A Brief Survey of Commonly Used Techniques in Robotic Swarms

Bas Metman, *Student, Delft University of Technology,*
Martin Jorn Rogalla, *Student, Delft University of Technology,*
and Sjoerd van Bekhoven, *Student, Delft University of Technology,*

(Draft)

***Abstract*—The abstract goes here. The abstract goes here. The abstract goes here. The abstract goes here. The abstract goes here. The abstract goes here. The abstract goes here. The abstract goes here. The abstract goes here. The abstract goes here. The abstract goes here. The abstract goes here.**

***Index Terms*—Robotic Swarms, Techniques, Survey**

I. Introduction

As robots become smaller and easier to produce, interest for robotic swarms is created. Many new applications for robotic swarms exist creating a strong growth in the field, indicated by a growing amount of paper written about robotics at for instance the AAMAS (International Conference on Autonomous Agents and Multi-Agent Systems). [1] Many different applications and techniques exist in the field of robotic swarms. This paper aims to deliver a concise review of these applications and techniques.

To avoid confusion, some terminology will be defined. Afterwards, in order to emphasize the importance of the connection between the technology and its applications, a top-down approach is used for this survey. Thus, in this paper there will first be decided on the terminology of the research area, after which some applications will be given. In the second part of the paper we will discuss the most used techniques for these applications and the algorithms behind these techniques. At the end a final overview and a discussion will be given.

II. Definitions from Literature

This section will provide some definitions so that there will exist no ambiguity for some terms. Firstly, we wish to define what robotic swarms are. A robotic swarm is a collection of robots. In this review, we will only consider something a swarm when the amount of robots is higher than two and the amount can be scalable; so a swarm of only two robots that doesn't interact with a third robot will not count as a swarm. Furthermore, in this review we will only consider robotic swarms in which each robot is not controlled individually; they should have some form of distributed intelligence. An exception is of course when a swarm of multiple robots is controlled by one control station; this swarm will still have some form of distributed intelligence to function and thus is considered a swarm.

Robotic swarm applications can roughly be characterised by two attributes; they are either *location-based* or *location-free*, or they are either *range-based* and *range-free*. A location-free approach does not exclude a range-free approach and vice-versa; they are two different ways of approaching an application. The definitions of these attributes may be interpreted ambiguously, which is why we will define it here. The definitions are:

- A robotic swarm is *location-free* if the swarm has no knowledge of the boundaries of te location it is in, whether it is provided at the beginning or is actively searched for during the execution of the algorithm.
- A robotic swarm is *location-based* if the swarm has the knowledge of predefined boundaries of the location it operates in, whether provided at the beginning of the execution of the algorithm or if it is actively searched for.
- A robotic swarm is *range-free* if each robot can detect the presence of other nearby robots or obstactles, but does not store or measure the distance towards the other object.
- A robotic swarm is *range-based* if each robot in the swarm keeps track of the exact distance between itself and the other robots in the swarm or obstacles.

III. Definitions and Applications

Robotic swarms can be used for many real-world applications which can be roughly split up in tasks that cover a region, tasks that are to dangerous for human beings, tasks that scale-up or scale-down in time or tasks that require redundancy [2]. In this chapter, for every technique the scope of the technique is defined and a few examples of practical applications will be named. We should mention that we chose a selection of techniques that we think are mostly used in the applications we have looked at. Some of these techniques partly overlap and/or make use of each other, but it is essential to compare them independently to identify the current problems and already found solutions in swarm robotics.

A. Exploration

Exploration is a technique in which a swarm of robots try to fully explore an environment, which is one of the fundamental problems faced in mobile robotics. The main goal is to minimize the overall exploration time while still exploring the whole environment. The main problem faced when trying

to achieve this goal is finding appropriate target points for each individual robot so that they simultaneously explore different regions of the environment. [3]
The exploration technique is found in many robotic swarms techniques, for example in *path-finding*, *collective transport* and *surveillance*. Practical applications that apply the exploration technique are for example rescue missions. [4], [5] In this particular paper, a robotic swarm applying the exploration technique is used to assist navigation for firefighters, used in situations in which their vision is blocked by smoke and obstacles. A last example of an application is cleaning. [6] Here, the exploration technique is used to clean a surface with cleaning robots, that communicate in a swarm, as fast and as efficient as possible. Of course, exploration techniques are used in many more different robotic swarm applications, and is the building block for many different other techniques.

B. Mapping

Mapping is a technique in which a swarm of robots try to make a map of the surrounding unknown environment. The mapping of an unkown environment can be efficiently done with the usage of robotic swarms; this is called collaborative mapping. As with other algorithms, this mapping should be done as fast as possible without losing accuracy. The core of the mapping technique looks very similar to exploration, but an important difference is that in mapping a map should be made and stored. This leads to more communication between robots and more distributed intelligence.

Mapping of an unknown environment is very useful in hazardous and unaccessable environments. [7] In this paper a practical application is mentioned, the detection of hazardous aerosol in a contaminated, confined area. Robotic swarms are used in this work because it is safer to use them. Of course, as mentioned earlier, some techniques overlap, and mapping is no exception to this. The mapping technique uses elements of exploration, dispersion and localization.

