MOUNT HOLYOKE COLLEGE

SENIOR THESIS

Persistent Multi-robot Formations with Redundancy

by

Alyxander Burns

Supervised by

Prof. Audrey St. John

Presented to the faculty of Mount Holyoke College in partial fulfillment of the requirements for the degree of Bachelor of Arts with Honors

May 2017

Abstract

For tasks such as collective transport, a multi-robot formation must preserve its global shape in order to prevent damage to the carried object. We focus on *persistent leader-follower formations*, which maintain local distance constraints in order to preserve the global shape of the formation. In this theoretical model of *persistence theory*, robots are modeled as vertices and dependencies between robots are represented as directed edges. To allow persistent formations to maintain persistence after mechanical failure, we incorporate *redundancy* into the existing theoretical model. We define *persistence circuits* to be persistent formations which become *minimally persistent* after the loss of any edge, present one method for creating these formations, and prove that persistent leader-follower formations cannot be redundantly persistent. For persistent leader-follower formations, we focus on a more restricted notion of redundancy and present 3 methods for constructing these formations. Finally, we present simulation results for 3 multi-robot formations simulated using Webots to evaluate behavior between formations with and without redundancy.

Acknowledgements

First and foremost, I would like to thank my thesis adviser, Audrey St. John, for her unwavering support, knowledge, and encouragement. I don't know where I would be without you. Working with you has truly been a pleasure and an honor.

I would like to thank the faculty and staff of the Mount Holyoke Computer Science department for their words of advice and encouragement over the years. You all inspire me to do my best work and for that I am eternally grateful.

To my wonderful family and friends, thank you for always being there for me. I could not have done this without you.

Contents

List of Figures

Chapter 1

Introduction

Consider the task of transporting the wing of an aircraft during assembly. Wings are large and heavy enough that humans cannot carry them unassisted. In addition, because of a wing's delicate nature, it cannot be held tightly with clamps or robotic arms. The simple problem of moving an object from point A to point B therefore becomes more complex - how should the wing be transported, without being dented or dropped and without the use of clamps or grips? One solution is to use multi-robot formations engaged in *collective transport*.

Multi-robot formations are groups of robots which are organized into a particular structure to collectively complete a task. This means that they are often able to accomplish tasks which could not be done, or so easily so, by a single robot. In the case of collective transport, defined as the transportation of an object by a group of robots which are not physically attached to each other, what one robot could not carry on its own, the group can move together. However, simply using any multi-robot formation does not sufficiently solve the problem. If the global shape of the formation changes too much as the formation moves, the carried item may be dropped or damaged.

In this thesis, we focus specifically on a subclass of multi-robot formations known as *persistent leader-follower formations* [8]. *Persistence* captures the notion that the formation should retain its structure as it moves, thus eliminating the problem of dropping or damaging a carried object due to changes in global structure. A *leader-follower formation* is organized such that there is a leader robot, which could be teleoperated by people or operated autonomously by an artificial intelligence, and follower robots which use sensors to analyze the position of other robots in the formation and adjust to follow a set of instructions called *constraints* which define dependence relationships between them [14].

A real-life analog of a persistent leader-follower formation is a skein of geese in flight. When the leader goose (the one at the front of the V) changes the heading of the group, the other geese observe this change, notice how it affects the positions of other geese, and then adjust accordingly. In Figure 1.1, a skein is drawn with a persistent leader-follower formation superimposed. Each arrow in the diagram has a fixed length called a *distance constraint*. Collectively, the arrows directed away from a goose represent the distance constraints that goose is charged with maintaining and to whom those distances are to be kept. One could imagine that the follower geese notice when a distance constraint has been violated and then take action to satisfy that constraint and maintain the formation's global shape.

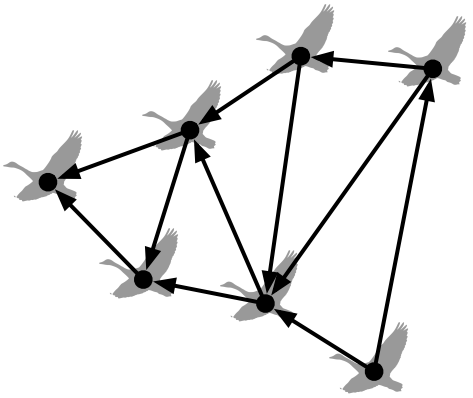


FIGURE 1.1: A skein of geese with a possible persistent leader-follower formation superimposed.

Incorporating *redundancy* to make systems robust to mechanical malfunction is an important consideration for robotics and other fields of engineering. If a sensor were to fail while a formation was engaged in collective transport, the formation should be able to continue and complete its assigned task– without damaging or dropping the carried item. However, current work on persistence theory does not explicitly address redundancy so, in this thesis, we expanded the existing persistence theory to incorporate redundancy.

1.1 Contributions

The main research question we address is: How can we construct persistent multi-robot formations which are able to retain persistence after the failure of sensors used to maintain constraints? The natural place to start is by defining *redundant persistence* which, in the same way that a *rigidity circuit* remains rigid after the loss of any one edge, retain persistence after the loss of any one edge in the formation. We present an algorithm for creating redundantly persistent formations and then prove that persistent leader-follower formations cannot also be redundantly persistent. Because we wanted

to focus specifically on leader-follower formations, we define a more restricted notion of redundancy. We present three methods for constructing persistent leader-follower formations with redundancy: derivation from rigidity circuits in Section 3.2, building larger formations by merging smaller ones in Section 3.3, and recursive construction through trilateration graphs in Section 3.4.

We provide results from experiments simulating the behavior of formations constructed using two of our construction approaches, validating that these formations are robust to sensor failure. Chapter 4 contains more specific information regarding the setup, results, and analysis of our simulation experiments. In particular, Section 4.1 contains information about the formations and robots we used, the path they followed, how the robots followed that path, and how we obtained minimally persistent formations from those containing redundancy. Section 4.2 contains information on what data we gathered, how we analyzed those data, and an analysis of the results.

1.2 Related Work

Mostly closely related to our work is the approach of [18], which utilizes leader-follower formations whose members maintain distance constraints for the application of collective transport. However, in [18] the robots maintain distances to the object they are carrying instead of to each other. This approach cannot be directly applied to the transport of delicate objects because these robots must tightly grasp the object they are carrying as to not drop it.

We note that our persistent formations necessarily are *decentralized* or *distributed*, meaning that each formation member handles its own decision-making as defined in [14]. Other work which includes distributed leader-follower formations are [17] and [13]. Some other approaches to collective transport which use distributed groups of robots (but not specifically persistent or leader-follower formations) are [1, 10, 15, 16]. In [6] and [5], distributed formations are utilized for collective transport, but focus specifically on control and path planning.

Persistence theory was introduced by Hendrickx et al in [7–9] and is closely related to rigidity theory. For further reading on rigidity theory, refer to [4]. Modeling robots as points in the plane allow us to use the rigidity theory model known as 2D *bar-and-joint* (as discussed in [19]). We provide the full details for two of the construction techniques in [2] for constructing persistent formations with redundancy. The third method utilizes theorems regarding trilateration graphs and rigidity [3].

1.3 Thesis Structure

Chapter 2 provides background for understanding the contributions and results of this thesis, covering foundations in graph theory, rigidity theory, and persistence theory. Theoretical and experimental results are contained in Chapter 3 and Chapter 4, respectively. Chapter 5 concludes with future directions.

Chapter 2

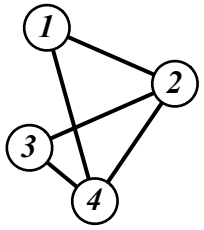
Background

The contributions of this thesis build upon persistence theory as introduced by Hendrickx et al in [8]. In this chapter, we cover the relevant basics of graph theory (Section 2.1), rigidity theory (Section 2.2), and persistence theory (Section 2.3) required for presenting our results. We also fix notation that will be used throughout this thesis.

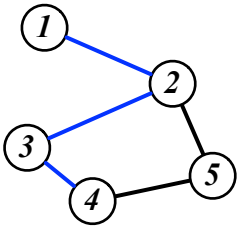
2.1 Graph Theory

In this thesis, we work with both *undirected* and *directed graphs*.

Formally, an *undirected graph* $G = (V, E)$ is a collection of vertices and edges where $V = [1..n]$ is the set of vertices and E is the set of edges represented by unordered pairs of vertices. We denote the edge between vertex i and j by ij and fix notation to require $i < j$. See Figure 2.1 for examples of undirected graphs.



(a) An undirected graph with $V = \{1, 2, 3, 4\}$ and $E = \{12, 14, 23, 24, 34\}$.

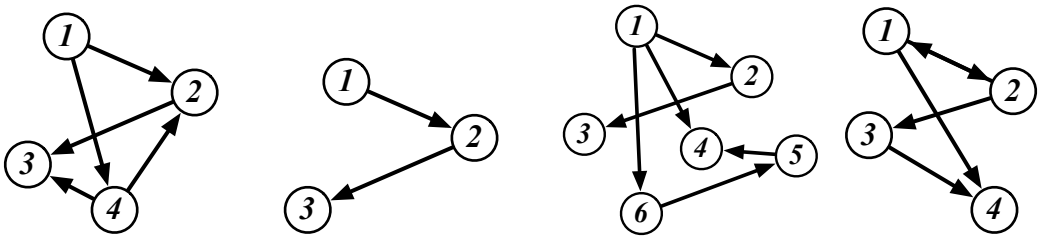


(b) An undirected graph with a path between vertex 1 and 4 highlighted in blue.

FIGURE 2.1: Examples of undirected graphs.

