2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Improving Aggregate Behavior in Parking Lots with Appropriate Local Maneuvers*

Samuel Rodriguez, Andrew Giese, and Nancy M. Amato
Parasol Lab., Dept. of Computer Science and Engineering, Texas A&M Univ., College Station, Texas, 77843-3112, USA
sor8786@tamu.edu, gieseanw@neo.tamu.edu, amato@tamu.edu

***Abstract*— In this paper we study the ingress and egress of pedestrians and vehicles in a parking lot. We show how local maneuvers executed by agents permit them to create trajectories in constrained environments, and to resolve the deadlocks between them in mixed-flow scenarios. We utilize a roadmap-based approach which allows us to map complex environments and generate heuristic local paths that are feasible for both pedestrians and vehicles. Finally, we examine the effect that some agent-behavioral parameters have on parking lot ingress and egress.**

I. Introduction

Many people simultaneously arrive and leave an environment every day all over the world [1]. Moving in a crowded space can be a source of stress for those involved, time consuming if there are many bottlenecks, and dangerous if some basic rules and restrictions are not followed [2], [3]. Being able to accurately simulate such a scenario would allow a designer to identify and solve problems in their conceived worlds. For example, it might be found that drivers should leave some minimum amount of space between neighboring vehicles or always allow someone backing out to have right of way. In another case, pedestrians on foot should perhaps stick to paths away from the aisles of a parking lot so that incoming and outgoing traffic is not blocked. We will refer to these situations where agents are arriving and leaving an area under normal conditions as ingress and egress, respectively.

To adequately support egress/ingress simulations, we utilize local maneuvers to resolve conflicts between agents. Our distributed approach to this scenario allows us to equip agents with local maneuvers that can be applied in a local area of the space without affecting or considering other agents that are far away. This is an intuitive approach and more scalable than a centralized approach that must consider every agent. We explore both cooperative and individual maneuvers with a focus on those needed to address local motion requirements for vehicles in parking lots. An example environment consisting of buildings and a parking lot is shown in Figure 1.

The main contributions of this work include:

- A tunable model for ingress and egress behaviors in our roadmap-based approach.
- Parking lot planning in complex structures with grouping and respecting agent dynamics.
- Maneuver planning within a parking structure.

Fig. 1: An actual environment from our college campus.

In parking lots, mixed flows of vehicles occur often as drivers identify different routes toward exits. As the drivers attempt to plan their paths, they are restricted by vehicle dynamics that must be respected as well as the presence of other vehicle agents and nearby pedestrians. Our tunable approach can model these higher level behaviors, factors that influence agent perception, and encode restrictions on different types of agent motion and dynamics. Another important factor to consider is groupings amongst agents. Agents may be statically grouped with one another, for example sharing a vehicle. Dynamic groups of agents must also be considered as local planning needs to be done, for example between nearby vehicles. The local maneuvering strategies we utilize allow us to model parking lot motions such as pulling out and in and resolving deadlocks that occur as vehicles move near one another. Our approach allows us to handle complex environments respecting pedestrian and vehicle motion and constraints.

The remainder of this paper is organized as follows. In Section II we describe work that is related to our approach. In Section III an overview of the important components of our system is described, Section IV describes the vehicle motion we consider, and in Section V the ingress and egress behaviors are described. Local interactions are then described in Section VI. Simulation results are presented in Section VII.

This research supported in part by NSF awards CRI-0551685, CCF-0833199, CCF-0830753, IIS-096053, IIS-0917266 by THECB NHARP award 000512-0097-2009, by Chevron, IBM, Intel, Oracle/Sun and by Award KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST).

II. Related Work

A number of issues are relevant when simulating agent egress in virtual environments. Here we describe a few of the most relevant ones, including modes of transportation considerations, environmental complexity, and collision avoidance. As described below, there has been a lot of work in virtual environments, some focused on evacuation and planning in dynamic environments. Our work is informed by and incorporates aspects of these approaches into a full system for general ingress and egress planning with varying agent types, addressing a broader range of problems.

This work has similarities to the approaches presented in [3], [4]. In [3], a similar approach to the scenario we are studying was presented at an airport parking lot. However, they considered a cellular automata-based approach with simplified agent dynamics. Movement is restricted in the direction of the grid cell, and each cell can only hold one vehicle at a time. They can capture elements of microscopic simulations including following, lane changes, protected turns and unprotected turns. Different types of steering algorithms were presented in [4] for pedestrians to allow them to choose the appropriate steering method depending on the scenario, however they focused solely on pedestrian agents. This included reactive steering and planned state-time steering.

The complex relationship between different modes of transportation is an issue in the full scale egress scenario. In [1], the need to consider the vehicle aspect of actual evacuations where vehicle and pedestrian flows are considered together is described. They claim that in an evacuation of an area, pedestrians can vastly influence the overall evacuation and prevent planned direction from optimizing movement. An underlying independent graph-based model is used for vehicles and pedestrians with the combined network used to analyze conflicts. In this work, motion was considered only on the graph which greatly simplifies actual motion constraints.