In [8] a model is described in which the robots have limited communication range and full knowledge of their location. By continuously looking for the closest undiscovered area in combination with a nearest measure it tends to explore the complete area by staying together and continiously updating each others maps. Some of the techniques used in these types of applications are localization, exploration, dispersion, mapping. [8], [9]

C. Dispersion

Dispersion is a technique in which a swarm of robots try to cover an area as large as possible. Dispersive applications make interactive use of autonomous robots in exploratory settings, and is thus often used in combination with exploration techniques. The robotic swarms are capable of supporting and enhancing operations co-operatively and are optionally coordinated by a single human supervisor. There is one big difference between dispersion and exploration, although at first they seem very much alike. The difference is that in exploration (and mapping) the whole area has to be explored, but in dispersion an area as large as possible should be covered. This means that in dispersion, the ultimate goal is to cover the whole area as fast as possible, but the robotic swarm has a possibility to stop exploring when reaching its maximum coverage.

An example of an application is, among others, planetary exploration. [5], [10] In this example, a swarm of robots tries to explore a part of the planet, but has no intent to explore the whole planet. So, a dedicated dispersion algorithm is used. Other examples of applications include military response and disaster response applications.

D. Localization

Source localization is a technique in which a swarm of robots try to locate a specific point, often used to find sources of disturbance. The main goal of this technique is to find all sources of disturbance as fast as possible. Source localization is used in virtually any other robotic swarms technique, because in most techniques something has to be found, whether it is a source or an object.

Obviously, this means that this technique has many practical applications. A few of these examples will be mentioned here. A first example is chemical plume tracing, in which localization is used to detect clouds of high density chemicals. [11] Another example which looks a lot like chemical plume tracing, is radiation source search. [12] The difference is that in this case the source of leaking radiation is searched for, and not only for high density clouds. A third example is searching for fire, used to assist fire-fighters in their every-day work. [13] As can be seen, this technique can be used to look for all kinds of emission sources, if the robots have the right sensors installed. [14] Although many practical applications can be found, a large amount of the work done in this field is purely theoretical. This is due to the fact that the price of these individual robots is still rather high and thus it is expensive to produce a swarm.

E. Path-finding

Path finding or path planning (in this paper we will use path finding) basically is the basis of every robotic swarm technique, since every distributed algorithm in mobile robotics tries to find the best location to go to from its current location. In this paper we consider path finding as the problem to find the optimal collision free path from the start state to the goal state. [15]. Path finding can be used in multiple real-life applications such as foraging (trying to find food sources and bring food back to the basis) [16] and searching (for spills, victims, targets) [17] and is often combined with some form of particle swarm optimization [18].

F. Collective transport

Collective transport of objects is the technique of a swarm of robots locating an object and collectively moving the object to another place, like a homebase. This can be compared to a

foraging technique; although this implies that a path is made to a certain place. This does not apply to all collective transport techniques. [16] It can be easily seen that localization is an important part of the collective transport technique, and thus has much overlap .
Transporting objects by robotic swarms has many potential applications in many settings, from agriculture to construction to disaster relief. Especially in dangerous settings like warzones or radio-active areas, robotic swarms can be a powerful tool to safely retrieve many objects. More so because the robots used for this application are cheap to produce.

G. Surveillance

Surveillance is a technique in which a swarm of robots continuously patrol the same space, and simultaneously keep an eye out for abnormalities which are specified beforehand. Surveillance with a single robot is already a powerful tool, but with a swarm of robots it becomes even more effective. A larger area can be covered, local communication can be handled efficiently and depleted robots can interchanged with other robots. Often this technique also includes the formation technique, because patrolling robots should be in the same formation when patrolling. Furthermore, in unknown locations mapping would be used, and in locating abnormalities a localization technique is often used. So, this technique uses many other techniques, and can be summarized to use formation, localization, exploration, mapping and dispersion.

The applications of swarm surveillance are diverse, and are already used in a vast range of applications like agricultural practices, police surveillance, inspecting unreachable locations, patrol missions and reconnaisance tasks. [19]
Of course, many more applications exist for surveillance, especially because it incorporates many other different techniques.

IV. In-Depth review of Techniques

After pointing out some applications implementing important techniques used in robotic swarms, we will now provide an in-depth study for these techniques. For every technique, we will make comparisons between different algorithms used by a technique, the problems that are solved and the the problems that are yet to be solved. This chapter thus serves to provide a good overview of ways to implement a technique.

A. Exploration

1) Collaborative multi-robot exploration [1]: [3] Should be divided into Mapping & Localization which are in my opinion two types of exploration with too much overlap.

B. Mapping

Mapping is a technique which is used for many applications and is incorporated in other techniques. Exploration is one of the techniques, which rely on mapping. The goal of mapping is to make a rough overview of the surrounding unknown environment. In this section we will focus on collaborative mapping algorithms for usage with robotic swarms.

