all the robots adapt their behavior to the new number of robots and they continue tackling the task and getting performance information, but after a predefined time, the performance of the team is compared with the performance stored before the new member inclusion, if it was increased then the robot is considered now as part of the team, else the robot is asked to leave the task.

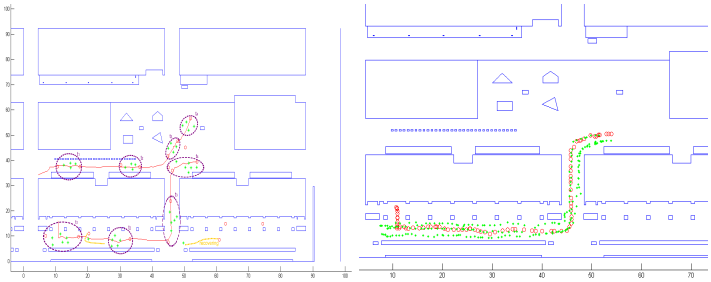
5 Implementation and Results

The results we will present correspond to different synthetic experiments, some of them within the previous map. In these experiments, the dynamical models of the persons –we considered a group of 5 persons– will follow the models described in the Section described before. In figure 3 some instants of time of a group of agent working cooperatively solving the task of guiding a group of people is shown.

We perform some experiments where one group was guided, we include in the simulated persons the ability to randomly leave the formation, to test our cost function and to study which is the recovery component and prove the behavior of the proposed architecture in group splitting. In fig. 2 it can be seen that when people leave the group a recover task is added to the list and some robot reacts to solve this task.

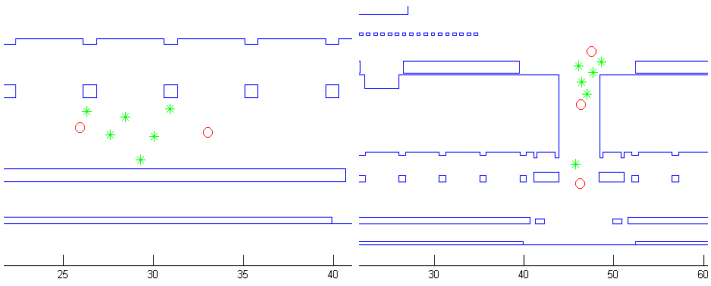
6 Conclusions

We have presented a new model to guide people in urban areas with a set of multi agents that work cooperatively and are able to adapt their behavior depending on people motion.



(a) (b)

Fig. 2 (a) Two different group of people are being guided by groups of cooperative agents. Several instants of time and the entire trajectory are shown. (b)Entire trajectory of people and robots in a guiding people mission.



(a) (b)

Fig. 3 (a) A group of people is being guided by two agents (red circles), in this occasion no human escapes from the formation, the task is being done correctly. (b) Two agents are guiding the group while a third agent is rescuing somebody who tries to escape.

In contrast to existing approaches, our method can tackle more realistic situations, such as dealing with large environments with obstacles, or regrouping people who left the group. For that reason, this work can be applied in some real robots applications, for instance, guiding people in emergency areas, or acting as a robot companion.

References

1. Helbing D and Molnar P (1995) Social force model for pedestrian dynamics. In: Physical Review E 51.
2. Arulampalam MS, Maskell S, Gordon N, Clapp T, Sci D, Organ T and Adelaide SA (2002) A tutorial on particle filters for online nonlinear/non-GaussianBayesian tracking. In: IEEE Transactions on Signal Processing
3. Gerkey BP and Matarić MJ (2004) A formal analysis and taxonomy of task allocation in multi-robot systems. In: The International Journal of Robotics Research
4. Lien JM, Rodriguez S, Malric JP and Amato NM (2005) Shepherding Behaviors with Multiple Shepherds. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA)
5. Tang Y and Parker LE (2008) Towards Schema-Based, Constructivist Robot Learning: Validating an Evolutionary Search Algorithm for Schema Chunking. In: ICRA'08.
6. Garrell A, Sanfeliu A and Moreno-Noguer F (2009) Discrete Time Motion Model for Guiding People in Urban Areas using Multiple Robots. In: IEEE/RSJ IROS'09.