A number of approaches for *pedestrian-only* navigation have been studied. A survey focused on virtual crowds [5] described many approaches and models that have been proposed for crowd simulation. One approach attempted to simulate agent panic when evacuating simple environments [2]. An approach to find the optimal evacuation time in simple 2D environments is described in [6] where the occupants have n possible exits and use an evacuation function to select routes. The idea of different levels of agent knowledge and planning ability is considered in [7]. In [8], a system is developed for simulating the local motion and global way finding behaviors of crowds moving in a natural manner within dynamically changing virtual environments. They are able to simulate levels of patience and pushing between agents. Improvements on previous work, [2], were made by considering factors that reduced shaking and vibration caused by social forces models in densely crowded areas. They also consider the challenge of avoiding bottlenecks. These approaches focus on pedestrian motion and usually only in basic two dimensional environments.

Physical factors are considered in [9], but the ability and need to include social grouping is also described. There are also known evacuation scenarios where agents have vastly different traveling speeds which includes people with disabilities [10] who may require evacuation in groups. Another work that describes the need to consider grouping in evacuation is [11], where depending on the population type, agents may be either individuals or be considered familial groups which may contain small children.

There has been some work focused on more complex environments which are encountered in the real world. Pedestrians evacuating a large stadium are shown in [12]. These agents operate on a simplified network graph of the stadium and generally follow the agent ahead of them given the constrained environment. Restricting motion within the graph results in ignoring the actual motion constraints of individual agents. Evacuation in high-rise buildings is shown in [13] using a cellular automata approach. The environment is discretized into grid-cells of free space or blocked space and agents select evacuation routes based on finding a path through unoccupied cells. They stated their limitations as being factors such as the grid size, uneven usage of stairwells, and their difficulty in simulating stairwells using this approach. While these approaches are very interesting given the complexity of the environment, the abstraction of the problem results in a great deal of factors about the environment being ignored.

There have been a number of approaches proposed where nearby agents coordinate with one another. Some approaches consider other nearby agents as velocity obstacles [14], [15]. This allows the agents to plan valid paths in very constrained environments while avoiding collisions with nearby pedestrian agents. In a similar approach [16], all agents use an optimization function to avoid congested areas by minimizing energy usage.

III. Overview

In this section we describe essential components of our system used to simulate ingress and egress scenarios. For more detailed descriptions of our roadmap-based multi-agent system approach for simulating group behavior we refer the reader to our prior work [17], [18], [19]

Agents in these scenarios consist of pedestrians and vehicles. Whereas pedestrians are modeled as holonomic agents that can move more freely, vehicles are nonholonomic and must obey dynamics restrictions. Each agent is also encoded with its own set of environmental knowledge, including semantically labelled areas and destinations as well as a volume of the environment that is mapped. All agents perceive the world given their abilities which we encode by view radius and angle values with the perception being updated at each time step of the simulation. The behavior the agent is executing dictates how the agent reacts throughout the simulation. The behavior is responsible for using the agent's perceived information and knowledge of the environment to determine a set of actions that the agent can take. The specific behaviors used in this work are described in Section V.

For pedestrians and vehicle agents we use a goal-based force model to integrate an agent's position along the path it is following. The force is in the direction of the last subgoal along the path. Our vehicle force rule is similar to the standard goal-based force rule but with added restrictions. One restriction is that agents must stop if another agent is within some predefined relative distance and angle to itself. Another is a restriction on the agent's maximum steering angle to respect the agent's minimum turning radius. A third restriction is that an agent must reverse occasionally to allow for more space to reach a subgoal along the global path. Vehicles use this reactive steering unless a local path (e.g., pull-in or pull-out) is required. Other forces we utilize for pedestrians include avoidance forces between walls and nearby agents.

We have the ability to encode both basic and complex environments using our roadmap-based approach. The roadmap encodes valid transitions between free regions in the environment. Nodes in the roadmap represent free areas in the environment, and transitions between nodes are allowed if the edge connecting them also lies in the free space. Node samples can be biased to be in certain portions of the environment. For example, by pushing nodes near the medial axis of the free space of the environment, we can emulate lanes that are traversed by vehicles.

At each step of the simulation cycle, the agents execute their behavior, attempt to move according to the planned action, resolve their state in the environment, and evaluate. This is the overall process that allows us to simulate agents with any number of behaviors. In resolving the state of the agent, we allow some amount of overlap between agents before requiring that the state between agents be resolved.

IV. Vehicle Dynamics

While our system can handle any type of vehicle motion, for simplicity we consider a planar unicycle model whose state **x** consists of its x-y position and heading angle θ, i.e., $\mathbf{x} = [x, y, \theta]$.

The control vector $\mathbf{u} = [V, \phi]$ consists of the linear velocity $V \in \{-v, v\}$ and steering angle $\phi \in (\frac{-\pi}{2}, \frac{\pi}{2})$. Note that v corresponds to forward motion and $-v$ to reverse motion.

