Multi-UAV Based Crowd Monitoring System

Rodrigo Saar de Moraes, and Edison Pignaton de Freitas

***Abstract*—This article presents the development of a multi-UAV based crowd monitoring system, demonstrating a system that uses UAVs to periodically monitor a group of moving walking individuals. Using auction paradigms to distribute targets among UAVs and genetic algorithms to calculate the best order to visit the targets, the system has shown capabilities to efficiently perform the surveillance, visiting all the targets during a surveillance period and minimizing the time between the visits made to each target. Moreover, the system showed robustness keeping the good performance under a variety of situations.**

***Index Terms*—Multiple-UAV systems; Task Auctioning; Crowd Monitoring; Intelligent Control System; UAV Monitoring;**

I. Introduction

THE usage of Unmanned Aerial Vehicles (UAVs) in civilian and military applications has increased exponentially since cheaper and more complete unmanned aircraft platforms have appeared on the market. This trend, allied to other technological developments in both software and hardware field has originated a great number of new applications and new uses for this kind of technology. [1].

Besides that, micro UAVs have become an important tool to law enforcement forces, allowing them to track and follow targets or survey areas or buildings quietly, safely and almost undetectably. These kinds of applications, however, depend today heavily on human control and supervision, increasing costs and complexity to perform missions and making them prone to human mistakes that would not occur if an autonomous control system was in place. Among these uses, crowd control and monitoring applications for law enforcement present a set of interesting and complex characteristics that turn the development of an autonomous controller for such scenarios a challenging task.

In general, crowd control and monitoring applications present more targets, or groups of targets, to be monitored in the crowd than resources to monitor them. While demonstrations may be composed of hundreds of individuals, police forces and resources are often bound to a much smaller number, restricting the capacity of law enforcement agencies to act in such situations. This fact exposes a need for capable and intelligent monitoring solutions to assist the law enforcement forces in large-scale monitoring scenarios, identifying a small number of potentially dangerous individuals in the crowd, e.g. individuals using masks or carrying potentially dangerous objects, and tracking them individually. Hence, in these scenarios, UAVs can be important assets, representing very useful tool to the law enforcement agencies. Due to their high degree of mobility, UAVs can move fast and assertively in order to follow and track targets, avoiding ground obstacles and efficiently reaching their mission goals. Due to these same characteristics, a single UAV can be used to monitor multiple targets at once, alternating and moving fast between different targets without losing sight of any of the subjects under its responsibility.

Observing this scenario, this work investigates the usage of rotatory-wing UAV platforms in this kind of crowd monitoring and control operation and presents a system capable of using multiple UAVs in this context. The system is able to distribute target monitoring responsibilities among the UAVs and to effectively monitor all the targets, minimizing the time each of them remains hidden from the system. As previously stated, the system has to be able to work in situations in which there are more targets than UAV assets, making sure that even small teams of UAVs are able to handle a moderate amount of targets (up to 20 targets per UAV), if necessary. On the other hand, it must still be able to provide a good degree of mobility for the UAVs so that they can explore the crowd in cases in which only a few targets are present or reorganize themselves as to avoid losing track of an escaping target.

To solve this specific problem, an auction protocol was implemented to assign targets to UAVs (agents). Once the assignment phase has taken place, each UAV executes a modified Genetic Algorithm (GA) [2] to decide the order in which it will visit the targets to grant continuous monitoring. Once the GA predicts a system failure or possible target disappearance, the UAVs initiate a hot-handover procedure to reallocate targets. The system then reorganizes as to not lose sight of the targets.

While several prior works have attempted to solve similar cooperative observation and tracking problems, most of them assumed fixed-wing UAV's dynamics and therefore are not directly comparable with the method proposed in this work.Thus, it is possible to state that the contribution of this paper is mainly three-fold: First, the presentation of a suitable model to handle the problem of crowd monitoring with a group of small rotatory-wing UAVs, characterizing it as a special version of the Traveling Salesman Problem (TSP), i.e. the Time Dependant TSP (TDTSP). Second, the presentation of a scalable efficient solution for non-fixed wing UAVs that distributes the targets among the UAVs and that is able to keep track of the said targets under varying conditions during the surveillance mission. And third, an extension to the tradition genetic algorithm heuristic to tackle the visit-

R. S. de Moraes is currently with the Graduate Program on Computer Science - Linköping University, SE-58183, Linköping, Sweden

Email: rodrigo.moraes@liu.se

E. P. de Freitas is with the Graduate Program on Electrical Engineering - Federal University of Rio Grande do Sul (UFRGS), Av. Osvaldo Aranha, 103, Bairro Bom Fim, 90035-190, Porto Alegre, RS, Brazil

Email: edison.pignaton@ufrgs.br

This work was developed when R. S. de Moraes was with the the Graduate Program on Electrical Engineering - Federal University of Rio Grande do Sul (UFRGS), Av. Osvaldo Aranha, 103, Bairro Bom Fim, 90035-190, Porto Alegre, RS, Brazil

oriented TDTSP defined in this work. Moreover, the proposed solution innovates in the handover algorithm that controls the transference of the responsibility of a given target from one UAV to another.

The remainder of this paper is organized as follows: Section 2 presents the problem definition and scope delimitation, while Section 3 reviews related works that tackle problems close to the one in the scope of this proposal as well as others that were used as inspiration. Section 4 describes the proposed approach to solve the problem. Section 5 presents the implementation details of the proposed solution and the setup of the simulation environment used to evaluate the proposal, while section 6 presents the obtained experimental results. Section 7 discusses the actual adequacy of the proposed solution to the described problem. Section 8, in turn, concludes the paper highlighting its main contributions and suggesting possible directions for future works to extend the proposal.

II. Problem Definition

As stated in the introduction, this work focuses on the development of a distributed and autonomous control system for UAVs used in crowd control and monitoring scenarios to support law enforcement operations. In this context, this work assumes crowd monitoring as an operation of threat or target identification and position tracking. That assumption means that the goal of the monitoring application is to track and record the position of the potential threats. However, this does not mean that the system continuously keeps recording their every single move and act, but it keeps track of their approximate location, and periodically returns to that location to verify their behavior. In terms of the system's behavior, the UAVs must distribute targets among themselves and keep track of these targets without losing them under any circumstance. This also means that the UAVs responsible for more than one target must be able to periodically visit each target to assure it is still where it ought or was predicted to be. Moreover, it has to verify if the target has not taken part in any threatening action.