1) Algorithms: **Range-Based**

- http://robots.stanford.edu/papers/thrun.map3d.pdf, http://robots.stanford.edu/papers/thrun.maps-multi.pdf
- http://link.springer.com/chapter/10.1007/978-3-540-30552-1_6
- http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1014804
- http://books.google.nl/books?hl=en&lr=&id=3c9w6XEUxIMC&oi=fnd&pg=PA1&dq=collaborative+mapping+robots&ots=ZB-hzyo0zK&sig=8oT_KIZnGViutyzkds4PvC-YwEE#v=onepage&q=collaborative%20mapping%20robots&f=false

TABLE I
Overview of Common Mapping Algorithms

Algorithm	Type	Performance	Scalability
DFLF	Location-Free	Medium-High	High
BFLF	Location-Free	High	High
Directed Dispersion	Range-Based	Medium	High
Random Walk	Range-Based	Low	Low
Follow Wall	Range-Based	Low	Low
Seek Open	Range-Based	Low	Medium
Fiducial	Range-Based	Medium	High
Clique-Intensity	Range-Based	High	High

C. Dispersion

One of the key techniques used in robotic swarms is dispersion. The goal of dispersion in the context of robotic swarms is to scatter the individual robots in an environment such that every section of the environment can be covered. In this section we will focus on non-location oriented algorithms. If prior knowledge about the exact location is provided, a formation technique should be used.

1) Algorithms: The following dispersion algorithms are the most widely used algorithms and thus will be taken into our analysis. A very brief description is given of each algorithm. For more information on the operation of these algorithms, please consult the references.

Location-Free Dispersion Algorithms

- **Depth-First Leader-Follower Strategy (*DFLF*)** [20]:
  A *Depth First Search*(DFS) inspired algorithm in which the swarm has one leader at any given point in time. The robotic swarm has the overview of a map in which specific regions, called *pixels*, are defined. *Frontier pixels*, are pixels which haven't been traversed yet. The leader robot keeps looking for frontier pixels until there are none left and stops and tells its *successor robot* to be the leader. If there are no successors left, the algorithm halts and the total dispersion of the area. The other robots always try to follow the leader and traverse the tiles around it.
- **Breadth-First Leader-Follower (*BFLF*)** [20]:
  A *Breadth First Search*(*BFS*) inspired algorithm, which does not exactly perform *BFS*, but approximates it. It is a more complex algorithm than the *DFLF* algorithm, but

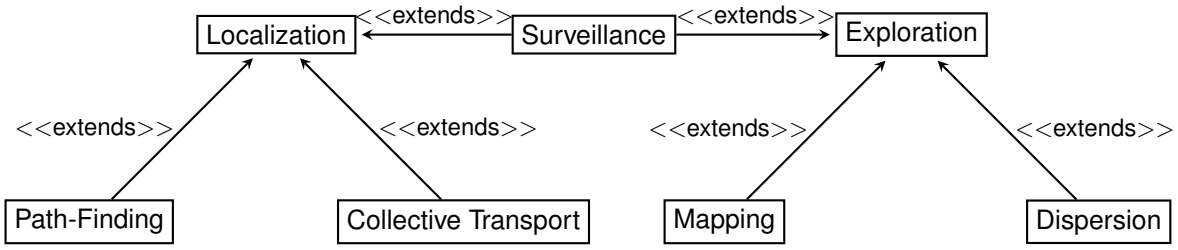

Fig. 1. Technique Hierarchy Overview

has to make fewer moves to fully cover the map. In an extension of the *DFLF* algorithm, the *BFLF* algorithm also contains a waiting state, in which a robot pauses to make the next move. Furthermore, instead of having only one leader, this algorithm allows for multiple leaders. The leaders again strive to find all the frontier pixels, but now also make sure that the follower robots don't stray to far away. Once no frontier pixels can be found by the leader, the leader waits for the followers to arrive and passes on its leadership to one of the followers, the successor. The *BFLF* strategy tries to create as many paths as possible. Visited pixels form a tree, the tree can be branched, which then represent the alternate pixels reachable from that pixel. This branching balances the flow by giving the possiblity to go through pixels multiple times, to be able to go into differerent directions.

Range-Based Dispersion Algorithms

- **Directed Dispersion/Disperse Uniformly** [21]:
  The directed dispersion algorithm has the goal to disperse the robots quickly and uniformly, while keeping the robots connected. The algorithm consists of two sub-algorithms: *disperseUniformly* and *frontierGuidedDispersion*. The *disperseUniformly* algorithm spreads the robots evenly with given constraints. It works by calculating an opposite direction vector of the positions of the nearest robots. This means that communication between robots is of vital importance. The *frontierGuidedDispersion* directs robots towards unexplored areas, while keeping the robots connected. It uses robots which are on the frontiers of explored space to guide the Swarm in unexplored space. An optimal path is created for the other robots to move optimally towards the frontier. For an exact description of the implementation of these algorithms please see [21].
- **Random Walk** [22]:
  This algorithm has 2 states: *walking* and *avoid obstacle*. In the *walking* state each robot keeps going straight with an orientation which randomizes over a certain time interval until there is an object and switches to the *avoid obstacle* state. In the case that it detects a possible collision, the robot changes orientation dramatically until the obstacle has been avoided and continues in the *walking* state.
- **Follow Wall** [22]:
  The follow wall algorithm has 4 states: *find wall*, *align to wall*, *follow wall* and *navigate corner*. The details of this algorithm will not be discussed, as there are quite a few major flaws in this algorithm. This algorithm does not take the existance of other robots into account and thus it is possible for robots to see each other as walls. The usage of this algorithm for dispersion is thus very limited and should be avoided.
- **Seek Open** [22]:
  By calculating an average obstacle vector with support of distance censors, a vector in the exact opposite direction is calculated and the robot, will follow this vector. Depending on the magnitede of the vector, a new assesment will be done once the robot reaches the approximated location. This means that the robot does not need to take further care of collisions with walls, but it is possible for robots to run into each other, or other dynamically moving objects, unless collision avoindance is separetely implemented.
- **Fiducial** [22]:
  By using a beacon system, every robot is able to get the relative location of other robots within a certain range. This information is the used to steer away from the other robots. If no robots are in range, the robot uses the *Random Walk* algorithm.
- **Clique-Intensity** [10]:
  This algorithm uses the connectivity in a cyclic graph for swarm robots to disperse the robots, by knowing their relative distance. By using multiple behaviours as many robots as possible try to stay connected to only one other robot, thus making a very spread out system, which still remains connected.