When a vehicle applies steering angle ϕ, it will circumscribe a circle with radius $r = \frac{l}{sin(\phi)}$, where l refers to the length between the front and rear axle. Knowing this turning radius r, we can derive the vehicle's angular velocity: $\omega = Vr^{-1}$

Vehicle dynamics can be modelled as a system of ODEs:

$$\dot{x} = V cos\left(\theta\right) \tag{1}$$

$$\dot{y} = V sin\left(\theta\right) \tag{2}$$

$$\dot{\theta} = \omega = V sin(\phi)/l \tag{3}$$

The above formulation allows us to modify parameters such as the velocity constant v, maximum steering angle ϕ_{max}, and wheelbase length l to model a variety of real world vehicles. To update the system, we use Eulerian integration with fixed timestep δ, yielding the update equations:

$$x_k = x_{k-1} + \delta V_{k-1} cos\left(\theta_{k-1}\right) \tag{4}$$

$$y_k = y_{k-1} + \delta V_{k-1} sin\left(\theta_{k-1}\right) \tag{5}$$

$$\theta_k = \theta_{k-1} + \frac{\delta V_{k-1}}{r_{k-1}} \tag{6}$$

V. Ingress and Egress Behavior

Our egress and evacuation behaviors have been described in [17], [19], [18]. The egress behavior is dependent on the exits and final destinations known to the agent. Using the roadmap, the agent extracts a path that will guide it from its current location through one of the known exits to a final destination. Given the agent's knowledge of the environment, there may be many routes that exist. The agent evaluates each route and selects the one with the lowest weight along its edges, which in the most basic case is the shortest route.

The ingress behavior utilized here is required for simulating normal conditions seen daily with some set of agents arriving. An arriving agent will consider the known entrances to the environment and the final destination when selecting a route to the goal configuration. It is important to note that we use the same behavior rule for both pedestrians and vehicles. For vehicle agents, the final configuration is important, and we utilize local maneuvers to enable the agent to reach it. In our current implementation of ingress, an agent has a single goal configuration. A more accurate behavior would allow an agent to have multiple goals in mind when arriving or even allow an agent to select a goal on the fly. We leave this for future work.

The roadmap-based approach is extremely beneficial in this type of scenario. It allows us to handle complex environments such as multi-level buildings and parking lots. We are able to bias sampling so that the majority of it is done near the medial axis of the environment. In this way, an agent can move through the environment near the center of the lane. Lanes can be adjusted to the right or left depending on the application (e.g., lanes in the US versus the UK). This allows agents coming toward one another to pass by safely. A time lapse of a simulation that uses this approach is shown in Figure 3.

The ingress and egress of agents causes mixed flow. We employ a stop-and-wait criteria among agents moving near one another to identify a potential collision. Without considering some of the local maneuvers (described next), a queue of vehicles would form resulting in complete deadlock and no flow in or out of the environment.

VI. Resolving Local Interactions

In this work we consider a parking lot scenario where vehicle-bound agents need to plan in a parking lot. While methods for motion planning under dynamics constraints certainly exist, many do not scale very well or fail to create realistic and predictable paths. Frazzoli et. al [20] showed that concatenating well-defined motion primitives facilitates solving even complicated motion planning problems.

The key benefit to this method is that it allows the designer to take advantage of *a priori* information about the given planning problem to shrink the solution space. A smaller solution space begets targeted maneuvers that solve a small class of motion planning problems for the agent quickly and efficiently.

A. Maneuver Planning

For our application, traditional kinodynamic planners are inadequate because we need to plan for many vehicles simultaneously on the order of a few milliseconds. We decided to combine our reactive force-based controller with a priority-based decoupled-planning approach in which agents would plan in order of priority. Agents effect low-overhead reactive behaviors until they reach a state at which they determine that reactive planning is insufficient. At that point the prioritized decoupled-planner is invoked to create a composite local plan for multiple agents.

In the composite local plan, some agents will plan before others, and later agents will treat preexisting agent plans as time-varying obstacles. This formulation results in an incrementally built composite local motion policy, Π_{comp}, for all active agents that guarantees no collisions, but potentially sacrifices some degree of completeness and optimality. In practice, this heuristic planner is fast and creates good quality paths when planning over short time windows.

The high level description of the Maneuver Planning algorithm is described in Algorithm 1. The routine is given an environment, E that describes the static obstacles, as well as a list of agents A, that need to plan a maneuver. It also receives starting configurations for agents, S_A, as well as goal positions G_A. The planner randomly staggers the start times of the agents. The staggered start strategy is a simple and useful one for modelling yielding. Planning is decoupled in that agents will only attempt to plan trajectories for themselves. It is prioritized by starting times; the agent scheduled to begin planning first will attempt to plan its entire local trajectory before any other agent, and agents that plan later must not collide with the first one's time-varying trajectory. The subroutine ATTEMPTLOCALMANEUVER called by MANEUVERSPLANNER is a stand-in for the local maneuvering routine that will actually be called (PULLOUT, PULLIN, DEADLOCK1, DEADLOCK2).

MANEUVERSPLANNER is convenient because it allows for partial solutions. That is, if one agent is unable to plan, the entire composite trajectory is not thrown out. Instead, those agents that were unable to find collision-free trajectories simply do not move during this planning interval, but will try to plan again later.