As this is a problem that can become prohibitively complex, some assumptions are made about the dynamic behavior of the targets to create a controlled scenario closest as possible to a real-world situation in which this monitoring system could be employed. It is supposed in the application scenario that targets belonging a mob infiltrated in a demonstration displaced on a map, will always have a constant movement behavior, not always keeping the same speed of movement, but having a controlled variability in their walking patterns. This means that they will never go from full-stop to full-sprint in a matter of seconds, for example.

The application scenario also assumes that the targets move in a relatively straight line. They do not abruptly change their direction of movement, acting, therefore, as if they were moving with the crowd along the streets or avenues, which can be divided into parts or segments, and each of these parts can be considered a straight line. Figure 1 illustrates these assumptions about the movement behavior.

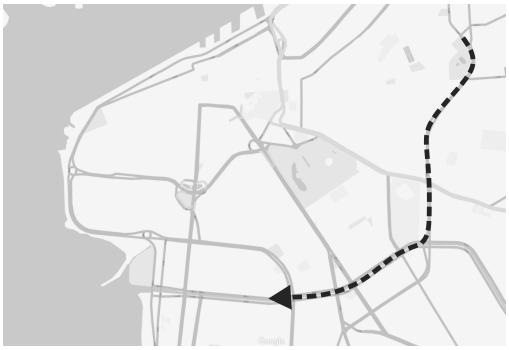

Fig. 1. Common Path of Protesters in the Streets of Porto Alegre - Brazil

These assumptions are very likely to be true in real life scenarios since people in a demonstration are not likely to move too fast nor to sprint due to the agglomeration of people. Demonstrations also tend to follow a well-defined path, with well-defined directions, normally following streets or avenues in a city, and not running around at will, reducing, therefore, the movement direction variability. Thus, by considering these assumptions, this work bases itself on a well-defined set of rules concerning the targets that ensure it's usability in real life situations, defining a solvable problem instead of an intractable one.

Another assumption made in this scenario is that the UAVs are going to be used to monitor pacific crowds, with the number of potential threats limited to a few individuals on the crowd. This can also be the case considering, for instance, lone wolf terrorist attacks. This system is not intended to operate in extreme situations where hundreds of threatening individuals are jointly operating, riots or violent manifestations are examples. The system can be used in such cases, but the success in monitoring great numbers of targets will not be guaranteed since it depends heavily on each target location and speed. The solution conceived in the next section is focused to work on situations where normally distributed targets on the crowd do not surpass 5 times the number of UAVs.

Finally, the UAVs are also constrained themselves. This study considers only rotatory-wings UAVs, which are the most suitable for operations in which there is a need for high maneuverability, sharp turns and back and forth movements, such as the scenario presented in Figure 1, where the UAVs would be flying in an urban environment, between buildings located over streets which are just a few meters wide.

The work in this paper is mainly focused on the functional behavior of the UAVs, leaving aside problems like movement control and image processing, which can be very complex on their fields, but that are out of the scope of this paper and then not addressed here. In the context of this work, it is supposed that UAVs are capable of recognizing the targets once they are on sight of the camera or image acquisition device mounted on them. Thus, image acquisition and processing are transparent to the handled problem. It is also assumed that the low-level control of the actuators is outside of the scope of this work.

Furthermore, in order to isolate the different parts of the system, the discussion on this paper addresses only the monitoring phase of the operation. It is assumed that, when the simulation starts, the system already has a list of the targets

and their current locations, acting as if a previous exploration had already been performed to find who are the targets and where they are. This situation simulates a real-life case in which an operator, i.e. a foot agent or a command center with access to cameras, has already identified the positions of the targets and just wants the UAVs to observe them. The outcome of running the exploration before or while the observation takes place can be subject to an extension of this work and was not addressed in this paper.

III. Related Work

Target monitoring with UAV or robotic-based systems is a widely studied research topic in robotics and automation literature. Most algorithms and solutions presented, however, focus themselves in spatial clustering [3] and ground monitoring, allocating each UAV to a specific portion of the monitored terrain and giving it the exclusivity responsibility for the targets present on that area, creating mobility restrictions that may not be desirable depending on the application. Variations of this approach allocate a group of targets to a given UAV, which ends up having the same mobility restrictions. Coordinating mobile robots to monitor moving targets also depends greatly on the communication resources available to the robots. While some robots, such as large UAVs or UGVs, may support complex and power demanding communication hardware, light commercial-off-the-shelf (COTS) UAVs, such as quad-copters, present an affordable solution for smart-surveillance applications, but cannot support power hungry hardware, limiting the communication range and capabilities of such assets. In addition to that, there is also the aspect of whether the coordination should be centralized, decentralized or distributed, inserting or not points of failure on the system and making it, or not, capable of taking self-healing actions.

Many of these different approaches are analyzed in [4], which organizes, categorizes and compares 20 years of studies in the usage of mobile robots for observing multiple moving targets. This work lists and addresses five factors, which are common in this area, and characterizes this kind of problem, providing a classification method to evaluate the different approaches. The authors further group the existing approaches based on four major control techniques: Cooperative Tracking (CT) [5], [3], [6], which has the objective to persistently track moving targets; Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT) [7], which aims to increase the collective time of observation for all targets; Cooperative Search, Acquisition, and Track (CSAT), which alternates between search and track states, both searching and tracking moving targets; and Multi-robot Pursuit Evasion (MPE) [8], whose objective is to capture evasive targets.

Following the classification proposed by [4] Cooperative Tracking (CT) approaches are those focused on reducing the time duration between two observations to the same target, updating the information the system has on the location of the targets on a more frequent basis. CT is very similar to the approach presented here on this current paper. This work, however, does not only implements CT, but also a distributed algorithm to do so, which does not require a master, or central, node to work. This fact reduces the chances of miss-functions due to single points of failure and communication problems, the same strategy used in [9], [10], [8], [11]. In addition, a comparison regarding the type of targets considered can also be made. According to the classification presented in [4], targets here are considered non-cooperative, as in [9], which means that they do not broadcast their positions, as in [12], but also does not necessarily mean they are hostile, or evasive, as [8], which is the case in the problem handled here.

An example of a proposal exploring a type of CT approach is presented in [5], in which a centralized algorithm is used to maximize the visibility of ground targets in urban environments. The algorithm proposed by the authors groups the targets to maximize each UAV's coverage area in order to reduce overlap. Then it assigns each UAV to a circular optimal path that maximizes the visibility given the shapes and the locations of obstacles on the ground. Their scenario considers that the number of moving targets to be tracked is much larger than the number of aircraft. However, they consider that the aircraft fly much higher than a common multi-rotor small UAV, being more suitable to be deployed on large fixed wing aircraft than on cheap COTS UAVs.

