2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Minimal Persistence Control on Dynamic Directed Graphs for Multi-Robot Formation

Hua WANG and Yi GUO

Abstract— **Given a multi-robot system, in order to preserve its geometric shape in a formation, the minimal persistence control addresses questions: (1) what pairwise communication connections have to be prescribed to minimize communication channels, and (2) which orientations of communication links are to be placed between robots. In this paper, we propose a minimal persistence control problem on multi-robot systems with underlying graphs being directed and dynamically switching. We develop distributed algorithms based on the rank of the rigidity matrix and the pebble game method. The feasibility of the proposed methods is validated by simulations on the robotic simulator Webots and experiments on e-puck robot platform.**

I. INTRODUCTION

In a coordination task of multi-robot systems, the topology control on (dynamic) graphs explicitly specifies how system-level interactions are organized, while the control law on each robot defines how the state variables update locally. In other words, with the topology control satisfying a specific topological property (i.e., connectivity), the control law on each robot will elegantly coordinate the multi-robot system (i.e., reaching concensus), in which the original coordination problem is transformed into a simplified problem with a combinatorial assumption. In this paper, we are interested in the persistence control over dynamic directed graphs, and its related formation control.

Among the literature on formation control, there exists two distinct categories according to the type of information that the control law requires –the *position-based* formation control in [1] [2] [3] [4], and the *distance-based* formation control in [5] [6] [7] [8]. The position-based formation control steers each vehicle upon the relative direction and distance information shared by adjacent vehicles. Instead, the distance-based formation control steers the vehicle upon the relative distance only. It maintains a set of vehicles in certain geometric shape by only keeping local relative distance, given the undirected/directed framework is rigid/persistent. Therefore, the rigidity/persistence control addresses the necessary topology property in formation control applications.

The framework persistence stretches the framework rigidity-like problem with specification on edge-orientations. The framework rigidity addresses the question "how many pairwise distances must be prescribed between points so that all the distances between these points are preserved". It is extensively studied in the distance-based formation control applications as in [5] and [9]. Olfati-Saber and Murray in the pinioneer work [5] propose a theoretical topological framework based on the notion of rigidity, which defines interesting formation control problems, e.g., split, rejoin, and reconfiguration maneuvers formally. In [9], Erev et al. propose a systematic, inductive construction method to build a provably rigid formation based on the Henneberg sequence and Delaunay triangulation. *On the other hand*, the persistence addresses the problem how to preserve all the pairwise distances with underlying directed graphs by controlling edge placements and edge orientations in [10] and [11]. In [10], Hendrickx et al. study the theoretical principle to construct a provably persistent graph in a planar space by a sequential elementary operations. Yu et al. in [11] provide a theoretical frameowrk to study persistence and structural persistence in a three and higher dimensional space. Approaches adopted in [10] and [11] to construct persistent graphs are parallel to the "Henneberg sequence" approach in undirected graphs.

In this paper, we consider effective schemes to control the minimal persistence over dynamic directed graphs. We develop minimal rigidity control by manipulating the rank of rigidity matrix, and constraint consistence control based on the pebble game theory. We verify the feasibility of the proposed methodology in simulations on robotic simulator Webots, and on a testbed of 6 e-puck robots. This work distinguishes peers' work in the following aspects. First, we consider "dynamic" directed graph. Also, the proposed control can be used to construct and maintain the persistence of a dynamic framework based on the rank of rigidity matrix. In contrast, the previous work in [10] and [11] proposed a sequential approach to construct a persistent framework based on Henneberg sequence. Last, our approach can be extended into arbitrary m-dimensional space.

II. PRELIMINARIES

A. Graph rigidity and Its sufficient conditions

Rigidity is the sufficient structure property that a group of robots maneuver as a cohesive whole, given the underlying communication graph is undirected. Persistence is formulated to address the similar problem in directed graphs. We first introduced in the basic notions in the context of rigidity.

We consider graphs with no self-loop nor multiple edges between two nodes. A *graph* $G(V, E)$ consists of a vertex set $V = \{1, 2, \ldots n\}$ and an edge set $E \subseteq V \times V$, which is a collection of pairs of vertices, note that $\times$ is the Cartesian product. If the pairs of vertices are unordered, the graph is called an *undirected graph*; otherwise, the graph is called a *directed graph*. In the context of this paper, we consider the leader-follower communication architecture as a directed

Hua Wang and Yi Guo are with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA wanghuahit@gmail.com; yi.guo@stevens.edu

graph, in which a directed edge points from the follower to the leader, i.e., head is the leader and tail is the follower[1]. The *indegree* $d_{in}^{(i)}$ of vertex i is the number of directed edges pointing to vertex i, the *outdegree* $d_{out}^{(i)}$ of vertex i is the number of directed edges leaving vertex i. A *framework* is a triple $(V, E, \mathbf{p})$ where (V, E) is a graph and $\mathbf{p} = [\mathbf{p}_1, \dots \mathbf{p}_n]^T$ is a coordinate vector corresponding to the vertices V in an m dimensional Euclidean space.

A *deformation* is a continuous one-parameter family $\mathbf{p}(t) = (\mathbf{p}_1(t), \dots, \mathbf{p}_n(t))$ on a framework. A deformation is trivial if it preserves the distance between any two points, whether they are adjacent or not. A framework is said to be *rigid* if it has only trivial deformations, e.g., translations and rotations. Otherwise, it is *flexible*.

