2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Meso-Scale Planning for Multi-Agent Navigation

Liang He Jur van den Berg

***Abstract*— We introduce a new concept; *meso-scale* planning in real-time multi-agent navigation. Whereas many traditional approaches to multi-agent navigation typically consist of two-levels — a macro-scale level providing agents with a global direction of motion around (large) static obstacles, and a micro-scale level in which agents seek to avoid collision with other agents — our approach adds a meso-scale level to give agents realistic behavior in scenarios where groups of other agents (e.g. families or crowds in a virtual world) form coherent entities. Rather than moving straight through such groups, our approach lets agents move around them. Our formulation considers each agent as an individual that may perceive sets of other agents as a group, and plans its motion accordingly. We base our approach on the *velocity obstacle* concept, and we show using simulation results that our method dramatically improves the quality of the trajectories computed for the agents.**

I. Introduction

Real-time multi-agent navigation in an important topic in robotics that has received considerable attention in recent years. Applications range from simulating human behavior in crowds to creating believable motion for animated characters in games and virtual worlds to letting robots autonomously navigate environments containing other robots, humans, and/or other (moving) obstacles. Many current approaches to real-time multi-agent navigation operate using two levels of planning [22]: a macro-scale level where each agent plans a path to its goal that avoids (large) static obstacles, e.g. using a PRM-roadmap [13], an RRT-tree [17], [10], or A* algorithms based on a cell-decomposition of the free space [16], [18]. And a micro-scale level where each agent executes its path while making sure collisions with surrounding agents and moving obstacles are avoided, e.g. using reciprocal collision avoidance [23], potential fields [15], [12], [9], or rule-based methods [21], [20]. Such multi-agent navigation approaches characterize themselves by computing motions for each agent independently, in a distributed fashion, as opposed to (non-real-time) centralized multi-robot planning approaches [11] that coordinate the motion of agents such that they achieve their goals.

A shortcoming of the two-level approach to multi-agent navigation is that when a group of other agents form a coherent entity, e.g. a family or a crowd in a virtual-character simulation, the computed motion may exhibit suboptimal or unrealistic behavior. The agent may try to find its way straight through the group, as it lacks the capability at the micro-scale level to consider the situation on a slightly higher level and realize that moving around the group is more efficient (see Fig. 1). To this end, we propose in this paper

The authors are with the of School of Computing, University of Utah. E-mail: `lianghe.hust@gmail.com`, `berg@cs.utah.edu`.

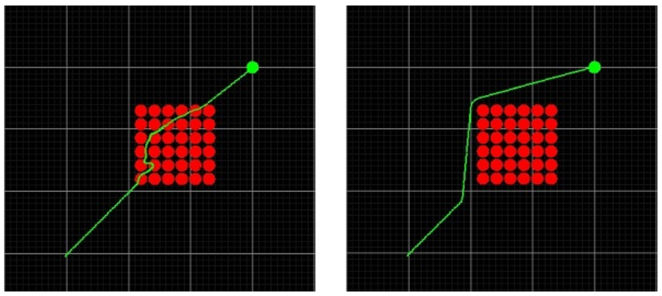


Fig. 1. The left picture shows the trajectory of an agent (green) using a traditional multi-agent navigation approach based on reciprocal collision avoidance. The agent moves straight through the group. The right picture shows the trajectory of the agent using our approach that includes meso-scale navigation. The agent moves around the group and reaches its target sooner. (The motion of the agents in the group (red) is not shown.)

an approach to multi-agent navigation based on three levels: macro-scale planning, micro-scale collision avoidance, and introducing a *meso-scale* level in which agents may consider sets of other agents as a coherent moving obstacle. The result is a navigation framework in which the three levels of planning seamlessly fit together, and that produces more realistic and efficient motion in multi-agent simulations.

We base our approach on the concept of *velocity obstacles* [2], [5]. On the macro-scale a path to the goal provides the agent with a *preferred velocity*; the velocity the agent would choose if no other agents were standing in the way. On the meso-scale each agent considers neighboring agents and clusters them according to similarity in position and velocity. These clusters of agents are then considered as coherent moving obstacles, and using the velocity obstacle concept an *adapted preferred velocity* is computed that would avoid collision with the groups of agents. On the micro-scale this adapted preferred velocity is then taken as an input and the actual velocity is computed based on immediately neighboring agents using a traditional multi-agent collision avoidance approach. In this paper we use *reciprocal collision avoidance* based on [23]. It should be noted that in our approach, each agent is considered as an individual who may perceive sets of other agents as coherent groups; each agent is not aware that it may be part of a group itself, and each agent has its own perception of whether agents constitute groups. Hence, our method remains a strictly distributed approach, in which there is no explicit coordination or communication among agents. Using simulation experiments, we show that our approach yields more realistic and efficient motions in several virtual-world scenarios.