The same kind of approach is used in [3]. In this work, a clustering algorithm is used to define an area of surveillance for each UAV, making them responsible only for the targets existing on the area they were assigned for. The authors, however, use in their work an interesting data fusion algorithm to estimate each target's position, giving a different insight into movement prediction problems and techniques. Another similar technique is used in [6], in which an offline algorithm calculates a terrain division and allocates each UAV to a given set of targets present on one of these divisions. Then, the UAVs assume an orbital movement behavior around these places. This kind of solution works against the main constraints of the problem defined in this paper, the need to have a group of mobile and unconstrained agents capable of moving around according to the possible changes in the operation scenario.

Besides Cooperative Tracking, another technique described in [4] is the Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT), which has the goal to dynamically position robots to maximize the collective time during which targets are observed. CMOMMT differs from CT once, instead of trying to keep track of the positions of the targets through successive visits, i.e. occasionally leaving the target and later returning back, it tries to maximize the time a target effectively stays into the field of view of the UAV. The problem addressed by CMOMMT is to maximize not only the number of targets under observation but also the duration of observation for each target, which is a much more complex problem. A variation of such algorithm is presented in [13], where the authors implement a "help" mode, in which a robot losing a target from its FOV, broadcasts a help request to the other robots. The robots available then respond to these help requests by approaching the robot that is in need of help, maximizing the use of resources, and granting that no target is lost.

Target tracking is also addressed in [14] where the authors proposed a dynamic UAV path-planning algorithm for tracking

a ground target. Their algorithm is a combination of a point-mass approximation and other techniques, such as vector fields and obstacle avoidance strategies. These techniques give the proposed method the ability to find the shortest path for a UAV tracking a target that may be moving in an environment that may include obstacles and/or wind.

Approaching specifically crowd control and pedestrian monitoring, [15] brings an important study in pedestrian groups behavior and image processing to evaluate potentially threatening behaviors. The work of these authors can be considered an important complement to the solution presented here in this present paper since there is no focus on image processing here. An additional module would be needed to be run on each UAV to identify targets and make sure they are found by the UAV cameras. This aspect is out of the scope of this work thus considered transparent here.

While still on the same topic, [16] distances itself from the UAV or aerial monitoring problem and discusses definitions and challenges to address TDTSP situations, presenting mathematical and complexity analysis, along with possible algorithms to solve such problems. [17] continues discussing this type of problem presenting a special case of the TDTSP in which not only the cost to move between cities change over time but the cities' (comparable to the targets in the present work) locations also change. In their work, the authors also present suitable techniques to solve this problem, as well as an intense discussion and analysis of this type of problem.

Closer to the work here presented, [18] provides an analysis of both greedy and LKH heuristics [19] to solve TSP problems focused on the pursuit of moving targets. This work has a very similar goal to the one here presented and many of the considerations presented in their work were also used to conceive the solution reported in this current paper. Their solution, however, is much less flexible than the one presented here, not allowing the mobility and temporal constraints required by pedestrian tracking problems.

Even tough Cooperative Tracking techniques share a common goal with the proposal here presented, which is reducing the interval between two observations of a same given target, they considerably differ from the work here proposed. For once, most of these algorithms and solutions focus themselves in spatial clustering or target grouping, allocating each UAV to a specific portion of terrain or group of targets and, mostly, forcing them into orbital or circular flight paths. This kind of approach may be a good fit to larger fixed-wing UAVs, which are designed to be used on higher altitudes, with reduced maneuverability, and covering large portions of ground through the use of complex and heavy image acquisition equipment. However, it might not be the best approach once a small, cheap and lightweight UAV that flies closer to the targets, covering smaller portions of the ground, but possibly in urban environments, is considered.

Since most approaches presented in literature work better with fixed wings UAVs, grouping and clustering targets, not taking full advantage of the maneuverability offered by small UAVs, this work proposes a novel coordination technique that explores the characteristics of small COTS UAVs to monitor multiple moving targets from a closer range. Besides that, this work presents a cooperative communication behavior, re-handling and auctioning targets to other UAVs when a sight loss is imminent, a different approach for Cooperative Tracing techniques, that usually handle target attributions as fixed throughout mission time. Consequently, the dynamic approach here proposed improves the overall performance of the system.

IV. Proposed Solution

The conceived solution is composed of three main modules or algorithms. First, the solution implements an auction algorithm to distribute threats, or targets, among the UAVs taking into account the current status of both UAVs and targets. A Genetic Optimization Solver accounts for the second module and it is responsible for defining which of the targets in the UAV's current queue it will visit next. Moreover, it also organizes the next visits to not lose sight of any of the targets the UAV is responsible for. The last module implements a target handover technique, similar to the the "help" technique proposed in [13] , granting that if a UAV finds out it will not be able to cover all its targets, it informs and negotiates with another UAV to take the responsibility over one or more targets, handing it(them) over and reducing its current target queue. These algorithms are presented in the following sections, discussing their inner workings and how they interact to build a complete solution.

A. Solution Overview

As stated above, the integrated solution uses three main modules, one providing an auction mechanism, another implementing a genetic solver and the last responsible for the target handover method. These three modules interact with each other to make sure all the mission requirements will be met, i.e. the modules interact and exchange information in order to make sure all targets will be monitored and will not be lost. Together these different algorithms work toward the same goal: ensuring that a given target is visited by some UAV every n seconds. The variable n here stands for the maximum time interval allowed between two visits to the same target without giving it time to disappear from the field of view of the system as a whole. The solution works to ensure that there will always be a UAV capable of monitoring or checking on a given target before the visit time window of that specific target expires.

To achieve this desired result, the first step of the proposed solution is the distribution of targets among the UAVs. Every target known by the system must be, at first, assigned to the UAV that fits better to monitor it at that given time. This allocation is made through an auction algorithm specifically tuned to the desired result of this system. Each auction section allocates a target to a UAV taking into account the state of both UAVs and targets, in terms of their location and speed, and the number of targets already assigned to each UAV. The Auction mechanism is asynchronous to the rest of the system. It is activated only when either a new target is discovered or informed to the system, or when a handover operation has to take place.

Independently from the auction sessions, a genetic solver runs internally on the UAVs that have at least one target assigned to them. The purpose of the genetic solver is to calculate and find the best order to visit the targets a UAV has assigned to itself. This is executed to make sure they will not be lost. To do this, each UAV simulates some possible different orders to visit the different targets. Then, it analyses each of these different orders and chooses the best tour (order), to visit the targets. It is important to notice that this algorithm is conceived to, and it tries to, minimize the UAV's displacement between targets in two successive visits. This minimization makes possible multiple visits to a faster UAV while only one to a slower UAV, if necessary (e.g. in systems in which there are a combination of UAVs with different capabilities and that can move with different maximum speeds). If the genetic solver finds a case in which it predicts that the best route or solution it finds will not ensure that the maximum time between visits to a target will be respected, it triggers the handover mechanism.

