# A Geometric Approach to Resilient Distributed Consensus Accounting for State Imprecision and Adversarial Agents

Christopher A. Lee and Waseem Abbas

***Abstract*— This paper presents a novel approach for resilient distributed consensus in multiagent networks when dealing with adversarial agents and imprecision in states observed by normal agents. Traditional resilient distributed consensus algorithms often presume that agents have exact knowledge of their neighbors' states, which is unrealistic in practical scenarios. We show that such methods are inadequate when agents only have access to imprecise states of their neighbors. To overcome this challenge, we adapt a geometric approach and model an agent's state by an 'imprecision region' rather than a point in $\mathbb{R}^d$. From a given set of imprecision regions, we first present an efficient way to compute a region that is guaranteed to lie in the convex hull of true, albeit unknown, states of agents. We call this region the *invariant hull* of imprecision regions and provide its geometric characterization. Next, we use these invariant hulls to identify a *safe point* for each normal agent. The safe point of an agent lies within the convex hull of its *normal* neighbors' states and hence is used by the agent to update it's state. This leads to the aggregation of normal agents' states to safe points inside the convex hull of their initial states, or an approximation of consensus. We also illustrate our results through simulations.**

## I. Introduction

In distributed multi-agent networks, agents work collaboratively to accomplish complex tasks by sharing information with each other. However, false or misleading information from a subset of agents–due to adversarial actions or other failures–can significantly compromise the network's operations. Thus, to realize the full benefits of distributed decision-making, resilience to abnormal agents is crucial for the correct functioning of multi-agent systems. As such, there has been an increased interest in characterizing and imposing resilience in distributed systems.

Recent works on resilient distributed networks, particularly in resilient distributed consensus—a canonical problem in networked and distributed control systems–present a suite of algorithms and conditions on the network structure. If these conditions are satisfied, they guarantee the normal operations of the overall system [1], [2]. The main idea of such algorithms is to minimize the impact of information from adversarial or faulty agents, even without knowing their identities [3]–[12]. However, existing resilient consensus solutions often make the unrealistic assumption that agents can observe their neighbors' exact, unperturbed states. In real-world applications, agents deal with imprecise data due to factors like sensor noise, environmental conditions, or hardware limitations [13], [14]. As we demonstrate in this paper, applying current resilient algorithms in these 'imprecise' scenarios can lead to failures, even if the number of adversarial agents is below the theoretical threshold allowed by the solution. Therefore, new methods are needed, and this paper introduces strategies to handle both adversarial agents and imprecise states in multi-agent networks.

This paper introduces a new method to tackle the resilient distributed consensus problem, focusing on the commonly overlooked issue of state imprecision, that is, the observed states vary from the true state by a known bounded measure. There are existing studies that address a similar problem in a static scenario [15], however our formulation is developed independently and tailored towards the application of resilient consensus. Our approach ensures that all normal agents remain within the convex hull of their initial states and each agent continuously converges towards a point contained in the convex hull of the normal agents' initial positions, which we refer to as *approximate consensus*.

We summarize our key contributions as follows:

- First, we show that current algorithms for resilient consensus do not work well when agents have imprecise states (Section III). This is important because imprecise states are common in real-world applications.
- Second, we introduce a new way to model these imprecise states. Instead of using points in $\mathbb{R}^d$, we use what we call 'imprecision regions' containing true states of agents. We then introduce the notion of *invariant hull* which is essentially the largest region that is contained in the convex hull of normal agents' true states when only their imprecision regions are known. We also provide a geometric characterization of invariant hulls and an efficient algorithm to compute them (Section IV).
- Third, we develop a method that leverages these invariant hulls to identify a 'safe point' in the neighborhood of a normal agent. This neighborhood may contain up to $\frac{N_v}{d+1} - 1$ adversaries. Here, $N_v$ is the total number of nodes in the neighborhood of a normal agent $v$ and $d$ is the state dimension. This safe point allows the normal agent to update its state while ignoring any adversaries. We also illustrate our results through simulations (Section V).

## II. Preliminaries

We consider a network of agents modeled by an undirected graph $G = (V, E)$, where $V$ represents agents and $E$ denotes interactions among agents. An edge between agents $u$ and $v$ is denoted by an unordered pair $(u, v)$. We use the terms *agent* and *node* interchangeably. Each agent $u \in V$ has a $d$-dimensional state vector whose value is updated over time.

C. A. Lee is with the Electrical Engineering Department, and W. Abbas is with the Systems Engineering Department at the University of Texas at Dallas, Richardson, TX, USA (Emails: christopher.lee@utdallas.edu, waseem.abbas@utdallas.edu).

