Collision Free Navigation with Interacting, Non-Communicating Obstacles

Mario Santillo, Mrdjan Jankovic

Abstract—In this paper we consider the problem of navigation and motion control in an area densely populated with other agents. We propose an algorithm that, without explicit communication and based on the information it has, computes the best control action for all the agents and implements its own. Notably, the host agent (the agent executing the algorithm) computes the differences between the other agents’ computed and observed control actions and treats them as known disturbances that are fed back into a robust control barrier function (RCBF) based quadratic program. A feedback loop is created because the computed control action for another agent depends on the previously used disturbance estimate. In the case of two interacting agents, stability of the feedback loop is proven and a performance guarantee in terms of constraint adherence is established. This holds whether the other agent executes the same algorithm or not.

I. INTRODUCTION

We consider the problem of an autonomous vehicle or robot (referred to as the "host") operating in an environment with other autonomous or human-operated agents (referred to as "targets") that it has no communication with, yet interacts through observed actions. In contrast to the problem of avoiding stationary obstacles, the host’s path depends on what other agents decide to do and this, in turn, depends on what the host agent does, and so on. In many less structured traffic situations – an open parking lot, a construction zone, a large roundabout – both the path and timing of each participant has to be modified in response to what everyone else is doing. A similar situation could occur with mobile robots that share an open space with other robots, humans, and/or human operated vehicles with no clear rules of precedence.

Most papers concerning collision avoidance with interacting agents are in the field of robotics. The Interacting Gaussian Process (IGP) method [12] is a position-based approach that tackles the so-called "frozen robot problem" with humans acting as interacting moving obstacles to the mobile robot. The algorithm computes "likely" future positions of the host and the targets using a joint probability density function obtained from empirical data. No kinematic or dynamic model was used – it is just assumed that the robot and the pedestrians could occupy these precomputed positions at the scheduled time.

Another approach is based on reciprocal velocity obstacles [11], [13] with each agent assuming half of the responsibility for collision avoidance. The agents need to reach an implicit agreement on what side to pass on. Otherwise, “reciprocal dance” occurs where the agents switch sides being unable to decide. If no permissible velocity can be found, the algorithm removes the farthest away robot or obstacle from consideration and repeats the calculation. Velocity is also used as the control variable in Lyapunov-barrier function (LBF) based methods, such as [8], [10], where a non-holonomic unicycle robot model is considered with communication assumed in [10]. The LBFS provide a guarantee of collision-free motion if their values are non-increasing. A side effect of LBFS being infinite at the boundary is that a region around an obstacle becomes inaccessible – the algorithm sees obstacles as larger than they actually are.

The acceleration-based approach with Control Barrier Functions (CBF) is considered in [3], [14]. The CBFS provide linear constraints on acceleration for a Quadratic Program (QP). In the centralized version, the controller knows everyone’s desired acceleration and computes everyone’s instantaneously-optimal control action while adhering to the constraints. Compared to LBF’s, the CBF with QP allows the distance to the obstacle boundary to decrease to 0, but at a rate that decreases with the distance. An advantage over the reciprocal velocity approach is that the optimization is not bilateral (i.e. each pair of agents handled independently), but multilateral (all at once). A disadvantage is that it requires communication to all the agents. In the decentralized, no-communication case also considered in [3], [14], the host computes only its own action assuming targets velocities are constant. Compared to the centralized controller that has control inputs for all the agents at its disposal, the decentralized QP is more likely to be infeasible producing no solution. In such a situation, [14] proposes the (host) agent enter a separate “braking mode” – i.e. stop with full deceleration.

In this paper, we set up a quasi-centralized QP based on Robust Control Barrier Functions (RCBFs) (see [6]) while still assuming no explicit communication between agents. Each agent computes optimal accelerations for all the agents, with information at its disposal, and implements its own control action. Because the agents, in general, will not agree, the method considers other agent actions as disturbances for RCBF. The disturbances could be assumed bounded or, as we have done in this paper, estimated on line as a difference between the actual target acceleration and the one computed by the host. This creates a static loop (the disturbance is used to compute the target acceleration, while the acceleration is used to compute the disturbance) that has
to be cut by inserting a unit delay, i.e. using the value from the previous sample, or a low pass filter creating internal controller states. Because the host predicts and then corrects the target acceleration, we refer to the algorithm as the Predictor-Corrector for Collision Avoidance (PCCA). Here are some properties of the PCCA controller:

1) The QP for the PCCA controller has the same feasibility as the centralized controller of [14] because it has the control action of all agents at its disposal, though only its own is actually applied.