The handover mechanism removes a target from a UAV current queue and reallocates the responsibility for this target to another UAV. By doing this, it is ensured that a UAV having problems to visit all its targets will be relieved from at least one target. Then, it recalculates its route, trying to visit all remaining targets within their visit time constraints. Basically, when the handover is triggered by the genetic solver, the UAV tries to sell one of its targets, the one located in a more distant position from the other targets it has to visit. Once an auction is triggered by one UAV, another one, that fits better to the auctioned target, will buy it. Once a UAV succeeds in selling a target, it will try to re-execute its genetic solver, recalculating its new route for the remaining target. If a feasible solution is not found, this handover process may be repeated until a solution is found, or until the UAV that triggered the handover keeps just one target under its responsibility.

A flowchart of the algorithm comprising the whole solution can be seen in Figure 2 for a better understanding of these mechanisms and how they interact.

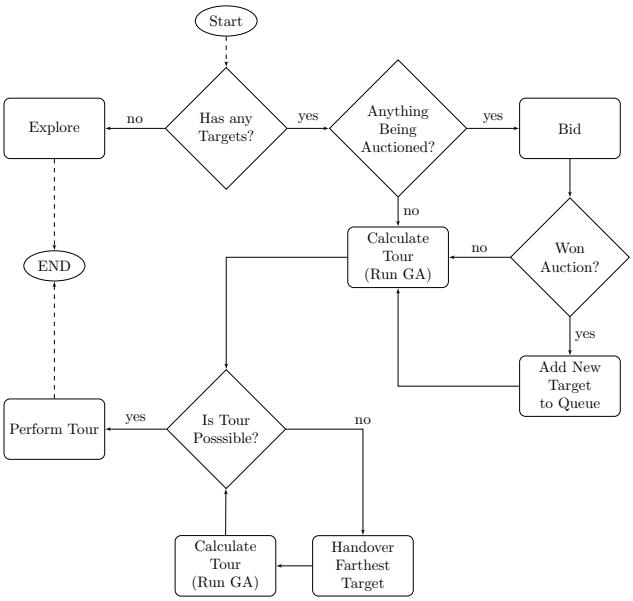

Fig. 2. Proposed Solution Flowchart

B. Auction Mechanism

As mentioned above, an auction mechanism is proposed to be used as a task allocator, allowing an efficient distribution of target monitoring responsibilities among the UAVs. The idea behind the algorithm is based on the behavior of buyers and sellers of an auction market, meaning that goods or resources will be put to auction and sold to the buyer who makes the best offer for them. In the presented context, the targets represent the goods being sold and each UAV represents a buyer that bids, or not, for a target it considers it is able to monitor efficiently.

The implemented mechanism works through a first price sealed-bid auction based on a simplified auction protocol. A first price sealed-bid auction (FPSBA) is one in which the highest bidder wins the auction and pays the price it submitted for the resource. In this type of auction, all bidders submit secret (sealed) bids, meaning that no bidder knows what other bidders offered until the bids are revealed.

In an auction-based approach, the decision of who wins or not the auction is made based on the value of the bids. These bids must be well calculated and formed as to ensure that, as wished, only the fittest bidder will win. Bids are calculated following a set of rules that define how each parameter of interest to the auction will influence the bid. To solve the problem of distributing the moving targets among UAVs, a mathematical rule was created based on both UAVs' and targets' current state, as already mentioned above. (1) shows how this rule was implemented.

$$Bid(u,t) = K_a \left(\frac{1}{v_t (D_{ut} + K_d \sum_{i=1}^{n_{uq}} D_{ti}) n_{uq})} \right) \quad (1)$$

Basically, as is shown in (1), each bid of a UAV u for a target t is inversely proportional to the number n_{uq} of targets u has queued for monitoring. This ensures that UAVs that already have targets leave new targets to other UAVs, not overloading themselves. The value of a bid is also inversely proportional to the velocity v_t of the target t, so UAVs bid less for fast targets that will potentially be a problem to be monitored. The other parameters in this rule are the distances between the UAV and the target (D_{ut}) and the distance (D_{ti}) between the target being auctioned and the other n_{uq} targets already assigned to the UAV. These two parameters ensure that a UAV will bid higher for targets that are close to each other and close to the UAV itself. The rationale for this way of functioning is distributing targets spatially close to each other to the same UAV. As a consequence, not only a UAV will have to move less to monitor it's assigned targets, but also free UAVs will tend to win the rights over targets that are not close to any other target. Therefore, the goal of this implementation is, mainly, forcing UAVs to choose to monitor

groups of targets that are near to each other, bidding less for isolated targets.

To balance the system and ensure the best possible outcome, different combinations and relations among the factors that influence the bid, i.e. v_t, n_{uq}, and the spatial grouping of the target, have been tried before coming to the proposed cost function. This function in (1) has presented itself as the best solution in part because of its simplicity, not requiring multiple weights or parameters that could be hard to tune and could easily influence the system robustness. Moreover, multiplying the terms attribute them the same level of importance. Likewise, in order to adjust the effects of the variables on the bid the constants K_a=1m^2/s and K_d=25 were empirically chosen accordingly, through experimental analysis, to guarantee that the these desired effects are achieved. While K_d is a dimensionless factor responsible for adjusting the effects of D_{ti} on the overall result, K_a is a constant used to balance both n_{uq} and v_t effects and adjust the dimensions as to make $Bid(u,t)$ a dimensionless value.

Each of the auction sections in this work is composed basically of four main phases. In the first phase, called Announcement, an auctioneer agent advertises all possible bidders about the starting of a new auction session. During the second phase, Bid Awaiting, the auctioneer waits for bids for a certain amount of time before proceeding to the next phase. On the third phase, bids are analyzed, a winner is chosen and the bidders are informed of the result. In the last phase, the contract is expedited to the winner, giving it the right to use the auctioned resources. The phases where communication is necessary the system implements a 3-time retry mechanism. This mechanism ensures that errors or non-responses on a section due to communication issues are less prone to occur.