A directed graph $H = (V, A)$ is a set of vertices and edges where $V = [1..n]$ is the set of vertices and A is the set of directed edges represented as ordered pairs of vertices. We call the edge from i to j $\vec{ij}$. We call i the *source* and j the *target*. Notice that for

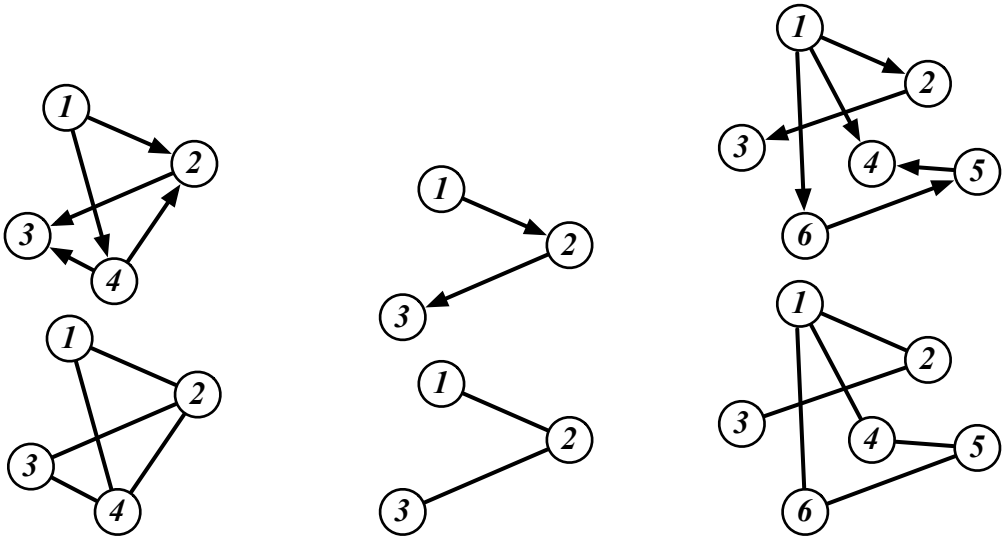
directed graphs, the edge $\overrightarrow{ij} \neq \overrightarrow{ji}$. We depict directed edges as arrows where the tail of the arrow is at the source and the tip is at the target; see Figure 2.2.



(a) (b) (c) (d)

FIGURE 2.2: Examples of directed graphs.

Every directed graph has an associated *underlying undirected graph*. Formally, if $H = (V, A)$ is a directed graph, then its underlying undirected graph is $G = (V, E)$, where an edge $ij \in E$ if $\overrightarrow{ij} \in A$ or $\overrightarrow{ji} \in A$. Figure 2.3 provides examples of directed graphs and their underlying undirected graphs.



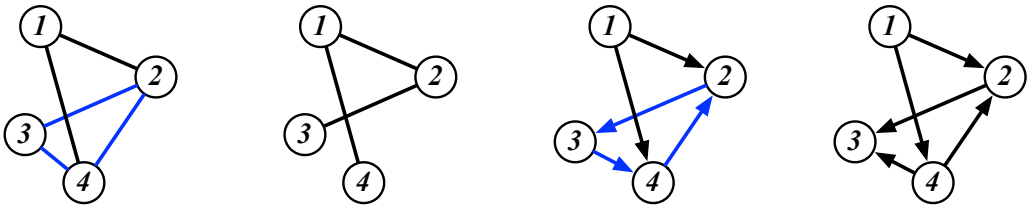
(a) (b) (c)

FIGURE 2.3: Examples of directed graphs with their associated underlying undirected graphs below.

The *degree* of a vertex v is the number of edges incident to v. In Figure 2.1(a), vertex 4 has a degree of 3 and vertex 1 has a degree of 2. For directed graphs, the *out-degree* of v is the number of edges whose source is v. For example, the vertex 1 in the graph shown in Figure 2.3(c) has out-degree 3, while vertex 4 has out-degree 0.

We say that there is a *path* between two vertices i and j if there exists a sequence of vertices $v_1, ..., v_p$ such that $v_1 = i, v_p = j$ and $v_k v_{k+1} \in E$ for all $1 \leq k < p \in \mathbb{R}$. In the

graph in Figure 2.1(b), the edges of a path between vertex 1 and 4 are highlighted in blue. A *cycle* is a path where $i = j$. For directed graphs, note that the edge direction must be maintained. For example, the graphs shown in Figure 2.4(a) and 2.4(c) contain cycles marked in blue, while the graphs shown in Figure 2.4(b) and 2.4(d) contain no cycles. We call a graph without a cycle *acyclic* or *cycle-free*.



(a) An undirected graph with a cycle marked in blue.

(b) An acyclic undirected graph.

(c) A directed graph with a cycle marked in blue.

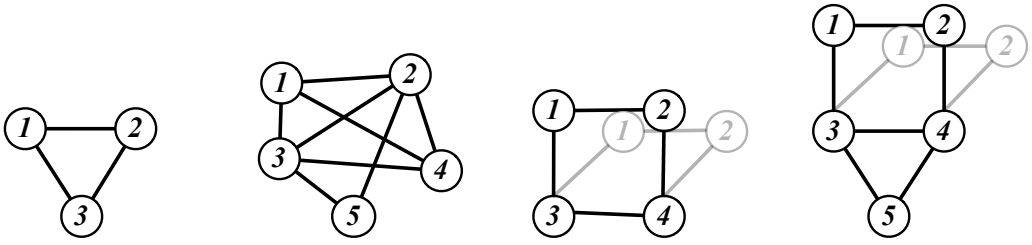
(d) An acyclic directed graph.

FIGURE 2.4: Examples of undirected and directed graphs with and without cycles.

2.2 Rigidity Theory

We now present the core concepts of rigidity theory as a foundation for persistence theory. For further reading on rigidity theory, we refer the reader to [4].

Conceptually, a structure is *rigid* if it does not bend or flex and is *flexible* otherwise. While there are many theoretical models within rigidity theory, we focus on the model known as *2D bar and joint*– where structures are composed of revolute joints connected by fixed-length bars. The 2D bar and joint structures shown in Figure 2.5(a) and 2.5(b) are rigid but the one in Figure 2.5(c) is flexible – the dashed shape shows a possible flexion. If some portion of a structure is rigid, but not all of it, we say that it contains a *rigid component*. In Figure 2.5(d) we see a structure made up of both Figure 2.5(a) and 2.5(c). Even if a structure contains a rigid component, it is still considered to be flexible.



(a) A 3-joint rigid structure.

(b) A 5-joint rigid structure.

(c) A flexible structure with a gray flexion.

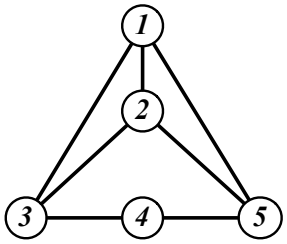
(d) A flexible structure with a rigid component.

FIGURE 2.5: Rigid and flexible structures in the 2D plane.

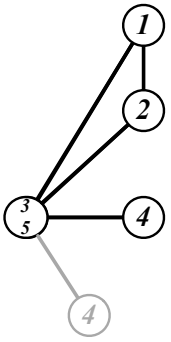
Let $G = (V, E)$ be an undirected graph. We call $\mathbf{p} \in (\mathbb{R}^2)^n$ an *embedding* in the Euclidean plane, where $\mathbf{p_i}$ is the position of vertex i. Two embeddings $\mathbf{p}$ and $\mathbf{q}$ are considered to be *congruent* if $||\mathbf{p_i} - \mathbf{p_j}|| = ||\mathbf{q_i} - \mathbf{q_j}||$ for all points i and $j \in V$. We call the pair $H = (G, \mathbf{p})$ a *framework*. Given a framework $H = (G, \mathbf{p})$, we can extract a distance function $d : E \to \mathbb{R}$ such that $d(ij) = ||\mathbf{p_i} - \mathbf{p_j}||$.

Definition 2.1. If all $\mathbf{q}$ in some neighborhood of $\mathbf{p}$ satisfying the distance function d are congruent, we say the framework H is *locally rigid* and *flexible* otherwise.

Definition 2.1 specifies local behavior to account for scenarios such as the following example, shown in Figure 2.6. Consider the two embeddings $\mathbf{p}$ and $\mathbf{q}$ shown in Figure 2.6(a) and 2.6(b), respectively. While $\mathbf{p}$ is associated with a locally rigid framework, $\mathbf{q}$ is not because vertex 4 is able to rotate as shown in grey in Figure 2.6(b). In this thesis, we are not concerned with cases like this as we expect our formations to maintain local behavior. Therefore, we omit the word "locally" and instead refer to frameworks as either being *rigid* or *flexible*.



(a) A locally rigid embedding of G.



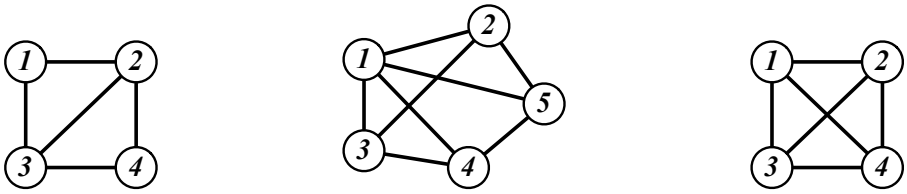
(b) A flexible embedding of G where vertices 3 and 5 are embedded at the same point.

FIGURE 2.6: Two embeddings of the same graph display different local behavior.

We say that a graph is *minimally rigid* if it is (1) rigid and (2) removal of any edge would render it flexible. The framework in Figure 2.7(a) is minimally rigid while the one shown in Figure 2.7(b) is not because edges 13, 23, or 34 could be removed and the framework would remain rigid. There is a special type of rigid graph known as a *rigidity circuit* which is formally defined as follows:

Definition 2.2. Let $G = (V, E)$ be an undirected graph. We say that G is a *rigidity circuit* if it is rigid and every sub-graph obtained by removing one edge from E is minimally rigid.

The graph shown in Figure 2.7(c) is an example of a rigidity circuit.



(a) A minimally rigid framework.

(b) A rigid framework which is not minimally rigid.