Mathematically, the distance-preserving between any two points in the edge set E can be written as

$$(\mathbf{p}_i(t) - \mathbf{p}_j(t)) \cdot (\mathbf{p}_i(t) - \mathbf{p}_j(t)) = c_{ij}, \forall (i,j) \in E \quad (1)$$

By taking the first derivative over time on both sides of the equation, we get

$$(\mathbf{p}_i(t) - \mathbf{p}_j(t)) \cdot (\mathbf{p}_i'(t) - \mathbf{p}_j'(t)) = 0, \forall (i,j) \in E \quad (2)$$

We define a framework $\{V, E, \mathbf{p}\}$ to be *first-order rigid* or *infinitesimally rigid* if its coordinates $\mathbf{p}$ meet (2). Infinitesimal rigidity is a natural approximation to rigidity.

A rigid framework $\{V, E, \mathbf{p}\}$ is called *minimally rigid* (MR) if any removal of a single edge will destruct the rigidity. Otherwise, $\{V, E, \mathbf{p}\}$ is called *redundantly rigid*.

Next, we introduce the sufficient condition on rigidity, the rank criteria of the rigidity matrix, which guarantees the infinitesimal rigidity of a framework in an m-dimensional space [12]. We start with the rigidity matrix R. Assume the coordinates $\mathbf{p}$ of the n vertices in an m-dimensional Euclidean space $\mathbb{R}^m$ are:

$$\begin{aligned} \mathbf{p} &= \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_n \end{bmatrix}^T \\ &= \begin{bmatrix} p_{11}\ p_{12} \dots p_{1m} \mid p_{21}\ p_{22} \dots\ p_{2m} \mid p_{n1}\ p_{n2} \dots p_{nm} \end{bmatrix}^T \end{aligned}$$

The *rigidity matrix* R is an $|E| \times nm$ matrix whose rows and columns are indexed by the edges and the vertices respectively, i.e., each row corresponds to an edge, and each consecutive m-column block from leftmost corresponds to a vertex. In rigidity matrix R, on the kth row corresponding to the edge (i, j), it has vector coordinates $(\mathbf{p}_i - \mathbf{p}_j)^T$ in the columns from $(i-1)m+1$ to im, $(\mathbf{p}_j - \mathbf{p}_i)^T$ in the columns from $(j-1)m+1$ to jm, and zero elsewhere. The kth row of R is:

$$[\ 0\ \cdots\ 0\ \overbrace{(\mathbf{p}_i - \mathbf{p}_j)^T}^{\text{vertex } i}\ 0\ \cdots\ 0\ \overbrace{(\mathbf{p}_j - \mathbf{p}_i)^T}^{\text{vertex } j}\ 0\ \cdots\ 0\ ] \quad (3)$$

We illustrate how to construct a rigid matrix R in following example. Fig. 1 shows a framework $\mathbf{F}$ in a 3-dimensional Euclidean space $\mathbb{R}^3$ that consists of 4 vertices V_1, V_2, V_3, V_4,

[1]Note that in the context of this paper, the directed edge origins from the information receiver, pointing to the information sender (exactly opposite the data flow).

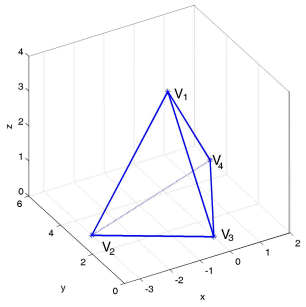


Fig. 1. A framework in a 3-dimensional space

i.e., the number of dimension m is 3; the number of nodes n is 4. Nodes' coordinates are $p_1 = (-1, 2, 4)^T, p_2 = (-3, 3, 0)^T, p_3 = (0, 1, 0)^T, p_4 = (2, 5, 0)^T$ respectively, and 6 edges are $e_1 = (V_1, V_2), e_2 = (V_1, V_3), e_3 = (V_1, V_4), e_4 = (V_2, V_3), e_5 = (V_2, V_4), e_6 = (V_3, V_4)$.

The rigidity matrix R is (6×12) shown below:

$$\begin{array}{c} \begin{matrix} V_1^x & V_1^y & V_1^z & V_2^x & V_2^y & V_2^z & V_3^x & V_3^y & V_3^z & V_4^x & V_4^y & V_4^z \end{matrix} \\ \left[\begin{array}{ccc|ccc|ccc|ccc} 2 & -1 & 4 & -2 & 1 & -4 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 4 & 0 & 0 & 0 & 1 & -1 & -4 & 0 & 0 & 0 \\ -3 & -3 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & -4 \\ 0 & 0 & 0 & -3 & 2 & 0 & 3 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -5 & -2 & 0 & 0 & 0 & 0 & 5 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & -4 & 0 & 2 & 4 & 0 \end{array}\right] \end{array} \quad (4)$$

The six rows from top to down in (4) represent edges e_1, e_2, e_3, e_4, e_5, and e_6 respectively. Next, we present Lemma 1 clarifying the connection between the rank of the rigidity matrix and the infinitesimal rigidity.

Lemma 1: [13] (Rank of the rigidity matrix) A framework on n vertices is infinitesimally rigid in an m-dimensional space if and only if its rigidity matrix R satisfies $\mathsf{rank}(R) = mn - \frac{m(m+1)}{2}$.