The state of agent $u$ at time $t$ is represented by a point $x_u(t) \in \mathbb{R}^d$. The *neighborhood* of $u$ is the set of nodes $N_u = \{v \in V : (u,v) \in E\} \cup \{u\}$ (node $u$ is included in its neighborhood). For a given set of points $X \subset \mathbb{R}^d$, we denote its *convex hull* by $\texttt{Conv}(X)$. Also, we use terms *points* and *states* interchangeably.

*Normal and Adversarial Agents* – Agents in the network can either be normal or adversarial. *Normal* agents, denoted by $V_n \subseteq V$, are the ones that interact with their neighbors synchronously and update their states according to a predefined state update rule, which is the consensus algorithm. *Adversarial* agents, denoted by $V_f \subset V$, are the ones that can change their states arbitrarily. Moreover, an adversarial node can transmit different values to its different neighbors, which is referred to as the *Byzantine* model. The number of adversarial nodes in the neighborhood of a normal node $u$ is denoted by $F_u$. For a normal node $u$, all nodes in its neighborhood are indistinguishable, that is, $u$ cannot identify which of its neighbors are adversarial.

*Agent Imprecision* – We consider that each agent has an imprecise information of its neighbor's state. In other words, an agent $u$ having agent $v$ as a neighbor, acquires/observes an imprecise value of agent $v$'s state, which could be due to perturbation as a result of improper calibration, hardware imprecision, or other measurement uncertainties. We denote this imprecise value of agents $v$'s state by $r_v$, and associate a *region of imprecision*, $B_v \subset \mathbb{R}^d$ such that the true value of agent $v$'s state, i.e., $x_v$, can be anywhere inside this $B_v$. An example of this imprecision model is a hypersphere with a center $r_v$ and radius $\delta$. The true state value $x_v$ can be anywhere inside this hypersphere of radius $\delta$ (indicating the extent of imprecision). Another example of imprecision model is a hypercube, where imprecision in each coordinate is at most $\delta$, as shown in Figure 1.

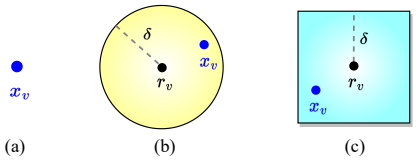



Fig. 1: (a) An exact point $x_v$ (no imprecision). (b) A disk imprecision region in $\mathbb{R}^2$. The observed point is $r_v$, which is the imprecise version of the exact point $x_v$ that lies somewhere inside the disk. (c) A square imprecision region in $\mathbb{R}^2$.

*Resilient Consensus Problem* – In a network with both normal and adversarial agents, the aim of resilient vector consensus is to design a mechanism that ensures all normal agents update their states to eventually converge on a common state. This common state must lie within the convex hull of their initial states, denoted as $X(0) = \{x_1(0), x_2(0), \cdots, x_n(0)\}$. The mechanism must satisfy the following conditions:

- *Safety:* At any time step $t$, the state of any normal node $v$ must lie within $\texttt{Conv}(X(0))$.
- *Agreement:* For every $\epsilon > 0$, there exists a time $t_\epsilon$, such that the states of any two normal agents $u$ and $v$ come within $\epsilon$ of each other for all $t > t_\epsilon$.

## III. Effect of Imprecision

In this section, first, we review a resilient distributed consensus algorithm that guarantees convergence of normal nodes despite adversarial agents without considering any imprecision in agents' states. Then, we consider imprecision and show that the existing resilient algorithms fail to achieve consensus and perform poorly under states' imprecision.

### A. Resilient Consensus with No Imprecision

Several resilient consensus schemes have been proposed in the literature for the scalar and higher dimensional cases, e.g., [1], [4], [5], [7], [8], [10], [11]. The main approach, especially in higher dimensions $d \geq 2$, relies on the principle that in each round of the consensus algorithm, each normal node computes a point that lies in the convex hull of its normal neighbors' state, and then updates its state by moving towards that point, often referred to as the *'safe point'*. The computation of this safe point is challenging, and various methods are employed. A useful way that results in superior resilience (in terms of the number of adversarial agents that can be allowed) is to obtain a safe point by computing a centerpoint of the agents' states. A centerpoint is a generalization of median in higher dimensions, and is defined below.