2) In the case of 2 agents, we prove that controller internal states are bounded and any error in constraint enforcement is of the order of the sampling time $\Delta t$. Quantitatively similar performance guarantee is established even if the target agent is not interacting, while the host, by running PCCA, assumes it is. Based on results observed in simulations, we believe these properties might hold for multi-agent cases as well.

3) The PCCA computational complexity (for each agent as the host) is observed to be similar to the centralized controller. With $N_a$ denoting the number of agents, complexity theoretically scales as $N_a^4$ using the interior point method [4].

The paper is organized as follows. Section II reviews the results related to QP with RCBF. Section III introduces the model for the agents, equations describing the controller, and contains the main results. The simulations in Section IV consider cases of 2 agents, both running the PCCA algorithm or one running PCCA and the other non-interacting.

**Notation:** For a differentiable function $h(x)$ and a vector $f(x)$, $L_f h(x)$ denotes $\frac{\partial}{\partial x} f(x)$. A function $\alpha: \mathbb{R}^+ \rightarrow \mathbb{R}^+$ is of class $\mathcal{K}$ if it is continuous, zero at zero, and strictly increasing. A function $\gamma(t, \varepsilon)$ is said to be $\mathcal{O}(\varepsilon)$ if $|\gamma(t, \varepsilon)| \leq \kappa|\varepsilon|$ for some $\kappa > 0$ and for all sufficiently small $\varepsilon$.

**II. ROBUST CONTROL BARRIER FUNCTIONS**

In this section, we review the concept of robust Control Barrier Functions introduced in [6]. RCBFs are an extension of CBFs introduced in [15] for systems with a bounded external disturbance $w(t) \in \mathbb{R}^n$, $\|w(t)\| \leq \bar{w} > 0$, of the form

$$\dot{x} = f(x) + g(x)u + p(x)w$$  \hspace{1cm} (1)

One control objective for (1) is to regulate the state close to the origin (e.g. input-to-state stability (ISS)) and we assume that there is a known baseline controller $u_0$ that achieves this objective. The other control objective is to keep the state of the system in an admissible set $\mathcal{C} = \{x : h(x) \geq 0\}$. Below is the “zeroing” version of Robust Control Barrier Functions (RCBF) definition.

**Definition 1:** (Robust-CBF) For the system (1), a differentiable function $h(x)$ is an RCBF with respect to the admissible set $\mathcal{C} = \{x : h(x) \geq 0\}$ if there exists a class $\mathcal{K}$, Lipschitz continuous function $\alpha_h$ such that

$$L_y h(x) = 0 \Rightarrow L_f h(x) - \|L_p h\| \bar{w} + \alpha_h(h(x)) > 0$$  \hspace{1cm} (2)

The definition states that, when the control input has no impact on $\dot{h} = L_f h(x) + L_p h w + L_y h u$, under the worst-case disturbance, the RCBF $\dot{h}$ cannot decrease towards 0 faster than $\alpha_h(\dot{h})$.

In the case of an unknown disturbance, we set up a barrier constraint for the worst case disturbance:

$$F_1 = L_f h(x) - \|L_p h(x)\| \bar{w} + L_y h(x) u + \alpha_h(h(x)) \geq 0$$  \hspace{1cm} (3)

With a good estimate or a measurement of the disturbance $\bar{w}$, one can be less conservative by using

$$F_2 = L_f h(x) + L_p h(x) \bar{w} + L_y h(x) u + \alpha_h(h(x)) \geq 0$$  \hspace{1cm} (4)

In both cases, the idea is to find a control $u$, that is as close as possible to the baseline control $u_0$ in the Euclidean distance sense, such that the barrier constraint is satisfied. Compared to [6], the Lyapunov (CLF) constraint is removed and replaced by $u_0$. While the Lyapunov constraint provides two tuning parameters to adjust responsiveness of the controller (see [6]), it would make the analysis carried out in the next section more difficult. From the theoretical standpoint, the change does not make a difference.

**Robust QP Problem:** Find the control $u$ that satisfies

$$\min_u \|u - u_0\|^2 \text{ subject to } F_i \geq 0, \quad i = 1 \text{ or } 2$$  \hspace{1cm} (5)

where we select $F_1$ or $F_2$ (see (3) and (4)) depending on whether we use an estimate/measurement of the disturbance or the worst case upper bound. In either case, the following result applies.