Lemma 1 implies that given a framework of n vertices, there are at most $mn - \frac{m(m+1)}{2}$ independent edges in the sense of rigidity. It can be used to determine the infinitesimal rigidity of frameworks in m-dimensional spaces.

We define critical rank rank_c as

$$\mathsf{rank}_c \stackrel{\text{def}}{=} m \cdot n - \frac{m \cdot (m+1)}{2} \quad (5)$$

A graph with n vertices is rigid in dimension m if it has at least rigidity-independent rank_c edges. We say a graph is *minimally rigid* if it has exactly rank_c edges.

From the rigidity matrix R of the framework $\mathbf{F}$ in Fig. 1, it is simple to compute that $\mathsf{rank}(R) = 6$, which equals rank_c. Thus, the framework $\mathbf{F}$ in Fig. 1 is infinitesimally rigid.

B. Persistence and Its sufficient conditions

Persistence is formulated to address the rigidity-like problem in directed graphs. We first present basic concepts regarding persistence.

In $\mathbb{R}^m$, a *presentation* of an undirected graph $G(V, E)$ is a function $\{p : V \to \mathbb{R}^m, p = \{p_1, p_2 \dots, p_n\}\}$, where $p_i \in p$ is the position of vertex i. A *distance set* $\bar{d}$ on $G(V, E)$ is

a set of distances $d_{ij} > 0$, for all edges $(i,j) \in E$. Given the desired distance set $\overline{d}$ corresponding to the formation constraints is properly defined, the position vector of the vertex set in a framework is called *fitting* for the distance set $\overline{d}$. More formally, given a representation p, the position of vertex i is fitting if there is no $p^* \in \mathbb{R}^m$ for which the following strict inclusion holds:

$$\begin{aligned} &\{\overrightarrow{(i,j)} \in E : \|p_i - p_j\| = d_{ij}\} \\ \subset &\{\overrightarrow{(i,j)} \in E : \|p^* - p_j\| = d_{ij}\} \end{aligned} \tag{6}$$

A representation p is *persistent* if there exists $\varepsilon > 0$ such that every representation p' fitting set induced by p and satisfying $d(p,p') < \varepsilon$ is congruent to p for $\mathfrak{R}^m$. A graph is then *generically persistent* if almost all its representations are persistent. The constraint consistence of a graph is the necessary and sufficient condition that a generically rigid graph is generically persistent. A representation p is *constraint consistent* if there exists $\varepsilon > 0$ such that any representation p' fitting for the distance set $\bar{d}$ induced by p and satisfying $d(p,p') < \varepsilon$ is a realization of $\bar{d}$. For vertex i, the position $p'(i)$ is the fitting if and only if the out-degree is $min(m, d_out(i))$. The constraint consistence of a representation prevents any node overruled by a set of redundant distance constraints.

Lemma 2: [11] A graph $G(V,E)$ in $\mathbb{R}^m (m \in \{2,3,\dots\})$ is always generically constraint consistent, if all the vertices have an out-degree smaller than or equal to m.

The above Lemma provides a out-degree upper bound criteria for the generally constraint consistency, and the following lemma from [11] connects the persistence, rigidity and constraint consistency.

Lemma 3: [11] A representation in $\mathbb{R}^m (m \in \{2,3,\dots\})$ is persistent if and only if it is rigid and constraint consistent. A graph in $\mathbb{R}^m$ is generically persistent if and only if it is generically rigid and generically constraint consistent.

The constraint consistence of a graph is related to the nodes' out-degree as presented in Lemma 4.

Lemma 4: [11] An m-dimensional graph is persistent if and only if all those subgraphs are rigid, and are obtained by successively removing outgoing edges from vertices with out-degree larger than m until all such vertices have an out-degree equal to m.

We last define the minimal persistence, which is analogous to minimal rigidity in undirected graphs. A framework in $\mathbb{R}^m$ is said to be *minimally persistent* if (1) it is persistent and (2) no single edge can be removed without losing persistence.

III. Problem formulation

Given a dynamic directed framework (V,E,p) consisting n vertices in m-dimensional space, the coordinations p of the vertices are fixed, the edge set E is time-varying. At time t_k, the corresponding edge set is $E_k \subseteq V \times V$. $|E_k|$ denotes the number of the edges in edge set E_k. We presume that each vertex in the framework is capable of processing information locally, localizing itself and receiving/transmitting data in a limited range with neighbors.

We have the following assumptions on the system.

- A1 A given framework (V,E,p) evolves in an m-dimensional space $\mathbb{R}^m$;
- A2 The underlying directed graph is reachable, that is, each state variable of the system is reachable by a path connecting all vertices from at least one vertex. We call the path connecting all vertices the "reachable path" P;
- A3 The edge set in the framework is under control sequence $\{\mathbb{C}^1 \times \mathbb{C}^2 \times \cdots \times \mathbb{C}^k\}$, control $\mathbb{C}^k$ on edge set is a graph operation.

Next, we define the basic graph operators on one single directed edge e_{ij}. We adopt notation $E \cup e_{ij}$ as edge e_{ij} addition onto edge set E, $E \Longleftarrow E \backslash e_{ij}$ as edge e_{ij} removal from edge set E, and $\circlearrowleft e_{ij}$ as the reverse operation on e_{ij}, which results e_{ji}. Now we define the persistence control problem as follows.