The rest of paper is organized as follows. Firstly we review the prior methods for multi-agent navigation in Section II.

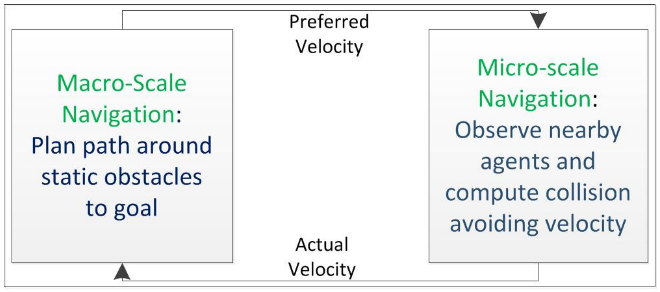
We formally define the problem we discuss in this paper in Section III. In Section IV we present our meso-scale planning approach, and in Section V we show how micro-, meso-, and macro-scale come together. In Section VI we present our simulation results and then conclude in Section VII.

II. Prior Work

There is a large body of work in distributed multi-agent navigation. Many frameworks assume the global path for each agent comes from some motion planning technique, e.g. PRMs [13], RRTs [17], [10], visibility graphs [4], or A* in a cell-decomposition of the free space [16], [18]. These global paths provide a preferred velocity or a preferred direction of motion for the agent, which then need to be adapted to avoid collisions with nearby agents. Many techniques exist for this local collision avoidance: So-called force-based or potential field techniques treat each agent as a point mass, where the direction of motion provides an attractive force, and nearby agents a repelling force [15], [9], [12]. Other techniques are rule-based, such as cellular automata [1], [21], or Reynolds flocking model [20], in which the motion of each agent is computed as a function of the motion of nearby agents. Reciprocal collision avoidance approaches [22], [7], [23] take a geometric approach based on velocity obstacles [5], assuming agents can observe each other's velocity and predict their future motion, and adapt their own assuming each agent takes 50% of the responsibility of avoiding pairwise collisions. Yet other collision-avoidance approaches are based on dynamic windows [6], rapid short-horizon replanning [19], etc.

Few multi-agent navigation approaches take into account meso-scale effects, where groups of agents can form coherent (moving) "obstacles" to other agents. The "PLEdestrians" approach [8] uses a grid overlaid on the free space in which for each cell agent density information is stored and updated over time. An A*-approach then finds a path for each agent through this density field that trades-off length and encountered density. Contrary to our method, this approach tackles the meso-scale obstacles in the macro-scale planning algorithm, which can be costly, because a new global path needs to be recomputed continually to account for changing agent densities. Also, the density map is assumed to be globally available to each agent, whereas our method takes a strictly distributed approach and reasons from the information that is available to each agent individually. Our approach to meso-scale planning can be combined with any macro-scale planning and micro-scale collision-avoidance approach mentioned above.

Other approaches related to our work consider a slightly different problem: how to plan a motion for a group of agents among other (groups of) agents. The approach of [3] replaces intermediate waypoints by "waylines" to make sure a group does not unrealistically funnel to a single point. In [14] it is explored how groups maintain coherence while each member of the group uses an individual collision-avoidance strategy based on velocity obstacles.


Preferred Velocity
Macro-Scale Navigation: Plan path around static obstacles to goal
Micro-scale Navigation: Observe nearby agents and compute collision avoiding velocity
Actual Velocity

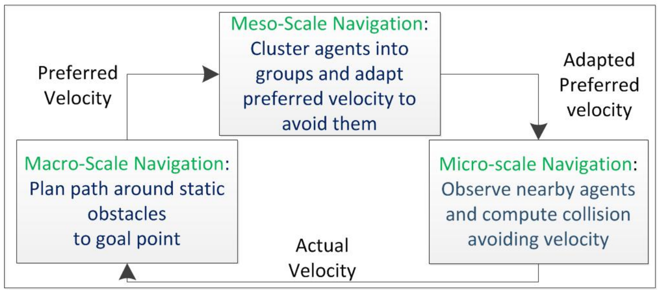

Meso-Scale Navigation: Cluster agents into groups and adapt preferred velocity to avoid them
Preferred Velocity
Adapted Preferred velocity
Macro-Scale Navigation: Plan path around static obstacles to goal point
Micro-scale Navigation: Observe nearby agents and compute collision avoiding velocity
Actual Velocity