Vital to all our strategies is the subroutine TRYTOPLAN(Environment E, TrajectoryPolicy Π_{comp}, Trajectory Π_i, Agent i, Controls u, TimeSteps t, Current Time $timestep$) shown in Algorithm 2, which will attempt to append additional states created using u over time interval t to agent i's current trajectory. The subroutine fails if an update causes the agent to be in collision with an obstacle or another agent at the current timestep. If it succeeds, the successful plan is returned to the caller. The subroutine NEXTSTATE applies the update equation to the current state to produce

Algorithm 1 MANEUVERSPLANNER

```
Input: $E, A, S_A, G_A$
Output: $\Pi_{comp}$
  for Agent $a \in A$ do
    $StartTimes.push\_back(a, RandInRange(0, max))$
  end for
  // $StartTimes$ is sorted by time
  $timestep \leftarrow 0$
  $\Pi_{comp}[timestep] \leftarrow X_A$
  for $(Agenti, StartTimest) \in StartTimes$ do
    $trajectorySuccess \leftarrow false$
    for $(attempts \leftarrow 0; attempts < maxAttempts$
    $\&\& \; !trajectorySuccess; attempts \leftarrow attempts + 1)$
    do
      $trajectorySuccess \leftarrow$
      ATTEMPTLOCALMANEUVER$(E, \Pi_{comp}, i, st, G_A[i])$
    end for
  end for
```

the new state x', and the subroutine ISVALID checks for collision between x' and obstacles in the environment, as well as other agent states at time $timestep$ in the local composite trajectory Π_{comp}.

Algorithm 2 TRYTOPLAN

```
Input: $E, \Pi_{comp}, \Pi_i, i, u, t, timestep$
Output: $\Pi_i$
  $x \leftarrow \Pi_i[timestep]$
  for $j = 1$ to $t$ do
    $x' \leftarrow$ NEXTSTATE$(x, u)$
    $timestep = timestep + 1$
    if ISVALID$(E, x', \Pi_{comp}, timestep)$ then
      $\Pi_i.push\_back(x')$
    else
      $return\ INVALID$
    end if
    $x \leftarrow x'$
  end for
  $return\ \Pi_i$
```

A similar subroutine to TRYPLAN called TRYTOPLANTOWARDGOAL in which the agent greedily attempts to move toward the goal assuming no obstacles. This subroutine is useful not only as a basic naïve movement strategy, but also to guide an agent's trajectory if it has used quasi-random turning to set its heading in the vicinity of the goal.

B. Pull-Out

The Pull-Out maneuver, like all maneuvers, takes in a description of the Environment, E, the current state of the composite trajectory, Π_{comp}, and the current $timestep$. It attempts to plan a trajectory out of a parking spot for agent i. Upon success, it will update Π_{comp} to account for agent i's plan using the subroutine MERGETRAJECTORYWITHCOMPOSITE. Any calls to TRYTOPLAN or TRYTOPLANTOWARDGOAL that fail will cause the entire local maneuver to fail, returning control to MANEUVERSPLANNER, which will try again for a preset number of attempts.

Algorithm 3 PULLOUT

```
Input: $E, \Pi_{comp}, i, timestep, x_{goal}$
Output: $\Pi_{comp}$
    LocalTrajectory $\Pi_i$
    if $|\text{ANGLETOTURN}(i, x_{goal}) \leq \frac{\pi}{2}|$ then
        $velocity \leftarrow v$
    else
        $velocity \leftarrow -v$
    end if
    $\text{TAXISTRAIGHT}(E, \Pi_{comp}, \Pi_i, i, timestep, velocity, \alpha)$
    $\text{TURNTOWARDGOAL}(E, \Pi_{comp}, \Pi_i, i, timestep, velocity, \beta)$

    if $veloicty == -v$ then
        $\text{TRYTOPLANTOWARDGOAL}(E, \Pi_{comp}, \Pi_i, i, \gamma, timestep)$
    end if
    $\text{MERGETRAJECTORYWITHCOMPOSITE}(\Pi_{comp}, \Pi_i, i)$
```

When an agent needs to exit a parking space, it assumes other parked agents may be present near its flanks. It may also assume that the goal configuration lies in the "aisle" of the parking lot. The routine ANGLETOTURN computes the angle between the agent's heading and the goal position. We assume that a goal position that would cause a forward turn of more than $\frac{\pi}{2}$ is placed behind the agent such that it should reverse out of the parking stall. This assumption allows us to not need to track any parking-stall specific information in the agent's state space.

If the agent determines that it must pull forward, then it first moves straight forward a short distance α using the TAXISTRAIGHT routine, and then turns in the direction of the goal position using TURNTOWARDGOAL as shown in Figure 2(a). TURNTOWARDGOAL uses randomness in the steering angle used to turn so that turns are wider or narrower to potentially plan around unseen obstacles. Generally speaking, all arcs that make up a maneuver are tuned to be within some threshold that gives both an adequate amount of randomness as well as good quality solutions. Subroutines like TAXISTRAIGHT TURNTOWARDGOAL utilize TRYTOPLAN to update Π_i and the current timestep.

If the agent will attempt to reverse out of the space it performs all the same maneuvering as a forward strategy, but in reverse. After the reverse turn is completed though, the agent attempts a short forward plan of length γ toward the goal position like in Figure 2(b).

