Resilient Multidimensional Sensor Fusion Using Measurement History

Radoslav Ivanov
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA 19104
rivanov@seas.upenn.edu

Miroslav Pajic
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104
pajic@seas.upenn.edu

Insup Lee
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA 19104
lee@cis.upenn.edu

ABSTRACT

This work considers the problem of performing resilient sensor fusion using past sensor measurements. In particular, we consider a system with n sensors measuring the same physical variable where some sensors might be attacked or faulty. We consider a setup in which each sensor provides the controller with a set of possible values for the true value. Here, more precise sensors provide smaller sets. Since a lot of modern sensors provide multidimensional measurements (e.g., position in three dimensions), the sets considered in this work are multidimensional polyhedra.

Given the assumption that some sensors can be attacked or faulty, the paper provides a sensor fusion algorithm that obtains a fusion polyhedron which is guaranteed to contain the true value and is minimal in size. A bound on the volume of the fusion polyhedron is also proved based on the number of faulty or attacked sensors. In addition, we incorporate system dynamics in order to utilize past measurements and further reduce the size of the fusion polyhedron. We describe several ways of mapping previous measurements to current time and compare them, under different assumptions, using the volume of the fusion polyhedron. Finally, we illustrate the implementation of the best of these methods and show its effectiveness using a case study with sensor values from a real robot.

Categories and Subject Descriptors

K.6.5 [**Security and Protection**]: Unauthorized access (e.g., hacking, phreaking); C.3 [**Special-purpose and Application-based Systems**]: Process control systems, Real-time and embedded systems

Keywords

CPS security; sensor fusion; fault-tolerance; fault-tolerant algorithms

1. INTRODUCTION

With the cost of sensors constantly falling in recent years, modern Cyber Physical Systems (CPS) can now be equipped with multiple sensors measuring the same physical variable (e.g., speed and position in automotive CPS). Their measurements can be fused to obtain an estimate that is more accurate than any individual sensor's, thus improving the system's performance and reliability. Furthermore, having diverse sensors increases the system's robustness to environmental disturbances (e.g., a tunnel or a mountainous region for automotive CPS that heavily rely on GPS navigation) and sensor limitations (e.g., low sampling rate, biased measurement noise).

The increase in sensor diversity naturally leads to different characteristics of the various sensors. In particular, some sensors may have drift (e.g., IMU), others may not be very accurate (e.g., smart phone applications [12]), yet others may be accurate but not always reliable (e.g., GPS). Hence, the goal of any effective sensor fusion algorithm is to account for these limitations and output a measurement that is robust and accurate.

The importance of reliable sensor fusion is further highlighted with the increase of autonomy of modern control systems. In automotive CPS, for instance, a malicious attacker may be able to corrupt the measurements of some sensors, thereby misleading the controller into performing an unsafe action. The consequences of such attacks may range from slight disturbances in performance to life-threatening situations [3, 8]. Resilience to sensor attacks, therefore, is an emerging additional requirement for modern sensor fusion algorithms.

There is significant amount of academic literature devoted to the problem of fault-tolerant sensor fusion. Problems investigated depend first and foremost on the sensor model in consideration. The most popular model is a probabilistic one: a sensor provides the controller with a numeric measurement that is corrupted by noise with a known distribution (e.g., uniform, Gaussian) [5, 13]. In an alternative approach, a set is constructed around the sensor's measurement containing all points that may be the true value [10, 11]. The pivotal work with this viewpoint considers one-dimensional intervals around measurements and assumes an upper bound on the number of sensors whose intervals do not contain the true value; the author then provides worst-case bounds on the size of the fusion interval [9]. An extension of this work considers intervals in d dimensions, i.e., d-rectangles, and obtains similar results [4]. Furthermore, researchers assume a distribution of the true value over the intervals so probabilistic analysis can again be performed [14]. Finally, this model can be used not only to aid control but also for fault detection [7, 9].

This paper considers the problem of attack-resilient and fault-tolerant sensor fusion with multidimensional *abstract sensors*, i.e., a set is constructed around each sensor's measurement that contains all points that may be the true value.

The size of the set depends on the sensor's precision – a larger set means less confidence in the sensor's measurement. Since most modern sensors employ internal filtering techniques (e.g., Kalman filters in GPS) these sets are not always as simple as d-rectangles; hence, we focus on d-dimensional polyhedra. Some camera-based velocity and position estimators used in urban robotics applications, for example, guarantee different position precisions for different robot velocities. Note that this paper extends our previous work in which we considered attack-resilient sensor fusion with one-dimensional intervals and investigated the effects of communication schedules on the size of the fusion interval (without incorporating past measurements) [6].

The sensor model considered in this work is very general as it does not make any assumptions about the distribution of sensor noise; instead, the polyhedron is constructed based on manufacturer specifications (e.g., worst-case guarantees about sampling rate and implementation limitations) and system dynamics. To deal with malicious and faulty sensors in these scenarios, we propose an extension to the sensor fusion algorithm for d-rectangles described by Chew and Marzullo [4]. Given this algorithm, we provide worst-case bounds on the size of the fusion polyhedron based on the number of assumed faulty or attacked (see Section 2 for a definition) polyhedra. Note that this approach could be extended to a set membership technique (e.g., [10, 11]) where some of the sensors or state estimators may be corrupted.

In addition, we note that most CPS have known dynamics. Therefore, this knowledge can be utilized to improve sensor fusion (i.e., reduce the size of the obtained region) by incorporating past measurements in the sensor fusion algorithm. To achieve this, we consider discrete-time linear systems with bounded measurement noise. Measurements are collected from sensors at each time step and are used for the remainder of the system's operation to reduce the size of the fused polyhedron. We consider all possible algorithms of using historical measurements (given our weak assumptions) and compare them by means of the volume of the fusion polyhedron. Finally, we provide a case-study with an autonomous vehicle, called the LandShark [1], to illustrate the effectiveness of the best of these methods. In particular, we consider four speed sensors, two of them also measuring position, that provide the controller with two-dimensional polyhedra. We then show the reduction in the volume of the fusion polyhedron when historical measurements are considered.

This paper is organized as follows. Section 2 introduces the problems considered in the paper, namely reliable multidimensional sensor fusion incorporating historical measurements. Section 3 provides a sensor fusion algorithm that meets the requirements outlined in Section 2. Section 4 extends the algorithm in Section 3 by incorporating system dynamics and past measurements. Section 5 presents an implementation of this algorithm and a case study to illustrate its effectiveness. Finally, Section 6 concludes the work.

2. PROBLEM FORMULATION AND PRELIMINARIES

This section describes the two problems considered in this paper. We start by formulating the multidimensional sensor fusion problem in a single time step (i.e., without taking history into account). Given this algorithm, we outline the problem of using system dynamics and past measurements to improve the system's sensor fusion.