**Theorem 1:** If $h(x)$ is an RCBF for the system (1) then

1) The Robust QP problem (5) is feasible and the resulting control law is Lipschitz continuous in $\mathcal{C}$.

2) $h(x) \geq -\alpha_h(h(x))$ for all $x \in \mathcal{C}$ and the set $\mathcal{C}$ is forward invariant.

3) If the barrier constraint is inactive, $u = u_0$. As a result, if the barrier constraint is inactive for all $t$ greater than some $t_*$ and $u_0$ is an ISS controller, the closed loop system is input to state stable with respect to the disturbance input $w$.

The proof follows from Theorem 2 of [6], with appropriate modifications for zeroing RCBF and $u_0$ replacing the CLF constraint.

In contrast to the standard definitions of CBF (e.g. [11], [15]) and RCBF [6], the barrier function considered for interacting agents later in this paper has relative-degree two. That is, the control input and the disturbance input do not appear in $h$, but in $\dot{h}$—hence, relative degree two. Here, we briefly review the approach [9], [16], which considers

$$\ddot{h} + l_2 \dot{h} + l_0 h \geq 0$$  \hspace{1cm} (6)

as the QP constraint. The parameters $l_0, l_1$ should be selected so that the two roots $\lambda_{1,2} = -\frac{l_2 \pm \sqrt{l_2^2 - 4l_0}}{2}$ of the polynomial $s^2 + l_1 s + l_0 = 0$ are negative real. Then, if the barrier
constraint \( \{ \} \) holds, it is not difficult to show that the set \( C^* = \{ x \) : \( h(x) \geq 0, h(x) \geq \frac{1}{\lambda_1} h(x) \} \), with \( \lambda_1 \) being either one of the two roots, is forward invariant. Picking the smaller (more negative one) makes \( C^* \) larger. With \( C^* \subseteq C \) the original constraint \( h(x) \geq 0 \) will be satisfied.

Taking into account the disturbance, the RCBF condition for the system \( \{ \} \) with a barrier \( u \) from \( h \) smaller (more negative one) makes \( u \) controller, requiring full knowledge of \( h \).

As follows:

\[ \text{Centralized QP: Find the controls } u_i, i = 1, \ldots, N_a \]

\[
\min_{u_i} \sum_{i=1}^{N_a} \|u_i - u_{0i}\|^2 \quad \text{subject to} \\
a_{ij} + b_{ij}(u_i - u_j) \geq 0 \forall i, j = 1, \ldots, N_a, i \neq j
\]

The QP solution could be computed by a central node and communicated to the agents, or each agent could solve it independently. If the QP problem is feasible, the control action would satisfy all the barrier constraints \( \{ \} \) and guarantee collision-free operation \( \{ \} \).

Without communication, the base control action \( u_{0ij} \) for the targets are not available to the host \( i \) (i.e. the agent doing the computation). In this case, each agent could implement an on-board decentralized controller:

\[
\text{Decentralized QP} \quad \text{(for agent } i) \quad \text{Find the control } u_i \text{ for the agent } i \text{ that satisfies} \\
\min_{u_i} \|u_i - u_{0i}\|^2 \quad \text{subject to} \\
\chi a_{ij} + b_{ij}u_i \geq 0 \forall j = 1, \ldots, N_a, j \neq i
\]

The value \( \chi = 1 \), as used in \( \{ \} \), implies that the agent \( i \) assumes full responsibility for avoiding all the targets (no reciprocal action assumed), while \( \chi = 1/2 \) assumes evenly shared responsibility. It was proven in \( \{ \} \) that, if all the agents execute the same reciprocal policy (e.g. \( \chi = 1/2 \) for agents with equal acceleration capability) and the QP is feasible for all of them, then collision avoidance is guaranteed. However, with only the host’s control available to avoid all the \( N_a-1 \) targets, either version of the decentralized policy might be infeasible, necessitating action external to the QP (e.g. max braking proposed in \( \{ \} \)).