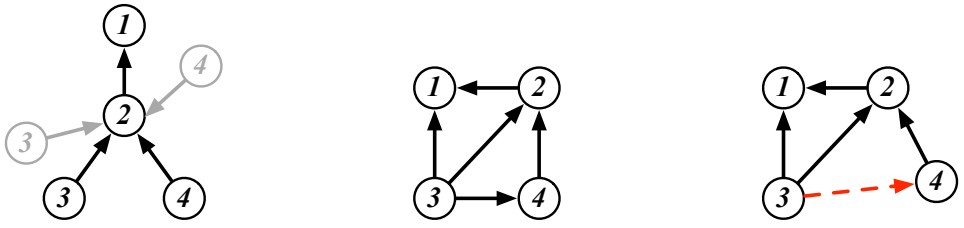
(c) A rigidity circuit, a special case of graph which is rigid, but not minimally rigid.

FIGURE 2.7: Examples of (minimally) rigid frameworks.

2.3 Persistence Theory

A straight forward application of applying rigidity theory to multi-robot formations would require both robots to maintain the distance constraint associated with the undirected edge between them. Since this would lead to increased communication and/or sensing costs, we instead choose to use the related notion of *persistence*, wherein the edges are directed. This thesis project builds upon persistence theory first introduced by Hendrickx et al in [7]. For containment, we present the relevant background, definitions, and results from [8] in this section.

Within the persistence model, imagine that each vertex represents a robot and each outgoing edge a distance constraint which it is responsible for satisfying. Conceptually, a formation is persistent if (1) every robot can find a location which satisfies all of its distance constraints and (2) the underlying, undirected graph is rigid. The frameworks shown in Figure 2.9 are persistent while the frameworks in Figure 2.8 are not. Figure 2.8(a) fails condition (1) because its underlying, undirected graph is not rigid, while Figure 2.8(b) fails condition (2) because it is possible to be in a position where not all of the distance constraints can be maintained simultaneously. In this example, it is possible for vertex 4 to move to a position where 3 cannot satisfy all 3 of its constraints at the same time, as shown in Figure 2.8(c).

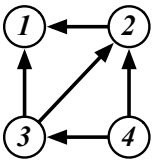


(a) A framework with a flexible underlying graph is not persistent.

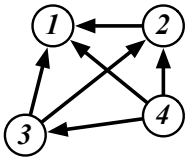
(b) A framework where it is not always possible for all vertices to satisfy the distance function d.

(c) A framework which displays how 2.8(b) is not persistent.

FIGURE 2.8: Examples of frameworks which are not persistent.



(a) A minimally persistent framework.



(b) A persistent framework which is not minimally persistent.

FIGURE 2.9: Examples of persistent frameworks.

Let $H = (V, A)$ be a directed graph and $\mathbf{p}$ be an embedding of H. We define a *formation* to be the pair $(H, \mathbf{p})$. Given the underlying undirected graph of H, we can extract the distance function d.

Let $\mathbf{q}$ be another embedding of H. If $\vec{ij} \in A$ and $||\mathbf{q_i} - \mathbf{q_j}|| = d(\vec{ij})$ then we say that the edge $\vec{ij}$ is *active*. In other words, if the distance constraint associated with edge $\vec{ij}$ is being satisfied, then it is considered active. We say that a vertex $i \in V$ in position $\mathbf{q_i}$ is *fitting* if there is no other position $\mathbf{q_i}* \in \mathbb{R}^2$ such that the set of active (outgoing) edges is larger, or, more formally, if $\{\vec{ij} \in A : ||\mathbf{q_i} - \mathbf{q_j}|| = d(\vec{ij})\} \subset \{\vec{ij} \in A : ||\mathbf{q_i}* - \mathbf{q_j}|| = d(\vec{ij})\}$. We say that an embedding is a *fitting embedding* if all of the vertices in V are fitting.

In its application to robotics, we can think of the position of a robot in a formation as fitting if there is no other location which the robot could move to which would allow it to strictly increase the number of constraints it is satisfying.

We can now state the technical definition of persistence:

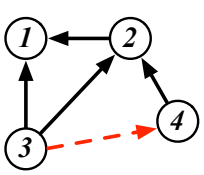
Definition 2.3. Let $(H, \mathbf{p})$ be a formation. If all fitting embeddings $\mathbf{q}$ in some neighborhood of $\mathbf{p}$ are congruent to $\mathbf{p}$, then the formation is *persistent*.

Note that the definition of persistence requires the embeddings of a formation to be both fitting and congruent. While it seems like a formation which has a fitting embedding also imply that its embeddings must be congruent, that is not necessarily the case.

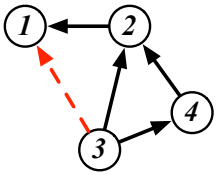
Consider the example shown in Figure 2.10. Let the embedding shown in Figure 2.10(a) be $\mathbf{p}$. Here, we see that vertex 3 has an out-degree of 3 and therefore must try to satisfy three different constraints. Assume that all of vertex 3's edges cannot be active simultaneously. In other words, assume that by satisfying two of these constraints it must necessarily violate the third. There are two other possible positions for 3 (shown in Figure 2.10(b) and 2.10(c)) such that 2 of its 3 edges are active.

Because the definition of fitting requires that a different position must strictly increase the number of active outgoing edges, we consider the position of 3 in $\mathbf{p}$ to be fitting and therefore we consider $\mathbf{p}$ to be a fitting embedding. However, the embeddings in

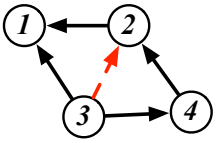
Figure 2.10(b) and 2.10(c) are also fitting embeddings by the same logic. As Figure 2.10(a), 2.10(b), and 2.10(c) are fitting embeddings of the same framework but are clearly not congruent, we can see that fitting is a necessary, but not sufficient, condition for persistence.



(a) In this position, vertex 3 satisfies its distance constraints to vertices 1 and 2 but not 4.



(b) In this position, vertex 3 satisfies its distance constraints to vertices 2 and 4 but not 1.



(c) In this position, vertex 3 satisfies its distance constraints to vertices 1 and 4 but not 2.

FIGURE 2.10: Examples of fitting embeddings which are not congruent.

Alternately, we can check for the persistence of a formation a different way – by checking for rigidity within the underlying undirected graph:

Theorem 2.4 (Theorem 3 of [8]). *A graph is persistent if and only if the underlying undirected graph of every subgraph obtained by removing out-edges from vertices with out-degree > 2 until all vertices have out-degree ≤ 2 is generically*[1] *rigid.*

In addition, graphs which are known to be persistent remain persistent when unnecessary edges are removed by the following Proposition.

Proposition 2.5 (Proposition 1 of [8]). *A persistent graph remains persistent after deletion of any edge $\overrightarrow{ij}$ for which the out-degree is ≥ 3.*

Analogously to rigidity, a formation is *minimally persistent* if it is (1) persistent and (2) removal of any edge will cause the resulting framework to no longer be persistent. See Figure 2.9 for examples of graphs which are and are not minimally persistent.

By the following theorem, if a graph is rigid, the degrees of its vertices determine if it is minimally persistent.

Theorem 2.6 (Theorem 4 of [8]). *A rigid graph (with more than one vertex) is minimally persistent if and only if one of the following two conditions is satisfied.*

1. *Three vertices have an out-degree 1 and all the others have an out-degree 2.*
2. *One vertex has an out-degree 0, one vertex has an out-degree 1, and all the others have an out-degree 2.*

[1] Genericity is outside of the scope of this thesis.

Persistent Leader-Follower Formations

This project focuses on a specific subset of persistent formations known as *persistent leader-follower* formations. Conceptually, a persistent leader-follower formation is a persistent formation where one vertex, known as the *leader*, maintains no constraints, and a second vertex, known as the *co-leader*, maintains a single constraint to the leader. All other vertices in the formation maintain 2 or more constraints and are known as *followers*. Persistent leader-follower formations are formally defined as follows:

Definition 2.7. Let $F = (H, \mathbf{p})$ be a persistent formation. We say that F is a persistent leader-follower formation if:

1. There exists some vertex $v_L \in V$ with out-degree 0,
2. There exists some vertex $v_C \in V$ with out-degree 1, with $\overrightarrow{v_C v_L} \in E$, and
3. The out-degree of all vertices $v \neq v_L, v_C \in V$ is ≥ 2.

Figures 2.9(a) and 2.9(b) depict examples of persistent leader-follower formations – vertex 1 is the leader, 2 is the co-leader, and 3 and 4 are followers. Both leader and co-leader vertices are necessary to control the movement of the formation; one point can determine the translation of a formation within the xy plane, while a second point is necessary to determine the rotation.

Acyclicity

As discussed in Chapter 4, we choose to focus specifically on acyclic persistent leader-follower formations for simulation considerations. The persistence of acyclic formations can be analyzed differently than that of formations which may contain cycles. For example, we are able to determine the persistence of an acyclic graph using its vertex degrees, as follows:

Theorem 2.8 (Theorem 5 of [8])**.** *A cycle-free graph having more than one vertex is persistent if and only if:*

1. *One vertex (called the leader) has an out-degree 0.*
2. *One vertex (called the first follower) has an out-degree 1 and the corresponding edge is incident to the leader.*
3. *Every other vertex has an out-degree larger or equal to 2.*

The proof of this theorem is inductive, requiring:

Proposition 2.9 (Proposition 5 of [8])**.** *A graph obtained by adding one vertex to a graph G = (V, E) and at least two edges leaving this vertex is persistent if and only if G is persistent.*

Chapter 3

Constructing Persistent Formations with Redundancy

In this chapter, we will build upon the background in Chapter 2 by expanding persistence theory to incorporate redundancy into the theoretical model. We begin by defining *persistence circuits*, and present an algorithm for creating them (Section 3.1) and then prove that persistence circuits cannot be leader-follower formations. Therefore, we present three different methods for the construction of persistent multi-robot formations with a more restricted notion of redundancy: derivation from a rigidity circuit (Section 3.2), merging existing persistent formations with redundancy (Section 3.3), and recursive construction through vertex addition (Section 3.4).

3.1 Persistence Circuits

The notion of minimal redundancy in rigidity theory is captured by rigidity circuits: any edge can be removed to obtain a minimally rigid graph. Therefore, we begin by defining the analogous concept for persistence.