2.1 Fusion Algorithm

We consider a system with n sensors measuring the same physical variables. Each sensor provides the controller with a d-dimensional measurement (e.g., position in three dimensions, or estimates of both position and velocity); a polyhedron is constructed around this measurement based on the sensor's precision and implementation guarantees (e.g., sampling jitter). Additionally, the sensor may have an internal filtering algorithm (e.g., Kalman filter) that will further affect the shape of the polyhedron. Thus, the controller will obtain n d-dimensional polyhedra $P_1, \dots, P_n$ of the form $P_i = \{x \mid B_i x \leq b_i\}$, where $x \in \mathbb{R}^d$, $B_i \in \mathbb{R}^{m \times d}$ and $b_i \in \mathbb{R}^m$.

Definition 1. *A sensor is said to be* correct *if its polyhedron contains the true value and* corrupted *(due to faults or attacks) otherwise.*

We assume an upper bound of f corrupted sensors; since the actual number of corrupted sensors is not known, f will usually be set conservatively high, e.g., $f = \lceil n/2 \rceil - 1$.

With these assumptions, the problem is to obtain an algorithm that, given n polyhedra as input, will output a polyhedron that is guaranteed to contain the true value and will be as small in volume as possible.

2.2 Fusing Past and Current Measurements

We note that most autonomous systems have known dynamics, hence previous measurements (i.e., measurement history) can be used to aid the fusion algorithm. In this paper we assume that sensors monitor a discrete-time linear system of the form:

$$x(t+1) = Ax(t) + w.$$

Here $x \in \mathbb{R}^d$ is the state of the system (e.g., position), $A \in \mathbb{R}^{d \times d}$, and $w \in \mathbb{R}^d$ is bounded noise such that $\|w\| \leq M$, where $\|\cdot\|$ denotes the L_∞ norm, and M is a constant.

Definition 2. *In this setting, a sensor is corrupted if there exists a time step t such that its polyhedron does not contain the true value at t.*

We still use f to denote the upper bound on the number of corrupted sensors. In particular, this means that there are at least $n - f$ sensors that are correct in all time steps. We relax this assumption in Section 5 when discussing which of the proposed methods of using history should be applied in real systems.

Given a sensor fusion algorithm that satisfies the requirements outlined in the previous section, in this scenario the problem is to find an algorithm to use past measurements that satisfies the following criteria:

- the final fusion polyhedron is guaranteed to contain the true value,
- the fusion polyhedron is never larger in volume than the fusion polyhedron obtained when no history is used,
- the fusion polyhedron is as small as possible.

Algorithm 1 Sensor Fusion Algorithm

```
Input: An array of polyhedra $P$ of size $n$ and an upper
    bound on the number of corrupted polyhedra $f$
1: $C \leftarrow combinations\_n\_choose\_n\_minus\_f(P)$
2: $R_{\mathcal{N},f} \leftarrow \emptyset$
3: for each $K$ in $C$ do
4:     add($R_{\mathcal{N},f}$, intersection($K$))
5: end for
6: return conv($R_{\mathcal{N},f}$)
```

2.3 Notation

Let $\mathcal{N}(t)$ denote all n polyhedra at time t. In Section 3 we drop the time notation and write $\mathcal{N}$ since only one time step is considered. We use $S_{\mathcal{N}(t),f}$ to denote the fusion polyhedron given the set $\mathcal{N}(t)$ and a fixed f. Let $|P|$ denote the volume of polyhedron P; in particular, $|S_{\mathcal{N}(t),f}|$ is the volume of the fusion polyhedron. We use $\mathcal{C}(t)$ to denote the (unknown) set of all correct polyhedra.

3. SENSOR FUSION ALGORITHM

In this section, we describe a sensor fusion algorithm that meets the criteria outlined in Section 2 before providing a bound on the size of the fusion polyhedron based on the number of assumed corrupted sensors.

The algorithm is described in Algorithm 1. It is based on the algorithm for d-rectangles described by Chew and Marzullo [4]. It computes the fusion polyhedron by finding all regions contained in $n-f$ polyhedra, denoted by $R_{\mathcal{N},f}$, and then taking their convex hull in order to return a polyhedron, i.e.,

$$S_{\mathcal{N},f} = \texttt{conv}(R_{\mathcal{N},f}), \tag{1}$$

where $\texttt{conv}(\cdot)$ denotes the convex hull. Intuitively, the algorithm is conservative – since there are at least $n-f$ correct polyhedra, any point that is contained in $n-f$ polyhedra may be the true value, and thus it is included in the fusion polyhedron; the convex hull is computed since the output should be in the same format as the inputs (i.e., a polyhedron).

The algorithm is illustrated in Figure 1. The system consists of three sensors, hence three polyhedra are obtained, and is assumed to have at most one corrupted sensor. Therefore, all regions contained in at least two polyhedra are found, and their convex hull is the fusion polyhedron (shaded).

Proposition 1. *The fusion polyhedron computed by Algorithm 1 will always contain the true value.*

Proof. Since there are at least $n-f$ correct polyhedra, the true value is contained in at least $n-f$ polyhedra, and hence it will be included in the fusion polyhedron. $\square$

Proposition 2. *The fusion polyhedron computed by Algorithm 1 is the smallest convex set that is guaranteed to contain the true value.*

Proof. We first note that any set that is guaranteed to contain the true value must contain $R_{\mathcal{N},f}$ since any point that is excluded may be the true value. This proves the proposition since $\texttt{conv}(R_{\mathcal{N},f})$ is the smallest convex set that contains $R_{\mathcal{N},f}$. $\square$

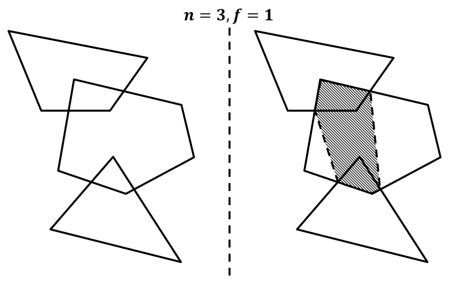


Figure 1: An illustration of the proposed sensor fusion algorithm.

Having shown the desired properties of the proposed algorithm, we comment on its complexity. There are two subprocedures with exponential complexity. First, finding all combinations of $n-f$ polyhedra is exponential in the number of polyhedra. Second, computing the convex hull of a set of polyhedra requires finding their vertices; this problem, known in the literature as vertex enumeration, is not known to have a polynomial algorithm in the number of hyperplanes defining the polyhedra (hence in their dimension) [2].

To prove a bound on the volume of the fusion polyhedron, for completeness we first prove the following lemma that will be useful in showing the final result.