Fig. 2. The top figure shows the general pipeline of traditional multi-agent navigation frameworks with a macro-scale planner and micro-scale collision avoidance. The bottom figure shows the pipeline of our approach, in which a meso-scale planner had been added.

III. Problem Definition and Global Approach

In the context of multi-agent navigation, we are given a (typically 2-D) environment containing static obstacles populated with multiple decision-making agents. Each agent i is considered to have a position $\mathbf{p}_i$, a velocity $\mathbf{v}_i$, and — assuming the geometry of the agent can be modeled as a disc— a radius r_i. Each agent has a particular goal position $\mathbf{g}_i$, and employs a continual cycle of sensing and acting, say with period Δt: in each cycle it observes nearby agents and static obstacles within a certain neighborhood, and based on this information the agent computes a new velocity for itself, which in turn leads to a new position for the agent in the next cycle. Each agent perfoms this cycle fully independently, as it cannot communicate or coordinate with other agents about its motion. The challenge is to compute the agents' new velocities in each cycle so that agents do not collide with each other and each agent ultimately reaches its goal.

In many traditional multi-agent navigation approaches, a macro-scale global planner provides the agent with a *preferred velocity* $\mathbf{v}_i^{\text{pref}}$ in each cycle, which would lead the agent around static obstacles and towards the goal if no other agents were around. A micro-scale collision-avoidance technique then takes this preferred velocity as input and adapts it to compute an actual velocity that lies close to the preferred velocity but also ensures collisions are avoided with nearby agents (see Fig. 2).

Our approach introduces a meso-scale planner in between the macro-scale and the micro-scale. That is, in each cycle the meso-scale planner takes as input the preferred velocity $\mathbf{v}_i^{\text{pref}}$ from the macro-scale planner, considers what groups of agents in its neighborhood form coherent moving obstacles, and computes an *adapted preferred velocity* $\mathbf{v}_i^{\text{adapt}}$, which is as close as possible to the preferred velocity but ensures that the agent would not collide with any (convex hull of a) group of agents if the groups would maintain their shape and continue moving with their current velocities (see Fig. 2). The micro-scale collision avoidance then takes this adapted preferred velocity as input and ensures collisions are avoided with any individual agent or individual members of groups of agents.

Our approach to meso-scale planning can be combined with any macro-scale planner and micro-scale collision avoidance technique, so we focus in this paper on the meso-scale planning, which takes as input a preferred velocity, and must compute as output an adapted preferred velocity. In our experiments, we use a visibility graph approach for macro-scale planning and reciprocal collision avoidance (RVO 2.0) for micro-scale collision avoidance.

IV. Meso-Scale Navigation

In this section we introduce our approach to meso-scale navigation. We describe our approach for a single agent a, whose preferred velocity is given as input, and whose adapted preferred velocity should be computed.

A. Approach

Our approach is as follows. First, the agent observes its neighborhood (the positions and velocities of all agents within a certain radius), and then clusters neighboring agents with similar positions and velocities into groups. These groups are then considered as obstacles of constant shape moving with a constant velocity equal to the average current velocity of the agents in the group. Using the concept of *velocity obstacles* [5], we then compute an adapted preferred velocity as the closest velocity to the preferred velocity that would avoid collisions with the groups if they move according to the above assumptions.

In the subsections below, we discuss our clustering method and detail the computation of the adapted preferred velocity based on velocity obstacles.

B. Clustering

Let N_a be the set of neighboring agents that are currently within the neighborhood (as defined by a given radius n_a) of the agent a under consideration:

$$N_a = \{b \mid \|\mathbf{p}_b - \mathbf{p}_a\| < n_a\}. \qquad (1)$$

Note that the neighbor radius n_a can be different for each agent. Then, we cluster the neighboring agents into groups according to the following rule. If for a pair of agents $b \in N_a$ and $c \in N_a$ it holds that both

- the position $\mathbf{p}_b$ of agent b and the position $\mathbf{p}_c$ of agent c are within a predefined distance ε_a^p, and
- the velocity $\mathbf{v}_b$ of agent b and the velocity $\mathbf{v}_c$ of agent c are within a predefined distance ε_a^v,

then agent b and agent c belong to the same group. The *transitive closure* of this relation uniquely defines the clustering into groups. Note again that the thresholds ε_a^p and ε_a^v can be different for each agent. Hence, each agent perceives groups independently, and potentially differently.