**Definition 3.1.** *(Centerpoint) Given a set $S$ of $N$ points in $\mathbb{R}^d$, a centerpoint $p$ is a point with the property that every closed halfspace of $\mathbb{R}^d$ containing $p$ must also contain at least $\frac{N}{d+1}$ points of $S$.*

The set of all centerpoints is referred to as the *centerpoint region*. Figure 2 illustrates an example. There are six points in $\mathbb{R}^2$, and any line passing through a centerpoint divides these six points into two regions, each containing at least two points, as in Figures 2(a) and (b). Figure 2(c) illustrates the centerpoint region for the given example. The notion of centerpoint provides a complete characterization of the safe point. It is shown in [4] that a safe point for an agent $v$ is essentially a centerpoint of its neighbors' states as long as the number of adversaries in the neighborhood of $v$ is bounded by $\frac{N_v}{d+1}$. Here, $d$ is the dimension of the state and $N_i$ is the number of neighbors of $v$. Moreover, a safe point may not exist if the number of adversaries is more than $\frac{N_v}{d+1}$. Thus, the centerpoint-based safe point computation is particularly useful. Now, a *resilient consensus algorithm*

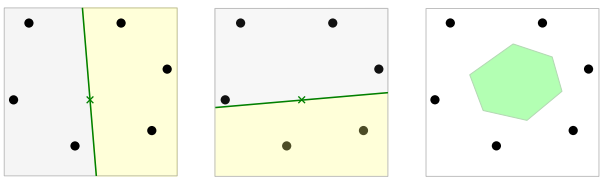



Fig. 2: Illustration of centerpoint. In (a) and (b), centerpoint is denoted by '$\times$' and lines are passing through the centerpoint. The green shaded region in (c) is the centerpoint region.

based on centerpoints and considering *no imprecision*—agents observe their normal neighbors' states exactly—can be designed as follows:

- In each iteration $t$, a normal agent $u$ gathers the state values of its neighbors in $N_u$, and computes a safe point $s_u(t)$ by computing a centerpoint agents' states.
- Agent $u$ updates its states as follows:

$$x_u(t+1) = \alpha_u(t)s_u(t) + (1-\alpha_u(t))x_u(t), \quad (1)$$

where $\alpha_u(t) \in (0\ \ 1)$ is a dynamically chosen parameter whose value depends on the application [3].

Analysis in [4] and [3] shows that the above scheme achieves resilient consensus as long as the number of adversaries in each normal agent $u$ is at most $\frac{N_u}{d+1} - 1$. Next, we see how the above resilient consensus (and other safe point based) algorithms perform if we consider imprecise agents' states.

### B. Resilient Solutions Do Not Work Under Imprecision

The fundamental premise of resilient consensus algorithms lies in computing safe points that are always inside the convex hull of *normal agents' true states*. In the case of imprecision, however, the true states are unknown. Thus, computing a safe point is challenging, and the existing approaches prove inadequate. Note that, in general, true consensus is not possible when there is fixed imprecision in state values, however resilient consensus algorithms can fail to achieve *approximate consensus* as defined in Section I. For example, a centerpoint computed by a normal agent based on its neighbors' *observed states* (due to imprecision) does not always lie in the convex hull of its normal neighbors' true states and, therefore, is not a safe point. Figure 3 illustrates this situation.

In Figure 3(a), we have six agents that are in the neighborhood of a normal agent $v$ (including $v$). One of the agents, $u$, is an adversary. Note that $v$ remains oblivious to the identity of the adversarial agent within its neighborhood. Each agent has a state in $\mathbb{R}^2$ accompanied by a region of imprecision, which we assume to be a square. The centerpoint region based on the observed states (represented by '•') is highlighted in green, whereas the convex hull of normal agents' true states (indicated by '×') is depicted as grey. Figure 3(b) reveals the challenge imprecision poses. The centerpoint region fails to remain entirely contained within the convex hull of the normal agents' true states. Consequently, agent $v$ may select a centerpoint (indicated by '○') that does not qualify as a safe point. As a result, $v$ may update its state by moving in a direction outside the convex hull of normal agents' true states, thus violating the safety condition of the resilient consensus.

Figure 4 demonstrates examples of the run of the resilient consensus algorithm with and without the imprecision model. All six agents are pairwise adjacent, and there is one adversary. With no imprecision, all normal agents converge inside the convex hull of their initial states despite a single adversary, as shown by the state trajectories of agents in Figure 4(a). However, with imprecision (square regions), the resilient consensus algorithm fails as normal agents do not remain inside their initial states' convex hull and continue moving farther away, as shown in Figure 4(b).