Lemma 1. *The vertices of the convex hull of a set of polyhedra are a subset of the union of the vertices of the polyhedra.*

Proof. Let p be any vertex of the convex hull. Then $p = \sum \theta_i v_i$, where the v_i are the vertices of the polyhedra, $\sum \theta_i = 1$ and $\theta_i \geq 0$ (i.e., p is a convex combination of the v_i's). This means that p lies on a hyperplane defined by some of the v_i's, hence it cannot be a vertex, unless it is one of the v_i's. $\square$

Before formulating the theorem, we introduce the following notation. Let $\min_p \mathcal{B}$ denote the p^{th} smallest number in the set of real numbers $\mathcal{B}$ with size $r = |\mathcal{B}|$. Similarly, we use $\max_p \mathcal{B}$ to denote the p^{th} largest number in $\mathcal{B}$. We note that $\min_p \mathcal{B} = \max_{r-p+1} \mathcal{B}$ (e.g., if $\mathcal{B} = \{14, 15, 16\}, \min_1 \mathcal{B} = 14 = \max_3 \mathcal{B}$). Finally, let v be the number of vertices in the fusion polyhedron.

Theorem 1. *If $f < n/v$ then*

$$|S_{\mathcal{N},f}| \leq \min_{vf+1}\{|P| : P \in \mathcal{N}\}.$$

Proof. We use a counting argument. Let $\mathcal{V}$ be the set of vertices of $S_{\mathcal{N},f}$. By Lemma 1, each vertex in $\mathcal{V}$ is a vertex of one of the polyhedra formed by the intersection of $n-f$ of the sensor polyhedra (in step 4 of Algorithm 1). Therefore, it is contained in at least $n-f$ polyhedra. For each $p \in \mathcal{V}$, let P_p denote the number of polyhedra containing p. Consequently, $P_p \geq n-f$. Then

$$v(n-f) \leq \sum_{p \in \mathcal{V}} P_p.$$

The sum in the right-hand side can be split into two sums. One contains the number of polyhedra where each of the polyhedra contains all v vertices (we denote this number by a). Then the number of the remaining polyhedra is $n-a$.

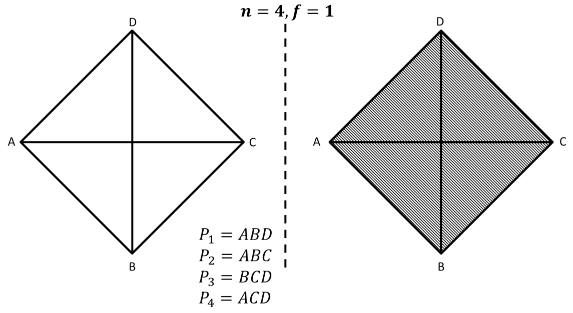


Figure 2: An example showing that the bound specified in Theorem 1 is tight.

The part of the sum due to the polyhedra that contain fewer than v vertices can be bounded from above by $(n-a)(v-1)$ since each of these polyhedra contains at most $v-1$ vertices. We then have

$$v(n-f) \leq av + (n-a)(v-1),$$

which implies that $a \geq n - vf$, i.e., at least $n - vf$ polyhedra contain the v vertices of the fusion polyhedron. Since polyhedra, including the fusion polyhedron, are convex, we conclude that at least $n - vf$ polyhedra contain the fusion polyhedron. This completes the proof, since

$$|S_{\mathcal{N},f}| \leq \max_{n-vf}\{|P| : P \in \mathcal{N}\} = \min_{vf+1}\{|P| : P \in \mathcal{N}\}.$$

$\square$

Theorem 1 suggests that if $f < n/v$ then the volume of the fusion polyhedron is bounded by the volume of some polyhedron. We note that this condition may not always hold as the number of vertices of the fusion polyhedron may be the sum of the number of vertices of the polyhedra. However, the condition is tight in the sense that if it does not hold, then the volume of the fusion polyhedron may be larger than the volume of any of the individual polyhedra. This is illustrated in Figure 2. In this case, each polyhedron (P_1, P_2, P_3 or P_4) is a triangle that is a half of the big square, so $n = 4$, and $f = 1 = n/v$. Hence the fusion polyhedron, i.e., square, is larger in area than any of the triangles. In cases like this one, we resort to the following bound.

THEOREM 2. *If $f < \lceil n/2 \rceil$, then $|S_{\mathcal{N},f}|$ is bounded by the volume of $\texttt{conv}(\mathcal{C})$ (i.e., the convex hull of all correct polyhedra).*

PROOF. Assume the opposite – that there exists a point $\mathbf{x}_A \in S_{\mathcal{N},f}$ that is not in $\texttt{conv}(\mathcal{C})$. Then for any convex combination $\sum \theta_i v_i = \mathbf{x}_A$, where $v_i \in P_j$ for some j, at least one v_i must not be in $\mathcal{C}$, meaning that it is contained in at most f polyhedra, where $f < n - f$. Therefore, there does not exist a convex combination $\sum \theta_i v_i = \mathbf{x}_A$ with all v_i contained in at least $n - f$ polyhedra, and hence $\mathbf{x}_A$ cannot be in $S_{\mathcal{N},f}$. $\square$

In conclusion, three different upper bounds on the volume of the fusion polyhedron exist based on different values of f. If $f > \lceil n/2 \rceil$, then the fusion polyhedron can be arbitrarily large. This is due to the fact that there are now enough corrupted sensors to include points not contained in any correct polyhedra in the fusion polyhedron (as opposed to Theorem 2). On the other hand, if $f \leq \lceil n/2 \rceil$, then $|S_{\mathcal{N},f}| \leq |\texttt{conv}(\mathcal{C})|$. In addition, if $f < n/v$, then the volume of $S_{\mathcal{N},f}$ is bounded from above by the volume of some polyhedron. Note that either of the last two bounds may be tighter than the other depending on the scenario.

4. FUSING PAST AND CURRENT MEASUREMENTS

Having developed a sensor fusion algorithm that produces a minimal polyhedron from n polyhedra in a given time step, we now consider the problem of incorporating knowledge of system dynamics to improve our resilient sensor fusion by using measurement history. As outlined in Section 2, we assume a discrete-time linear system with bounded noise of the form: $x(t+1) = Ax(t) + w$. Furthermore, as outlined in Definition 2, the definition of a corrupted sensor is now modified – a sensor s is corrupted if there exists a time t at which s provides a polyhedron that does not contain the true value.

To simplify notation, we use the mapping m defined as

$$m(P(t)) = AP(t) + w,$$

where $P(t)$ is any polyhedron in time t. Then let $m(\mathcal{N}(t)) \cap_p \mathcal{N}(t+1)$ denote the intersection of each sensor s_i's measurement in time $t+1$ with the mapping of s_i's measurement from time t. Note that this object again contains n polyhedra, some of which may be empty. We will refer to $\cap_p$ as *pairwise intersection*.