In this paper, we present a different QP policy (predictor-corrector for collision avoidance, PCCA) that is computed by each agent independently. It takes into account everyone’s constraints (\( a_{ij} \)’s and \( b_{ij} \)’s are known to all), but only the host’s own base control – the others are not known and 0’s are used instead. Even with the known set of constraints, the agents’ computed actions need not agree and the constraint maybe still be violated. Instead of increasing the radius margin to avoid potential collisions, we add a fictitious disturbance \( \tilde{w} \) to each target’s acceleration in \( \{ \} \) and use the RCBF design of Section \( \{ \} \). It is easy to check that \( b_{ij} \)’s are RCBFs because \( b_{ij} \neq 0 \) in \( C \), and, in fact, in any set that does not contain \( \xi_{ij} = 0 \) (that is, the circles don’t completely overlap). Applying the RCBF design described in Section \( \{ \} \) we obtain:

\[
\text{PCCA QP} \quad \text{(as computed by agent } i) \quad \text{Find control actions} \\
u_{ij}, j = 1, \ldots, N_a \text{ such that} \\
\min_{u_{ij}, \ldots, u_{N_a}} \left( \|u_{ii} - u_{0i}\|^2 + \sum_{j=1}^{N_a} \|u_{ij}\|^2 \right) \quad \text{subject to} \\
\chi a_{ik} + b_{ij}(u_{ij} + \tilde{w}_{ij} - u_{ik} - \tilde{w}_{ik}) \geq 0 \\
\forall j = 1, \ldots, N_a, j \neq k
\]

and implement its own: \( u_i = u^{**}_i \), where \( u^{**} \) denotes the solution to \( \{ \} \).
As one could ascertain by comparing to Section II, we have used the known-disturbance RCBF setup in (16). The disturbance estimate comes from comparing the control action for agent $j$ ($u^*_{ij}$) computed by the host (agent $i$) with the action agent $j$ actually implemented ($u_{ij}$):

\[ \hat{w}_{ij} = u_j - u^*_{ij} \]  

(17)

where, obviously, $\hat{w}_{ij} \equiv 0$. Target acceleration could be obtained from an estimator (e.g. \cite{2}) or by approximate differentiation of the velocity signal. To compute $\hat{w}_{ij}$ we need to know $\dot{w}_i$ and vice versa, creating a static (algebraic) feedback loop. We break the static loops by either using the value from the previous sample (the controller is implemented in discrete time) as illustrated in Fig. 1 or inserting a low-pass filter. In this paper we consider the former because it illuminates the stability mechanism more clearly.

The latter would correspond to the implementation where the target control action $u_j$ is obtained by approximate differentiation of its velocity.

\[ \hat{w}_{ij} = u_j - u^*_{ij} \]  

(17)

where, obviously, $\hat{w}_{ij} \equiv 0$. Target acceleration could be obtained from an estimator (e.g. \cite{2}) or by approximate differentiation of the velocity signal. To compute $\hat{w}_{ij}$ we need to know $\dot{w}_i$ and vice versa, creating a static (algebraic) feedback loop. We break the static loops by either using the value from the previous sample (the controller is implemented in discrete time) as illustrated in Fig. 1 or inserting a low-pass filter. In this paper we consider the former because it illuminates the stability mechanism more clearly.

The latter would correspond to the implementation where the target control action $u_j$ is obtained by approximate differentiation of its velocity.

\[ \hat{w}_{ij} = u_j - u^*_{ij} \]  

(17)

where, obviously, $\hat{w}_{ij} \equiv 0$. Target acceleration could be obtained from an estimator (e.g. \cite{2}) or by approximate differentiation of the velocity signal. To compute $\hat{w}_{ij}$ we need to know $\dot{w}_i$ and vice versa, creating a static (algebraic) feedback loop. We break the static loops by either using the value from the previous sample (the controller is implemented in discrete time) as illustrated in Fig. 1 or inserting a low-pass filter. In this paper we consider the former because it illuminates the stability mechanism more clearly.

The latter would correspond to the implementation where the target control action $u_j$ is obtained by approximate differentiation of its velocity.

\[ \hat{w}_{ij} = u_j - u^*_{ij} \]  

(17)

where, obviously, $\hat{w}_{ij} \equiv 0$. Target acceleration could be obtained from an estimator (e.g. \cite{2}) or by approximate differentiation of the velocity signal. To compute $\hat{w}_{ij}$ we need to know $\dot{w}_i$ and vice versa, creating a static (algebraic) feedback loop. We break the static loops by either using the value from the previous sample (the controller is implemented in discrete time) as illustrated in Fig. 1 or inserting a low-pass filter. In this paper we consider the former because it illuminates the stability mechanism more clearly.