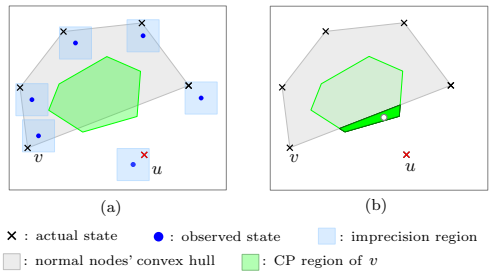



Fig. 3: Centerpoint region based on the observed states (due to imprecision) is not contained entirely in the convex hull of normal agents' initial states.

Imprecision presents a significant obstacle to successfully operating resilient distributed algorithms. The primary hurdle arises from the *difficulty in computing a safe point when dealing with imprecise observed states*. Thus, it is imperative to explore new approaches to ensure the accurate computation of safe points, even in the presence of imprecision. We address this problem in the next section.

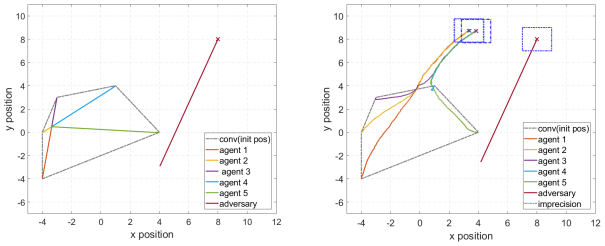


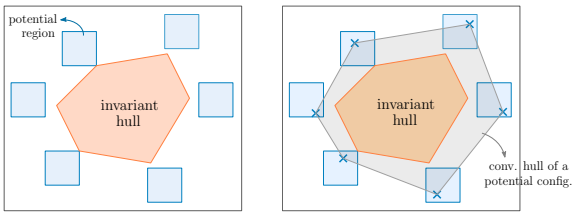

Fig. 4: (a) Normal agents achieve resilient consensus with no imprecision. (b) Normal agents do not converge and move outside the convex hull of normal agents' initial states due to imprecision.

## IV. Resilience in the Presence of Imprecision

In this section, we address the challenge of dealing with imprecision in resilient consensus algorithms and present our approach to mitigate its impact. Our main objective is to enable normal agents to *compute safe points even when they encounter imprecise state values from their neighbors*.

When a normal agent $v$ observes an imprecise state of its neighbor $u$, the agent $v$ essentially observes an imprecision region associated with $u$. This imprecision region contains the true state value of $u$, and is termed as a *potential region* (as any value in this region can potentially be a true value of the agent). Consequently, agent $u$ effectively observes a collection of potential regions linked to its neighbors. By selecting one value (a point) from each potential region, we construct a *potential configuration* for the given set of potential regions. It is worth noting that the true values of the agents represent just one potential configuration among an infinite set. Our focus lies in identifying the largest region contained within the convex hull of *any* potential configuration from a given set of potential regions. We refer to this set as an *invariant hull*. Importantly, an invariant hull, associated with a given set of potential regions is a subset of the convex hull of the true states of agents. Figure 5 illustrates these concepts. Figure 5(a) shows a set of potential regions (blue boxes) and the associated invariant hull. Figure 5(b)

shows one potential configuration (set of points in potential regions indicated by '$\times$') and the associated convex hull of the potential configuration (grey shaded region). Note that the invariant hull is contained in the convex hull of the potential configuration shown. Similarly, if we choose any other potential configuration, the invariant hull will be contained in the convex hull of such a configuration.




(a) (b)

Fig. 5: (a) Invariant hull of a set of potential regions. (b) Invariant hull is a subset of the convex hull of an arbitrary potential configuration.

Next, we introduce some notations.

| | |
|---|---|
| $x_v$ | Actual/true state of agent $v$; |
| $r_v$ | Observed state of agent $v$; |
| $B_v$ | potential region of agent $v$; |
| $B_V$ | Set of potential regions of agents in the set $V = \{v_1, \cdots, v_n\}$. |

**Definition 4.1.** *(Potential Configuration) For a given set of potential regions $B_V = \{B_1, \cdots, B_n\}$, a potential configuration is a set of points $P(B_V) = \{p_1, \cdots, p_n\}$ such that $p_v \in B_v$ for each $v$.*

Now, we define the notion of *invariant hull*.