C. Pull-In

PULLIN is the only local maneuver where we attempt to plan all the way to a goal configuration. The algorithm is described in Algorithm 4. The configuration's rotation, however is assumed to be constrained to that of a vehicle facing forward in the parking stall (i.e, it did not reverse into it). This does not constrain the agent's final rotation in the stall, but rather instead provides useful information to the planner.

Before we call the maneuver, we assume that the agent has positioned itself to be in the aisle the parking spot is located. A basic driving behavior is sufficient for taxiing the agent until it is in a position such that the maneuver is valid.

In PULLIN, the goal configuration x_{goal} is a full configuration that not only specifies an (x, y) position but also a rotation about the vertical axis, θ.

PULLIN first attempts to maneuver the agent such that its heading constitutes a vector that is perpendicular to the heading of the goal configuration using the TURNTOFACEPERPGOAL subroutine. Afterwards it chooses a turning radius at which to turn into the parking stall. This radius may be equivalent to the smallest turning radius the agent is capable of, or something slightly larger (but less than infinite). With this turning radius in mind, the agent taxis forward toward the intersection point between a ray in the direction of its heading and a ray from the goal position in the opposite direction of the goal configuration's heading using the TAXIFORWARD subroutine. This subroutine will stop planning r distance short of the intersection point between the perpendicular rays, at which point the maneuver decides whether to pull into the stall normally or in reverse.

The user can alter probability parameters that affect the likelihood the agent pulls into the stall in a normal forward manner as depicted in Figure 2(c), or attempts to reverse into the stall as shown in Figures 2(d) and 2(e). PULLINFORWARD will turn at the input turning radius until the agent is in front of the goal configuration, and then simply plan forward until the goal is reached. PULLINREVERSE1 causes the agent to actually overshoot the parking stall, and then turn backwards at the same turning radius as a forward pull-in. PULLINREVERSE2 reverses the steering angle of the agent from the PullInForward scenario to turn away from the goal until it can reverse in a straight line into the stall.

Algorithm 4 PULLIN

```
Input: $E, \Pi_{comp}, i, timestep, x_{goal}$
Output: $\Pi_{comp}$
    LocalTrajectory $\Pi_i$
    $\text{TURNTOFACEPERPGOAL}(E, \Pi_{comp}, i, timestep, x_{goal})$
    $distToInt \leftarrow RayIntersection(x_{goal}.position,$
    $-x_{goal}.\theta, i.position, i.\theta)$
    $r \leftarrow RandInRange(r_{min} + \alpha, r_{min})$
    $\text{TAXIFORWARD}(E, \Pi_{comp}, i, timestep, x_{goal}, distToInt - r)$
    $randNum \leftarrow RandInRange(0, 1)$
    if $randNum \leq forwardProbability$ then
        $\text{PULLINFORWARD}(E, \Pi_{comp}, i, timestep, x_{goal}, r)$
    else
        $randNum \leftarrow RandInRange(0, 1)$
        if $randNum \leq normalReverseProb$ then
            $\text{PULLINREVERSE1}(E, \Pi_{comp}, i, timestep, x_{goal}, r)$
        else
            $\text{PULLINREVERSE2}(E, \Pi_{comp}, i, timestep, x_{goal}, r)$
        end if
    end if
    $\text{MERGETRAJECTORYWITHCOMPOSITE}(\Pi_{composit}, \Pi_i, i)$
```

D. Deadlock 1

When many car-like agents attempt to enter and exit a parking lot simultaneously, deadlock situations occur where two agents may not move toward their next goal without causing collision. Such a situation calls for a resolution planner that will allow the agents to avoid collision as well as still be reasonably on course with their goal. In one very common case, two agents are attempting to reach goals, each with its goal behind the opposite agent. In such a situation, one agent simply yielding to the other agent does not help — both agents must plan together to resolve the deadlock. A common rule in western countries to avoid deadlocks is "stay to the right". The Deadlock1 local maneuver described by Algorithm 5 encodes this heuristic to an extent.

DEADLOCK1 first chooses how hard of a right hand turn the agent should make. With a small input probability, the agent will actually decide to turn left. After completing the turn of arclength α, the agent plans forward a short distance β, and then attempts a turn equal but opposite the initial turn. Shortly thereafter the agent attempts another short forward plan of length γ.

The algorithm is described graphically in Figure 2(f). The DEADLOCK1 maneuver is intended to only resolve deadlocks between pairs of agents, which we found to be the most prominently occuring deadlock scenario.