Definition 3.1. A persistent formation is a *persistence circuit* if all subgraphs obtained by removing a single edge are minimally persistent.

We propose the following algorithm for the construction of *persistence circuits*. Note that this algorithm uses the $(2,3)$-pebble game algorithm of [11, 12] as a subroutine. The pebble game determines rigidity and outputs a directed graph. If the input graph is rigid, the output is a minimally persistent directed graph (whose vertices have at most out-degree 2). Pebbles are used as a mechanism for controlling the out-degree of

a vertex. Specifically, for the (2,3)-pebble game used in the following algorithms, the following result holds.

Lemma 3.2 (Invariant (1) of Lemma 10 of [12])**.** *For any vertex v, the sum of the number of pebbles on v and the out-degree of v is 2.*

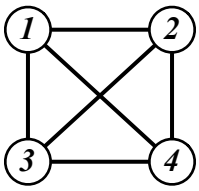
Algorithm 1 Constructing a persistence circuit from a rigidity circuit.

Input: a rigidity circuit $G = (V, E)$
Output: a persistence circuit.

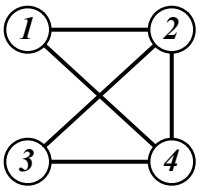
1. Remove any edge $e = ij \in E$.
2. Play the $(2, 3)$-pebble game on $G' = (V, E \setminus \{e\})$ to obtain a directed graph H.
3. If i has 0 pebbles, use pebble collection moves on H to collect a single pebble on i.
4. Output the resulting directed graph with the additional edge $\overrightarrow{ij}$.

Proof of correctness of Algorithm 1. Let $H = (V, A)$ be the output of Algorithm 1. By construction and Lemma 3.2, H has either 1 vertex with out-degree 0 or 2 vertices with out-degree 1; all other vertices have out-degree 2. We show that H is a persistence circuit. Choose some edge $\overrightarrow{ij} \in A$ to remove and call the resulting graph J. J either has (1) 3 vertices with out-degree 1, and all other vertices with out-degree 2; or (2) 1 vertex with out-degree 0, 1 vertex with out-degree 1, and all others with out-degree 2. By Theorem 2.6, J is minimally persistent if its underlying, undirected graph of J, J', is rigid. Because the underlying, undirected graph G of H is a rigidity circuit and J' is obtained by removing one edge from G, J' must also be rigid and so J is minimally persistent. □

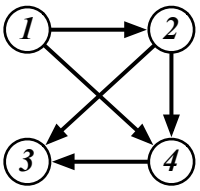
We now work through an example of this technique with a graph with 4 vertices. All steps of this example are shown in Figure 3.1. We begin with the K_4 rigidity circuit and choose to remove edge 13. We play the pebble game to obtain the directed graph shown in Figure 3.1(c). Note that because this orientation has 2 pebbles on vertex 3 already, we do not need to reorient the graph. We add the edge we removed back in as $\overrightarrow{31}$ and output the final graph. For the purpose of this example and to display an alternate persistence circuit of the same K_4 rigidity circuit, we have reoriented the graph output by the pebble game so that 2 pebbles are on 1. This new orientation is shown in Figure 3.1(e). In this example, the removed edge is added back in as $\overrightarrow{13}$ and is shown in Figure 3.1(f).



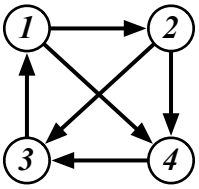
(a) Example input to Algorithm 1.



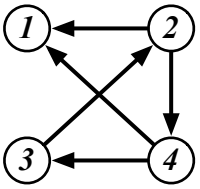
(b) Graph after removing edge 13 in Step 1.



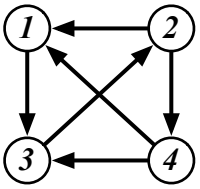
(c) Graph output by (2,3)-pebble game, oriented such that vertex 3 has out-degree 0.



(d) Edge 13 is added back in as $\overrightarrow{31}$ and output as the final persistence circuit.



(e) Alternately, the graph in Figure 3.1(c) could be oriented so that vertex 1 has out-degree 0.



(f) 13 is added back as $\overrightarrow{13}$.

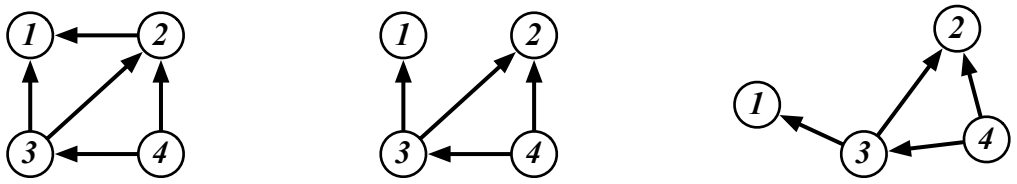
FIGURE 3.1: Example of Algorithm 1 with input of rigidity circuit K_4.

Unfortunately, the formations created using Algorithm 1 are never leader-follower formations, which are the type of formation needed for our simulation approach. In fact, we can show the following:

Lemma 3.3. *A persistent leader-follower formation cannot be a persistence circuit.*

Proof. Let $F = (H, \mathbf{p})$ be a persistent leader-follower formation, where $H = (V, A)$ is a directed graph with n vertices and $\mathbf{p} \in (\mathbb{R}^2)^n$ an embedding from which we extract the distance function d; note that this implies that $\mathbf{p}$ is a fitting embedding of the formation. Let the vertex $v_L \in V$ be the leader and $v_C \in V$ be the co-leader. By the definition of a leader-follower formation, $\overrightarrow{v_C v_L} \in A$. Assume, for the purpose of contradiction, that F is a persistence circuit. If F is a persistence circuit, then it must be true that the removal of any one edge $\overrightarrow{v_i v_j} \in A$ does not result in a loss of persistence. We remove edge $\overrightarrow{v_C v_L} \in A$ and let F' be the resulting formation. We construct a second embedding $\mathbf{p}'$. Without loss of generality, let $\mathbf{p_{v_C}} = (0, 0)$ and $\mathbf{p'_{v_C}} = (1, 1)$. For all other vertices $v \neq v_C \in V$, let $p'_v = p_v$. Note that because the out-degree of v_C is 0, any position $\mathbf{p'_{v_C}} \in \mathbb{R}^2$ is fitting for v_C. By construction, since p_v was fitting for v, p'_v is still fitting for all $v \neq v_C \in V$. Thus, $\mathbf{p}'$ is fitting for F'. However, $||\mathbf{p_{v_L}} - \mathbf{p_{v_C}}|| \neq ||\mathbf{p'_{v_L}} - \mathbf{p'_{v_C}}||$ and so $\mathbf{p}$ and $\mathbf{p}'$ are not congruent. Thus, F' is not persistent, resulting in the contradiction. □

Consider the example shown in Figure 3.2. Figure 3.2(a) shows a persistent leader-follower formation. After the removal of the edge $\overrightarrow{21}$, we obtain the formation shown in Figure 3.2(b). Because vertex 2 is maintaining no constraints, it is free to move anywhere



(a) Example of a persistent leader-follower formation.

(b) The leader-follower formation with the edge from co-leader to leader removed.

(c) Flexion of the framework possible with edge 21 removed.

FIGURE 3.2: Example of proof of Lemma 3.3.

in $\mathbb{R}^2$. Visually, we can see in Figure 3.2(c) that it is possible for the formation to change shape, and thus confirm that the formation is not persistent after the removal of the edge $\overrightarrow{21}$. Therefore, it cannot be a persistent leader-follower formation.

3.2 Derivation from Rigidity Circuits

The simulation of persistent formations which are not leader-follower formations is outside of the scope of this project and so we present a class of redundant formations with a more restricted definition of redundancy which can occur in persistent leader-follower formations.

Proposition 3.4 (Proposition 1 of [2])**.** *For every rigidity circuit G, there exists some persistent, leader-follower orientation of it.*

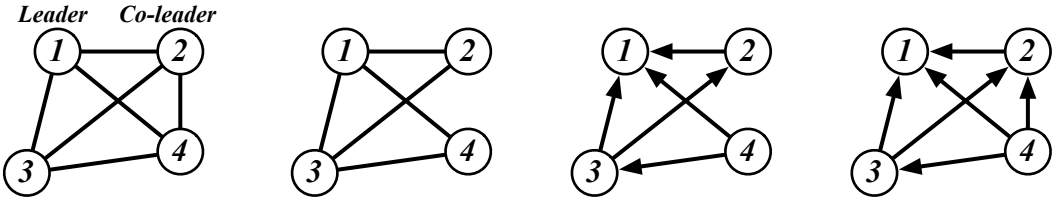
The correctness of this proposition is proved by construction in [2] through the algorithm that follows.

Algorithm 2 [Algorithm 1 of [2]], Constructing a persistent leader-follower formation from a rigidity circuit.

Input: a rigidity circuit $G = (V, E)$, a desired leader vertex v_L and a desired co-leader vertex v_C incident to v_L.
Output: a persistent leader-follower formation.

1. Remove any edge $e = ij \neq v_C v_L \in E$.
2. Play the $(2, 3)$-pebble game on $G' = (V, E \setminus \{e\})$ to obtain a directed graph H.
3. Use pebble collection moves on H to collect 2 pebbles on v_L and one pebble on v_C.
4. Output the resulting directed graph with the additional edge $\overrightarrow{ij}$, if $i \neq v_C, v_L$, or $\overrightarrow{ji}$ otherwise.

For example, consider the rigidity circuit K_4 as shown in Figure 3.3(a). We designate vertex 1 to be v_L and vertex 2 to be v_C. We choose to remove edge 24 to obtain the minimally rigid graph in Figure 3.3(b). After playing the pebble game, we obtain the directed persistent graph shown in Figure 3.3(c). As vertex 2 is v_C, we add back in the edge we removed as $\overrightarrow{42}$ and obtain the final persistent formation with redundancy shown in Figure 3.3(d).



(a) Example input for Algorithm 2 with 4 vertices.