TABLE II
OVERVIEW OF COMMON DISPERSION ALGORITHMS

Algorithm	Type	Performance	Scalability
DFLF	Location-Free	Medium-High	High
BFLF	Location-Free	High	High
Directed Dispersion	Range-Based	Medium	High
Random Walk	Range-Based	Low	Low
Follow Wall	Range-Based	Low	Low
Seek Open	Range-Based	Low	Medium
Fiducial	Range-Based	Medium	High
Clique-Intensity	Range-Based	High	High

2) Problems: **Range-Based**

Starting from the simpler Algorithms such as random walk and wall following, we show the problems that the dispersion

technique has faced in the past and how the newer algorithms have solved these problems. The *Random Walk* algorithm has to be one of the simpelest algorithms, the problem this algorithm faces though, is that dispersiveness of the robots does not guarantee uniform dispersion. The algorithm can be seen as a brute-force approach, it possibly achieves our end-goal, but it does not do this in an optimal and scalable way.

Wall Following is an algorithm which is used a lot in robotic swarms, however it is not very effective by itself and faces many problems with scalability. One of the main problems with scalability is the fact that the robot does not distinguish other robots and actual walls. This causes an extereme amount of collisions and thus with large amounts of robots, this algorithm becomes near to useless.

The *Seek Open* algorithm is a good example of the real usage of Range-Based navigation, due to the fact that it actually uses the magnitudes of the data provided and does calculations in order to find the best position to move towards. The problem that this algorithm faces however is that it should be implemented with a collision avoidance algorithm. The algorithm is not really adapted to work with other dynamic objects, such as other robots. It is therefore not a very scalable algorithm by itself.

The *Fiducial* algorithm uses the Random Walk algorithm. A problem with the Random Walk algorithm has been solved here: the beacon like system, prevents the robots from running into each other. The Fiducial algorithm does not have any specific problems, however it still is a brute-force approach and thus does not guarantee uniform dispersion.

The main problem with the *Clique-Intensity* algorithm is the fact that due to high amounts of noise in the wireless intensity signals there is a lot of uncertainty in some real world applications. In a perfect situation, the algorithm would also work near perfectly. The work in this area has mostly been theoretical, real-world application is very different compared to theoretical situations.

Location-Free

The *BFLF* algorithm, requires the robots to travel less compared to the *DFLF* algortihm. The DFLF algorithm is furthermore also more computationally expensive than the DFLF algorithm. There is no big difference further during the execution of both algorithms, and thus BFLF has the preference. The problems that these types of algorithms face are not theoretical, but are coming forth from the category that they're in: often it is impossible to know the exact location, even if it's relative. There are quite a few ways that people try to achieve to create a relative position grid, however it difficult if not impossible to achieve in many real world applications.

3) Remaining-Problems: The remaining problems in dispersion algorithms can be generally categorized into range-based problems and location-free problems, since all of the algorithms that are in these categories, are facing similar problems. The focus needed for the range-based approach needs to be on the uniformity of the dispersion. So how can we guarantee uniformity when dispersing the robots. The focus needed for location-free approaches is: how do we actually implement this in real world applications. There are minor to no problems in theory, however to actually bring the relative positioning into a grid is quite difficult. Research in this area should be focussed on how to create these relative positioning grids into a dependable and accurate manner.

D. Localization

Robotic swarm search is an area which has been recieiving a lot of research attention in the past few years. The main goal is to design an algorithm that effectively allows a swarm of robots to explore an unknown area and find the target(s).

1) Swarm Optimization: The robotic swarm search problem described above is often treated as an optimization problem. Therefore we will discuss three optimization algorithms that are closely related and have been used for swarm robotic approaches.

Particle Swarm Optimization

In PSO a number of particles are randomly placed in an unkown space of a problem or function. Each particle evalues its current location according to a certain fitness function and then calculates the best position to go to according to its own history and the history of the particle(s) that it can communicate with at that moment. To prevent the particles from agglomeration a certain randomness is often implemented. When continuously looking for a better position by helping each other, the swarm of particles eventually positions itself at the position of target. [18]

Glowworm Swarm Optimization

In Glowworm Swarm Optimization (GSO) the idea is to distribute "glowworms" randomly over the area and let them, according to the fitness function, carry a certain lumeniscence quantity called luciferin. The closer they get to the target the more luciferin they contain - thus the brighter they are - and the more they attract other glowworms. In every movement step each glowworm moves towards a neighbour within a certain range that carries more luciferin, so they eventually conglomerate at the target. T he glowworms have a varying communication that changes each step with a certain randomness, to make sure multiple targets can be found. [23]