Problem 1: Given a dynamic framework (V,E,p) satisfying A1-A3, find a control sequence $\{\mathbb{C}^1 \times \mathbb{C}^2 \times \cdots \times \mathbb{C}^k\}$ under which the framework is (generically) minimally persistent.

IV. Minimal persistence control over dynamic frameworks

In this section, we propose a distributed scheme to control the persistence of a given directed framework. We assume that each robot is equipped with sensors to accurately localize itself and robots share information by point-point communications. Pairs of nodes communicate in a simple protocol of *send*, *request* or *wait*. Our motivation is to control the persistence of a framework by local communications and computations.

The minimal persistence of a given framework is subject to that (1) the framework is minimally rigid, and (2) the representation is constraint consistent. Since we consider frameworks in high dimensions, we adopt the "rank of rigidity" criteria to determine the infinitesimal rigidity of a given framework. The control on persistence is divided into the following sequential procedures: (1)locally constructs the rigidity matrix block based on a directed graph; (2) locally estimate the rank of the rigidity matrix to ensure the minimal rigidity of the framework; and (3) locally control the constraint consistence of the framework. Note that we design local algorithm with the directional communications.

A. Local construction of the rigidity matrix

Each robot i is equipped with sensors to acquire its location coordinations $p_i(t)$ itself at time t, $p_i(t)$ is an m-dimensional stack vector. By simply sending/receiving protocols, each robot is capable of constructing its own rigidity matrix block. If robot i sends its information to j, robot j constructs the m entries, which is the position difference vector $[x_j - x_i \;\; y_j - y_i \dots]$ defined in (3), and otherwise zeros.

To construct the rigidity matrix of a directed graph in a decentralized fashion, it is subject to the directional pairwise communication between (i,j) and the data flows along

the reachable path. For instance, if the communication link between pair (i, j) is directional, and node i is an ancestor of node j along the presumable reachable path, entries on the (i, j) row in rigidity matrix block corresponding to robot i, can be copied from the entries of jth rigidity matrix block, once the data flow gets to j.

B. Minimal rigidity control algorithm

After the rigidity matrix is locally built, we propose decentralized schemes to determine the infinitesimal rigidity of the framework by the rank of the rigidity matrix as described in Lemma 1. We present an agent-based distributed algorithm to determine the rigidity of dynamic frameworks based on multifrontal sparse QR factorization in Algorithm 1. The data flow passes on via point-point communications until the framework has been determined to be infinitesimally rigid or not. If the framework is rigid, Algorithm 1 returns 1.

The detail of the Algorithm 1 is outlined in the following. Initially, each robot constructs its local rigidity matrix block $R_{rgd}^{(i)}$ by exchanging their position information with its communicating neighbors. On Line 2, the QR factorization function 2 is called, given parameters $(i, R_{rgd}^{(i)}, T)$, where $R_{rgd}^{(i)}$ is the rigidity matrix block acquired by robot i, T is an topological ordering of the data flow in order to factorize the rigidity matrix. If the rank of the rigidity matrix is rank_c, the algorithm returns 1. Otherwise, it searches potential neighbors by extending communication range. By testing the triviality of each potential neighbor, a control sequence is generated to guarantee the framework is rigid. By testing the triviality of edge set E, a control decision is made to guarantee the rigidity-dependent edges are removed, therefore, the framework is minimally rigid.Once a minimally rigid framework is found, the algorithm returns with the control sequence $\{\mathbb{C}^1 \times \mathbb{C}^2 \times \cdots \times \mathbb{C}^k\}$.

To factorize a sparse matrix, we employ a multifrontal sparse QR factorization method. The operations can be categorized into the *symbolic phase*, which determines the supernodal elimination tree (the structure of the data flow) and *numeric phase* (which carries out the numerical operations) [14]. We mainly consider the numeric phase in Function 2, which describes the data processing procedure of the multifrontal sparse QR factorization. Let R be a rectangular sparse matrix to be QR factorized. A as a symmetric matrix satisfying $A = (PR)^T PR$, where P is a row ordering matrix. A_j corresponds to the jth row in the ordering of the supernode[2] S of matrix A. Based on the matrix structure of A, a topological ordering (a.k.a. supernodal elimination tree T) is defined, which determines the data flow. Each supernode S is associated with a *frontal matrix* F_S and an *update matrix* U_S.

In Function 2, the key operation in this algorithm is to compute the frontal matrix, in other words, to iteratively merge the update matrix from its children nodes into the frontal matrix F_S. The frontal matrix F_S is assembled and updated at each supernodal terminal (in the context of this paper, each robot is a supernodal terminal) until fully merged i.e., all the update matrices from children nodes and the matrix block it stands for. The "assembly" of an update matrix in Line 3 can be done by a matrix extend-add operation. The " merge" and "transform" operations on Lines 4, 8 and Line 9 can be done by Givens rotations[3]. The update matrix U_S can be calculated by a row/column elimination operation on the frontal matrix F_S in Line 11, then, it is passed on to the next parent node in the topological order.

The correctness of Algorithm 1 is straightforward. The infinitesimal rigidity of the framework is subject to the rank

Algorithm 1 The minimal rigidity control algorithm

Require:
A1 and a topological ordering of the elimination tree T
Goal: output the rigidity status and the control sequence $\{\mathbb{C}^1 \times \mathbb{C}^2 \times \cdots \times \mathbb{C}^k\}$.