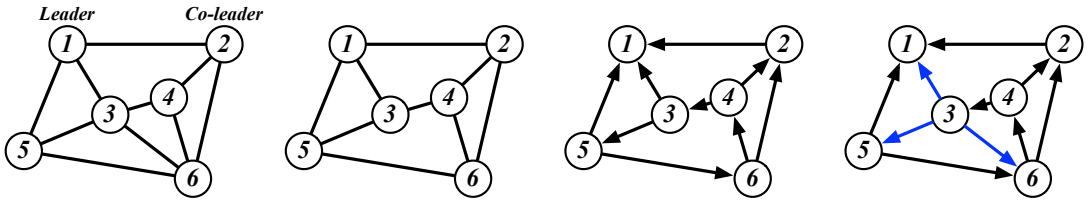
(b) Graph after removing edge 24 in step 1 of Algorithm 2.

(c) Graph after playing the pebble game in step 2.

(d) Algorithm 2 output.

FIGURE 3.3: Example of Algorithm 2 with input of the rigidity circuit K_4.

We also demonstrate this technique with a second rigidity circuit as shown in Figure 3.4(a). Again, we designate vertex 1 to be v_L and vertex 2 to be v_C. We remove edge 36 to obtain the minimally rigid graph in Figure 3.4(b). After playing the pebble game, we obtain the graph shown in Figure 3.4(c). We add edge 36 back in as $\overrightarrow{36}$ to obtain the final persistent directed graph with redundancy as shown in Figure 3.4(d).



(a) Example input for Algorithm 2 with 6 vertices.

(b) Graph after removing edge 36 in step 1 of Algorithm 2.

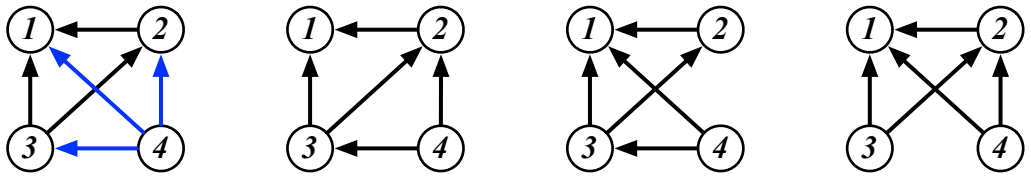
(c) Graph after playing the pebble game in step 2.

(d) Algorithm 2 output with redundant edges colored in blue.

FIGURE 3.4: Example of Algorithm 2 with input of the rigidity circuit with 6 vertices.

Formations created using this technique contain a single vertex with out-degree 3. We call the set of edges leaving this vertex *redundant edges*, as the removal of any one of these edges will make the formation minimally persistent. For example, Figure 3.5(a) shows a persistent formation with redundancy with the set of redundant edges highlighted in blue. The minimally persistent formations created by removing one edge from that set are shown in 3.5(b), 3.5(c), and 3.5(d).

The redundancy in formations created using this method is necessarily restricted as there is only 1 set of 3 edges which can be safely removed without a loss of persistence. The probability that the loss of a random edge will not effect the persistence of the formation



(a) Persistent formation with redundancy whose redundant edges are colored blue.

(b) Minimally persistent graph after removing edge $\overrightarrow{41}$ from graph in Figure 3.5(a).

(c) Minimally persistent graph after removing edge $\overrightarrow{42}$ from graph in Figure 3.5(a).

(d) Minimally persistent graph after removing edge $\overrightarrow{43}$ from graph in Figure 3.5(a).

FIGURE 3.5: Example of a persistent formation with redundancy.

is $3/m$, where m is the number of edges in the formation. For small formations such as the formation shown in Figure 3.3(d), the probability of preserving persistence is good – here, there is a 50% chance that the loss of an edge will not affect the persistence of the formation. Note, however, that as m grows, the probability that the loss of an edge will not affect the persistence of the formation approaches 0.

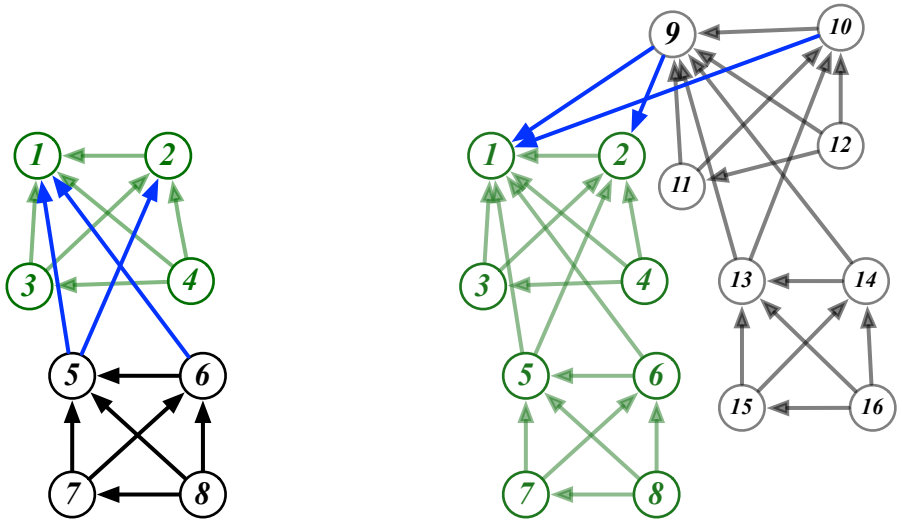
3.3 Merging Formations

As a method for creating persistent leader-follower formations with a larger number of redundant edges, we propose the following in [2]:

Proposition 3.5 (Proposition 3 of [9]). *Let $H = (U, E)$ and $I = (V, F)$ be persistent leader-follower graphs with leaders $u_L \in U$, $v_L \in V$ and co-leaders $u_C \in U$ and $v_C \in V$. Then the graph $U \cup V$, $E \cup F \cup \{e = \overrightarrow{v_L u_i}, f = \overrightarrow{v_L u_j}, g = \overrightarrow{v_C u_k}\}$ is a persistent leader-follower graph, where $u_i, u_j, u_k \in U$ and $|\{u_i, u_j, u_k\}| > 1$.*

Using this method, we are able to treat existing formations as "seeds" and join them together to create larger formations. If redundant sets of edges are evident in the seed formations, they are preserved in the larger merged formation. Refer to [2] for a formal proof of correctness.

For example, we could join two copies of the persistent formation with redundancy shown in Figure 3.3(d) by adding edges $\overrightarrow{51}$, $\overrightarrow{52}$, and $\overrightarrow{61}$ (highlighted in blue). This new, merged formation is the one shown in Figure 3.6(a). Additionally, this technique can be used recursively. This time, we begin with two copies of the formation obtained in Figure 3.6(a). We choose to add edges $\overrightarrow{91}$, $\overrightarrow{92}$, and $\overrightarrow{101}$ (highlighted in blue) to obtain final formation in Figure 3.6(b). Notice that this formation is four of the formation shown in Figure 3.3(d).



(a) Formation obtained by merging two persistent formations with redundancy from Figure 3.3(d).

(b) Formation obtained by merging two persistent formations with redundancy from Figure 3.6(a).

FIGURE 3.6: Examples of merged persistent formations with redundancy using Proposition 3.5.

With this construction technique, we calculate the probability that the loss of an edge does not affect the persistence of the overall formation as $3w/m$, where w is the number of vertices with out-degree 3 and m is the total number of vertices in the graph. For the formation in Figure 3.6(a), we see that the probability that the loss of an edge will not result in a loss of persistence is 40%, and $\approx$ 36% for Figure 3.6(b).

3.4 Construction using Trilateration

In this section, we propose the application of trilateration graphs to build persistent formations with redundancy. We build upon the work of [3] by extending the application of trilateration graphs to persistence instead of rigidity.

Here, we construct persistent leader-follower formations by using directed edges instead of undirected edges. We propose the following algorithm for the construction of persistent leader-follower formations with redundancy:

Algorithm 3 Constructing a persistent leader-follower formation by recursive addition

Input: A set of vertices $V_0 = [1..n]$, a leader vertex $v_L \in V_0$ and a co-leader vertex $v_C \in V_0$.

Output: a persistent leader-follower formation.

1. Initialize a set of vertices $V = \{v_L, v_C\}$ and a set of edges $E = \{\overrightarrow{v_C v_L}\}$
2. For each vertex $v_i \in V_0 \neq v_L, v_C$:
   (a) If $V = \{v_L, v_C\}$, add edge $\overrightarrow{v_i v_L}$ and $\overrightarrow{v_i v_C}$ to E
   (b) Else randomly select 3 unique vertices $v_j, v_k, v_l \in V$ and add edges $\overrightarrow{v_i v_j}$, $\overrightarrow{v_i v_k}$, and $\overrightarrow{v_i v_l}$ to E.
   (c) Add v_i to V
3. Output the directed graph $G = (V, E)$

Note that the running time of this algorithm is O(n), where n is the number of vertices in the graph. We now prove the correctness of this construction technique:

Proof. We proceed by induction on n, the number of vertices.

Base case: When $n = 2$, the output of Algorithm 3 is $G = (V, E)$, where the set of vertices is $V = \{v_L, v_C\}$ and the set of edges is $E = \{\overrightarrow{v_C v_L}\}$. Because G is acyclic, G is persistent by Theorem 2.8. In addition, G satisfies the definition of a persistent leader-follower formation.

Induction step: Assume that the output of Algorithm 3 is a persistent leader-follower formation for $k \geq 2$ vertices. We show that the output $G' = (V', E')$ is correct for $k+1$ vertices.

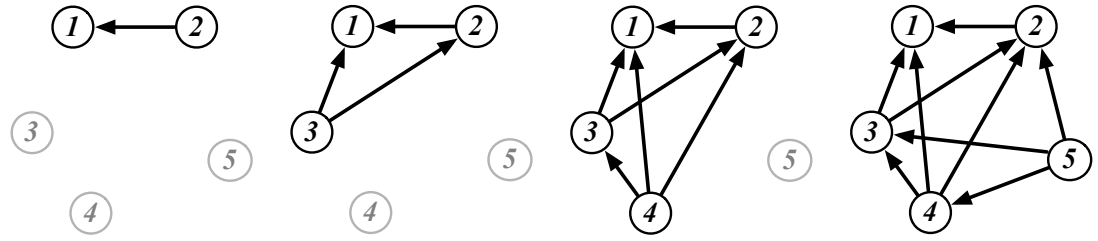
Let $G = (V, E)$ be the constructed graph before the last iteration of Step 2. Then $V' = V \cup \{v_{k+1}\}$.