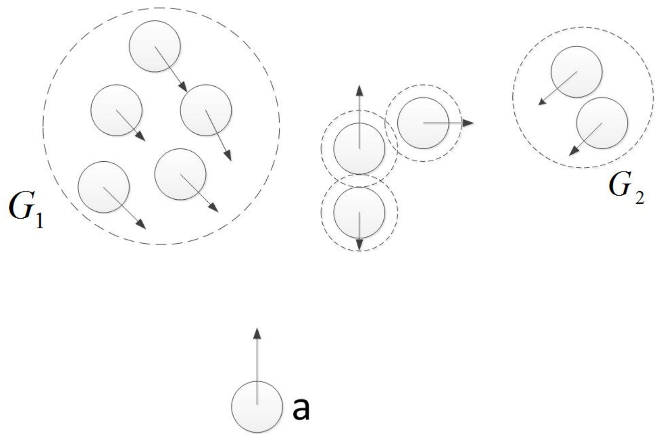


Fig. 3. Our algorithm clusters nearby agents into groups if they have similar positions and velocities.

More formally, let

$$(b \sim c) \equiv (\|\mathbf{p}_c - \mathbf{p}_b\| < \varepsilon_a^p \wedge \|\mathbf{v}_c - \mathbf{v}_b\| < \varepsilon_a^v), \qquad (2)$$

where $\sim$ is a binary relation operator defining the above rule. Now, the transitive closure $\simeq$ of the relation $\sim$ is defined as:

$$(b \simeq c) \equiv \exists[\{d_1, d_2, \dots, d_i\}|(b \sim d_1) \wedge (d_1 \sim d_2) \wedge \cdots \wedge (d_{i-1} \sim d_i) \wedge (d_i \sim c)]. \qquad (3)$$

Hence, agent b and agent c are in the same group if and only if $b \simeq c$. Note that $\simeq$ is a formal *equivalence relation*, i.e. it is reflexive ($b \simeq b$), symmetric ($(b \simeq c) \Leftrightarrow (c \simeq b)$), and transitive ($(b \simeq c) \wedge (c \simeq d) \Rightarrow (b \simeq d)$). Hence, the relation $\simeq$ defines a unique *partition* of the set N_a into disjoint nonempty subsets that form the groups of agents (see Fig. 3).

A greedy algorithm finds these groups in $O(nk)$ time where n are the number of agents in N_a and k is the number of groups in the partition.

C. Velocity Obstacles

Given the groups of agents as computed above, we want to adapt the given preferred velocity $\mathbf{v}_a^{\mathrm{pref}}$ such that when the agent follows the adapted preferred velocity, it will not "collide" with any of the groups. For this, we use the concept of *velocity obstacles* [5]. Let us focus the discussion on a single group of neighboring agents G. We assume that the shape of the group is given by the *convex hull* $\mathcal{CH}(G)$ of the set of agents constituting the group G, and that the velocity $\mathbf{v}_G$ of the group is given by the average of the velocities of the agents within the group:

$$\mathbf{v}_G = \sum_{b \in G} \mathbf{v}_b / |G|, \qquad (4)$$

where $|G|$ is the number of agents in G. We assume that group's shape and velocity will remain constant into the (near-term) future. Even though this assumption is not precisely correct, the agent can adapt to changes in groups and

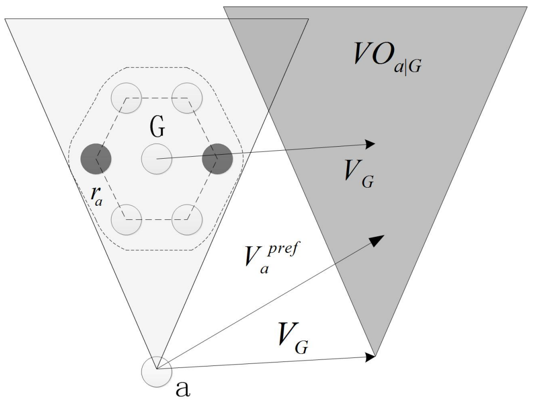


Fig. 4. The velocity obstacle $VO_{a|G}$ for agent a with respect to group G. The velocity obstacle is fully defined by the radially most extreme agents of the group (dark grey), so the convex hull of the group need not be explicitly computed. The preferred velocity for a shown in the figure would result in collision with group G at some point in time.