For robot i, $1 \leq i \leq n$

1: obtain rigidity matrix block $R_{rgd}^{(i)}$ by local communications
2: call Function 2 $(i, R_{rgd}^{(i)}, T)$ to obtain F_i
3: evaluate rank $\hat{\mathsf{r}}(R_{rgd})$ of matrix R_{rgd} by judging F_i.
4: **if** $\hat{\mathsf{r}}(R_{rgd}) \neq \mathsf{rank}_c$ **then**
5: search potential neighbor set Υ^i
6: **for all** $j \in \Upsilon^{(i)}$ **do**
7: update rigidity matrix $R_{rgd}^{(i)'}$ with edge e_{ij}
8: call Function 2 $(i, R_{rgd}^{(i)'}, T)$ to obtain F_i
9: evaluate rank $\hat{\mathsf{r}}(R'_{rgd})$ of R_{rgd} by judging F'_i
10: **if** edge e_{ij} is trivial (e.g., $\hat{\mathsf{r}}(R_{rgd}) == \hat{\mathsf{r}}(R'_{rgd})$) **then**
11: discard the potential edge e_{ij};
12: **elseif** e_{ij} is nontrivial (e.g., $\hat{\mathsf{r}}(R_{rgd}) > \hat{\mathsf{r}}(R'_{rgd})$) **then**
13: update edge with $\{\mathbb{C}^w : E \leftarrow E \cup e_{ij}\}$
14: set $\hat{\mathsf{r}}(R_{rgd})$ as $\hat{\mathsf{r}}(R'_{rgd})$
15: **end if**
16: **end for**
17: **elseif** $\hat{\mathsf{r}}(R_{rgd}) == \mathsf{rank}_c$ **then**
18: **for all** $e_{ij} \in E$ **do**
19: **if** $|E| == \mathsf{rank}_c$ **then**
20: break;
21: **elseif** edge e_{ij} is trivial to rigidity **then**
22: discard edge e_{ij}, update edge set with $\{\mathbb{C}^m : E \leftarrow E \backslash e_{ij}\}$
23: **else**
24: preserve edge e_{ij}
25: **end if**
26: **end for**
27: **return** 1 with $\{\mathbb{C}^1 \times \mathbb{C}^2 \times \cdots \times \mathbb{C}^k\}$;
28: **end if**

[2]In the context of this paper, each robot is a supernode, which occupies equal-column block of rigidity matrix

[3]Givens rotatation is an operation on pair of rows to introduce single zero. By systematically applying it to successive pairs of rows, it can be used to annihilate elements in a matrix

Function 2 The parallel multifrontal sparse QR factorization

Input:

(i) a supernode S

(ii) an equal-column partitioning block R_S (corresponding to supernode S) of the sparse rectangular matrix R

(iii) the supernodal elimination tree T, the data flows from leaf nodes to the root

Output: frontal matrix F_S

1: **for** each supernode S in a topological ordering T **do**
2: allocate space for U_S and F_S
3: retrieve update matrices U_D from its children nodes
4: **for** each $D \in children(S)$ **do**
5: merge update matrix U_D into F_S and discard U_D;
6: **end for**
7: compute the symmetric matrix A_S on R_S
8: **for** each column j that S covers, $j \in S$ **do**
9: transform A_S to an upper trapezoidal matrix T_j;
10: merge T_j into F_S
11: **end for**
12: compute the update matrix U_S on F_S
13: send U_S to its parent
14: **return** F_S
15: **end for**

Algorithm 3 The constraint consistence control algorithm

Require:

(i) ***A1*** and ***A3***

(ii) a given undirected framework (V, E, p) is minimally rigid.

(iii) the out-degree quota $q_{out}^{(i)}$ is initialized as m.

Goal: find the control sequence $\{\mathbb{Q}_1 \times \mathbb{Q}_2 \times \ldots \mathbb{Q}_k\}$ orienting the edge set E with constraint upper-bounds.

For a local robot i, $1 \leq i \leq n$

1: **for** each edge e_{ij} in the edge set E^i **do**
2: **if** $q_{out}^{(i)} + q_{out}^{(j)} \geq (m+1)m/2 + 1$ **then**
3: orient e_{ij} as $\mathbb{Q}_k : \{i \to j\}$, and set $q_{out}^{(i)} \leftarrow q_{out}^{(i)} - 1$;
4: **else**
5: mark i and j as visited for the searching by others
6: searching for quotas from other neighbors
7: **if** a quota is found at robot k **then**
8: reverse all the edges along directed path i to k $\mathbb{Q}_k : \{\circlearrowleft e_t, \forall e_t \in \widehat{ik}\}$, and set $q_{out}^{(i)} \leftarrow q_{out}^{(i)} + 1$, $q_{out}^{(k)} \leftarrow q_{out}^{(k)} - 1$;
9: **end if**
10: **end if**
11: **return** control sequence $\{\mathbb{Q}_1 \times \mathbb{Q}_2 \times \ldots \mathbb{Q}_k\}$
12: **end for**

of its rigidity matrix. The rank of the rigidity matrix is determined by the multifrontal sparse QR factorization. For each robot, its rigidity matrix block is built by pairwise position exchange. The decentralized sparse QR algorithm processes the data, and passes the update information along until it reaches a concluding rigidity status y. Therefore, Algorithm 1 is simply applied to determine the rigidity of a given framework.