The latter would correspond to the implementation where the target control action $u_j$ is obtained by approximate differentiation of its velocity.

\[ \hat{w}_{ij} = u_j - u^*_{ij} \]  

(17)

where, obviously, $\hat{w}_{ij} \equiv 0$. Target acceleration could be obtained from an estimator (e.g. \cite{2}) or by approximate differentiation of the velocity signal. To compute $\hat{w}_{ij}$ we need to know $\dot{w}_i$ and vice versa, creating a static (algebraic) feedback loop. We break the static loops by either using the value from the previous sample (the controller is implemented in discrete time) as illustrated in Fig. 1 or inserting a low-pass filter. In this paper we consider the former because it illuminates the stability mechanism more clearly.

The latter would correspond to the implementation where the target control action $u_j$ is obtained by approximate differentiation of its velocity.

\[ \hat{w}_{ij} = u_j - u^*_{ij} \]  

(17)

where, obviously, $\hat{w}_{ij} \equiv 0$. Target acceleration could be obtained from an estimator (e.g. \cite{2}) or by approximate differentiation of the velocity signal. To compute $\hat{w}_{ij}$ we need to know $\dot{w}_i$ and vice versa, creating a static (algebraic) feedback loop. We break the static loops by either using the value from the previous sample (the controller is implemented in discrete time) as illustrated in Fig. 1 or inserting a low-pass filter. In this paper we consider the former because it illuminates the stability mechanism more clearly.

The latter would correspond to the implementation where the target control action $u_j$ is obtained by approximate differentiation of its velocity.

\[ \hat{w}_{ij} = u_j - u^*_{ij} \]  

(17)

where, obviously, $\hat{w}_{ij} \equiv 0$. Target acceleration could be obtained from an estimator (e.g. \cite{2}) or by approximate differentiation of the velocity signal. To compute $\hat{w}_{ij}$ we need to know $\dot{w}_i$ and vice versa, creating a static (algebraic) feedback loop. We break the static loops by either using the value from the previous sample (the controller is implemented in discrete time) as illustrated in Fig. 1 or inserting a low-pass filter. In this paper we consider the former because it illuminates the stability mechanism more clearly.

The latter would correspond to the implementation where the target control action $u_j$ is obtained by approximate differentiation of its velocity.

\[ \hat{w}_{ij} = u_j - u^*_{ij} \]  

(17)

where, obviously, $\hat{w}_{ij} \equiv 0$. Target acceleration could be obtained from an estimator (e.g. \cite{2}) or by approximate differentiation of the velocity signal. To compute $\hat{w}_{ij}$ we need to know $\dot{w}_i$ and vice versa, creating a static (algebraic) feedback loop. We break the static loops by either using the value from the previous sample (the controller is implemented in discrete time) as illustrated in Fig. 1 or inserting a low-pass filter. In this paper we consider the former because it illuminates the stability mechanism more clearly.

The latter would correspond to the implementation where the target control action $u_j$ is obtained by approximate differentiation of its velocity.
from (23), but we shall write out the dynamics for $\hat{w}_2$ needed later:
\[
\hat{w}_2(k) = \left[D\hat{w}_2 + (I - D)u_{02} + \frac{ab^T}{2bb^T}\right]_{k-1} + D(k-1)(u_{01}(k-2) - u_{02}(k-2))
\]
\[+ D(k-1)(u_{01}(k-2) - u_{02}(k-2))
\]
(25)

Case C ($\mu_1(k-1) \leq 0, \mu_2(k-1) > 0$): This case is symmetric to Case B and the same conclusion follows.

Case D ($\mu_1(k-1) \leq 0, \mu_2(k-1) \leq 0$): Define $\sigma_w = \hat{w}_1 + \hat{w}_2$ and note that
\[
\hat{w}_1(k-1) = \frac{1}{2}[\sigma_w(k-1) + u_{01}(k-2) - u_{02}(k-2)]
\]
\[
\hat{w}_2(k-1) = \frac{1}{2}[\sigma_w(k-1) - u_{01}(k-2) + u_{02}(k-2)]
\]
Using these two inequalities to replace $\hat{w}_i$’s with $\sigma_w$ in the $\mu_i \leq 0$ inequalities that define the Case D and rearranging terms we obtain
\[
(b \sigma_w)_{k-1} \leq (2bu_{02} - 2a)_{k-1} - b(k-1)(u_{01} - u_{02})_{k-2}
\]
\[
(b \sigma_w)_{k-1} \geq (2a + 2bu_{01})_{k-1} - b(k-1)(u_{01} - u_{02})_{k-2}
\]
(26)