It is worth noting here that our assumptions impose a restriction on the number of ways in which history can be used. In particular, we only assume an upper bound on the number of corrupted sensors; thus, it is not possible to map subsets of the polyhedra while guaranteeing that the fusion polyhedron contains the true value. In other words, such mappings would require additional assumptions on the number of corrupted sensors in certain subsets of $\mathcal{N}(t)$; hence, all permitted actions in this work are: a) computing fusion polyhedra for all n polyhedra in a given time step; b) mapping this fusion polyhedron to the next time step; c) or mapping all polyhedra to the next time step, thus doubling both n and f. Formally, following are the ways of using past measurements considered in this work:

1. *map_n*: In this approach we map all polyhedra in $\mathcal{N}(t)$ to time $t+1$, and obtain a total of $2n$ polyhedra in time $t+1$. We then compute their fusion polyhedron with $2f$ as the bound on the number of corrupted polyhedra. This is illustrated in Figure 3a. Formally the fusion polyhedron can be described as

$$S_{m(\mathcal{N}(t)) \cup \mathcal{N}(t+1), 2f}.$$

2. *map_S_and_intersect*: This algorithm computes the fusion polyhedron at time t, maps it to time $t+1$, and then intersects it with the fusion polyhedron at time $t+1$, as illustrated in Figure 3b. Formally we specify this as

$$m(S_{\mathcal{N}(t),f}) \cap S_{\mathcal{N}(t+1),f}.$$

3. *map_S_and_fuse*: Here the fusion polyhedron from time t is mapped to time $t+1$, thus obtaining a total of $n+1$ polyhedra at time $t+1$, as presented in Figure 3c. Note that f is still the same because $S_{\mathcal{N}(t),f}$ is guaranteed

to contain the true value by Proposition 1. Formally this is captured by

$$S_{m(S_{\mathcal{N}(t),f}) \cup \mathcal{N}(t+1),f}.$$

4. *map_R_and_intersect*: This is similar to *map_S_and_intersect*, but instead we map $R_{\mathcal{N}(t),f}$ to time $t+1$, intersect with $R_{\mathcal{N}(t+1),f}$, and compute the convex hull as illustrated in Figure 3d. Formally we describe this as

$$\texttt{conv}(m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f}).$$

5. *pairwise_intersect*: This algorithm performs pairwise intersection as shown in Figure 3e. Formally we capture this as

$$S_{m(\mathcal{N}(t)) \cap_p \mathcal{N}(t+1),f}.$$

The obvious way to compare these algorithms is through the volume of the fusion polyhedra. We provide below a series of results that relate the sizes of the fusion polyhedra for the aforementioned methods used to incorporate measurement history.

Theorem 3. *The region obtained using map_R_and_intersect is a subset of the region derived by map_n.*

Proof. Consider any point $p \in m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f}$. Then p lies in at least $n-f$ polyhedra in $\mathcal{N}(t+1)$, and there exists a q such that $m(q)=p$ that lies in at least $n-f$ polyhedra in $\mathcal{N}(t)$. Thus, p lies in at least $2n-2f$ polyhedra in $m(\mathcal{N}(t)) \cup \mathcal{N}(t+1)$, i.e., $p \in R_{m(\mathcal{N}(t)) \cup \mathcal{N}(t+1),2f}$, implying

$$\begin{aligned}\texttt{conv}(m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f}) &\subseteq \texttt{conv}(R_{m(\mathcal{N}(t)) \cup \mathcal{N}(t+1),2f}) \\ &= S_{m(\mathcal{N}(t)) \cup \mathcal{N}(t+1),2f}.\end{aligned}$$

□

Theorem 4. *The polyhedron derived by map_R_and_intersect is a subset of the polyhedron obtained by map_S_and_intersect.*

Proof. Note that for any sets $\mathcal{A}$ and $\mathcal{B}$, $\texttt{conv}(\mathcal{A} \cap \mathcal{B}) \subseteq \texttt{conv}(\mathcal{A})$, and thus

$$\begin{aligned}\texttt{conv}(m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f}) &\subseteq \texttt{conv}(R_{\mathcal{N}(t+1),f}) \\ &= S_{\mathcal{N}(t+1),f}.\end{aligned}$$

Furthermore, any point $p \in \texttt{conv}(m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f})$ is a convex combination of points q_i in $m(R_{\mathcal{N}(t),f})$. But $m(R_{\mathcal{N}(t),f}) \subseteq m(S_{\mathcal{N}(t),f})$ (since $R_{\mathcal{N}(t),f} \subseteq S_{\mathcal{N}(t),f}$) and the fact that $m(S_{\mathcal{N}(t),f})$ is convex imply $p \in m(S_{\mathcal{N}(t),f})$. Accordingly,

$$\begin{aligned}&\texttt{conv}(m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f}) \subseteq S_{\mathcal{N}(t+1),f} \text{ and} \\ &\texttt{conv}(m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f}) \subseteq m(S_{\mathcal{N}(t),f})\end{aligned}$$

implying

$$\texttt{conv}(m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f}) \subseteq m(S_{\mathcal{N}(t),f}) \cap S_{\mathcal{N}(t+1),f}.$$

□

Theorem 5. *The polyhedron obtained by map_R_and_intersect is a subset of the polyhedron derived using map_S_and_fuse.*

Proof. Note that, since the fusion interval is always guaranteed to contain the true value, the number of corrupted polyhedra in *map_S_and_fuse* is still at most f, but the number of correct ones is now at least $n+1-f$. In addition, note that

$$m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f} \subseteq m(S_{\mathcal{N}(t),f})$$

since $m(R_{\mathcal{N}(t),f}) \subseteq m(S_{\mathcal{N}(t),f})$. Furthermore, any point $p \in R_{\mathcal{N}(t+1),f}$ is contained in $n-f$ polyhedra in $\mathcal{N}(t+1)$. Thus, all points in $m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f}$ are contained in $n+1-f$ polyhedra in $m(S_{\mathcal{N}(t),f}) \cup \mathcal{N}(t+1)$, and hence in $R_{m(S_{\mathcal{N}(t),f}) \cup \mathcal{N}(t+1),f}$. Since the fusion polyhedron is convex,

$$\texttt{conv}(m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f}) \subseteq S_{m(S_{\mathcal{N}(t),f}) \cup \mathcal{N}(t+1),f}.$$

□

Theorems 3, 4 and 5 suggest that *map_R_and_intersect* is the best of the first four methods enumerated above as can also be seen in Figure 3. This intuitively makes sense since it is only keeping enough information from previous measurements to guarantee that the true value is preserved. In particular, it is not computing the convex hull at time t as *map_S_and_intersect* and *map_S_and_fuse* do (and potentially introduce additional points to the fused region), nor is it mapping potentially corrupted polyhedra as does *map_n*.