C. Time Dependent Genetic Solver

Since a UAV on the monitoring system may be responsible for several targets, it must be able to decide in which order it will visit these targets. The visits must be organized in a way to make sure that no target can move enough to be lost between two consecutive visits. This means that, by having visited a target A and proceeded to visit others targets, a UAV must return to visit A again before this target is able to move out of this UAV's field of view. The problem presented by this challenge is similar to a Travelling Salesman Problem (TSP) [20], In the TSP, an agent, called the traveler, must be able to decide the best route (meaning order of visits) to visit all the points of interest, usually referred to as cities in a map. The optimization problem found in this work is, actually, a special case of TSP called Time-Dependent TSP. In this type of TSP, the nodes representing the cities move, or change their location over time, exactly what happens with the targets in the problem tackled here.

Exact solutions for TSP problems are hard and very resource consuming to find. In order to obtain the optimal solution to a TSP, it would be needed to analyze all possible different routes and choose the best one. However exact, this kind of brute force approach is not suitable for real-world applications, presenting a computational complexity cost of $O(n!)$. They quickly become too time-consuming to run on commercial machines or embedded processors, even for small TSP problems. Because of these high costs to find the optimal solution, many different AI, Optimization and Heuristic methods are found in the literature to find acceptable local optimal solutions to specific cases of the problem.

Among the optimization methods used for solving TSP problems, Genetic Algorithms have been shown to be pretty efficient once they are able to find good local solutions to optimization problems [2], [21]. Genetic Algorithms (GAs) use the concepts of natural Darwinian evolution to solve complex optimization problems. The concept behind this approach is to evolve an initial population of candidate solutions to a problem until one of these solutions in this pool or population is considered good enough by the algorithm. [22]

Besides a good solution to solve TSP Problems, genetic algorithms are also proved to be effective solutions to control, coordinate, or generate paths to UAVs or mobile robots, as in [23] and [24]. Another example of such usage is [25], where a Genetic-Based Path Planning algorithm addresses the problem of designing the path a vehicle is supposed to follow to maximize the collected information from desired regions while avoiding flying over forbidden regions.

As stated on the beginning of this subsection, the main goal to be achieved by the genetic optimization algorithm is the organization of the visits in a way to make sure that no target can move enough to be lost between two consecutive visits. The optimization problem here is the minimization of the average maximum displacement between visits to the same target, considering all targets under the responsibility of a given UAV. Mathematically, this objective function can be expressed as follows:

$$min(average(f_A, f_B, f_C, f_D, ...)) \tag{2}$$

,where f_X $(f_A, f_B, f_C, f_D, ...)$ is the maximum distance target X (A,B,C,D...) moves between any two visits to it's position. In other words f_X can be calculated as:

$$f_X = max(d_1, d_2, d_3, ...) \tag{3}$$

,where d_1 is the distance that target X moves between the first to visits considered, d_2 the distance X moved between the second and third visits, and so on.

To solve this problem, a GA algorithm was specifically implemented to deal with it. Figure 3 shows a flowchart presenting the basic steps of this algorithm. Basically, each time a UAV runs the genetic solver, an initial population of 25 possible solutions to that specific UAV problem, considering its assigned targets, is randomly created. Each solution is represented by an array of targets expressing that specific solution genome. The arrangement of the targets in these genomes represents the visitation order of that specific solution. For instance, a solution represented by the genome ACD will guide the UAV to visit first the target A, then the target C and finally target D. Figure 4 shows a sample of a genome and the graphic representation of the solution it represents. The numbers close to the arrows provide information about the order each node of the graph is visited.

As seen in Figure 4, each genome is twice as long as the number of targets the UAV running the GA has assigned to itself. This means that if a UAV has n targets in its monitoring queue, the genome of the possible solution to that UAV specific TDTSP problem will have size $2n$. The double length of the genomes was implemented so that faster targets, that can move a lot in short intervals, can be visited more frequently than slower ones, which are almost standing still and do not move much over time. This approach lets the system address the different pooling rates required by these two different types of targets, granting that a fast target will not be lost during the interval between visits and that a slow target is visited enough so that it is not lost either. The consequence of this approach is that each target may be visited twice on the same solution, or visiting tour, however, it is also possible that some targets are visited multiple times while others are only visited once. This design choice will also lead to a longer revisit time between some of the targets than simply choosing the shortest path, trading the interval between visits for fewer target misses.

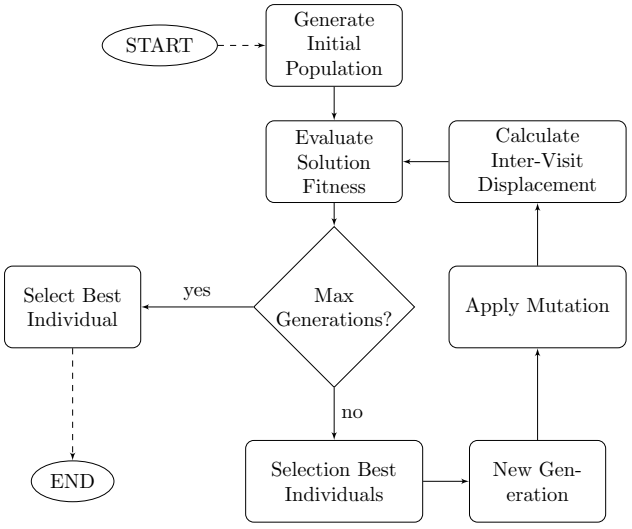

Fig. 3. Genetic Algorithm Flowchart

Once the first 25 solutions are created, they are analyzed according to the fitness function presented in (2) and (3). In the example presented in Figure 4, target A is visited three times on the tour. Supposing that between these visits A moved d_1 meters between the last visit and the first visit in this tour, d_2 meters between the first and the second visit, and d_3 meters between the second and third visit, the maximum value $f_A = max(d_1, d_2, d_3)$ will be chosen as the fitness of A in this solution. The fitness F_{sol} of that particular solution is then calculated as $F_{sol} = average(f_A, f_B, f_C, f_D)$. Having all those 25 solutions been analyzed, the three best solutions are chosen to minimize the value of F_{sol}, then the iteration is complete.