Ant Colony Optimization

Ant Colony Optimization (ACO) is often used in foraging algorithms and is based on the way ants work together in ant colonies. In ant colonies ants do not have one particular function or goal to achieve, but have to work together to complete certain tasks, for example moving a large object. Ant colonies use a concept called "stigmergy" to coordinate their activities. When ants work on a task and move, they leave pheromone behind. The more pheromone a task or location, the more it will attract other ants. To make sure that paths found in the beginning that have become irrelevant will eventually disappear, an evaporation factor is often introduced to let the pheromone evoporate over time. [24]

2) Algorithms: **Location-free and range-based**
In [25] a decentralized application of the PSO algorithm is developed to find multiple targets at unkown locations. The developed algorithm is *location-free* and *range-based*. The targets are equipped with a cell phone that radiates a Radio Frequency signal that can be detected by the robot, which can wirelessly communicate with limited range. The paper shows that a distributed algorithm based on PSO can easily overshoot targets, but with a dynamically weighted wireless coefficient applied to the standard PSO formula this can be prevented. Furthermore it concludes by experiments that the variation in received signal strenghts (RSS) in an indoor environment significantly increases the robot search time in finding a target.

A robotic implementation based on GSO is succesfully ipmlemented and described in [23]. Not only is it *location-free* and *range-based*, but also *memoryless*.

In [16] two foraging algorithms are discussed which both are *location-free* and *range-based*. The main concept is based on ant colony foraging in which ants search for food and when searching food whilde holding food, continuously drop pheromone so other ants can follow their trail backwards. In this algorithm some robots decide to become pheromone robots, which means other robots can store virtual pheromone information. Other robots can sense this information and therefore follow the path. The phereomone robots decay at a specific rate, just as pheromone does in basic ACO.

Location-based and range-based
A model of an implementation of PSO in which every robot has perfect knowledge of its location and can communicate and sense other robots within a certain range can be found in [17]. To model robotic swarm search a couple of modifications had to be made, for example: changing PSO's discret time to continious time, handling the movement limitations and collisions of robots and limiting the particle neighbourhood (range) of each robot, which is unlimited in general PSO. With the model the effect of the communication range and the number of robots have been investigated. Main results are that the algorithm indeed achieved better results (smaller distance to source) when enlarging the number of robots. Furthermore the detection of the source with small communiation achieved poor results, but improved dramatically as the range increased. At the maximum range the average position to the target was best compared to all other communication ranges.

In [26] we find an implementation of PSO combined with bio-inspired search.

TABLE III
OVERVIEW OF COMMON LOCALIZATION ALGORITHMS

Algorithm	Paper	Range-type	Location-type	Performance	Scalability
PSO-based	[18]	Range-based	Location-free	High	High
GSO based	[23]	Range-based	Location-free	Medium	High
PSO based	[17]	Range-based	Location-based	High	High
ACO based	[16]	Range-based	Location-free	Medium	High

3) Other algorithms?: In [27] a framework called *artificial physics* is provided for distributed control of agents.

In [11] a toxic plume is being searched by a roboti swarm using only local information.

In [28] blabla...

Multi-agent sweeping with limited sensors and communication, gas model in [29].

4) Problems: The main differences between PSO and GSO lie in the fact that GSO does not use any memory element. More importantly, GSO makes its decisions based on a continiously varying range, while PSO moves itself according to the k nearest neighbours. Furthermore standard PSO is limited to numerical optimization models, while GSO is also able to effectively detect multiple peaks or sources.

5) Remaining problems:

E. Path-finding

F. CollectiveTransport

Robotic transportation is an interesting technique. Recently Amazon, a large online retailer, announced it would make use of unmanned flying robots to deliver parcels at everyone's door. (www.amazon.com/b?node=8037720011) But, as is obviously true, each robot can only carry a certain weight, which poses a problem. With a robotic swarm of robots it could be possible to transport larger weights in a scalable system, adding swarm robots until you can carry the desired weight. Of course, this does not limit itself to aerial robots, but also includes ground robots and even underwater robots.
The main problem though of collectively transporting with robotic swarms is to move an object from the start to the destination. A simple distinction will be made in light of the different algorithms considered: the swarm is completely autonomous or the swarm is controlled by a single ground station. If a homogeneous swarm is controlled by a single ground station, the algorithm is not considered with finding an object, but is instead focused on safe, accurate and scalable transport. In algorithms in which swarms are completely autonomous, the focus is on finding the object and transporting it to its destination. When considering these two types seperately, both types of algorithms are expanded give a good view of the current state of transporting algorithms.

1) User-controlled swarm transportation algorithms: As listed before, first we consider some algorithms concerned with safe and scalable transport of objects, without actually finding the objects. Because of this, the algorithms listed here are all *location-based.*; the location is assumed to be known. Two different types of swarm transport are considered here: ground transport and aerial transport.

Range-free user-controlled swarm transportation

- **Aerial Equilibrium**
  An algorithm for stable aerial transport is given in the paper Cooperative manipulation and transportation with

aerial robots. [**?**]. In this paper, a scalable solution for stable transport is presented, which can manipulate a payload to a desired pose (position and orientation). This is done by attaching cables from the flying swarm robots, in this paper quadrocopters are used, to a given object. This paper formulates general conditions for the system equilibrium and thus can be scaled for many swarm robots, but is focused on three aerial swarm robots. To accomodate for aerial transport, the workspace and the payload stability are very important, and are extensively calculated. The mathematical model that is defined in the paper is tested with three quadrocopters and offers a stable transportation for objects, with a degree of freedom related to the amount of robots used and can thus be implemented in the field. Because the swarm robots do not calculate the exact distance between them, this algorithm can be defined as a range-free algorithm. Because The actual path-finding of the transport from the object to the goal location is not accounted for, as mentioned earlier.