**Definition 4.2.** *(Invariant Hull) Consider a set of potential regions $B_V$. Let $\mathcal{P}(B_V)$ be the set of all possible potential configurations, and $\texttt{Conv}(P)$ denotes the convex hull of a potential configuration $P \in \mathcal{P}(B_V)$. Then, the invariant hull of $B_V$ is defined as,*

$$\texttt{IHull}(B_V) = \bigcap_{P \in \mathcal{P}(B_V)} \texttt{Conv}(P). \tag{2}$$

In simpler terms, for the invariant hull, we find all possible potential configurations of $B_V$, compute the convex hull of each such configuration, and finally compute the intersection of all such convex hulls. Consequently, the invariant hull is essentially a subset of the convex hull of any potential configuration of $B_V$, as Figure 5 illustrates. Also, the invariant hull is a convex set.

Next, we provide a geometric characterization of the invariant hull. Then, in Section IV-B, we present an efficient method to compute the invariant hull of $B_V$.

### A. Characterization of Invariant Hull

First, we need to introduce notations to denote *combinatorial families* of sets.

- Consider a set of potential regions $B_V$, *then* $B_V^{d+1}$ *denotes a family of all* $(d+1)$*-member subsets of* $B_V$ (i.e., subsets of $B_V$ consisting of $d+1$ potential regions). For example, consider $V = \{v_1, v_2, v_3, v_4\}$, the corresponding $B_V = \{B_1, B_2, B_3, B_4\}$, and $d = 2$, then
$$B_V^3 = \{\ \{B_1, B_2, B_3\},\ \{B_1, B_2, B_4\}\ \{B_1, B_3, B_4\},\ \{B_2, B_3, B_4\}\ \}.$$
- Similarly, for $Q \in B_V^{d+1}$, the notation $Q^d$ denotes the family of all $d$-member subsets of $Q$. For example, in the above example, if $Q = \{B_1, B_2, B_3\} \in B_V^3$, then
$$Q^2 = \{\{B_1, B_2\},\ \{B_1, B_3\},\ \{B_2, B_3\}\ \}.$$

So, in general, a *superscript* on a set notation denotes the combinatorial family of the set.

The following lemma identifies the essential property for a point to be in the invariant hull of a set of potential regions.

**Lemma 4.1.** *Let $Q \in B_V^{d+1} = \{q_1, .., q_{d+1}\}$ be a $(d+1)$-member subset of potential regions. If $\texttt{IHull}(Q)$ is non-empty, then a point $p \in \texttt{IHull}(Q)$ if and only if $p$ has the following property:*

***Property 1.*** *For every hyperplane $h$ passing through $p$, at least one potential region $q_i \subset Q$ is contained in each of the associated halfspaces of $h$.*

For proof of Lemma 4.1, see [16].

**Corollary 4.2.** *For $(d+1)$-region set $Q$, it follows from Lemma 4.1 that $\texttt{IHull}(Q) = \texttt{Conv}(Q) \setminus \texttt{Conv}(\bigcup_{q_i \in Q^d} q_i)$. This can be seen by observing that no point with Property 1 may exist in the convex hull of a $d$-member subset of $Q$. Any hyperplane through $p$ and each of the $d$ members may only contain the remaining region of $Q$ in one or the other of its halfspaces, leaving one halfspace with no potential region as a subset.*

We now present an auxiliary lemma that will be used in proving Theorem 4.4 characterizing the invariant hulls.

**Lemma 4.3.** *If $\texttt{IHull}(B_V)$ is a convex polytope and point $p$ is a vertex of $\texttt{IHull}(B_V)$, then for all $d$-region subsets $f \in B_V^d$, $p \notin int(\texttt{Conv}(f))$, where $int(\texttt{Conv}(f))$ refers to the interior of $\texttt{Conv}(f)$.*

The proof of Lemma 4.3 is available in [16]. We now present the main result, which states that the invariant convex hull of $n$ potential regions is the convex hull of the union of invariant hulls of each of the $\binom{n}{d+1}$ subsets of $B_V^{d+1}$.

**Theorem 4.4.** *Let $B_V = \{B_1, \cdots, B_n\}$ be a collection of $n$ potential regions of $\mathbb{R}^d$ and $B_V^{d+1}$ denote the family of $(d+1)$-member subsets of $B_V$. Then,*

$$\texttt{IHull}(B_V) = \texttt{Conv}(\bigcup_{Q \in B_V^{d+1}} \texttt{IHull}(Q)). \tag{3}$$

For a proof of Theorem 4.4, see [16].

### B. Computing the Invariant Hull

We now outline a procedure for computing the invariant hull of $B_V$, given that $|B_V| > d+1$. The procedure involves computing the equation of a tangent hyperplane equation to $d$ potential regions of $\mathbb{R}^d$. In the case that potential regions are

polygons in $\mathbb{R}^2$, linear-time algorithms have been developed to find *outer* tangent lines, that is the tangent lines that have all participating polygons on one side [17].