We note, however, that without additional assumptions about the rank of A, *map_R_and_intersect* and *pairwise_intersect* are not subsets of each other. Counter-examples are presented in Figure 4. In Figure 4a, $R_{\mathcal{N}(t),f}$ is a single point that is projected onto the x axis. Hence *map_R_and_intersect* is a subset of *pairwise_intersect*, which produces an interval of points. Conversely, Figure 4b shows an example where *pairwise_intersect* is a point, and *map_R_and_intersect* is an interval containing that point. It is worth noting, however, that regardless of which of the two approaches is used, *pairwise_intersect* can be used as a preliminary step to detect corrupted sensors – if the two polyhedra of a certain sensor have an empty intersection, then the sensor must be corrupted (faulty or tampered with) in one of the rounds; thus, it can be discarded from both, effectively reducing n and f by one.

Finally, we note that if A is a full rank matrix and $w=0$, then *pairwise_intersect* is the best of all five methods, as shown in the following theorem.

Theorem 6. *If A is full rank and $w=0$, the polyhedron obtained by pairwise_intersect is a subset of the polyhedron derived using map_R_and_intersect.*

Proof. Let p be any point in $R_{m(\mathcal{N}(t)) \cap_p \mathcal{N}(t+1),f}$. Then p lies in at least $n-f$ polyhedra in $m(\mathcal{N}(t))$ and at least $n-f$ polyhedra in $\mathcal{N}(t+1)$. Hence,

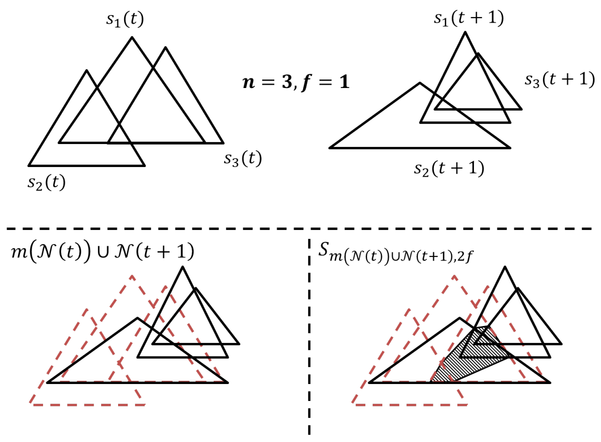
$$R_{m(\mathcal{N}(t)) \cap_p \mathcal{N}(t+1),f} \subseteq R_{\mathcal{N}(t+1),f}.$$

Furthermore, there exists a point $q = A^{-1}p$ that is contained in $n-f$ intervals in $\mathcal{N}(t)$. Therefore, p is also contained in $m(R_{\mathcal{N}(t),f})$. Then

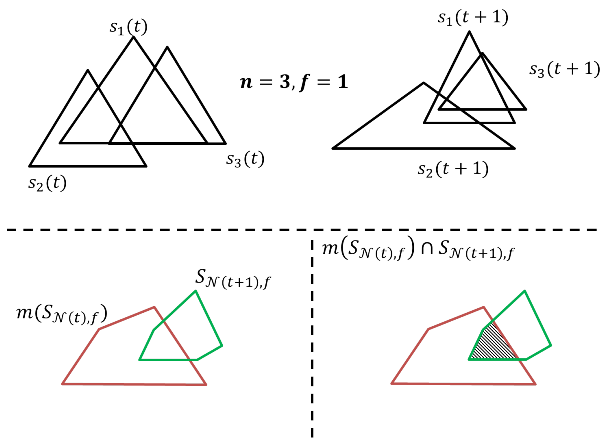
$$\begin{aligned}&R_{m(\mathcal{N}(t)) \cap_p \mathcal{N}(t+1),f} \subseteq m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f}, \text{ i.e.,} \\ &S_{m(\mathcal{N}(t)) \cap_p \mathcal{N}(t+1),f} \subseteq \texttt{conv}(m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f}).\end{aligned}$$

□

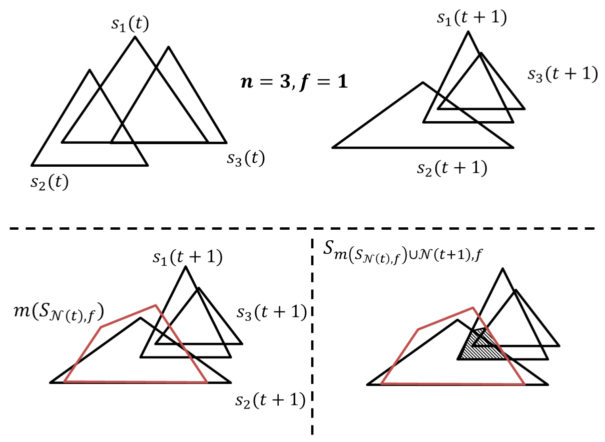
The conditions in Theorem 6 may seem too strong at a first glance. In reality, however, the difference between a


$s_1(t)$
$s_2(t)$
$s_3(t)$
$\boldsymbol{n = 3, f = 1}$
$s_1(t+1)$
$s_2(t+1)$
$s_3(t+1)$
$m(\mathcal{N}(t)) \cup \mathcal{N}(t+1)$
$S_{m(\mathcal{N}(t)) \cup \mathcal{N}(t+1), 2f}$

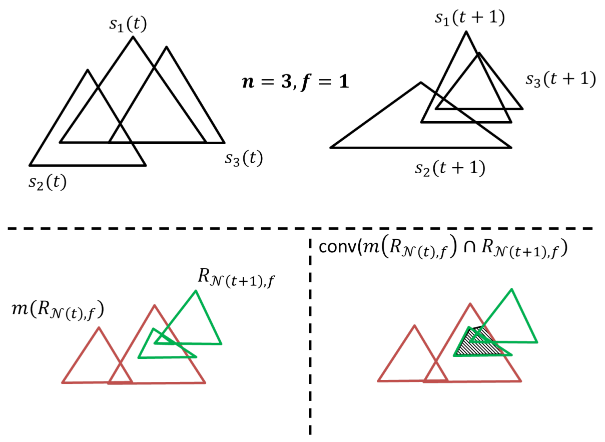
(a) *map_n*


$s_1(t)$
$s_2(t)$
$s_3(t)$
$\boldsymbol{n = 3, f = 1}$
$s_1(t+1)$
$s_2(t+1)$
$s_3(t+1)$
$m(S_{\mathcal{N}(t),f})$
$S_{\mathcal{N}(t+1),f}$
$m(S_{\mathcal{N}(t),f}) \cap S_{\mathcal{N}(t+1),f}$