Range-based user-controlled swarm transportation

- **Cluster space control**
  A second algorithm, used for transporting large objects on the ground, in the water and in the air, considers a multi-robot formation control framework, necessary to coordinate the motions of the robots in the group. It is mentioned in the paper Object Manipulation Using Cooperative Mobile Multi-Robot Systems. [**?**] This particular control approach is called *cluster space control*, and is utilized to control swarm of four four-wheeled robots. A user using this type of transportation uses a joystick to input user controls to the whole swarm, after which the control framework specifies the formation and the position and shape of the swarm. Because the framework calculates the exact range between each robot in the swarm, this This control framework allows the user to effectively transport large objects with a scalable set of robots.

2) Autonomous swarm transportation algorithms: Before transporting an object with a swarm of robots, first, the swarm has to find the object. Then, after finding it, the swarm has to get a hold on it. Lastly, they have to move it to the target location. In these types of algorithms, only algorithms that have to return one object are considered; we do not concern ourselves with foraging algorithms, which find a path to the target to collect multiple objects. Because these algorithm require that the target object localization and the target goal location are unknown, these algorithms are *location-free.*

Range-free autonomous transportation

- **Flocking**
  The first algorithm is simple; find the location of the object through a flocking algorithm and let the swarm robots push the object to the goal location. This algorithm is explained in the paper Collective Transport of Complex Objects by Simple Robots: Theory and Experiments. [**?**] Basically, this algorithm like many of the algorithms defined in this section can be divided by two parts: finding the object and moving it to the goal location. The algorithm used for finding the location of the object is called *flocking*. The algorithm is quite simple; every robot looks if it can see the object location. If so, move towards the location. If not, the robot compares its own heading with the other robots nearby and corrects it until the robot finds the object location. So, algorithmically, each robot uses one of two behaviors. A robot that can see the goals aligns itself towards the goal while moving forward. A robot that cannot see the goal uses its IR sensors to determine the heading difference between itself and its neighbors. In this paper, the object location and the goal location are found with light sensors, communicates headings with infrared sensors and uses bump sensors to avoid obstacles. After finding the object, the robot pushes from an arbitrary side, calculating the amount of force needed in relation to the middle point of mass with its bump sensors and transporting it to the goal location with its light sensor. This algorithm can thus be seen as a location-free . Because the robots latch onto an arbitrary side, scaling of the swarm is extremely effective.

- **Pheromone**
  Another widely used algorithm for transporting objects is the *pheromone* algorithm. Found in nature, used by ants, this algorithms relies on leaving a trail of marks that help guide other swarm robots. It is often used for foraging techniques to retrieve multiple objects, but can also be used for retrieving a single object, with which a trail will guide many robots to make for an easier retrieval. This algorithm is used in the paper Cooperative Transportation by Swarm Robots Using Pheromone Communication. [**?**] In this article, this method is tested with swarm-bots which can release and detect spots of ethanol left on the ground, indicating a spot they have to follow. Although the pheromone algorithm implies that pheromones should be used, different varieties of markings can be used, including wireless sensor beacons, drops of paint and even objects. This algorithm is particularly useful for guiding many robots to an object, if for example the object is very heavy or if many robots are needed for accurate and/or stable transportation. This algorithm is location-free but range-based; it calculates the exact distance to a trail. This algorithm is pretty difficult to implement in dynamic environments, as marks made can dissapear.

- **Granular Convection**
  In looking for swarm algorithms, you want the robots to be as simple as possible. An algorithm that only uses very simple robots is an algorithm that is based on granular convection, als known as the Brazil Nut Effect. This novel method is described in the paper Granular convection to transportation. [**?**] The algorithms described in this paper

are either intended for:

- A swarm with homogeneous robots, with no explicit communication
- A swarm of heterogeneous robots, composed of robots that change their direction with two different probabilities
- A swarm of heterogeneous robots that uses local communication to adjust the fraction of robots that change their direction with higher probability

These algorithms listed above are ranked in effectiveness, in which the last one obviously is the most effective, but keep in mind that the robots used to implement this are harder to produce than for the first.

The way this algorithm works is as follows. The goal of the swarm robots is to transport the objedct to the goal location. Each swarm robot vibrates with a random force. The goal location outputs a repulsive force. As the robots are randomly placed around the object and the goal location, the object gets kicked around by the vibrating swarm robots, until the object is kicked to the target location. When the swarm increases, the amount of time it costs to transport the object significantly decreases. With heterogenuous robots, there is more communication between robots to lower the density of the robots around the object, because slows the process of moving the object. This algorithm has different implementations, but is mostly location-free and range-free.

A short summarization of these algorithms will be given in the form of this table.

TABLE IV
OVERVIEW OF COLLECTIVE TRANSPORT ALGORITHMS

Algorithm	Type	Performance	Scalability
Flocking	Range-free	High	High
Pheromone	Range-free	Medium	Medium
Homogeneous granular convection	Range-free	Low	High
Heterogeneous granular convection	Range-free	Medium	High
Heterogeneous granular convection with local communication	Range-based	High	High

3) Problems: **User-controlled swarm transportation algorithms**

The main differences between these two algorithms is that in the first mathematical model that is used, each swarm robot individually calculates its position to the other robots and the object, and is calculating the resulting tension on the object. This way, the rotation and position can be controlled by the position and tension of each robot. With the second algorithm, that uses the cluster *cluster space control*, the calculations are not done individually. Instead, the framework calculates the position of each robot and the orientation of the object. Thus, these methods both provide stability to the transported object but in two completely different ways.