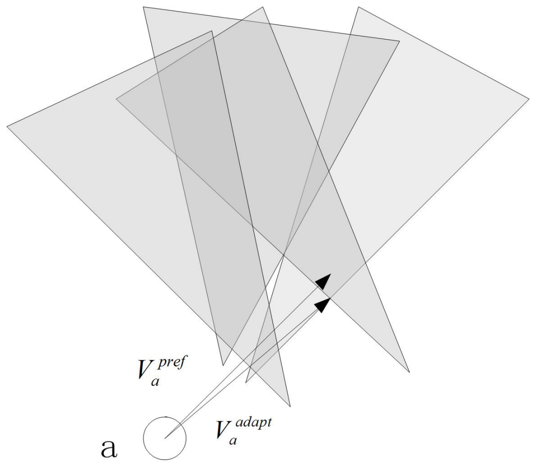


Fig. 5. The adapted preferred velocity is the velocity closest to the preferred velocity outside the union of velocity obstacles of agent a with respect to the groups of neighboring agents.

velocities since an adapted preferred velocity is computed in every sensing-acting cycle.

Given the group G and its velocity $\mathbf{v}_G$, the velocity obstacle $VO_{a|G}$ for agent a induced by group G is defined as the set of velocities $\mathbf{v}_a$ for agent a that will result in a collision with G at some point in time assuming that group G maintains its velocity $\mathbf{v}_G$:

$$VO_{a|G} = \{\mathbf{v} \mid \exists[t > 0|\mathbf{p}_a + (\mathbf{v} - \mathbf{v}_G)t \in \mathcal{CH}(G) \oplus \mathcal{D}(\mathbf{0}, r_a)]\}, \tag{5}$$

where $\mathcal{D}(\mathbf{0}, r_a)$ is a disc centered at the origin with radius r_a, and $A \oplus B$ denotes the Minkowski sum of sets A and B. The definition implies that if agent a chooses a velocity outside the velocity obstacle, it will not collide with group G, and if it chooses a velocity on its boundary, it will "touch" group G at some point in time.

The velocity obstacle is geometrically constructed as a cone with apex in $\mathbf{p}_a$ and its sides tangent to $\mathcal{CH}(G)$ enlarged with the radius r_a of agent a, which is then translated by $\mathbf{v}_G$ (see Fig. 4). This means that the convex hull $\mathcal{CH}(G)$ need not be computed explicitly, but that the velocity obstacle is fully defined by the radially most extreme agents in the group as seen from $\mathbf{p}_a$ (the most "clockwise" agent and the most "counterclockwise" agent).

D. Computing the Adapted Preferred Velocity

Given the various groups of neighboring agents as computed by our clustering algorithm, the preferred velocity $\mathbf{v}_a^{\mathrm{pref}}$, and the definition of velocity obstacles, we compute the adapted preferred velocity $\mathbf{v}_a^{\mathrm{adapt}}$ as follows. First of all, we ignore all singleton groups (i.e. groups consisting of a single agent) that result from the clustering algorithm, since single agents can be dealt with at the micro-scale agent-agent collision-avoidance level. Then we construct the velocity obstacles of all groups that remain, and select the adapted preferred velocity as the closest velocity to the preferred velocity that is outside the union of the velocity obstacles with respect to each of the groups (see Fig. 5):

$$\mathbf{v}_a^{\mathrm{adapt}} = \mathrm{argmin}\{\mathbf{v} \notin \bigcup_G VO_{a|G}\} \, \|\mathbf{v} - \mathbf{v}_a^{\mathrm{pref}}\|. \tag{6}$$

This velocity $\mathbf{v}_a^{\mathrm{adapt}}$ is either the preferred velocity itself, or the projection of the preferred velocity onto the side of a velocity obstacle, or the intersection point of the sides of two velocity obstacles [7]. Enumerating over these points and selecting the one closest to the preferred velocity that is not inside any of the velocity obstacles will give the adapted preferred velocity for agent a.