Algorithm 5 DEADLOCK1

```
Input: $E, \Pi_{comp}, i, timestep, x_{goal}$
Output: $\Pi_{comp}$
    LocalTrajectory $\Pi_i$
    $randVal \leftarrow RandInRange 0, 1$
    if $randVal \leq rightTurnProbability$ then
        TURNRIGHT$(E, \Pi_{comp}, \Pi_i, i, timestep, \alpha)$
    else
        TURNLEFT$(E, \Pi_{comp}, \Pi_i, i, timestep, \alpha)$
    end if
    ControlVector $u$
    TAXIFORWARD$(E, \Pi_{comp}, i, timestep, x_{goal}, \beta)$
    if $randVal \leq rightTurnProbability$ then
        TURNLEFT$(E, \Pi_{comp}, \Pi_i, i, timestep, \alpha)$
    else
        TURNRIGHT$(E, \Pi_{comp}, \Pi_i, i, timestep, \alpha)$
    end if
    TAXIFORWARD$(E, \Pi_{comp}, i, timestep, x_{goal}, \gamma)$
    MERGETRAJECTORYWITHCOMPOSITE$(\Pi_{comp}, \Pi_i, i)$
```

E. Deadlock 2

Another common deadlock situation occurs when two agents' paths toward their goals are intersecting, and the agents happen to arrive at the intersection point at nearly the same time. Fortunately, this situation *can* be solved via one agent yielding to the other. We add some randomness to the maneuver by randomly choosing who should yield as well as varying the evasive steering angle.

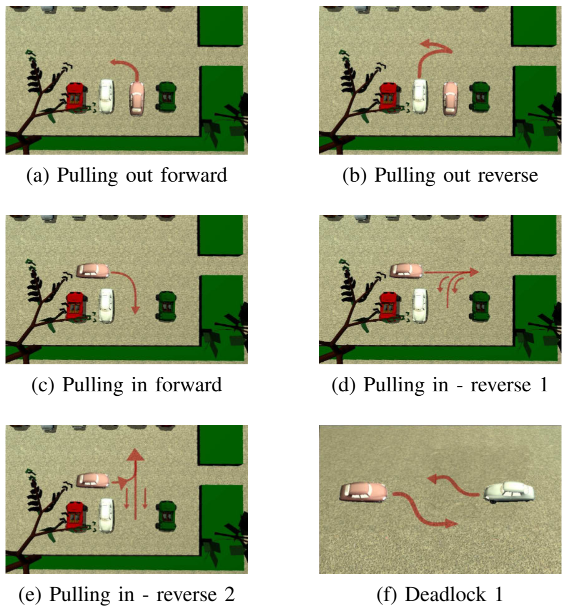
(a) Pulling out forward (b) Pulling out reverse (c) Pulling in forward (d) Pulling in - reverse 1 (e) Pulling in - reverse 2 (f) Deadlock 1

Fig. 2: Visual depiction of local maneuvers

F. Combining Local and Global Trajectories

In the previous section we used the subroutine MERGETRAJECTORYWITHCOMPOSITE, which we explained merged an agent's locally planned trajectory with a composite trajectory. The composite trajectory Π_{comp} can be considered a 2D matrix where rows consist of individual trajectories and columns are composite configurations for individual timesteps, as shown in Table I. The bold configurations in a row indicate those at which the agent started and completed planning, respectively.

At problem start, this matrix only contains a single column containing start configurations for each agent. If all agents planned trajectories of identical duration and began at the same time, then the matrix would always remain consistent and MERGETRAJECTORYWITHCOMPOSITE would simply consist of replacing an entire row in the composite trajectory. In reality, though, agents can begin and finish planning at any time. Our only guarantee is that agents with earlier starting times will plan their trajectories first.

For data consistency when creating a local trajectory Π_i we must account for the possibility that it extends beyond the planning interval for some higher-priority trajectory Π_j. For example, consider the relationship between Π_A and Π_B in Table I. Agent B begins planning one timestep after A completes. When B wishes to perform a collision check against the time-varying trajectory of A it assumes that A simply stopped moving after completing its local maneuver – we do not perform any guessing about or extending of Π_A. We also assume that A remains stationary for the duration of Π_B. B's planned trajectory may therefore be invalid against what A may potentially do after control exits the local maneuvers planner, and the maneuvers planner may need to be called again for B. Because agent A has a higher priority than B, its assumption that B would remain stationary during the creation of Π_A is valid; Π_B must consider Π_A an obstacle

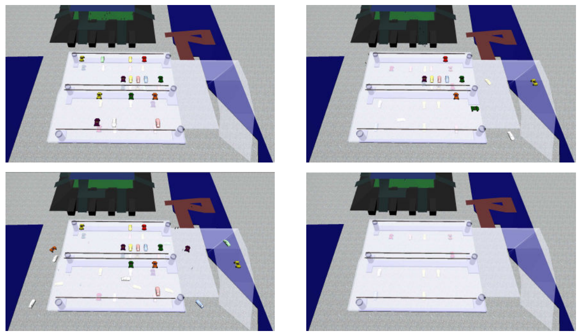

Fig. 3: An example scenario where agents exit and park in a parking lot

but not vice-versa.

TABLE I: Sample composite trajectory for three agents

Time	0	1	2	3	4	5	6
Π_A	A0	**A1**	**A2**	A2	A2	A2	A2
Π_B	B0	B0	B0	**B1**	B2	B3	**B4**
Π_C	C0	C0	C0	C0	**C1**	**C2**	

VII. Simulation Results

The example scenario we study for comparison consists of a building containing pedestrian agents and a two level parking lot, shown in Figure 3. Some vehicles are initially in the parking lot with agents in the building leaving to their vehicles. Another set of vehicles are arriving. We use this environment because it represents a very complex planning problem and it allows us to highlight some of the motion strategies we propose in these ingress and egress scenarios. We have successfully applied our techniques to achieve ingress and egress with collision avoidance in this scenario. We initialize the problem with 25 pedestrian agents, 30 parked vehicles, and 10 vehicles arriving, with a single pedestrian assigned to each vehicle. The vehicles have predefined initial and goal locations in the parking structure. The best representation of our work which shows planning in complex structures, respecting agent dynamics, groupings of agents, and necessary local maneuvers is through our animations. Along with the supplied video, additional simulation results can be found on our webpage:
http://parasol.tamu.edu/groups/amatogroup/research/flock/.