Autonomous swarm transportation algorithms

These three algorithms are very different from each other, but they all serve a common goal. A problem these algorithms ran into was how to locate the object, how to move the object and how to find and reach the goal location. The main problem is locating the object, something which the granular convection algorithm actually never does. What these three algorithms have in common is that the transportation process speeds up significantly when adding more swarm robots, but reaches a cap when it becomes ineffective due to maximal density. For this reason, small, easy and cheap to produce swarm robots should be required. The granular convection algorithm has the cheapest robots, but is not necessarily the best, because when the robots become slightly larger and more complicated, more sophisticated and faster algorithms could work.

4) Remaining problems: **User-controlled swarm transportation algorithms**

The problems that these algorithms had was that the robots should be coordinated to rely on the location of other robots. With the mathematical model these are calculated through the tension of the cable attached to each robot, but in the framework it is calculated centrally. Problems that are not treated in these articles is how these swarms can locate an object and autonomously can deliver it to a location. Another problem for which no solution is provided is that when these swarms grow to account for heavier objects, the robots will get in each other's way. Especially considering transportation, a technique in which every robot must latch on to an object, this can be hard to practically scale.

Autonomous swarm transportation algorithms

In the mentioned algorithms, moving the object is mostly done by pushing. In real-life applications this would not always be a good option and a more sophisticated method of attaching should be used. When considering search-and-rescue operations for example, a solution to effectively grab the person to be rescued should be thought of. This is a problem that is yet to be identified in these algorithms and poses a good question for the future.

G. Surveillance

Surveillance is the last technique that is going to be discussed in this paper, and for good reason. As can be seen in the figure 1, surveillance is a technique that uses both the techniques exploration and localization. This is obvious, because when a robotic swarm is surveilling an unknown environment, it has to explore the environment and has to localize disturbances. So it makes sense to review this technique as last, as it references algorithms already used in other techniques. Surveillance as a technique is also very close to the application domain, as surveillance is more of a goal than a technique. But, because it combines other techniques and is used in many applications, it is deemed useful to review a few popular algorithms. In some surveillance algorithms, we also consider the formation technique, in which the robot swarms have to dynamically retain their formation which is important for most of the surveillance algorithms.

1) Surveillance Algorithms: **Location-free surveillance**

- **Kinetic Theory of Gases Algorithm**
  The first surveillance algorithm which will be expanded

on is a novel one, based on the article Robotic Simulation of Gases for a Surveillance Task. [**?**]. The goal that is defined for this algorithm is that it requires a swarmof robots to monitor a corridor, by sweeping through it while avoiding obstacles. The problem reaching this goal is maintaining spatial coverage, especially after passing obstacles, while the swarm robots are only equipped with limited sensors and communication. The algorithm that is used is described as a Kinetic Theory algorithm, which is modelled after the movement of gases. In this algorithm, every molecule of a gas cloud is modeled as a robot, but because robots are not that small, this system is made for a larger scale. Every robot has 24 different sonar sensors to detect walls/obstacles, plus the capability to communicate with low bandwidth radio frequency communication. The goal direction is detected with a light sensor.

The algorithm works as follows. Each robots tries to detect the goal destination. If a robot finds it, it communicates this to all other to specify the direction in which they have to go. It is pretty similar to the flocking algorithm, but the difference in algorithms is the way they handle internal and external collisions (i.e. collisions between robots and walls). When collisions in the Kinetic Theory algorithm are detected, the robots calculates its new position and orientation depending on the speed and orientation of the obstacle (robot or wall). This is modelled after the way particles of a gas cloud move when colliding.

- **Scouts and Rangers**
  2000 - Rybski - A Team of Robotic Agents for Surveillance

Location-based surveillance

- **Networked Robotic Surveillance**
  2012 - Ghaffarkhah - Path Planning for Networked Robotic Surveillance
- **Surveillance Event Agents**
  2006 - Roman-Ballesteros - A Framework for Cooperative Multi-Robot Surveillance Tasks

Range-free surveillance

- **Stochastic Strategies**
  2005 - Grace - stochastic Strategies for Autonomous Robotic Surveillance

Range-based surveillance

- **Dynamic Directed Movement Behaviour**
  2013 - Mullen - Reactive Coordination and Adaptive Lattice Formation in Mobile Robotic surveillance Swarms

TABLE V
OVERVIEW OF SURVEILLANCE ALGORITHMS

Algorithm	Type	Performance	Scalability
Zooi1	Range-free	High	High
zooi2	Range-free	Medium	Medium
Zooi3	Range-free	Low	High

2) Problems: There are a lot of problems. Deal with it.

3) Remaining problems: Yeah, a lot of them.

V. DISCUSSION

The discussion goes here.

APPENDIX A

Appendix one text goes here.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] F. Amigoni, V. Schiaffonati, and M. Verdicchio, "Methods and Experimental Techniques in Computer Engineering," 2014. [Online]. Available: http://link.springer.com/10.1007/978-3-319-00272-9

[2] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *Swarm robotics*. Springer, 2005, pp. 10–20.