V. Multi-Agent Navigation with Meso-Scale

Our meso-scale module can, in principle, be combined with any macro-scale planner and micro-scale collision avoidance technique. In our implementation, we use for the macro-scale planner a *visibility graph* [4] that provides in each sensing-acting cycle a preferred velocity pointing along the shortest path around the static obstacles to the agent's goal with a magnitude equal to the agent's preferred speed. This preferred velocity is adapted using our meso-scale approach as discussed above. The adapted preferred velocity is then given as input to the micro-scale agent-agent collision avoidance module, for which we use reciprocal collision avoidance based on ORCA [23]. The micro-scale collision avoidance computes the agent's actual velocity, and makes sure the agent avoids collisions with nearby individual agents, and is oblivious to the notion of groups. The neighbor radius used in the micro-scale collision avoidance is typically smaller than the neighbor radius used to construct groups; the agent need only avoid pairwise collisions with immediately neighboring agents. This cycle repeats indefinitely, and is carried out by each agent individually.

VI. Results

We evaluate our approach in simulation in three illustrative scenarios. In all scenarios we compare our approach with a two-level multi-agent navigation approach that uses the same macro-scale planner and micro-scale collision avoidance

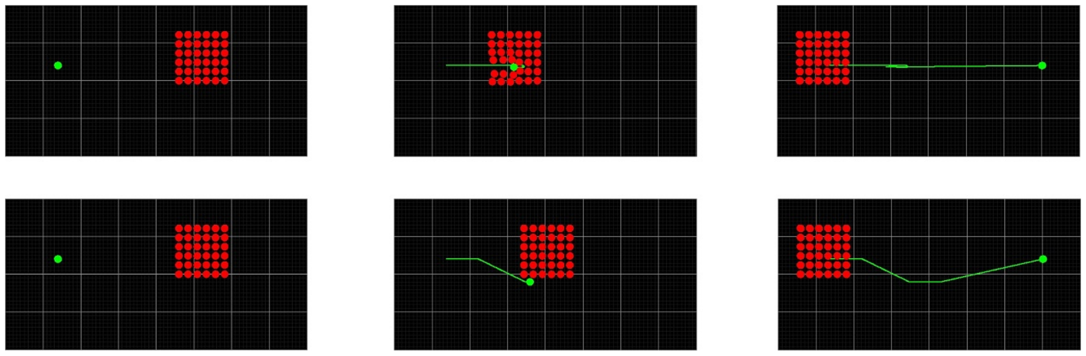


Fig. 6. Results for Scenario A in which a single agent (green) exchanges positions with a group of agents (red). The top three images show the path resulting from the traditional multi-agent navigation approach, in which the agent tries to move straight through the group. In the bottom three images show the resulting motion with our meso-scale approach: the agent smoothly moves around the group.

(ORCA). In scenario A, a single agent moves in the opposite direction of a group of agents that must pass each other (see Fig. 6). In scenario B, multiple smaller groups of agents as well as a number of individual agents simultaneously traverse the environment. Some of the individual agents temporarily form a group themselves as a result of their motion (see Fig. 7). In scenario C we simulate several larger groups and several individual agents navigating an environment with static obstacles (see Fig. 8). In our experiments, we focus on the length of the trajectories of each agent and compare it with the traditional multi-agent navigation approach. We implemented our algorithm using C++ and evaluated it on an Intel $i7$ PC.

Below, we discuss the qualitative results of each of the scenarios in detail. Table I shows the quantitative results. The column "path length" refers to the sum of the lengths of the paths taken by the individual agents (those not part of a group) in each scenario. The column "run time" gives the time per frame of the entire simulation. In terms of running time, we see across all scenarios that the results suggest that the addition of the meso-scale level causes about a 30% computational overhead compared to traditional multi-agent navigation approaches.

A. Single Agent and Single Group

The first experiment is relatively simple, but illustrates the effect of our approach. A single agent is moving from the left side of the environment to the right side, while a group consisting of 49 agents moves in the opposite direction, from the right to the left (see Fig. 6). In the figure, it can be seen that with a traditional multi-agent navigation approach the agent tries to cross straight through the group, which results in the agent being "dragged along" with the group in the opposite direction it wants to move. In contrast, in our approach the agent is aware that the agents approaching him do in fact form a group and finds a smooth clean path around the group towards its goal. Table I indicates that the agent takes a dramatically shorter path in this scenario.