A. *Parameters Influencing Scenario*

There are a number of agent and behavioral parameters that will affect the overall outcome of a scenario. These parameters would allow us to encode those observed from actual drivers (if the values were available) in order to simulate a more realistic scene. They would also allow a user to model scenarios involving self driving cars as they may eventually become mainstream. One of the parameters that influences the agents' ability to plan successfully is the amount of time it will wait before attempting to resolve a deadlock (WT). A low wait time represents less patient agents while agents with a higher wait time can be considered more patient. The view radius (VR) parameter dictates the level of environmental knowledge the agent uses when generating locally planned paths. Agents with a very low view radius will only consider their local area in the planning process while agents with a higher view radius will consider more local agents in the planning process. Additionally, vehicle agents only attempt to plan out of a parking spot if the number of vehicles moving nearby is less than a predefined number; in the experiments this is set to one.

B. *Tuning Scenario*

Results for this scenario are shown in Table II. Given the deadlock wait time and view radius for vehicle agents, we show the success rate (SR) of the scenario where all agents reach their goal locations, the time (in time steps) for a scenario to complete, and the attempt and success rate of each of the local strategies for vehicle motion.

It is important for agents to consider the appropriate amount of local information when creating local paths. This was evident for agents with a view radius of 25 where the agents would often plan local paths that were correct given this local information, but when following the local path would cause them to reach unresolvable deadlock with agents not in the previous planning area. In these cases the agents had low success rates.

Once an agent has generated a local path, it followed the path until completion or collision with nearby vehicle agents was imminent. If a collision is detected, the colliding agents were reset to their last valid configurations.

As the view radius increases for vehicle agents, the overall success rate of completing the scenarios also increases. This is due to the agents including more local information when generating a local path, and waiting longer if they detect too many nearby moving vehicles. Increasing the view radius also results in longer times for the scenario to complete. Larger view radius values also result in more successful pull-out rates and less need for deadlock resolution planning.

Increasing wait time also had an effect on the scenario given a defined view radius (VR). For example, for VR 50, increasing wait time from 40 to 200 time steps resulted in better success rate, but when the wait time got too large the success rate dropped back down. A similar drop in success rate occurred for VR 150. This is due to a build up of agents that need local resolution until the number of deadlocks becomes too high or the deadlock resolution leads agents to unresolvable states.

To successfully plan this type of scenario, the values given to agents impact the overall success rate. In these scenarios, agents with a view radius larger than 50 had better results. To our knowledge, this type of approach has not been used to study to ingress/egress while considering agent dynamics and planning constraints on the agents.

C. *Larger Scenarios*

We have been able to simulate a larger scale parking lot scene with ingress and egress behaviors, shown in Figure 1. This is a simulated model of an actual parking lot and buildings on our campus. Simulation results are presented in

TABLE II: Simulation results reporting time for completed scenario varying wait time (WT) and view radius (VR).

				successful/attempts		
WT	VR	SR	Time	PullOut	PullIn	Deadlock
40	25	0.0	-	-	-	-
80	25	0.2	5990	32 (94%)	10 (100%)	102 (28%)
200	25	0.2	5974	32 (94%)	10 (100%)	40 (45%)
400	25	0.2	5483	31 (97%)	10 (100%)	9 (89%)
40	50	0.0	-	-	-	-
80	50	0.8	6530	39 (77%)	11 (91%)	28 (98%)
200	50	1.0	6609	35 (86%)	29 (34%)	12 (97%)
400	50	0.8	7236	36 (84%)	60 (17%)	18 (71%)
40	100	0.8	9934	32 (94%)	10 (100%)	5 (100%)
80	100	0.8	10,114	32 (94%)	10 (100%)	3 (100%)
200	100	0.8	11,020	39 (77%)	40 (25%)	6 (94%)
400	100	1.0	10,806	31 (97%)	10 (100%)	5 (90%)
40	150	1.0	17,145	30 (100%)	35 (29%)	3 (100%)
80	150	1.0	17,180	30 (100%)	10 (100%)	1 (100%)
200	150	1.0	17,142	30 (100%)	10 (100%)	5 (78%)
400	150	0.8	18,340	30 (100%)	10 (100%)	1 (100%)

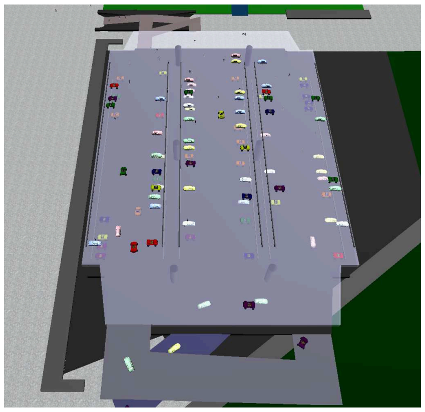

Fig. 4: An actual environment with a parking garage replacing the previous parking lot.

our animations. This represents a larger environmental model in the size of the area. Our roadmap-based approach scales well as do the resulting motions. We have also explored a parking garage structure, shown in Figure 4, that could replace the lot shown in Figure 1. This type of planning could allow an urban designer to see the effect of higher capacity of pedestrians and vehicles and issues that might arise in expanding an area.