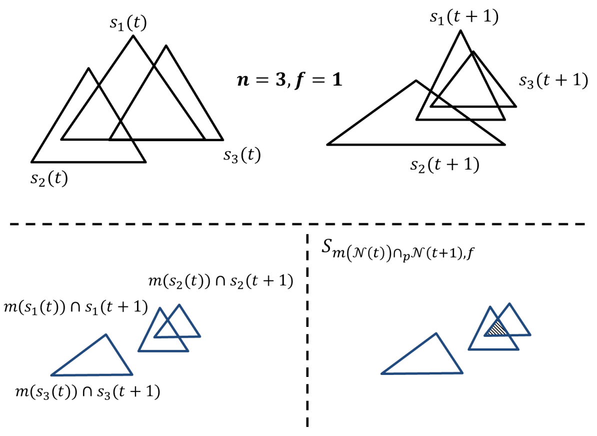
(b) *map_S_and_intersect*


$s_1(t)$
$s_2(t)$
$s_3(t)$
$\boldsymbol{n = 3, f = 1}$
$s_1(t+1)$
$s_2(t+1)$
$s_3(t+1)$
$m(S_{\mathcal{N}(t),f})$
$s_1(t+1)$
$s_3(t+1)$
$s_2(t+1)$
$S_{m(S_{\mathcal{N}(t),f}) \cup \mathcal{N}(t+1),f}$

(c) *map_S_and_fuse*

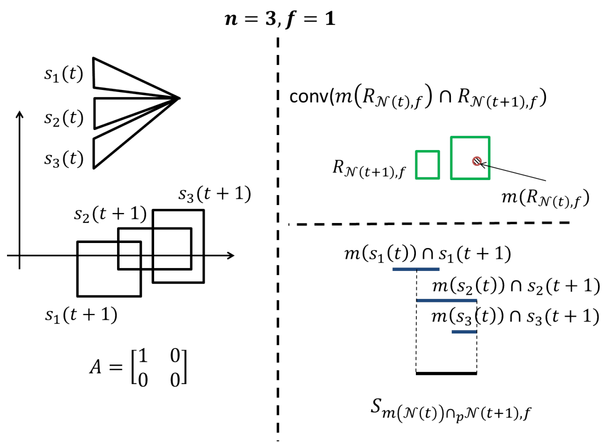

$s_1(t)$
$s_2(t)$
$s_3(t)$
$\boldsymbol{n = 3, f = 1}$
$s_1(t+1)$
$s_2(t+1)$
$s_3(t+1)$
$m(R_{\mathcal{N}(t),f})$
$R_{\mathcal{N}(t+1),f}$
$\mathrm{conv}(m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f})$

(d) *map_R_and_intersect*


$s_1(t)$
$s_2(t)$
$s_3(t)$
$\boldsymbol{n = 3, f = 1}$
$s_1(t+1)$
$s_2(t+1)$
$s_3(t+1)$
$m(s_2(t)) \cap s_2(t+1)$
$m(s_1(t)) \cap s_1(t+1)$
$m(s_3(t)) \cap s_3(t+1)$
$S_{m(\mathcal{N}(t)) \cap_p \mathcal{N}(t+1),f}$

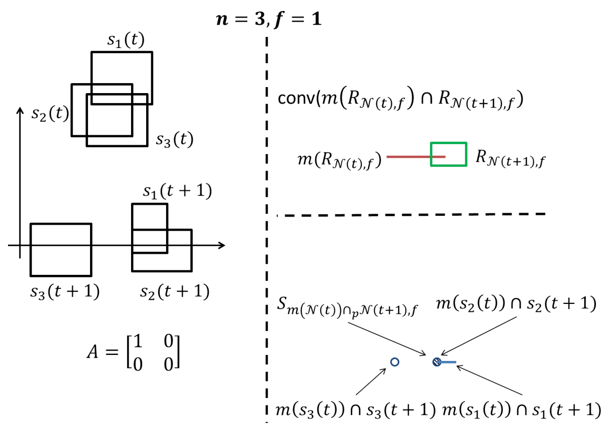
(e) *pairwise_intersect*

Figure 3: Illustrations of the different methods of using history. For simplicity $A = I$, the identity matrix, and $w = 0$.



$n = 3, f = 1$
$s_1(t)$
$s_2(t)$
$s_3(t)$
$\text{conv}(m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f})$
$R_{\mathcal{N}(t+1),f}$
$m(R_{\mathcal{N}(t),f})$
$s_3(t+1)$
$s_2(t+1)$
$s_1(t+1)$
$m(s_1(t)) \cap s_1(t+1)$
$m(s_2(t)) \cap s_2(t+1)$
$m(s_3(t)) \cap s_3(t+1)$
$A = \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}$
$S_{m(\mathcal{N}(t))\cap_p\mathcal{N}(t+1),f}$

(a) *map_R_and_intersect* is not a subset of *pairwise_intersect*.



$n = 3, f = 1$
$s_1(t)$
$s_2(t)$
$s_3(t)$
$\text{conv}(m(R_{\mathcal{N}(t),f}) \cap R_{\mathcal{N}(t+1),f})$
$m(R_{\mathcal{N}(t),f})$
$R_{\mathcal{N}(t+1),f}$
$s_1(t+1)$
$s_3(t+1)$
$s_2(t+1)$
$S_{m(\mathcal{N}(t))\cap_p\mathcal{N}(t+1),f}$
$m(s_2(t)) \cap s_2(t+1)$
$A = \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}$
$m(s_3(t)) \cap s_3(t+1)$
$m(s_1(t)) \cap s_1(t+1)$

(b) *pairwise_intersect* is not a subset of *map_R_and_intersect*.

Figure 4: Examples showing that, in general, polyhedra obtained using *map_R_and_intersect* and *pairwise_intersect* are not subsets of each other if A is not full rank.

singular and a nonsingular matrix is almost negligible. In particular, if A is singular, where λ_{min} is its smallest (in magnitude) nonzero eigenvalue, then $A + \varepsilon I$ is nonsingular, where $0 < \varepsilon < |\lambda_{min}|$. Therefore, systems with small noise in their dynamics will closely approximate the above requirements.

Therefore, we argue that systems that incorporate past measurements in their sensor fusion algorithms should use the *pairwise_intersect* method. We now show that it satisfies the requirements outlined in Section 2.