For each of the next iterations the 2 best solutions from the previous generation are copied until there are again 25 solutions on the new iteration solution pool, creating 12 copies of the best solution, 12 of the second best and 1 totally new random solution. These 25 solutions now on the pool suffer,

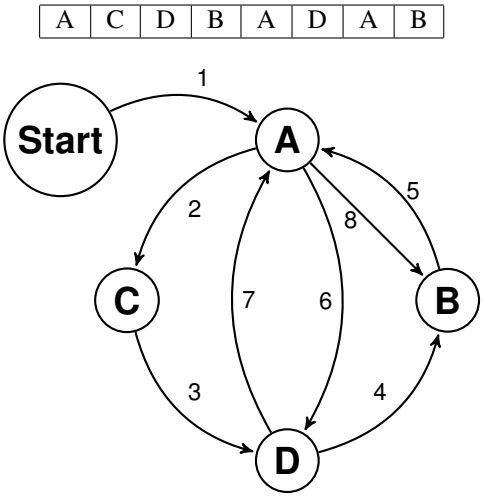

Fig. 4. Visit Diagram Representing a Possible GA Solution with Genome ACDBADAB

then, a mutation process. With a mutation chance rate of 20%, some of the targets on each genome are changed creating new solutions based on the copied ones. This mechanism makes sure that the newly generated solutions float around the best ones found on the previous iteration, presenting only small changes, but keeping their core almost unchanged. To implement this mutation mechanism, a Reverse Sequence Mutation [26] operator, was used. In this method, a random sequence of targets is taken from a tour, copied and then inversely placed in the sequence again. Besides simple, the RSM mutation operator has shown promising results in terms of convergence compared to other methods [26].

After mutating, the solutions are re-analyzed, a new iteration started, the best solutions copied and the mutation process repeated. The algorithm runs for a total of 5 iterations, analyzing roughly a total of 125 solutions for the problem. It is important to notice that the newly generated solutions are analyzed for completeness both at random creation and mutation phases of the algorithm. If a solution is found that does not contain all the targets the UAV has queued, this solution is discarded and a new one generated. This process guarantees that at the start of each iteration all 25 candidate solutions on the pool will visit each target at least once.

Concerning the parameters of the genetic algorithm, observations during the algorithms validation phase have shown that for the problem in hand, more than 5 iterations, or 25 individuals per generation, are not efficient parameters in terms of computational cost, consuming more time to run and resulting on very small fitness improvements compared to the implemented solution.

D. Handover Algorithm

The Handover algorithm is simply a target assignment transference mechanism. The implemented solution is activated once a UAV finds out, by analyzing the solution output provided by the genetic solver, that it will not be able to visit all targets on the next tour without losing one or more of them.

Once this fault is detected and the mechanism activated, the UAV chooses the target that is farthest from all the others. This is done by calculating the mean distance between each target to all the others. Then, the UAV starts the auction for this target, granting that some other UAV will bid and assign itself to handle this target.

The auction mechanism used here is the same auction mechanism used in the first allocation phase. The only difference is that now, the UAV auctioning the target is not allowed to bid for it since it would make no sense for this UAV to try buying something it already owns. As shown in Figure 2, it is important to notice that the handover algorithm may be run more than once, removing as many targets as necessary from a UAV responsibility and stopping only when a viable solution is found by the GA or when the UAV has only one target assigned to it.

In order to avoid a difficult target to be passed around infinitely, i.e. when a UAV hands over a target to another UAV, which by its turn immediately after also decides to hand over the same target, and this continues indefinitely back and forth, once a UAV takes the responsibility for a given target, it has to keep this for at least ten seconds. The UAV may consider skipping visits to this target, i.e. the UAV may not handle it during this time interval, but it is not authorized to pass it over before this timeout. This behavior is usually enough to change the system status, affecting bids and tours, thus avoiding this possible infinite handover situation, just by the natural movement of the UAVs and the targets.

As stated above, the conditions for activating this mechanism are related to the solution output provided by the GA solver, which is supposed to be the optimal, or quasi-optimal, solution for the set of the targets it was run over. The trigger is associated to a threshold on the solution fitness number, if the fitness of the GA solution is higher than this threshold level, meaning the average maximum distance targets move on that turn is greater than the threshold distance, the handover method is called. The threshold value of 20 meters was chosen according to the field of view of a RaspBerry PI Sony camera mounted on a UAV flying at an altitude of 45m.

V. Solution Implementation and Simulation Environment Setup

In order to correctly evaluate the proposed solution and analyze its behavior as realistically as possible, a simulated environment capable of modeling the behavior of both UAVs and targets was needed. Such a simulator needed to be able to mimic the real-time constraints of this type of system, considering targets' and UAVs' characteristics and ensuring that the processing platform would be able to handle all the concurrent tasks involved in the simulations. To achieve this goal, a custom ad hoc simulator was developed. Since the environment and scenarios proposed in this work involve more than just the UAV movement, simple commercial simulators were not fitted to be used in the evaluation, thus the need for a custom ad hoc solution.

In the proposed evaluation scenarios, when working with high-speed vehicles and moving targets, for example, it is important to assure that no outside variables influence in the simulation results. At high speeds and strict temporal constraints, even the slightest delay in processing some information or making some decision can heavily influence the activity outcome, sometimes crossing the line between success and failure. When monitoring threatening individuals on a crowd, for example, the system must be able to process a mission and target related information as soon as they come, treating them as a priority, risking to loose targets otherwise.With these constraints in mind, a Linux-based environment, featuring a controlled Java Virtual Machine that avoids non-deterministic preemptions to the main code, was set up to run the simulations and to ensure that no outside variables would influence them or their results.

Besides considering the real-time constraints of the application, the system was also supposed to deal with the high level of concurrency presented by the simulated scenarios. At the same moment during a simulation, both UAVs and targets are supposed to be moving. UAVs are supposed to be exchanging information over a communication network and they are also supposed to run their own control loops and internal algorithms. To achieve such concurrency, each UAV instance was modeled as a completely separated thread, acting, thus, independently from all other threads, sharing with them just the simulation environment parameters. Besides the UAV instances, three other modules were also implemented as threads to maintain concurrency: A UAV movement engine, responsible for managing UAVs movement behavior; A target movement engine, responsible for managing target's movements; and a Simulation Control Module, responsible for controlling the simulation parameters and simulate the communication network, besides supporting all other modules. For better understanding, the four types of threads and the general multithreaded architecture are diagrammed on Figure 5. It is important to notice that since the system is supposed to mimic real-time constraints and concurrency, the simulation must be run on a computational platform capable of running all threads simultaneously, running any two of them sequentially would break the concurrency principle, breaking the very concurrency conception.

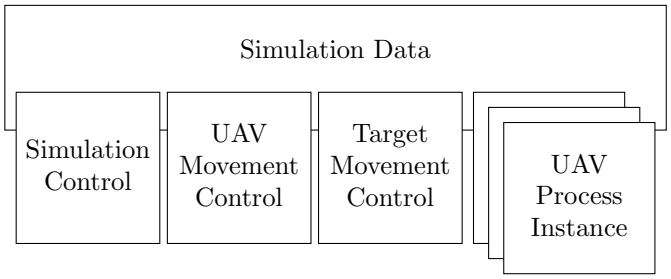

Fig. 5. Threaded Simulation Environment

The simulator and the experiments were also parametrized accordingly to mimic the behavior of the 3DR Iris+ quadcopter [27], a mini-UAV manufactured by 3D Robotics. The Iris+ 950 kV motors are capable of accelerating the 1282g UAV to around 20m/s, the flight autonomy stands between 16

and 22 minutes, depending on the payload. Considering this information, and the assumptions made in Section II of this work, the initial test environment was parametrized for 5 UAVs and 25 targets as presented in Table I. Target speeds were based on common movement speeds of individuals walking on a crowd.