If $k = 2$, then $E' = E \cup \{\overrightarrow{v_3 v_1}, \overrightarrow{v_3 v_2}\}$.

Otherwise, if $k > 2$, $E' = E \cup \{\overrightarrow{v_{k+1} v_i}, \overrightarrow{v_{k+1} v_j}, \overrightarrow{v_{k+1} v_l}\}$, where $v_i, v_j, v_l \in V$.

By induction, G is a persistent leader-follower formation. By Proposition 2.9, we know that G' must be persistent because G was persistent. Additionally, we know that G' satisfies the definition of a persistent leader-follower formation.

Let vertex $v_i \in V$ have out-degree 3 and let E_{v_i} be the set of edges out of v_i. Let E_r be the set of all E_{v_i} for all $v_i \in V$. Let G'' be the graph obtained after removing at most 1 edge from each $E_{v_i} \in E_r$. By Proposition 2.5, G'' is persistent. Therefore, G' contains a set of redundant edges $E_r \in E'$. □



(a) Begin by adding vertex 1 and 2 to V and edge $\overrightarrow{21}$ to E.

(b) Graph after adding vertex 3 to V and edges $\overrightarrow{31}$ and $\overrightarrow{32}$ to E.

(c) Graph after adding vertex 4 to V and edges $\overrightarrow{41}$, $\overrightarrow{42}$, and $\overrightarrow{43}$ to E.

(d) Final graph, after adding vertex 5 to V and edges $\overrightarrow{52}$, $\overrightarrow{53}$, and $\overrightarrow{54}$ to E.

FIGURE 3.7: Construction of a persistent formation with redundancy using Algorithm 3.

We now work through an example formation with 5 vertices using the above algorithm. Diagrams of each step are shown in Figure 3.7. We begin by designating vertex 1 to be v_L and 2 to be v_C, respectively. We construct two new sets $V = \{1, 2\}$ and $E = \{\overrightarrow{21}\}$. We select $v_i = 3 \in V_0$, add edges $\overrightarrow{31}$ and $\overrightarrow{32}$ to E and add vertex 3 to V. We then select $v_i = 4 \in V_0$. Because $V \neq \{v_L, v_C\}$, we randomly select 3 vertices from V: $v_j = 1$, $v_k = 2$ and $v_l = 3$. We add edges $\overrightarrow{41}$, $\overrightarrow{42}$ and $\overrightarrow{43}$ to E and vertex 4 to V. Finally, we select $v_i = 5 \in V_0$ and select 3 vertices at random from V: $v_j = 4$, $v_k = 2$, and $v_l = 3$. We add edges $\overrightarrow{54}$, $\overrightarrow{52}$, and $\overrightarrow{53}$ and add vertex 5 to V. Finally, we output the graph $G = (V, E)$.

With this construction method, we see that the proportion of the number of edges to the number of vertices is much higher than in other construction techniques. Notice that in formations with greater than 3 vertices, all edges are redundant except three: $v_C v_L$ and the two edges added in step 1 of Algorithm 3. We therefore determine that the probability that the loss of an edge will not affect the persistence of the formation at $1 - (3/m)$, where m is the number of edges. Even for small formations, such as the one built in Figure 3.7 the probability of that persistence will not be affected is good – in this example, it is $\approx 67\%$. Notice that as m grows large, the probability that persistence will not be affected approaches 1.

We note that because this construction technique is purely additive and relatively simple, it could be used to rebuild a persistent formation after the loss of a vertex in a group of robots which are all able to maintain the same number of constraints. In addition, we note that because of the large number of edges, the total sensing cost associated with formations created using this method is relatively high. We leave further investigation and simulation of this construction as future work.

Chapter 4

Simulating Persistent Formations with Redundancy

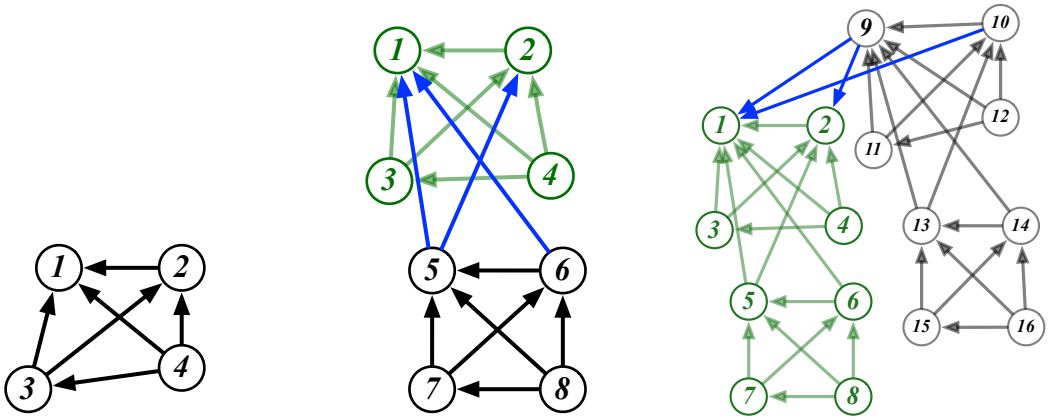
To verify and evaluate the performance of our theoretical results, we simulated formations constructed with two of the three construction algorithms. In addition, we utilize the simulations in order to identify interesting research questions. In this chapter, we discuss the set-up, results, and analysis of our simulation experiments. Our goal was to analyze the behavior of the formations by specifying a repeatable path for the leader. The followers then used on-board sensors to determine the distance between themselves and other robots in the group and then moved to a position which satisfied the pre-determined distance constraints they were trying to maintain. We repeated this experiment using persistent formations with and without redundancy in order to quantitatively compare their performance.

In Section 4.1, we discuss more specific information pertaining to the robots, formations, and paths used in the simulations. Section 4.2 contains information on the data collected and the analysis of those data.

4.1 Set-up

Formations

We simulated three different formations whose graphs are shown in Figure 4.1(a), 4.1(b), and 4.1(c). We number the formations 1, 2, and 3, where 1 is the smallest formation with 4 vertices (Figure 4.1(a)) and 3 is the largest formation with 16 vertices (Figure 4.1(c)). We note that for all three formations, vertex 1 has been denoted the leader and vertex 2, the co-leader.



(a) Formation 1. (b) Formation 2. (c) Formation 3.

FIGURE 4.1: Formations used in simulation experiments.

Notice that the persistent leader-follower orientation of the K_4 rigidity circuit serves as a "seed" formation for all of our simulated formations. Formation 1 is the persistent orientation of the K_4 rigidity circuit itself, while formation 2 contains 2 K_4 seed formations, and formation 3 contains 4.

For this thesis project, we did not simulate formations with different seeds. Because of this, we do not know how the shape of the seed formation affects the performance of the formation overall and leave this as an avenue for future investigation.

Robots

For our experiments, the type of robot we chose to use was the iRobot Create. The iRobot Create is a circular differential drive robot; it has a circular body with two parallel, separately powered wheels on the left and right side of the robot and two unpowered wheels on the front and back for stability. A diagram of the robot and the placement of its wheels is shown in Figure 4.2. Because the movement of differential drive robots is bounded by the direction of their two powered wheels, they can only move directly forward and backward. To travel in any other direction, the robot must either first rotate and then move along a straight path or travel along a curved path. While the center of rotation for a differential drive robot depends on the speed of its two wheels, the robots in our experiments were programmed to rotate about their center.

Because omnidirectional robots can move directly in any direction without first rotating, we hypothesize that an omnidirectional robot would be able to reach a fitting location faster than a differential drive robot. We leave the replication of these experiments using omnidirectional robots as future work.

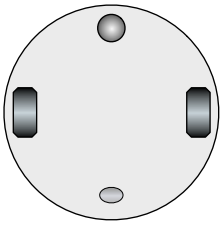


FIGURE 4.2: Diagram of the location of the wheels on a iRobot Create. The two driven wheels are on the left and right. The two wheels on the top and bottom are used for stability.

Followers

In our experiments, each follower vertex is associated with a single Create. We name this robot after its associated vertex. For example, vertex 4 is associated with robot 4. At the center of rotation of each follower robot there is:

1. a GPS, used for tracking the movement of each robot throughout the simulation for data collection;
2. an emitter, which broadcasts an omnidirectional signal on a channel unique to each robot; and
3. one receiver for each edge out of that vertex "listening" to the channel of the edge's target.

For example, the robot associated with vertex 3 in Figure 4.1(a), has two receivers at its center: a receiver listening to channel 1 and a receiver listening to channel 2. Each receiver is able to determine both the strength and direction of the received signal.

Leaders & Co-leaders

We treated the leader and co-leader vertices differently from the follower vertices. Instead of being two separate robots, we chose to attach both the leader and co-leader vertices to the same robot, which we will refer to as the *leader robot*. See Figure 4.3 for a diagram of the placement of vertices on the leader robot. We made this choice to simplify the control of the formation and let the leader robot control both the translation and rotation of the formation as a whole. We can set the distance constraint between the co-leader and leader vertices to be the fixed distance between them on the robot and thus leave the underlying graph intact.

We place a GPS and emitter at the location of the leader and co-leader vertices on the leader robot to be used for the same purposes as those on the follower robots. Note that

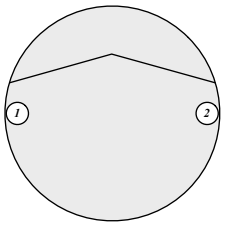


FIGURE 4.3: Location of the leader vertex (vertex 1) and the co-leader vertex (vertex 2) on the leader robot.

because the leader robot has a fixed size, the distance constraint associated with the $\overrightarrow{v_C v_L}$ edge will always be satisfied. Thus, we did not include a receiver at the location of the co-leader vertex.