Note here that the right-hand sides are bounded by assumption. If we add the two equalities in (24) we obtain
\[
\sigma_w(k) = [2D\sigma_w + (I - 2D)(u_{01} + u_{02})]_{k-1}
\]
Multiplying both sides by $b_1(k-1)$ and its orthogonal with the same magnitude $b_1^*(k-1)$ we obtain
\[
b(k-1)\sigma_w(k) = (b \sigma_w)_{k-1}
\]
\[
b_1(k-1)\sigma_w(k) = [b^*(u_{01} + u_{02})]_{k-1}
\]
Because the right-hand sides are bounded (the first from (24), the second by assumption) and the matrix $\begin{bmatrix} b & b_1 \end{bmatrix}_{k-1}$ is invertible with determinant $\geq 4r_0^2$, $\sigma_w$, the sum of the $\hat{w}_i$’s, is bounded. The difference $\hat{w}_2 - \hat{w}_1$ is also bounded by (24) and, hence, $\hat{w}_i$’s are bounded in Case D.

Thus far, we have shown that $\hat{w}_i(k)$’s are bounded if they are computed in Cases A and D. They are also bounded if they stay in Cases B and C due to their input-to-state stable dynamics. Finally, we consider the possibility of switching between Cases B and C. By considering (24), (25) and using the symmetry, we obtain that in both cases the dynamics are between Cases B and C.

**Remark 1:** The proof establishes that in all four cases $\hat{h} + l_1 \hat{h} + l_0 \hat{h} \geq \Delta u_{0i}$. With $|b| \leq \beta$ and $\|u_{0i}\| \leq M$, $i = 1, 2$ for some $\beta, M > 0$ (as assumed in Theorem 2), we have that $\hat{h} + l_1 \hat{h} + l_0 \hat{h} \geq -2\beta M \Delta t$ providing a way to assess how small the sampling time needs to be, or, for a given sampling time, compute the necessary radius margin.

The above result tells us what happens when the agents are cooperating. Next, we analyze the PCCA controller when the situation is the opposite – agent 2 is not interacting, while agent 1 assumes it is. That is, the control action of agent 1 is computed by (19) while agent 2 simply applies its unconstrained control action $u_2 = u_{02}$. Note that only $\hat{w}_2$ is computed in this case.

**Proposition 1:** The system (11), with the controller $u_1$ computed by the controller (19) and $u_2 = u_{02}$ assumed bounded and differentiable, satisfies the following: for any $T > t_0$ there exists $\Delta t^*$ such that for all sampling times $\Delta t < \Delta t^*$
\[
h(\xi(t)) \geq O(\Delta t), \forall t \in [t_0, T]
\]
(27)

**Proof:** The proof relies on the singular-perturbation approach (see, for example, Chapter 11 in [22]). To this end, we introduce a singular perturbation parameter $\varepsilon: \Delta t = \varepsilon \Delta t_0$. To simplify notation, we assume $\Delta t_0 = 1s$. Using (24), the symmetry between Cases B and C, and rearranging terms, we obtain the fast dynamics of the singularly perturbed system:
\[
\varepsilon \hat{w}_2 = A\hat{w}_2 - Au_{02} + \min\{0, \mu_{cc}\} \frac{b^T}{2bb^T} + O(\varepsilon)
\]
(28)
where $A = D - I$ with $D$ and $\mu_{cc}$ defined for (11) in the proof of Theorem 2. The fast $\hat{w}_2$ dynamics is exponentially stable uniformly in the "slow" variables $a$, $b$, and $u_{0i}$. The slow dynamics is given by (11) with
\[
u_{01} - u_2 = u_{01} - u_{02} - \min\{0, a + bu_{01} - bw_{02}\} \frac{b^T}{2bb^T} + O(\varepsilon)
\]