Let $Q = \{q_1, q_2, \cdots, q_{d+1}\} \in B_V^{d+1}$. As stated in Corollary 4.2, $\texttt{IHull}(Q)$is equivalent to $\texttt{Conv}(Q) \setminus \texttt{Conv}(Q^d)$. A simple approach is to compute the vertices of $\texttt{IHull}(Q)$:

1) Select $q_i \in Q$.
2) For each $j = 1, 2, \cdots, d+1$, and each $f_j \in Q^d$ containing $q_i$, compute hyperplane $h_j$ tangent to the $d$ members of $f_j$ such that $f_j \subset h^{outer}$ and $Q \setminus f \subset h^{inner}$.
3) Compute point of intersection:$\{x \in \mathbb{R}^d : h_1 \cdot x = h_2 \cdot x = \cdots = h_{d+1} \cdot x\}$. Label $x$ as $w_i$.
4) Repeat steps 1-3 for all remaining $q_i \in Q$.
5) $\texttt{IHull}(Q) = \texttt{Conv}(\bigcup_{\forall q_i \in Q} w_i)$.

After repeating steps 1-5 for all $Q \in B_V^{d+1}$, the invariant hull of $B_V$ is computed with any convex hull routine as:

$$\texttt{IHull}(B_V) = \texttt{Conv}(\bigcup_{Q \in B_V^{d+1}} \texttt{IHull}(Q)).$$

## V. Resilient Consensus through Centerpoint using Invariant Hulls of Potential Regions

In this section, we present a way for a normal agent $v$ to compute a safe point despite access to only imprecise states of neighbors. Moreover, the proposed approach allows $F$ number of adversaries in the neighborhood of $v$, where $F \leq \frac{N_v}{d+1} - 1$. The core of our method relies on the intersection of the invariant hulls of potential regions.

First, consider the case with no imprecision, i.e., normal agents observe true states of their neighbors (as discussed in [4], [18]). In this ideal situation, a safe point for a normal agent $v$ is effectively the centerpoint of its neighbors' states, provided the number of adversarial agents in the neighborhood of $v$ is $F \leq \frac{N_v}{d+1} - 1$. This proposition is rooted in geometric principles. Specifically, a centerpoint of a given set of $N_v$ points in $\mathbb{R}^d$ lies within the convex hull of any subset comprising more than $\frac{d}{d+1}N_v$ of those points [19]–[21]. To put it practically, if we select an arbitrary subset of neighbors of $v$—comprising more than $\frac{d}{d+1}N_v$ neighbors—and compute the convex hull of their states, a centerpoint is guaranteed to lie within this hull. Since one such choice of more than $\frac{d}{d+1}N_v$ neighbors of a normal agent $v$ consists of only normal neighbors of $v$, a centerpoint of the states of agent $v$'s neighbors also lies within the convex hull of *only* the normal neighbors' states.

In cases where the observed states are imprecise, the approach to determining a safe point for a normal agent $v$ must be modified. As elaborated in Section III-B, the inherent imprecision precludes the use of convex hulls calculated from observed states. Instead of knowing a true state of the neighbor, $v$ only knows the potential region of the neighbor (containing the true state). So, instead of computing convex hulls of subsets of neighbors' observed states, a normal agent computes the invariant hulls of subsets of potential regions of neighbors. Assume that a normal agent $v$ has $N_v$ neighbors, of which at most $\frac{N_v}{d+1} - 1$ are adversarial. For any subset of neighbors containing more than $\frac{d}{d+1}N_v$ neighbors, agent $v$ computes the invariant hull of the potential regions of the corresponding neighbors, and then computes the intersection of all such invariant hulls. This resulting intersection serves as a modified 'centerpoint region', constructed from the invariant hulls of potential regions. Importantly, every point in this new centerpoint region is guaranteed to lie within the convex hull of the true states of agent $v$'s normal neighbors, and is thus a safe point (despite imprecise observed states). We illustrate this through an example below.