C. The constraint consistence control algorithm

Given a directed network topology, the persistence of a framework is subject to the rigidity and the constraint consistence. We propose Algorithm 3 to control the constraint consistence of a given undirected framework. By orienting the edge set, the out-degree of any robot is bounded by the space dimension m from Lemma 4. Algorithm 3 describes an efficient decentralized scheme balancing the out-degree of each node in the network. The basic idea origins from pebble game algorithm [15]. In order to guarantee that the node's out-degree is upper-bounded by the space dimension m, each robot has a state variable called out-degree quota $q_{out}^{(i)}$.

Algorithm 3 is run on each edge e_{ij}, where $e_{ij} \in E^i$, E^i is the edge set consisting all the edges of which node i is one end. Robot i locally orients its edge set E^i if the quota criteria is met on Line 2. Once an outward edge is set, the out-degree quota $q_{out}^{(i)}$ is reduced by one. If the quota criteria is not satisfied, it searches for potential out-degree quota from adjacent neighbors in Line 6, i.e., depth-first search. Once a out-degree quota is found at robot k, reverse the edges from i to k, and adjust the out-degree quotas accordingly in Line 8. Obviously, the out-degree quota of robot i is reduced by one once an outward edge is oriented, and the overall out-degree quotas of the network is a constant over time. The decentralized algorithm orients each edge only once, it terminates once the overall out-degree quotas of the network is (m+1)m/2.

The correctness of Algorithm 3 over minimally rigid framework (V, E, p) is guaranteed as of a typical case of the pebble game algorithm on a tight $(m, m(m+1)/2)$-sparse graph[4]. The initial graph is a tight $(m, m(m+1)/2)$-sparse graph since it is minimally rigid. From Theorem 8 in [15], it is proven that given a tight sparse graph, the pebble game algorithm will result in a *well-constraint* graph, that is, it has no greater than $(m+1)m/2$ out-degrees and the outward degrees of any node is "well-constraint" with the orientations in the graph. To summarize, Algorithm 3 guarantees that the framework is consistent constraint.

V. Simulation results

In this section, we show simulation results of the persistence control schemes introduced in Sections IV. We use robotic simulator Webots v6.1.5, a platform providing a virtual environment and virtual commercially-available robot types. We build a multi-robot system, each robot prototype is the e-puck robot. We show the impact of persistence control in a distance-based formation application. We apply distance-based formation control on the dynamic of robotic system, in which each robot updates its states upon its distance

[4]A graph G on n vertices is (k, l)-sparse if every subset of $n' < n$ vertices spans at most $kn' - l$ edges. G is tight if, in addition, it has exactly $(kn - l)$ edges [15].

measurement between robots. Note that the distance-based formation control [7] and [8] updates robotic dynamics upon the relative distance between robots.

In Fig. 2, we show the kinetic and topology dynamics of the multi-robot system. We label the robots from the top left counterclockwise as 1 to 6. In Fig. 2(a), 6 robots start to maneuver in the same direction (to the left) while maintaining a triangular shape formation. In Fig. 2(b), robot 1 stops moving. As a consequence, the link between robots 1 and 6 drops, the persistence of the framework is violated (since the framework is not rigid any more). By increasing communication radius on robot 1, robot 1 reaches robot 6. The request to add the link between robot 1 and 6 is approved based on the persistence correction algorithm. In Fig. 2(c), the orientations of the communication links are shown as the pointing arrows. In Fig. 2(d), the link $(2, 4)$ suddenly drops (we set it as unrecoverable), and the framework persistence is violated according to the rigidity checkup algorithm. In Fig. 2(e), as robot 2 keeps extending its communication range, few more links $(4, 6)$ and $(2, 5)$ are rebuilt. The framework is infinitesimally minimally rigid. In Fig. 2(f), under the persistence correction control and formation control, the 6-robot system reaches the original formation shape again.

VI. Experimental validation

In this section we demonstrate the experimental results carried out on a 6-robot mobile e-puck platform, in order to illustrate the impact of persistence control on distance-based formation control. The e-puck robot is equipped with a distributed microprocessor, two differential-drive step motors, and 8 IR sensors which serve dual functions: communication and distance sensing. Communication modules include an IR emitter and an IR receiver with adjustable communicating range; and distance sensors measure the distance between robots based on the IR sensor readings. In the distance-based formation control introduced in [7] and [8], the distances between robots are requisite, and the communication is critical in formation application. In order to eliminate the conflict on distance sensor and communication module, we use remote control functionality from Webots v6.1.5, which provides a bluetooth remote control between robots and computer. The Webots provides a *supervisor node* which can read all the systematic information in the world (the simulation environment in Webots) to measure the distance between robots and a *emitter/receiver node* which models the radio, serial or infra-red emitters/receivers to wirelessly exchange data between robots in the simulator. It also provides a Bluetooth remote control interface to synchronize the robotic motion between the virtual robot models and real world E-puck robots.