Paths

The path that the leader robot followed is the S-shaped path shown in Figure 4.4. This path was chosen because it has a combination of straight sections, a right turn, and a left turn. The radius of curvature is different for the left and right turn as to test how the formation performed on different types of curves. While we did not test our formations on other paths, we hypothesize that the type of path matters to the performance of the formation as a curve whose radius of curvature is small is harder to travel along than a curve with a larger radius of curvature.

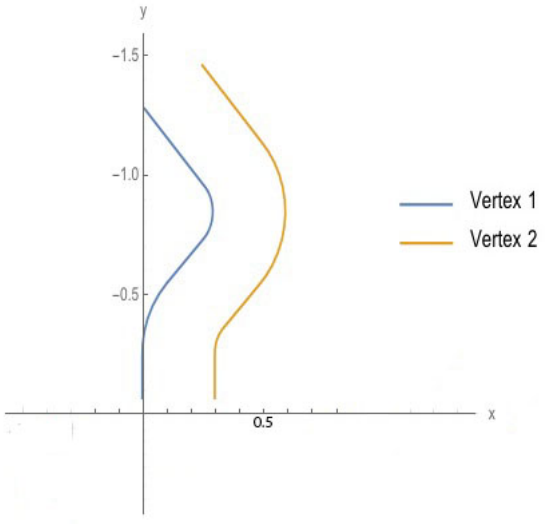


FIGURE 4.4: Path followed by the leader vertex (vertex 1) and the co-leader vertex (vertex 2) shown in blue and yellow respectively.

The leader robot is hard-coded to follow this path so that precisely the same path can be replicated for each experiment. Each simulation experiment can be broken down

into 2 alternating movement phases: (1) leader movement and (2) follower movement. During the leader movement phase, the leader robot moves a short distance along the pre-determined path and then signals that it has reached its final position by changing the message it is emitting. At this point, the follower movement phase begins. During this phase, each follower waits until it receives a "go" signal from all of the emitters which it is listening to. Because we have enforced that our formations are acyclic, we know that there must exist some ordering of the robots such that the position of each robot only depends on those which are earlier in the order. If there were a cycle in the graph, then robots in that cycle would never move because they are each waiting for a "go" from another robot in the cycle. See Figure 4.5 for an example of how a cycle in a graph would cause this kind of deadlock. Because our aim was to specifically evaluate the theoretical model, we chose to use acyclic formations to simplify the control of the formation, as formations with cycles prompt a more challenging problem of developing a control scheme that converges.

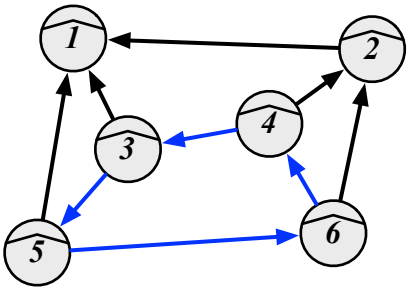


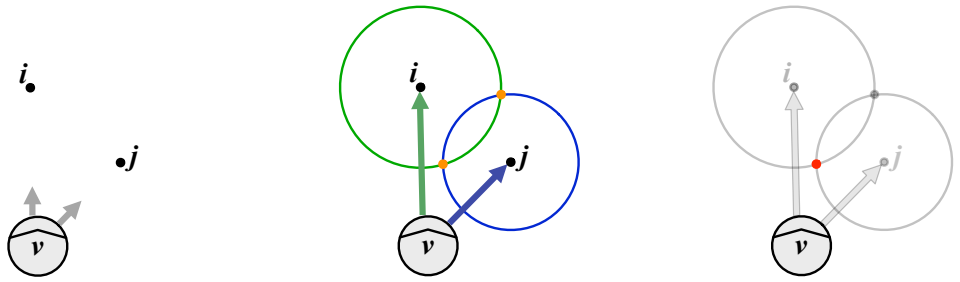
FIGURE 4.5: Notice the cycle marked in blue. Robots 3, 4, 5, and 6 are deadlocked because each is waiting for another in the cycle to reach its final destination.

Once this condition has been met, the follower robot first calculates the location which would satisfy its constraints. We begin with the pseudocode algorithm for determining that location for a robot v which is associated with a vertex with out-degree 2 to edges with targets i and j:

1. Using the signal strength and direction received from the emitters on robot i and j, calculate the location of i and j on a local coordinate system with the origin at the center of v.
2. Calculate the intersection point(s) of two circles centered at the local coordinates of i and j whose radii are the length of the distance constraints associated with edges $\overrightarrow{vi}$ and $\overrightarrow{vj}$.
3. Select the closer intersection point.

See Figure 4.6 for a diagram of this procedure. Note that the numerical equations for this procedure are rather complicated and can be found in Appendix A. For followers

with an out-degree of 3, the above algorithm is repeated three times, once with each pair of 2 edges. The final 3 points are averaged together to obtain the location which this follower robot will attempt to move to. We note that theoretically, the 3 optimal points determined by the procedure will be exactly the same, but because of error in sensing and satisfaction of constraints, we average the points to get the best approximation possible.



(a) Begin with two unit vectors, each pointed in the direction of emitters at the location of vertices i and j.

(b) Create a circle centered at the locations i and j with radii equal to the d(vi) and d(vj) respectively and find the intersection points (marked here in orange).

(c) Select the closer intersection point (marked here in red).

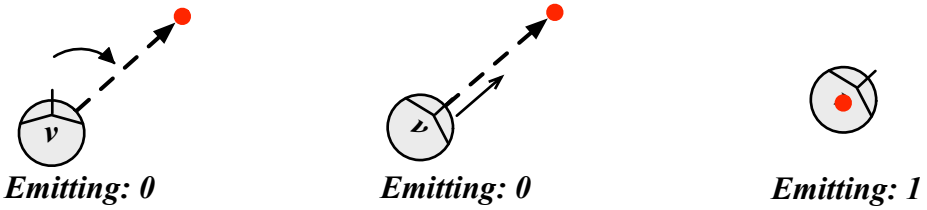
FIGURE 4.6: Example of how follower robots determine the next fitting location.

Once a robot has determined its target location, it begins the process of moving to that location, as follows:

1. Calculate the vector between current location and target location.
2. Rotate until robot is parallel to the vector (within a 0.02 m margin).
3. If the target point is in front, move forward.
4. If the target point is behind, move backward.
5. Stop when within 0.01 m of target location.
6. Emit signal confirming robot has reached its target location.

See Figure 4.7 for a diagram of each step of this pseudocode.

The goal of our experiment was to compare the performance of formations with redundant edges to the formations obtained by removing a selection of those redundant edges. In our experiments, we chose to simulate a mechanical failure by randomly "disabling" one receiver on each robot with an out-degree of 3. For each of our formations with redundancy, we ran our experiment once with all redundant edges intact to obtain a *control*. Then, we randomly removed an edge from each set of redundant edges to



(a) In Step 1 and 2, the robot calculates the vector between itself and the location which satisfies its constraints and then rotates so its heading is parallel to that vector.

(b) In Step 3, the robot moves forward to the point.

(c) Finally, once the robot has reached that location, it begins emitting a confirmation signal.

FIGURE 4.7: Example of how a follower robot reaches the nearest location which satisfies all of its constraints.

obtain a random minimally persistent formation and simulated that formation. We repeated this 20 times for each formation, randomly removing different redundant edges at the start of each simulation.

4.2 Results & Analysis

During each experiment, we recorded the location of each robot every 64 milliseconds using the GPS units attached to them. At the end of each experiment, the lists of locations were output as text files and stored for further analysis.

In order to compare the performance of each formation, we define a metric for comparison. Because we aim to analyze the persistence of the formations, we chose to focus on the distance between where a robot was at a particular time in the simulation and where it should have been if the global shape of the formation did not change at all. For a robot i, we call the distance between the predicted location $p*_i$ and the realized location p_i the *error*. See Figure 4.8 for a diagram of the error.

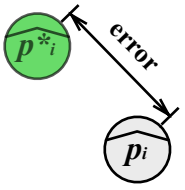


FIGURE 4.8: Let p_i be the actual location of the robot in simulation and let $p*_i$ be the predicted location. The distance between p_i and $p*_i$ is the error.

Because our simulation experiments alternated between two movement phases (leader movement and follower movement) and robots wait for a "go" signal from all of the robots upon which their location depends before they move during those phases, there

are portions of the simulation where the formation is distorted while robots move to their next location or wait for the correct signal. For the analysis of the formation, we filter out these steps and instead focus only on samples from the time step immediately before the leader phase begins again - a moment where we assume that all follower robots have moved to the location which satisfies their constraints best.

Because the leader robot (containing both the leader and co-leader vertices) was following hard-coded instructions, we did not analyze its behavior. Instead, we used the locations of the leader and co-leader vertices to define translation and rotation matrices which, when applied to the initial locations of each follower robot, gave a list of points where each robot should have been if the formation had not changed shape at all. Using that list, we calculated the error for each robot over every selected time stamp. We found the maximum, minimum, and mean error for each follower robot during each simulation and then found the mean of maximums, mean of minimums, and mean of means for all robots in a particular formation over all simulation experiments. We mark the formations with redundant edges as the *control* and the formations which had edges randomly removed the *random*. The results of our data analysis are shown in Table 4.1.

	Mean of minimums	Mean of means	Mean of maximums
Formation 1 control	0.0132058 ± 0.004	0.0173324 ± 0.006	0.021459 ± 0.010
Formation 1 random	0.0131554 ± 0.004	0.0179166 ± 0.006	0.0235468 ± 0.010
Formation 2 control	0.00826846 ± 0.004	0.0810221 ± 0.011	0.191657 ± 0.034
Formation 2 random	0.00769588 ± 0.004	0.0872681 ± 0.012	0.261483 ± 0.104
Formation 3 control	0.0087836 ± 0.004	0.0766168 ± 0.051	0.362899 ± 0.361
Formation 3 random	0.00835823 ± 0.004	0.0608315 ± 0.045	0.197745 ± 0.162

TABLE 4.1: Simulation data: Mean values with standard deviation across all simulations with a constraint accuracy threshold of 0.02m.[1]

Notice that for each formation, the mean of means error is roughly the same for the control as the random. The theoretical model predicts that there should be no difference between the control and random formations, and while we do see a slight difference between the two, it is within a reasonable margin of error. Therefore, our results validate the behavior predicted by the theoretical model.