Proposition 3. *The fusion polyhedron computed using* pairwise_intersect *will always contain the true value.*

Proof. Note that pairwise intersection does not increase the number of corrupted polyhedra. If a sensor is correct, then both of its polyhedra (in time t and $t+1$) contain the true value; in addition, the map m preserves the correctness of polyhedra, hence any pairwise intersection will also contain the true value. Thus, the number of corrupted and correct sensors is the same, therefore Proposition 1 implies that the fusion polyhedron contains the true value. □

Proposition 4. *The fusion polyhedron computed using* pairwise_intersect *is never larger than the fusion polyhedron computed without using history.*

Proof. Each of the polyhedra (e.g., $m(P_1(t)) \cap P_1(t+1)$) computed after pairwise intersection is a subset of the corresponding polyhedron when no history is used (e.g., $P_1(t+1)$). Consequently, the fusion polyhedron will always be a subset of the fusion polyhedron obtained when no history is used. □

Note that *pairwise_intersect* and *map_R_and_intersect* do not add significant computational complexity to the sensor fusion algorithm described in Section 3. While they still suffer from the exponential procedure of computing the fusion polyhedron at each time, each of the two methods requires storing at most n polyhedra to represent historical measurements - intuitively they are the "intersection" of all past measurements. Thus, implementing any of these methods will not add substantial computational or memory cost for the system. The algorithm's implementation is discussed in greater detail in the following section.

5. APPLICATIONS

Given our results in Section 4, we argue that systems with linear dynamics should use the *pairwise_intersect* method. This section provides an algorithm that implements this method and a case study to illustrate its usefulness.

5.1 Implementation

The implementation is shown in Algorithm 2. In essence, at each point in time n polyhedra (the pairwise intersections) are stored. Thus, *past_meas* represents the "pairwise intersection" of all previous measurements of each sensor. In addition to being more efficient in terms of the size of the fusion polyhedron, the algorithm also needs very little memory – the required memory is linear in the number of sensors irrespective of how long the system runs.

An important detail that is hidden behind the `pair_inter` function is how corrupted sensors are dealt with. If a sensor s_i's two polyhedra have an empty intersection then that sensor must be corrupted. This is where we use the assumption about the same set of polyhedra that are corrupted over time. In particular, if we relax this assumption, then *pairwise_intersect* does not guarantee that the true value is contained in the fusion polyhedron. If this is the case, then we revert to *map_R_and_intersect*, the best of the methods that do not rely on this assumption.

On the other hand, if that assumption is satisfied, both polyhedra are discarded and n and f are reduced by one. Furthermore, the system has the option of discarding all future measurements provided by the sensor s_i; alternatively, the system may update *past_meas* with s_i's measurement in the next round. Which choice is made depends on the system's trust in the sensor – if it is believed to be often faulty or under attack, then discarding all or some of its future measurements is the better option. However, if it is faulty rarely, then its future measurements should be kept and incorporated in the algorithm. Quantification of sensor

Algorithm 2 Implementation of the *pairwise_intersect* algorithm

```
Input: $f$, the number of corrupted sensors
 1: $past\_meas \leftarrow \emptyset$
 2: for each step t do
 3:     $cur\_meas \leftarrow$ get_meas($t$)
 4:     if $past\_meas == \emptyset$ then
 5:         $past\_meas \leftarrow cur\_meas$
 6:     else
 7:         $past\_meas =$ pair_inter($cur\_meas, past\_meas$)
 8:     end if
 9:     $S \leftarrow$ fuse_polyhedra($past\_meas, f$)
10:     send_polyhedron_to_controller($S$)
11: end for
```

Figure 5: LandShark vehicle [1].

trust, however, is not within the scope of this paper, hence we take this choice as a design-time decision (input) and leave its analysis for future work.

5.2 Case Study

To show the effectiveness of the *pairwise_intersect* approach we use the sensors on a real autonomous vehicle, called the LandShark [1] (the robot is shown in Figure 5). The LandShark is capable of moving at a high speed on different surfaces and is usually used to carry humans or other cargo in hostile environments.

The LandShark has four sensors that can be used to estimate velocity – GPS, camera and two encoders. In addition, GPS and the camera can be used to estimate the vehicle's position. Therefore, the encoders provide the controller with interval estimations of the vehicle's velocity only, whereas GPS and the camera send two-dimensional polyhedra as estimates of the velocity and position.[1] The sizes of the encoders' intervals were obtained based on the manufacturer's specification, whereas the shapes and sizes of GPS and camera's polyhedra were determined empirically – the LandShark was driven in the open, and largest deviations from the true values (as measured by a high-precision laser tachometer) were collected.

Given this information, the following three scenarios were simulated. The LandShark is moving in a straight line at a constant speed of 10 mph. In each scenario, a different sensor was attacked such that a constant offset of 1 mph was introduced to the sensor's speed estimate. The sensors' speed tolerances were as follows: 0.2 mph for the encoders, 1 mph for GPS and 2 mph for the camera. GPS's tolerance for position was 30 feet, whereas the camera's tolerance varies with speed (hence its polyhedron is a trapezoid) and was 100 feet at 10 mph. At each point in time, we compute the fusion polyhedron in two ways – using only current measurements and using the *pairwise_intersect* method. Finally, we record the differences and the improvement achieved by using history.

To illustrate consistence with earlier one-dimensional works (e.g., [9]), for each of the three scenarios we first computed the size of the fusion interval in one dimension. Figure 6 presents the results. For each scenario, the size of the fusion interval was never larger when using *pairwise_intersect*, while the gain was significant at certain times. This is particularly apparent when the encoder was under attack. The reason for this, as we have shown in our recent work, is that it is in general beneficial for the attacker to corrupt the most precise sensors [6].

Figure 7 presents the results when two-dimensional polyhedra are considered. Note that in this case there are only two sensors estimating the robot's position – when one is corrupted, the size of the fusion polyhedron can grow dramatically. Consequently, *pairwise_intersect* is greatly beneficial for the system as it identifies the corrupted sensors and discards their measurements when their polyhedra do not intersect. It is worth noting here that in all simulated scenarios if a sensor is found corrupted in any step we do not disregard its measurement in the next step. Note also that the volumes in Figure 7 are much larger than those in Figure 6 – this is a result of the fact that position tolerances are measured in feet and are larger than 10. (i.e., 30 feet for GPS). Finally, as consistent with Proposition 3, all fusion polyhedra contained the actual value of velocity (i.e., 10 mph).