*Example:* Consider a set of six potential regions in a plane, each corresponding to an agent. Note that the true states of these agents are 'hidden' within these potential regions. Among these six, one potential region belongs to an adversarial agent, though we don't know which one. The green-shaded region in Figure 6(a) is the centerpoint region constructed using the intersection of invariant hulls of subsets of potential regions. To see this, consider a subset of five potential regions—calculated as $\frac{d}{d+1}N+1 = 5$—and determine their invariant hull. As demonstrated in Figures 6(b)–6(g), the green centerpoint region is consistently enclosed within the invariant hull, regardless of which five potential regions are chosen. Notably, one of these choices in Figures 6(b)–6(g) includes the actual adversary. In this case, the computed invariant hull—by definition—remains a subset of the convex hull formed by the true states of normal agents. Therefore, the points in the green-shaded centerpoint region, which is contained entirely in the invariant hull, are safe points.

Next, we outline the steps of the state update for each normal agent. An essential step is the computation of a safe point using the idea of invariant hulls, as described previously. The steps of the algorithm, which we call *centerpoint of invariant hulls (CPIH)* method, are as follows:

For each node $v \in V_n$.

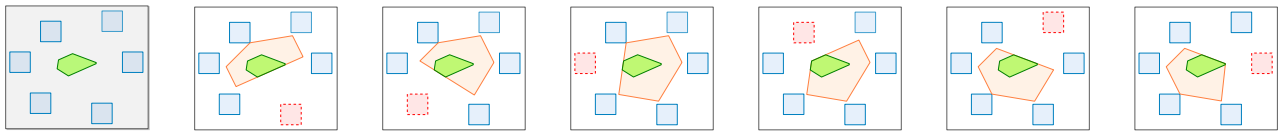
1) At time $t$ compute $k = \frac{dN_v}{d+1} + 1$.
2) For $C \in B_V^k$, compute $\texttt{IHull}(C)$. ( Compute invariant hull of each $k$-tuple of $B_V$).
3) If $\bigcap_{C \in B_V^k} \texttt{IHull}(C) = \emptyset$, set $x_v(t+1) = x_v(t)$. Otherwise, proceed to step 4. (Do nothing at time $t$ if safe region does not exist).
4) Select safe point $p_v(t) \in \bigcap_{C \in B_V^k} \texttt{IHull}(C)$.
5) $x_v(t+1) = \alpha_v(t)p_v(t) + (1 - \alpha_v(t))x_v(t)$.

The value of $\alpha_v(t)$ is a dynamic weight that may be chosen in range $[0, 1]$ according to the application.

The CPIH algorithm identifies a region that is a generalization of the centerpoint region in that the property of a CPIH safe point has identical properties to that of of a centerpoint with compact sets replacing the role of points. Thus each halfspace of a hyperplane intersecting a CPIH safe point contains at least $\lfloor \frac{N}{d+1} \rfloor$ potential regions. As such, every $\frac{dN}{d+1} + 1$ potential regions contain a CPIH safe point.

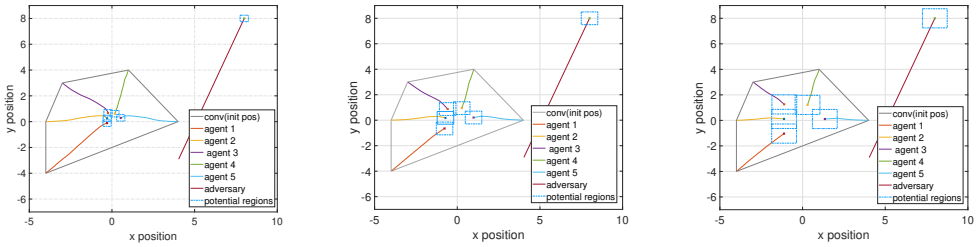
### A. Simulations and Discussion

In order to verify that the CPIH Algorithm succeeds where the ordinary centerpoint-based consensus algorithm fails (as illustrated in Figure 4), we executed a series of simulations



(a) (b) (c) (d) (e) (f) (g)

Fig. 6: (a) Centerpoint region based on the intersection of the invariant hulls of subsets of potential regions. In each of (b)–(g), the green shaded region is contained entirely in the invariant hull of the five potential regions.




(a) $\delta = 0.5$ (b) $\delta = 1$ (c) $\delta = 1.5$

Fig. 7: Implementation of CPIH algorithm. Normal agents remain inside the convex hull of their initial states and converge to a degree that depends on the level of state imprecision, as quantified by the parameter $\delta$.

using the above algorithm. For simplicity, potential regions were chosen to be a square of width $\delta$ centered around each node, and $G$ was chosen to be a complete graph. At each time step, agents received a uniformly random state estimate within the potential regions of each of its neighbors. For illustrative purposes, we chose a small range of values for $\delta$. For each value of $\delta$, a simulation was run for $5000$ time steps. Figure 7 shows the results. As evidenced in the simulations, the CPIH algorithm ensures that the final states of normal nodes are safe points. The mean disagreement of the safe points varies according to the magnitude of $\delta$. As a result, CPIH algorithm presents a tradeoff between network safety and convergence precision.