Using the standard singular-perturbation approach, we assume that the slow variables are frozen and compute the quasi steady-state behavior of $\hat{w}_2$ at $\varepsilon = 0$. For the slow dynamics $(\xi, v)$, we are only interested in $bw_{02}$. Solving (28)
for the equilibrium of $\dot{w}_2$ at $\varepsilon = 0$, denoted by $\hat{w}_2$, and using $b(I - D) = \frac{1}{2}$, we have

$$b\hat{w}_2 = \begin{cases} b_0u_02 & \text{if } \mu_{cc} > 0 \\ -a - b_0u_01 + 2b_0u_02 & \text{if } \mu_{cc} \leq 0 \end{cases}$$

(29)

Using the quasi steady-state value $b\hat{w}_2$ in the reduced model, we obtain

$$\dot{\hat{v}} = \begin{cases} u_01 - u_02 & \text{if } \mu_{cc} > 0 \\ u_01 - u_02 - \mu_{cc}^{\frac{\varepsilon}{\varepsilon}} & \text{if } \mu_{cc} \leq 0 \end{cases}$$

(30)

With $\hat{\xi} = \hat{v}$, the reduced $(\hat{\xi}, \hat{v})$ dynamics are identical to the dynamics that the centralized QP controller would have produced. It is easy to check that they satisfy the barrier constraint (13) and, hence, achieve $h(\xi) \geq 0$.

The singular-perturbation result (Theorem 11.1 in [10]) states that, on any finite time interval $t \in [t_0, T]$, for sufficiently small $\varepsilon$, the difference between the solution $\xi(t, \varepsilon)$ of the original system and the solution $\xi(t)$ of the reduced system is $O(\varepsilon)$ uniformly in $t \in [t_0, T]$. As a result,

$$h(\xi(t, \varepsilon)) \geq O(\varepsilon)$$

(31)

Substituting $\Delta t = \varepsilon\Delta t_0 = \varepsilon$ completes the proof of the Proposition.

Because $O(\varepsilon)$ is not sign definite, with one agent non-interacting, the PCCA controller requires a radius margin of the order $\Delta t$ to ensure collision-free operation. The results of item 3 in Theorem 2 and the singular-perturbation result of Proposition 1 look qualitatively very similar, but are quantitatively different. With both agents cooperating, (23) provides that, for constraint adherence, the control adjusts to changes with one sample lag. With one agent non-interacting, the other agent takes only a part of the responsibility in each step. Because the other agent is not reacting, it eventually takes full responsibility with the time constant of $2\Delta t$. We will see the difference in simulations in the next section where the former case does not require any radius margin, while the latter does.

IV. Simulation Results

We illustrate the results by simulation in the cases of two agents maneuvering in an enclosed area. In each case, the radius of the agents’ circles is taken to be $r_0 = 2$, and hence, including the radius margin, $r \geq 4$ is used for computation of the barrier constraint $h$. The controller sample time is $\Delta t = 50$ ms. For computation of the QP constraints (13), we choose $l_0 = 6$ and $l_1 = 5$ to sify $l_1^2 \geq 4l_0$. The baseline controller $u_{0i}$ for each agent is computed by the Linear Quadratic Regulator (LQR) with $Q = 4I_4$ and $R = I_2$ intended to bring the agent to a preassigned destination.

Example 1: [Two Interacting Agents] In this example, we illustrate the controller operation by considering the completely symmetric case of two agents approaching each other head-on while heading towards each other’s initial position. Because the PCCA controller is continuous and the case completely symmetric, the agents brake instead of steer and stop without colliding (second plot from the top in Fig. 2). Stimulated by numerical rounding error or, more likely, precision setting of the solver, at some point (third plot in Fig. 2) they start moving again and pass one another without collision. Note that the agents only run the PCCA algorithm, with no preference for the passing side and no external de-conflicting mechanism such as in [5] or [14].

Example 2: [Two Agents as Pursuer/Evader] In this example agent 1 (evader) acts as before, that is, executes the controller (19), but agent 2’s (pursuer) goal is modified to intercept agent 1. That is, agent 2 operates without any collision avoidance action and its only control input is the LQR control $u_{02}$ with the destination being the current location of agent 1.

Figure 3 displays time snapshots of the pursuer/evader scenario. We see that, since agent 2 continually pursues agent 1, it is impossible for agent 1 to reach its destination and come to a rest. With only one agent interacting, the analysis at the end of Section III suggests that we must add a margin ($r > 2r_0$) to ensure collision-free operation. In this case, we
add a margin equivalent to just over 1% of the agent’s radius such that the barrier for actual physical collision avoidance, $h_{r_0}(x) = \xi^T \xi - 2r_0^2$, remains positive. This is shown in Figure 4 where the barrier $h_{r_0} \geq 0$. Additionally, as the simulation sample time reduces by a factor of 5 (50 ms to 10 ms), the radius margin necessary to ensure collision-free operation reduces by a factor of 6, confirming the results presented in Section III.