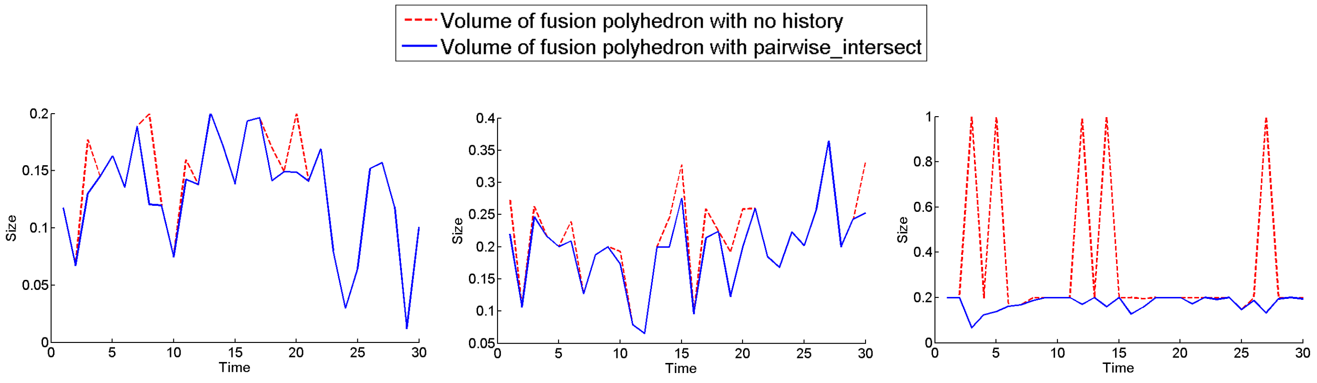
6. CONCLUSION

This paper considered the problem of resilient multidimensional sensor fusion using measurement history. We focused on the setup where each sensor provides a multidimensional polyhedron as a measurement of the plant's state. We presented a fusion algorithm for this case and gave bounds on the volume of the fusion polyhedron based on the number of corrupted sensors. In addition, we investigated several methods of using history to improve sensor fusion and identified the best ones depending on what conditions the system in consideration satisfies. Finally, we presented a case study to illustrate the improvement in sensor fusion that can be obtained when history-based fusion is employed.

There are two main avenues that we plan to explore in the future. Firstly, one could consider extending this approach to nonlinear systems; in this scenario polyhedra may no longer be mapped into polyhedra. Thus, one can utilize a set membership technique in order to compute the set of possible true values [10, 11]. Secondly, one could take into consideration sensor trust, i.e., how often a sensor has been faulty over time, and incorporate this information in the mapping algorithm.

[1]For this case study we only require one-dimensional position as will become clear in the next paragraph. However, our approach could easily be applied to multidimensional measurements.

7. ACKNOWLEDGEMENTS

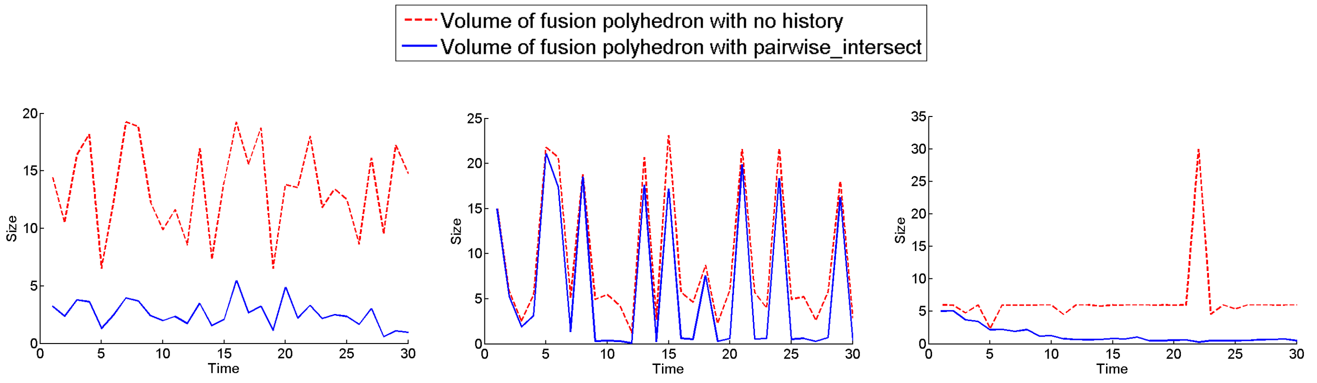


(a) GPS under attack.

(b) Camera under attack.

(c) Encoder under attack.

Figure 6: Sizes of velocity (ONLY) fusion intervals for each of the three simulated scenarios; Dashed line – volume of the fusion polyhedra when measurement history is not considered, Solid line – volume of the fusion polyhedra obtained using *pairwise_intersect*.



(a) GPS under attack.

(b) Camera under attack.

(c) Encoder under attack.

Figure 7: Sizes of fusion polyhedra of velocity and position for each of the three scenarios simulated; Dashed line – volume of the fusion polyhedra when measurement history is not considered, Solid line – volume of the fusion polyhedra obtained using *pairwise_intersect*.

This material is based on research sponsored by DARPA under agreement number FA8750-12-2-0247. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

8. REFERENCES

[1] The LandShark. http://blackirobotics.com/LandShark_UGV_UC0M.html.

[2] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete and Computational Geometry*, 8(1):295–313, 1992.

[3] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *SEC'11: Proc. 20th USENIX conference on Security*, pages 6–6, 2011.

[4] P. Chew and K. Marzullo. Masking failures of multidimensional sensors. In *SRDS'91: Proc. 10th Symposium on Reliable Distributed Systems*, pages 32–41, 1991.

[5] V. Delouille, R. Neelamani, and R. Baraniuk. Robust distributed estimation in sensor networks using the embedded polygons algorithm. In *IPSN'04: Proc. 3rd International Symposium on Information Processing in Sensor Networks*, pages 405–413, 2004.

[6] R. Ivanov, M. Pajic, and I. Lee. Attack-resilient sensor fusion. In *DATE'14: Design, Automation and Test in Europe*, 2014.

[7] D. N. Jayasimha. Fault tolerance in a multisensor environment. In *SRDS'94: Proc. 13th Symposium on Reliable Distributed Systems*, pages 2–11, 1994.

[8] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. Experimental security analysis of a modern automobile. In *SP'10: IEEE Symposium on Security and Privacy*, pages 447–462, 2010.

[9] K. Marzullo. Tolerating failures of continuous-valued sensors. *ACM Trans. Comput. Syst.*, 8(4):284–304, 1990.

[10] M. Milanese and C. Novara. Set Membership identification of nonlinear systems. *Automatica*, 40(6):957–975, 2004.

[11] M. Milanese and C. Novara. Unified set membership theory for identification, prediction and filtering of nonlinear systems. *Automatica*, 47(10):2141–2151, 2011.

[12] X. Niu, Q. Zhang, Y. Li, Y. Cheng, and C. Shi. Using inertial sensors of iPhone 4 for car navigation. In *Position Location and Navigation Symposium (PLANS), 2012 IEEE/ION*, pages 555–561, 2012.

[13] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *IPSN'05: Proc. 4th International Symposium on Information Processing in Sensor Networks*, pages 63–70, 2005.

[14] Y. Zhu and B. Li. Optimal interval estimation fusion based on sensor interval estimates with confidence degrees. *Automatica*, 42(1):101–108, 2006.