TABLE I
SIMULATION PARAMETERS

Simulation Time	900s
Simulation Area	500x500m
Number of UAVs	5
Number of Targets	25
Target Initial Speed	random from 0 to 2m/s
Target Initial Heading	random
Target Speed Variation	random from $\pm$0m/s to $\pm$0.5m/s changing each 20s
UAV Speed	20m/s

VI. SIMULATION RESULTS AND ANALYSIS

A. Correct Behavior Tests

To properly evaluate the developed solution, a set of initial tests were designed to test if the system has a correct behavior according to what was expected, using the parameters presented in Table I. With this goal in mind, a set of 6 simulated scenarios was created. In the first four scenarios, the objective was to test the solution comparing 4 different cases with part of the solution modules active or deactivated. The first scenario is a simple run of the algorithm with no handover operation turned on and also no prediction on the targets' movement, meaning that at each visit the UAV searches a target on it's last known position, where it was last seen. The second simulation scenario has the handover mechanism still off, but the movement prediction is enabled. Now the UAVs search for targets based on their actual movement patterns, decreasing the chance of losing a target. The third and fourth scenarios follow the same pattern of scenarios one and two, one with no movement prediction, and the other predicting the targets' movement, but this time both using the handover mechanism. In all these four first scenarios, the targets maintain a constant speed during the whole simulation, not changing their speed under any circumstance. Table II summarizes these simulation setup scenarios.

TABLE II
SIMULATION SETUP

Scenario	Handover	Movement Prediction
Scenario 1	disabled	disabled
Scenario 2	disabled	enabled
Scenario 3	enabled	disabled
Scenario 4	enabled	enabled

The two last scenarios are variations of the third and fourth scenarios featuring small controlled variations on the targets movement patterns. In scenarios 5 and 6, targets change their speed in $\pm$0.5m/s every 20 seconds, simulating the subtle speed changes that are likely to occur in real life situations of people walking. These two last scenarios are summarized in Table III along with scenarios 3 and 4 with whom they are comparable.

TABLE III
SIMULATION SETUP

Scenario	Handover	Movement Prediction	Speed
Scenario 3	enabled	disabled	constant
Scenario 4	enabled	enabled	constant
Scenario 5	enabled	disabled	variable
Scenario 6	enabled	enabled	variable

All these scenarios were primarily evaluated in terms of two principal parameters, the mean number of visits to a target and the mean time between visits. The first statistic shows how many times a target was visited during the simulation, or how many times a UAV found that target during the simulation. It is important to notice that once a UAV may have been allocated exclusively for the same target for a long time, a low visit count is not necessarily a bad result. The second statistic is important to know how frequent these visits were, or how well distributed they were. A target with a low visit count, but a higher time between visits, for example, may have gotten less coverage than one with a lower interval between visits, meaning that the last had a UAV allocated to it for some time while the other one did not.

For each scenario, 100 simulations runs were performed and the presented results express an average from these runs. Table IV shows these results. Comparing the obtained results from scenarios one and two, it can be verified that the movement prediction mechanism activated on scenario 2 works efficiently to improve target coverage, augmenting the visit count in 29% and reducing the time between visits in almost 6 seconds. Such improvements mean that targets are visited more times and more frequently. Both improvements are important for the monitoring mission. These enhancements were already expected since the movement prediction mechanism leads the UAVs to intercept targets based on their movement patterns, avoiding the misses that would occur if the UAVs went looking for targets on their last known position.

TABLE IV
ALGORITHM EVALUATION SIMULATION RESULTS

Scenario	Average Visit Count	Average Time Between Visits (s)
Scenario 1	26.0	33.7
Scenario 2	33.6	27.8
Scenario 3	31.5	29.1
Scenario 4	35.9	25.4
Scenario 5	23.9	36.7
Scenario 6	24.4	34.2

Analyzing scenarios 3 and 4, that have the handover mechanism activated, it is possible to observe significant gains in coverage compared to the results with scenarios 1 and 2. Comparing scenarios 1 and 3, both with no movement prediction, an improvement of 20% can be seen in visit count statistics when the handover is activated. The handover activation also causes a reduction of 4.5 seconds in the

time between visits, confirming the expected improvement on coverage due to the redistribution of the targets between the UAVs. Similarly, comparing scenarios 2 and 4, both featuring movement prediction techniques, an improvement of 6% on visit count and less 2.4s on the time between visits confirm the handover positive effects on the target coverage.

Also, comparing scenarios 3 and 4 between them, the last one featuring movement prediction, the same expected positive effect caused by this mechanism can be verified. Targets in scenario 4 receive 4.4 more visits per simulation and are visited 3.7s more frequently than in scenario 3, reinforcing the idea that fewer targets are missed when the prediction mechanism is on.

Looking into the results from scenarios 5 and 6 it can be seen that, as expected, these simulations feature both fewer visits to the targets and greater times between visits when compared to 3 and 4. This result was expected because on these simulations a lot of misses occur due to the changes in the target speeds that are unknown to the UAVs and their movement prediction and genetic algorithms, making them look for targets on the wrong locations, or calculate a bad visit order. Scenario 5, for example, performs 7.5 fewer visits to targets than scenario 3, while scenario 6 performs 11.5 fewer visits than scenario 4, showing that even subtle changes in the movement pattern of the targets may result on serious degradation on the performance of the solution.

In order to illustrate the distribution of target to the UAVs, Figure 6 shows an example of how the UAVs (represented by the big dots circumscribed by a circle representing their FOV) spatially organize themselves and calculate the best routes to visit the targets (represented by the small dots in the figure). This a screenshot from a simulation of scenario four, and represents part of the scenario in which the system can be used. In this figure, it is possible to observe that with the movement prediction turned on during that simulation, the paths of the UAVs are pretty distant from a target's current position and to the point of a future position of that target.