V. Conclusions

We considered the problem of navigation and motion control in an area shared with other agents. Without explicit communication, the control algorithm utilizes observed acceleration of each target agent and compares it to its “best” action as computed by the host. The difference between observed and computed accelerations guides a modification to the action taken by the host and a recomputed best action for the targets. The algorithm is shown to be stable and to avoid collisions if the sampling is fast enough. The result applies in the case the other agent is cooperating, as well as when it is not cooperating as assumed. The design allows a tight representation of obstacles which is beneficial in densely populated operating areas.

REFERENCES

[1] A.D. Ames, X.W. Grizzle, P. Tabuada, “Control Barrier Function Based Quadratic Programs for Safety Critical Systems,” IEEE Trans. on Automatic Control vol. 62, pp. 3861 - 3876, Aug. 2017.

[2] A. Ansari, D. Bernstein, “Input Estimation for Nonminimum-Phase Systems with Application to Acceleration Estimation for a Maneuvering Vehicle,” IEEE Trans. Contr. Sys. Techn., Vol. 27, pp. 1590-1607, 2019.

[3] U. Borrmann, L. Wang, A. D. Ames, M. Egerstedt, “Control Barrier Certificates for Safe Swarm Behavior,” in Proc. IFAC Conf. Anal. Des. Hybrid Syst., pp. 68-73, Oct. 2015.

[4] S.P. Boyd, L. Vandenberghe Convex Optimization, Cambridge University Press, 2004.

[5] F. Celi, L. Wang, L. Pallottino, M. Egerstedt, “Deconfliction of Motion Paths with Traffic Inspired Rules in Robot-Robot and Human-Robot Interactions,” Robotics & Automation Letters, Vol. 4, pp. 2227-2234, 2019.

[6] M. Jankovic, “Robust Control Barrier Functions for Constrained Stabilization of Nonlinear Systems,” Automatica, Oct. 2018.

[7] H.K. Khalil, Nonlinear Systems, Prentice Hall, 2002.

[8] S. Masellone, D. Stipanovic, C. Graunke, K. Intlekofer, M. Spong, “Formation Control for Collision Avoidance for Multi-agent Non-holonomic Systems: Theory and Experiments,” Int. J. of Robotics Research, vol. 27, pp 107-126, Jan 208.

[9] Q. Nguyen, K. Sreenath, “Exponential Control Barrier Functions for Enforcing High Relative-Degree Safety-Critical Constraints,” American Control Conference, Boston, 2016.

[10] D. Panagou, D. Stipanovic, P. Vulgaris, “Distributed Coordination Control for Multi-robot Networks using Lyapunov-like Barrier Functions,” IEEE Trans on Automatic Control, vol. 61, pp. 617-632, Mar 2016.

[11] J. Snape, J. van den Berg, S. Guy, “The Hybrid Reciprocal Velocity Obstacle,” IEEE Transaction on Robotics, vol. 27, pp. 696-706, Aug. 2011.

[12] P. Trautman, A. Krause, “Unfreezing the Robot: Navigation in Dense, Interacting Crowds,” IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2010.

[13] J. van den Berg, S. Guy, M. Lin, D. Manocha, “Reciprocal n-body Collision Avoidance,” 14th Int. Symposium on Robotics Research, 2009.

[14] L. Wang, A. D. Ames, M. Egerstedt, “Safety Barrier Certificates for Collision-Free Multirobot Systems,” IEEE Transaction on Robotics, vol. 33, pp 661-674, June 2017.

[15] P. Wieland, F. Allgower, “Constructive Safety using Control Barrier Functions,” Proceedings of IFAC Symp. on Nonlinear Control Systems, 2007.

[16] X. Xu, “Constrained Control of Input-output Linearizable Systems using Control Sharing Barrier Functions,” Automatica, vol. 87, pp 195-201, 2018.

[17] X. Xu, P. Tabuada, J.W. Grizzle, A.D. Ames, “Robustness of Control Barrier Functions for Safety Critical Control,” IFAC Conference Analysis and Design of Hybrid Systems, 2015 (updated version at https://arxiv.org/abs/1612.01554).