[1]For a comparison of scale, the smallest constraint satisfied by a robot is 1.0m.

Chapter 5

Future Work

The main research question addressed in this thesis is: How can we construct persistent multi-robot formations which are able to retain persistence after the failure of sensors used to maintain constraints? By approaching the problem in two different ways, first by extending the existing persistence theory to incorporate redundancy and then validating the theory through simulation, we were able to identify future research directions in both theory and simulation.

We defined a new class of formations called persistence circuits which become minimally persistent after the loss of any edge and presented an algorithm for their construction. However, we also showed that persistence circuits cannot be leader-follower formations. Because we wanted to focus specifically on leader-follower formations for this project, we defined a more restricted notion of redundancy that is compatible with them and proposed three different methods for their construction. Through tracking robots in simulation experiments we compared the performance of formations with redundant edges and those without. After analyzing the data we collected, we showed that there was no significant difference between the execution of the two groups of formations.

This project begins to lay down the foundation for the simulation of persistent multi-robot formations and the inclusion of redundancy in those formations, but it leaves many opportunities for further investigation and research. For example, the simulation experiments that we ran for this thesis revealed that there are factors which are not accounted for by the pure theoretical model such as the path followed, collisions, sensor error, robot type, and noise. We do not yet understand what effects these other factors have or what other external factors exist that we haven't even identified yet.

We can continue to use simulation as a tool for evaluating behavior and identifying other research problems, but modify the set-up in order to focus on specific aspects of

the problem. For example, we can choose to focus on simulating formations which are based upon seed formations other than the persistent orientation of K_4 to see what affect that has on the performance of the formation overall, or simulate formations constructed using the trilateration construction technique discussed in Section 3.4.

Appendix A

Java code for determining points which satisfy two distance constraints

```
double lStrength = rec1.getSignalStrength();
double rStrength = rec2.getSignalStrength();

double lDistance = java.lang.Math.sqrt(1/lStrength);
double rDistance = java.lang.Math.sqrt(1/rStrength);

double[] lDirection = rec1.getEmitterDirection();
double[] rDirection = rec2.getEmitterDirection();

// (a,c) is the local coordinate of the emitter received by rec1. Likewise, (b,d) is the local coordinate of the emitter received by rec2.
double a = lDirection[0] * lDistance;
double b = rDirection[0] * rDistance;

double c = lDirection[2] * lDistance;
double d = rDirection[2] * rDistance;

//length1 and length2 are the lengths of the distance constraint to be kept from the robots at (a,c) and (b,d) respectively.
double r = length1;
double s = length2;

//The following is calculating the two intersection points of two circles: (1) centered at (a,c) with radius r, and (2) centered at (b,d) with radius s
double f = Math.pow(c, 2) - 2*c*d + Math.pow(d, 2) - Math.pow(r, 2) - Math.pow(s, 2);
double g = Math.pow(c-d, 2);
```

```
double term1 = Math.pow(a, 3) - Math.pow(a, 2)*b + Math.pow(b, 3) + b * (Math.pow(c,2)
- 2*c*d + Math.pow(d, 2) + Math.pow(r,2) - Math.pow(s,2));

double term2 = a * (-Math.pow(b,2) + Math.pow(c,2) - 2*c*d + Math.pow(d,2) -
Math.pow(r, 2) + Math.pow(s, 2));

double term3 = Math.pow(a,4) - (4 * Math.pow(a, 3) * b) + Math.pow(b,4) + Math.pow(c,4)
- (4 * Math.pow(c, 3) * d) + (6 * Math.pow(c,2) * Math.pow(d, 2)) - ( 4 * c *
Math.pow(d, 3)) + Math.pow(d, 4) - (2 * Math.pow(c, 2) * Math.pow(r, 2)) +(4 *
c * d * Math.pow(r ,2)) - (2 * Math.pow(d, 2) * Math.pow(r, 2)) + Math.pow(r, 4) - (2
* Math.pow(c,2) * Math.pow(s, 2)) + (4 * c * d * Math.pow(s, 2)) - (2 * Math.pow(d,
2) * Math.pow(s, 2)) - (2 * Math.pow(r, 2) * Math.pow(s, 2)) + Math.pow(s, 4) +(2 *
Math.pow(b,2) * f) - (4 * a * b * (Math.pow(b,2) + f)) + (2 * Math.pow(a, 2) * (3 *
Math.pow(b, 2) + f));

double term4 = 2 * (Math.pow(a, 2) - 2*a*b + Math.pow(b, 2) + g);
term3 = Math.sqrt((-1/g) * term3);

// (x1, y1) and (x2, y2) are the points where both length1 and length2 are satisfied.
double x1 = (term1 + term2 - g * term3)/term4;
double x2 = (term1 + term2 + g * term3)/term4;

double y1Numerator = (Math.pow(r,2) - Math.pow(s, 2)) - (Math.pow(a,2)- Math.pow(b,
2)) - (Math.pow(c, 2) - Math.pow(d, 2)) + 2 * x1 * ( a - b);
double y2Numerator = (Math.pow(r,2) - Math.pow(s, 2)) - (Math.pow(a,2)- Math.pow(b,
2)) - (Math.pow(c, 2) - Math.pow(d, 2)) + 2 * x2 * ( a - b);

double denominator = -2 * (c - d);

double y1 = y1Numerator / denominator;
double y2 = y2Numerator / denominator;
```

Bibliography

[1] Berman, S., Lindsey, Q., Sakar, M. S., Kumar, V., and Pratt, S. C. (2011). Experimental study and modeling of group retrieval in ants as an approach to collective transport in swarm robotic systems. *Proceedings of the IEEE*, 99(9):1470–1481.

[2] Burns, A., Schulze, B., and St. John, A. (In Press). Persistent multi-robot formations with redundancy. In *Proceedings of DARS 2016*, Springer Tracts in Advanced Robotics.

[3] Eren, T., Goldenberg, O. K., Whiteley, W., Yang, Y. R., Morse, A. S., Anderson, B. D. O., and Belhumeur, P. N. (2004). Rigidity, computation, and randomization in network localization. In *IEEE INFOCOM 2004*, volume 4, pages 2673–2684 vol.4.

[4] Graver, J., Servatius, B., and Servatius, H. (1993). *Combinatorial rigidity.* Graduate Studies in Mathematics. Providence, RI.

[5] Habibi, G., Kingston, Z., Xie, W., Jellins, M., and McLurkin, J. (2015). Distributed centroid estimation and motion controllers for collective transport by multi-robot systems. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1282–1288.

[6] Habibi, G., Xie, W., Jellins, M., and McLurkin, J. (2014). Distributed path planning for collective transport using homogeneous multi-robot systems. In *DARS*.

[7] Hendrickx, J. M., Anderson, B. D. O., and Blondel, V. D. (2005). Rigidity and persistence of directed graphs. In *In Proceedings of the 44th IEEE Conference on Decision and Control*, pages 2176–2181.

[8] Hendrickx, J. M., Anderson, B. D. O., Delvenne, J.-C., and Blondel, V. D. (2007). Directed graphs for the analysis of rigidity and persistence in autonomous agent systems. *International Journal of Robust and Nonlinear Control*, 17(10-11):960–981.

[9] Hendrickx, J. M., Yu, C., Fidan, B., and Anderson, B. D. O. (2008). Rigidity and persistence for ensuring shape maintenance in multiagent meta-formations. *Asian Journal of control (special issue on Collective Behavior and Control of Multi-Agent Systems)*, 10(2):131–143.

[10] Hichri, B., Adouane, L., Fauroux, J.-C., Mezouar, Y., and Doroftei, I. (2016). *Cooperative Mobile Robot Control Architecture for Lifting and Transportation of Any Shape Payload*, pages 177–191. Springer Japan, Tokyo.

[11] Jacobs, D. and Hendrickson, B. (1997). An Algorithm for Two-Dimensional Rigidity Percolation: The Pebble Game. *Journal of Computational Physics*, 137(CP975809):346 – 365.

[12] Lee, A. and Streinu, I. (2008). Pebble game algorithms and sparse graphs. *Discrete Mathematics*, 308(8):1425–1437.

[13] Loria, A., Dasdemir, J., and Alvarez Jarquin, N. (2016). Leader-follower formation and tracking control of mobile robots along straight paths. *IEEE Transactions on Control Systems Technology*, 24(2):727 – 732.

[14] Mas, I. and Kitts, C. A. Centralized and decentralized multi-robot control methods using the cluster space control framework. In *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics, Montreal, QC, Canada, 2010*, pages 115–122.

[15] Mellinger, D., Shomin, M., Michael, N., and Kumar, V. (2013). *Cooperative Grasping and Transport Using Multiple Quadrotors*, pages 545–558. Springer Berlin Heidelberg, Berlin, Heidelberg.

[16] Sugawara, K., Correll, N., and Reishus, D. (2014). *Object Transportation by Granular Convection Using Swarm Robots*, pages 135–147. Springer Berlin Heidelberg, Berlin, Heidelberg.

[17] Vilca, J., Adouane, L., and Mezouar, Y. (2016). *Adaptive Leader-Follower Formation in Cluttered Environment Using Dynamic Target Reconfiguration*, pages 237–254. Springer Japan, Tokyo.

[18] Wang, Z. and Schwager, M. (2016). *Multi-robot Manipulation Without Communication*, pages 135–149. Springer Japan, Tokyo.

[19] Whiteley, W. (2004). Rigidity and scene analysis. In Goodman, J. and O'Rourke, J., editors, *Handbook of Discrete and Computational Geometry, 2nd ed.*, pages 1315–1254. Chapman & Hall/CRC.