B. Multiple Agents and Multiple Groups

In the second experiment, we set up several groups of 9 agents each which are instructed to move from one side of the environment to the other as shown in Fig. 7. In addition, several individual agents traverse the scenario simultaneously. As can be seen from the figure, the individual agents smoothly avoid the groups by traveling around them. Interestingly, the three individual agents moving from the bottom of the environment to the top at some point temporarily form a group themselves to the agents in the blue group, as they have similar positions and velocities. This causes the blue group to slightly change its course to avoid the individual agents. Also, in this scenario, the length of the trajectories of the individual agents is significantly less for our approach than with traditional multi-agent navigation, as can be seen in Table I.

TABLE I

QUANTITATIVE RESULTS

Scenario	Method	Path length	Run time (ms/frame)
A	Traditional	37.7	4.89
	with Meso-Scale	29.0	6.52
B	Traditional	152.3	11.58
	with Meso-Scale	137.2	14.21
C	Traditional	128.6	7.69
	with Meso-Scale	112.7	10.05

C. Multiple Agents and Groups with Static Obstacles

In the final experiment, we set up three groups of agents that each have 16 agents and three static polygonal obstacles in the environment (see Fig. 8). In addition, three individual agents try to pass through the center of the environment to arrive at their goals. Also in this scenario, the individual agents smoothly avoid both the groups and the static obstacles, resulting in shorter paths compared to traditional multi-agent navigation approaches (see Table I).

We see that the benefit of our approach is more pronounced in this scenario than in scenario B. This is due to the larger size of the groups. Whereas small groups can still be handled relatively well by the micro-level collision avoidance, this is not the case for larger groups, which need to be handled on the meso-scale.

VII. CONCLUSION

We introduced a method for multi-agent navigation to handle meso-scale obstacles: i.e. obstacles formed by groups

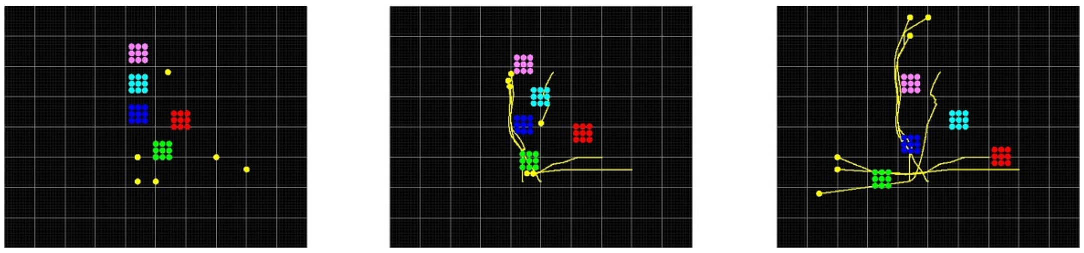


Fig. 7. Results for Scenario B in which five groups of agents (red, green, blue, cyan, magenta) share a workspace with six individual agents (yellow) each moving from one side of the environment to the other. The individual agents smoothly avoid the groups by walking around them. The three individual agents moving from the bottom to the top temporarily form a group themselves to the agents in the blue and magenta groups.

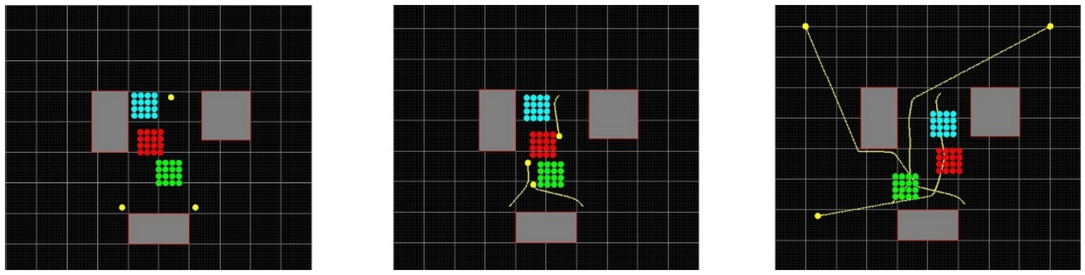


Fig. 8. Results for Scenario C in which three groups of agents (red, green, cyan) and three individual agents (yellow) cross an environment with three polygonal static obstacles (grey). The red and cyan groups move from the top to the bottem while the green group moves towards the bottom-left. The individual agents cross through the center of this scenario, smoothly avoiding the groups.

of individual agents that many traditional collision avoidance approaches see as a collection of individual agents rather than a coherent entity. Our approach to meso-scale planning is fully independent of the macro-scale planner and the micro-scale collision avoidance, and can be used in combination with any of them. Our simulation experiments show that our approach, at a small computational cost, can dramatically improve the realism and efficiency of motions of individual agents when they encounter groups of other agents, by letting the agents smoothly move around the groups.