Under the remote control from the computer side, we apply the persistence control and formation control laws on the physical e-puck robotic system. We show the snapshots of the robotic platform in Fig. 3. The underlying communication links are the same as shown in Fig. 2 to facilitate comparisons. At t=0, the 6 e-puck robots maneuver in a triangle-shape formation as in Fig. 3(a). Then, the top left robot is stuck at its location. Therefore, the formation of the system is interrupted shown in Fig. 3(b). The formation is backed up in Fig. 3(c). The formation of the system is interrupted with certain links dropping of as shown in Fig. 3(d), and Fig. 3(e). But the robot team finally recovers its triangular shape formation in Fig. 3(f).

VII. Conclusion

In this paper, we study the framework persistence control of multi-robot systems with a dynamic directed communication framework. We first introduce the rigidity and persistence notions in the infinitesimal, generic, and minimal senses, and the sufficient conditions on rigidity and persistence. We then proposed a minimally persistent problem to control the persistence of a given dynamic framework. Based on the "rank of the rigidity matrix" criterion and pebble game algorithm, we developed distributed control schemes to solve the persistence control problem. The simulation and experimental results show the feasibility and effectiveness of the algorithms we developed.

References

[1] A. Williams, G. Lafferriere, and J. Veerman, "Stable motions of vehicle formations," in *Proceedings of the 44th IEEE conference on decision and control, european control conference. (CDC-ECC'05)*, Dec. 2005, pp. 72 –77.

[2] Z. Lin, M. Broucke, and B. Francis, "Local control strategies for groups of mobile autonomous agents," *IEEE Transactions on Automatic Control*, vol. 49, pp. 622–629, 2004.

[3] Z. Lin, B. Francis, and M. Maggiore, "Necessary and sufficient graphical conditions for formation control of unicycles," *IEEE Transactions on Automatic Control*, vol. 50, no. 1, pp. 121 –127, Jan 2005.

[4] J. Marshall, M. Broucke, and B. Francis, "Formations of vehicles in cyclic pursuit," *IEEE Transactions on Automatic Control*, vol. 49, no. 11, pp. 1963 – 1974, Nov. 2004.

[5] R. Olfati-Saber and R. M. Murray, "Graph rigidity and distributed formation stabilization of multi-vehicle systems," in *Proceedings of the 41st IEEE conference on decision and control*, vol. 3, Las Vegas, NV, Dec. 2002, pp. 2965–2971.

[6] L. Krick, M. E. Broucke, and B. Francis, "Stabilization of infinitesimally rigid formations of multi-robot networks," in *Proceedings of the 47th IEEE conference on decision and control*, Cancun, Mexico, Dec. 2008, pp. 477 –482.

[7] D. Dimarogonas and K. Johansson, "On the stability of distance-based formation control," in *Proceedings of the 47th IEEE conference on decision and control*, Cancun, Mexico, Dec. 2008, pp. 1200–1205.

[8] ——, "Further results on the stability of distance-based multi-robot formations," in *Proceedings of the american control conference*, St. Louis, MO, June 2009, pp. 2972–2977.

[9] T. Eren and P. Belhumeur, "A framework for maintaining formations based on rigidity," in *Proceedings of the IFAC world congress*, Barcelona, Spain, July 2002, pp. 2752–2757.

[10] J.M.Hendrickx, B. Fidan, C. Yu, B. Anderson, and V. Blondel, "Formation reorganization by primitive operations on directed graphs," *IEEE Transactions on Automatic Control*, vol. 53, no. 4, pp. 968–979, May 2008.

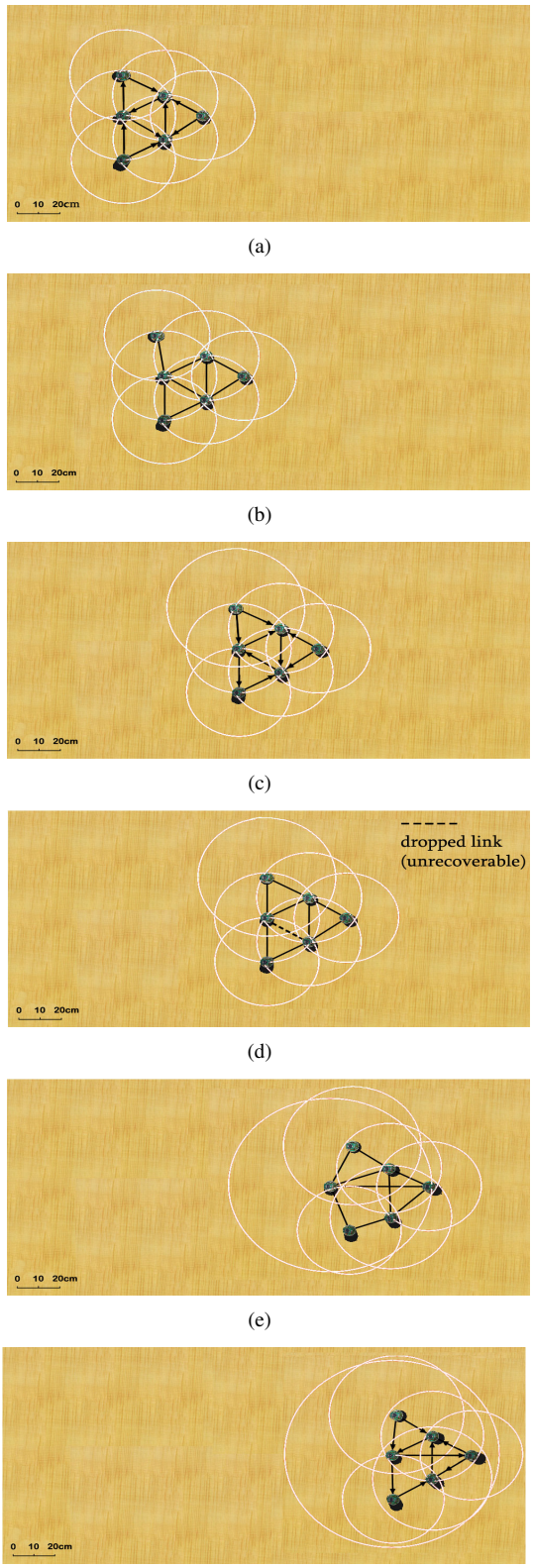
[11] C. Yu, J. M. Hendrickx, B. Fidan, B. Anderson, and V. D. Blondel, "Three and higher dimensional autonomous formations: Rigidity, persistence and structural persistence," *Automatica*, vol. 43, no. 3, pp. 387–402, Mar. 2007.