[3] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 376–386, 2005.

[4] A. M. Naghsh, J. Gancet, A. Tanoto, and C. Roast, "Analysis and design of human-robot swarm interaction in firefighting," *RO-MAN 2008 - The 17th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 255–260, Aug. 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4600675

[5] J. Penders, L. Alboul, and U. Witkowski, "A robot swarm assisting a human fire-fighter," *Advanced Robotics*, 2011. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1163/016918610X538507

[6] I. A. Wagner, Y. Altshuler, V. Yanovski, and A. M. Bruckstein, "Cooperative cleaners: A study in ant robotics," *The International Journal of Robotics Research*, vol. 27, no. 1, pp. 127–151, 2008.

[7] C. Hardin, X. Cui, R. Ragade, J. Graham, and A. Elmaghraby, "A modified particle swarm algorithm for robotic mapping of hazardous environments," in *Automation Congress, 2004. Proceedings. World*, vol. 17. IEEE, 2004, pp. 31–36.

[8] W. Sheng, Q. Yang, J. Tan, and N. Xi, "Distributed multi-robot coordination in area exploration," *Robotics and Autonomous Systems*, vol. 54, no. 12, pp. 945–955, 2006.

[9] J. A. Rothermich, M. İ. Ecemiş, and P. Gaudiano, "Distributed localization and mapping with a robotic swarm," in *Swarm Robotics*. Springer, 2005, pp. 58–69.

[10] L. Ludwig and M. Gini, "Robotic swarm dispersion using wireless intensity signals," in *Distributed Autonomous Robotic Systems 7*. Springer, 2006, pp. 135–144.

[11] D. Zarzhitsky, D. F. Spears, and W. M. Spears, "Distributed robotics approach to chemical plume tracing," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 4034–4039.

[12] S. Bashyal and G. K. Venayagamoorthy, "Human swarm interaction for radiation source search and localization," in *Swarm Intelligence Symposium, 2008. SIS 2008. IEEE*. IEEE, 2008, pp. 1–8.

[13] A. Marjovi, J. G. Nunes, L. Marques, and A. de Almeida, "Multi-robot exploration and fire searching," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 1929–1934.

[14] X. Cui, C. T. Hardin, R. K. Ragade, and A. S. Elmaghraby, "A swarm approach for emission sources localization," in *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*. IEEE, 2004, pp. 424–430.

[15] Y.-Q. Qin, D.-B. Sun, N. Li, and Y.-G. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, vol. 4. IEEE, 2004, pp. 2473–2478.

[16] N. R. Hoff, A. Sagoff, R. J. Wood, and R. Nagpal, "Two foraging algorithms for robot swarms using only local communication," in *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*. IEEE, 2010, pp. 123–130.

[17] J. Pugh and A. Martinoli, "Inspiring and modeling multi-robot search with particle swarm optimization," in *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*. IEEE, 2007, pp. 332–339.

[18] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.
[19] A. Bürkle, F. Segor, and M. Kollmann, "Towards Autonomous Micro UAV Swarms," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1-4, pp. 339–353, Oct. 2010. [Online]. Available: http://link.springer.com/10.1007/s10846-010-9492-x
[20] T.-R. Hsiang, E. M. Arkin, M. A. Bender, S. P. Fekete, and J. S. Mitchell, "Algorithms for rapidly dispersing robot swarms in unknown environments," in *Algorithmic Foundations of Robotics V*. Springer, 2004, pp. 77–94.
[21] J. McLurkin and J. Smith, "Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots," in *Distributed Autonomous Robotic Systems 6*. Springer, 2007, pp. 399–408.
[22] R. Morlok and M. Gini, "Dispersing robots in an unknown environment," in *Distributed Autonomous Robotic Systems 6*. Springer, 2007, pp. 253–262.
[23] K. Krishnanand and D. Ghose, "Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications," *Multiagent and Grid Systems*, vol. 2, no. 3, pp. 209–222, 2006.
[24] D. Yingying, H. Yan, and J. Jingping, "Multi-robot cooperation method based on the ant algorithm," in *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*. IEEE, 2003, pp. 14–18.
[25] K. Derr and M. Manic, "Multi-robot, multi-target particle swarm optimization search in noisy wireless environments," in *Human System Interactions, 2009. HSI'09. 2nd Conference on*. IEEE, 2009, pp. 81–86.
[26] J. Pugh and A. Martinoli, "Distributed adaptation in multi-robot search using particle swarm optimization," in *From Animals to Animats 10*. Springer, 2008, pp. 393–402.
[27] D. Zarzhitsky, D. Spears, D. Thayer, and W. Spears, "Agent-based chemical plume tracing using fluid dynamics," in *Formal Approaches to Agent-Based Systems*. Springer, 2005, pp. 146–160.
[28] W.-M. Shen, P. Will, A. Galstyan, and C.-M. Chuong, "Hormone-inspired self-organization and distributed control of robotic swarms," *Autonomous Robots*, vol. 17, no. 1, pp. 93–105, 2004.
[29] W. Kerr, D. Spears, W. Spears, and D. Thayer, "Two formal gas models for multi-agent sweeping and obstacle avoidance," in *Formal Approaches to Agent-Based Systems*. Springer, 2005, pp. 111–130.