VIII. Conclusion

In this paper, we proposed a framework for improving the overall motion in environments that include parking lots consisting of pedestrians and vehicles arriving and leaving. We identified a number of local motion maneuvers that could be applied in such a scenario to improve agent motion and resolve deadlocks that occur. Our system can be beneficial to planners who design parking lots and urban areas. These tools may be applied to identify bottleneck situations that may occur during the design process, and propose strategies to alleviate them in a post-occupancy setting. This system represents a complex simulation system that accounts for dependent motion between multi-modal agents. In the future, we could build upon our library of maneuvers, and more tightly integrate our local motion planner with the agent executing the plan.

References

[1] G.-L. C. Xin Zhang, "Optimal guidance of pedestrian-vehicle mixed flows in urban evacuation network," in *The 90th annual meeting of the Transportation Research Board*, 2011.

[2] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," in *NATURE*, 2000, pp. 487–490.

[3] J. Blum and A. Eskandarian, "The impact of multi-modal transportation on the evacuation efficiency of building complexes," in *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, oct. 2004, pp. 702 – 707.

[4] S. Singh, M. Kapadia, B. Hewlett, G. Reinman, and P. Faloutsos, "A modular framework for adaptive agent-based steering," in *Symposium on Interactive 3D Graphics and Games*, ser. I3D '11. New York, NY, USA: ACM, 2011, pp. 141–150. [Online]. Available: http://doi.acm.org/10.1145/1944745.1944769

[5] N. Pelechano, J. Allbeck, and N. Badler, *Virtual Crowds: Methods, Simulation, and Control*. Synthesis Lectures on Computer Graphics and Animation, Morgan & Claypool, 2008.

[6] S. C. Pursals and F. G. Garzon, "Optimal building evacuation time considering evacuation routes," *European Journal of Operational Research*, vol. 192, pp. 692–699, 2009.

[7] N. Pelechano and N. Badler, "Modeling crowd and trained leader behavior during building evacuation," in *IEEE Computer Graphics and Applications*, vol. 26, no. 6, 2006, pp. 80–86.

[8] N. Pelechano, J. Allbeck, and N. Badler, "Controlling individual agents in high-density crowd simulation," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2007.

[9] P. Thompson and E. Marchant, "A computer model for the evacuation of large building populations," *Fire Safety Journal*, vol. 24, pp. 131–148, 1995.

[10] K. Christensen and Y. Sasaki, "Agent-based emergency evacuation simulation with individuals with disabilities in the population," *Journal of Artificial Societies and Social Simulation*, vol. 11, no. 39, 2008.

[11] C. W. Johnson and L. Nilsen-Nygaard, "Extending the use of evacuation simulators to support counter terrorism," in *International Systems Safety Conference*, 2008.

[12] Z. Fang, Q. Li, Q. Li, L. D. Han, and D. Wang, "A proposed pedestrian waiting-time model for improving space-time use efficiency in stadium evacuation scenarios," *Building and Environment*, vol. 46, pp. 1774–1784, September 2011.

[13] N. Pelechano and A. Malkawi, "Evacuation simulation models: Challenges in modeling high rise building evacuation with cellular automata approaches," *Automation in Construction*, vol. 17, pp. 377–385, 2008.

[14] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: Highly parallel collision avoidance for multi-agent simulation," in *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2009, pp. 177–187.

[15] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Proceedings of Robotics Research: The 14th International Symposium, ISRR*, Madrid, Spain, 2011, pp. 1–16.

[16] S. J. Guy, J. Chhugani, S. Curtis, P. Dubey, M. Lin, and D. Manocha, "Pledestrians: A least-effort approach to crowd simulation," in *SCA '10: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Madrid, Spain: Eurographics Association, 2010.

[17] S. Rodriguez and N. M. Amato, "Behavior-based evacuation planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2010, pp. 350–355.

[18] S. Rodriguez, A. Giese, N. M. Amato, S. Zarrinmehr, F. Al-Douri, and M. Clayton, "Environmental effect on egress simulation," in *Proc. of the 5th Intern. Conf. on Motion in Games, 2012, Lecture Notes in Computer Scien ce (LNCS) (to appear)*, 2012.

[19] S. Rodriguez and N. M. Amato, "Utilizing roadmaps in evacuation planning," *24th Intern. Conf. on Computer Animation and Social Agents (CASA), 2011, in Intern. Journal of Virtual Reality*, pp. 67–73, 2011.

[20] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Trans. on Robotics*, vol. 21, no. 6, pp. 1077–1091, December 2005.