Our method was not designed to let groups of agents coherently avoid other groups. In such scenarios, our approach performs similarly to traditional approaches, since in our approach each agent still makes individual decisions and has no notion of belonging to a group: it only may perceive sets of other agents as a group and act accordingly. In future work, we will consider group coherence as well to let multiple groups smoothly avoid each other.

References

[1] C. Behring, M. Bracho, M. Castro, J. Moreno. An algorithm for robot path planning with cellular automata. *Proc. Int. Conf. on Cellular Automata for Research and Industry*, 2000.

[2] A. Chakravarthy, D. Ghose. Obstacle avoidance in a dynamic environment: a collision cone approach. *IEEE Trans. on Systems, Man and Cybernetics* 28(5):562-574, 1998.

[3] S. Curtis, J. Snape, D. Manocha. Way Portals: efficient multi-agent navigation with line-segment goals. *Proc. ACM SIGGRAPH Symp on Interactive 3D Graphics and Games*, 2012.

[4] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars. *Computational geometry: algorithms and applications*. Springer-Verlag, 2008.

[5] P. Fiorini, Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *Int. Journal on Robotics Research* 17(7):760-772, 1998.

[6] D. Fox, W. Burgard, S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Mag.* 4(1):23-33, 1997.

[7] S. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, P. Dubey. Clearpath: highly parallel collision avoidance for multi-agent simulation. *Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 2009.

[8] S. Guy, J. Chhugani, S. Curtis, P. Dubey, M. Lin, D. Manocha. PLEdestrians: a least-effort approach to crowd simulation. *Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 2010.

[9] D. Helbing, P. Molnar. Social force model for pedestrian dynamics. *Physical Review E* 51(5):4282?286, 1995.

[10] D. Hsu, R. Kindel, J.-C. Latombe, S. Rock. Randomized kinodynamic motion planning with moving obstacles. *Int. Journal on Robotics Research* 21(3):233-255, 2002.

[11] K. Kant, S. Zucker. Towards effcient trajectory planning: path velocity decomposition. *Int. Journal of Robotics Research*, 5:72-89, 1986.

[12] I. Karamouzas, P. Heil, P. Beek, M. Overmars. A predictive collision avoidance model for pedestrian simulation. *Proc. Motion in Games*, 2009.

[13] L. Kavraki, P. Svestka, J.-C. Latombe, M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. on Robotics and Automation* 12(4):566-580, 1996.

[14] A. Kimmel, A. Dobson, K. Bekris. Maintaining Team Coherence under the Velocity Obstacle Framework. *Proc. Int. Conf. on Autonomous Agents and Multiagents Systems*, 2012.

[15] Y. Koren, J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. *Proc. IEEE Int. Conf. on Robotics and Automation*, 1991.

[16] J.-C. Latombe. *Robot Motion Planning*. Springer, 1991.

[17] S. LaValle, J. Kuffner. Randomized kinodynamic planning. *Int. Journal on Robotics Research* 20(5):378-400, 2001.

[18] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, S. Thrun. Anytime Dynamic A*: An anytime, replanning algorithm. *Proc. Int. Conf. on Automated Planning and Scheduling*, 2005.

[19] S. Petti, T. Fraichard. Safe motion planning in dynamic environments. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2005.

[20] C. Reynolds. Flocks, herds, and schools: a distributed behavioral model. *ACM SIGGRAPH Computer Graphics* 21:25?4, 1987.

[21] S. Sarmady, F. Haron, A. Talib. Simulating Crowd Movements Using Fine Grid Cellular Automata. *UK-Sim*, 2010.

[22] J. van den Berg, S. Patil, J. Sewall, D. Manocha, M. Lin. Interactive Navigation of Individual Agents in Crowded Environments. *Proc. ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games*, 2008.

[23] J. van den Berg, S. Guy, M. Lin, D. Manocha. Reciprocal n-body collision avoidance. *Proc. Int. Symp on Robotics Research*, 2011.