## VI. Conclusion

In this paper, we have demonstrated that traditional resilient consensus algorithms can fail when they do not account for state imprecision. To address this, we proposed an uncertainty-aware geometric solution ensuring resilience against adversaries and state imprecision. Specifically, we extended the centerpoint-based resilient consensus algorithm by introducing the concept of an 'invariant hull', which we call the CPIH algorithm. Through illustrative simulations, we have showcased the effectiveness of our algorithm under varying levels of uncertainty. In the future, we aim to explore the resilience-accuracy trade-off further by determining the conditions under which the invariant hull of imprecise regions becomes empty.

## References


[1] H. Ishii, Y. Wang, and S. Feng, "An overview on multi-agent consensus under adversarial attacks," *Annual Reviews in Control*, vol. 53, pp. 252–272, 2022.

[2] M. Pirani, A. Mitra, and S. Sundaram, "Graph-theoretic approaches for analyzing the resilience of distributed control systems: A tutorial and survey," *Automatica*, vol. 157, p. 111264, 2023.

[3] H. Park and S. A. Hutchinson, "Fault-tolerant rendezvous of multirobot systems," *IEEE Transactions on Robotics*, vol. 33, pp. 565–582, 2017.

[4] W. Abbas, M. Shabbir, J. Li, and X. Koutsoukos, "Resilient distributed vector consensus using centerpoint," *Automatica*, vol. 136, 2022.

[5] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 4, pp. 766–781, 2013.

[6] L. Su and N. Vaidya, "Multi-agent optimization in the presence of byzantine adversaries: Fundamental limits," in *American Control Conference (ACC)*, 2016, pp. 7183–7188.

[7] H. Mendes, M. Herlihy, N. Vaidya, and V. K. Garg, "Multidimensional agreement in byzantine systems," *Distributed Computing*, vol. 28, no. 6, pp. 423–441, 2015.

[8] J. Yan, X. Li, Y. Mo, and C. Wen, "Resilient multi-dimensional consensus in adversarial environment," *Automatica*, vol. 145, 2022.

[9] Y. Wang, H. Ishii, F. Bonnet, and X. Défago, "Resilient consensus for multi-agent systems under adversarial spreading processes," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 5, pp. 3316–3331, 2022.

[10] J. Li, W. Abbas, and X. Koutsoukos, "Resilient distributed diffusion in networks with adversaries," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 1–17, 2019.

[11] J. Zhu, Y. Lin, A. Velasquez, and J. Liu, "Resilient distributed optimization," in *American Control Conference (ACC)*, 2023, pp. 1307–1312.

[12] K. Kuwarananchoroen, L. Xin, and S. Sundaram, "Byzantine-resilient distributed optimization of multi-dimensional functions," in *American Control Conference (ACC)*, 2020, pp. 4399–4404.

[13] M. Löffler, *Data imprecision in computational geometry*. Utrecht Univesity, 2009.

[14] J. Park, R. Ivanov, J. Weimer, M. Pajic, S. H. Son, and I. Lee, "Security of cyber-physical systems in the presence of transient sensor faults," *ACM Transactions on Cyber-Physical Systems*, vol. 1, pp. 1–23, 2017.

[15] T. Nagai, S. Yasutome, and N. Tokura, "Convex hull problem with imprecise input," in *Discrete and Computational Geometry*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 207–219.

[16] C. A. Lee and W. Abbas, "A geometric approach to resilient distributed consensus accounting for state imprecision and adversarial agents," *arXiv:2403.09009*, 2024.

[17] M. Abrahamsen and B. Walczak, "Common tangents of two disjoint polygons in linear time and constant workspace," *ACM Trans. Algorithms*, vol. 15, no. 1, dec 2018.

[18] J. Li, W. Abbas, M. Shabbir, and X. Koutsoukos, "Byzantine resilient distributed learning in multirobot systems," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3550–3563, 2022.

[19] J. Pach and P. K. Agarwal, *Combinatorial Geometry*. John Wiley & Sons, 2011.

[20] J. Matousek, *Lectures on Discrete Geometry*. Springer Science & Business Media, 2013, vol. 212.

[21] S. Ray and N. Mustafa, "An optimal generalization of the centerpoint theorem, and its extensions," in *Annual Symposium on Computational Geometry*, 2007, pp. 138–141.