[12] B. Hendrickson, "Conditions for unique graph realizations," *SIAM J. Comput.*, vol. 21, no. 1, pp. 65–84, 1992.

[13] B. Servatius and H. Servatius, *Rigidity theory and applications*, ser. Fundamental Materials Science series. Plenum Press, 1999.

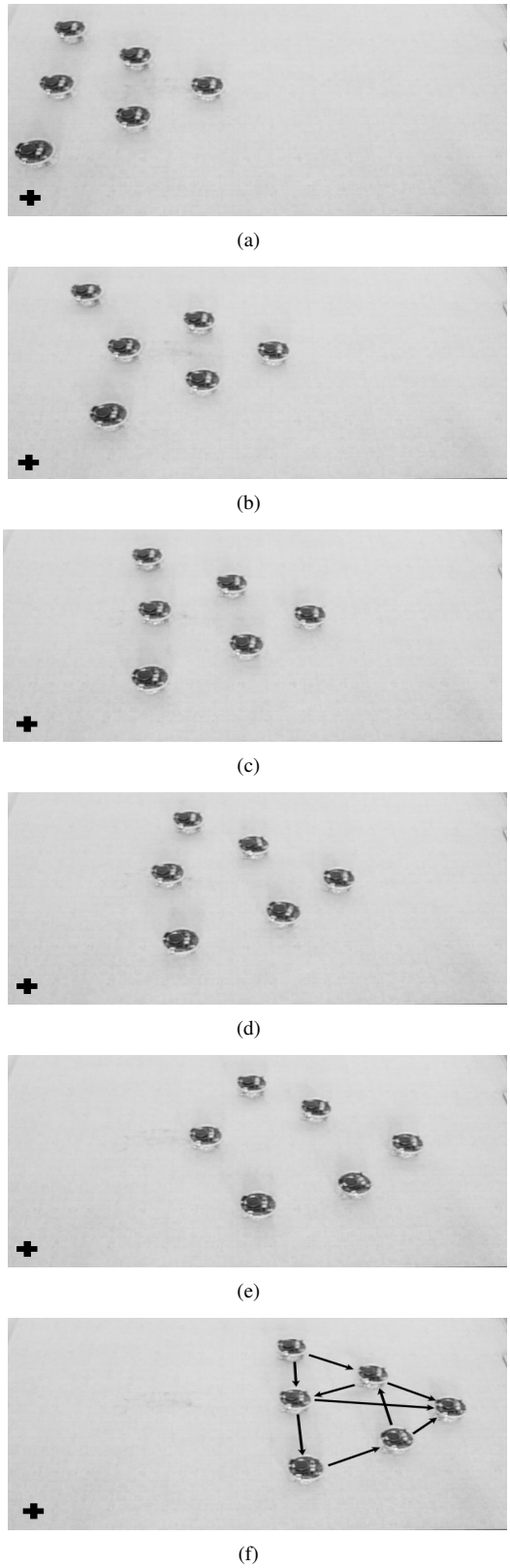
[14] C. Sun, "Parallel sparse orthogonal factorization on distributed memory multiprocessors," *SIAM Journal on Scientific Computing*, vol. 17, no. 3, pp. 666–685, May 1996.

[15] A. Lee and I. Streinu, "Pebble game algorithms and sparse graphs," *Discrete Mathematics*, vol. 308, no. 8, pp. 1425 – 1437, 2008.



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

Fig. 2. Simulation results of a 6-robot system under formation control and persistence control at time instances. (a) at t=0s, the robot system maneuvers in a triangular shape; (b) at t=7s, one link is dropped; (c) at t=30s, robot system backed up the original formation; (d) at t=36s, one link is dropped between robots 2 and 4, rigidity/persistence is lost; (e) at t=54s, the topology is rigid; (f) at t=67s. a persistent framework is shown.



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

Fig. 3. Experimental results of a 6-robot system under formation control and persistence control. (a) at t=0, the 6 E-puck robots maneuver in a triangle-shape formation; (b) the triangular formation has altered; (c) the triangle-shape formation is backed up; (d) the formation of the system at certain link-drop(s); (e) the formation of the system at certain link-drop(s); (f) the system again reaches the triangle-shape formation.