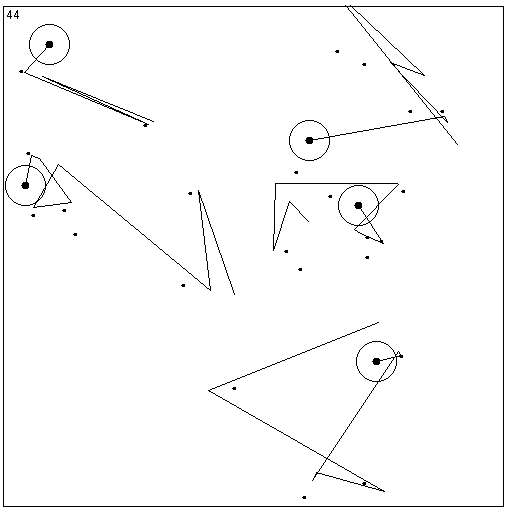

Fig. 6. Target Visit Paths in a Simulation Run from Scenario 4

B. Scalability Analysis

In order to evaluate how the number of targets per UAV reflects on the solution performance, scenarios 7 to 13 were designed changing the number of targets from 20 to 100, or, in other words, 4 to 20 targets per UAV. The setup of these scenarios regarding the number of targets and which algorithms are enabled is shown in Table V.

TABLE V
VARIABLE TARGET COUNT SIMULATION SETUP

Scenario	Number of Targets	Handover	Movement Prediction	Target Speed
Scenario 7	20	enabled	enabled	constant
Scenario 8	25	enabled	enabled	constant
Scenario 9	30	enabled	enabled	constant
Scenario 10	35	enabled	enabled	constant
Scenario 11	40	enabled	enabled	constant
Scenario 12	60	enabled	enabled	constant
Scenario 13	100	enabled	enabled	constant

Table VI shows the results obtained from these scenarios. It can be seen from these results that, as expected, the scenarios with fewer targets presented an increased performance on both the number of visits and the average time between visits. These results also showed a trend in how well the system performs when a high number of targets is confined to a small area. Observing the results from scenarios 10 and 11, it can be seen that no performance reduction was observed between these two cases. In fact, the performance drop from scenario 10 to scenario 11 was observed to be of only 0,0016%. Further inspection comparing these simulations showed that this result was observed due to the relation between the number of targets and the simulation space. According to the observed results, confining more than 35 targets in 250,000m^2 of simulation area led the targets to be packed close together, meaning that, in average, tours that visit 7 or 8 targets per UAV are not very different, causing the average time between visits to be the same.

Moreover, in packed crowds, the simple movement of a UAV searching for a target may cause it to run into another target, visiting it by hazard. This fact ends by increasing the observation parameter of the system as a whole, reducing the average time between visits and increasing the visit count. This behavior can be well observed analyzing the results of scenarios 12 and 13, in which even a great number of targets per UAV, i.e. 20, was not able to degrade the system results below a certain level. These results, however, will not be true to a case in which targets are not limited to such a restricted area. In larger environments, where targets are not so restrained and are free to move at will, these hazards visits would not occur (or rarely occur), meaning that such high target counts might indeed degrade the system until it is not capable to observe multiple targets anymore. However, this is a different type of scenario, compared to the one aimed to be tackled in this work.

In addition to these results, scenarios 14, 15, 17 and 18 were designed to verify how the system deals with the arrival of new targets. Basically, as shown in Table VII, these scenarios start with 20 target and new targets, 5 or 10 of them, are inserted

TABLE VI
VARIABLE TARGET COUNT SIMULATION RESULTS

Scenario Scenario	Number of Targets	Average Visit Count	Average Time Between Visits (s)
Scenario 7	20	36.05	24.97
Scenario 8	25	33.75	26.67
Scenario 9	30	30.60	29.41
Scenario 10	35	29.15	30.88
Scenario 11	40	29.15	30.88
Scenario 12	60	30.00	29.99
Scenario 13	100	30.03	29.95

either 1 by 1, evenly spaced during 5 minutes of simulation (300s), or all packed together at 150s. In all these scenarios target speeds are considered constant, and both movement prediction and handover algorithm are enabled, ensuring that the best performance is always achieved. Scenarios 16 and 19, which do not spawn new targets, but start already with the final target count, were also introduced as a baseline for comparisons.

Analyzing the results presented in Table VIII, it can be observed that scenarios that spawn targets during execution present, in average, better performance than those that already start with all the targets. Such behavior can be easily explained since those scenarios that start with 20 and spawn new targets, i.e. 5, have fewer targets to deal with on the beginning, presenting a good initial performance that slowly degrades as new targets are inserted. This behavior is even more remarkable when the cases with 1 by 1 and packed insertion are compared. Pack insertion presents high visits counts during simulation, and consequently an overall reduced time between visits, it can be assessed from the results that this interval is significantly increased after new targets are spawned. The 1 by 1 insertion, on the other hand, presents a graceful degradation in performance after targets are inserted. This is expected behavior since the targets are incrementally added, instead of abruptly inserted. These results show that the system is able to deal with new targets as their appearance, trying always to minimize the effects of these situations on the overall performance.

C. Experiments Considering Ordinary Urban Walking Scenario

Once the system was primarily designed to be used on urban environments, scenario 20 was developed to verify how the system handles a group of targets walking on a large urban avenue. The scenario is parametrized to simulate a 1500m x 50m city avenue, with 25 pedestrian targets moving along it. The targets were programmed to adopt simple ordinary human mobility behaviors [28], [29], moving longitudinally along the avenue with different speeds and movement patterns. Table IX provides details on how the simulation scenario was set-up, highlighting both UAVs' and targets' characteristics, showing which techniques and mechanisms of the algorithm are enabled.

For this scenario, 50 different simulation runs were executed. In all these runs, as can be seen in Table X, a 100% target observation coverage was verified, meaning that every time a UAV went after a target, it was able to generate a tour to find it, and actually find it where it was expected, not losing a single scheduled observation. It can also be seen, looking at these results, that the system has achieved a high visit count, visiting each target more than 70 times in 15 minutes, with an average interval of 11.76 seconds between visits. These results corroborate the idea that this system is highly able to efficiently perform in urban law enforcement tasks, i.e. demonstration or street party monitoring, completely fulfilling its primary goal. Moreover, notice that the results reported in this section are even better than those previously reported in the sections above. These better results are understandable due to the spatial distribution of the targets in this last scenario, that force them to stay closer, compared to those areas considered before.

VII. DISCUSSION OF THE RESULTS

In order to validate the solution adequacy, the results obtained on the previous section can be compared against the theoretical expected target movement behavior. Since this work focuses on monitoring targets that keep moving in known paths or straight paths (what usually happens in urban environments once they are limited to the streets) and targets that do not change their speed too often, under plausible conditions, some time constraints or requirements for the system can be defined as baselines for comparison. For a target that does not change its direction of movement, the maximum distance a target can deviate from the expected observation position between two observations is primarily defined in terms of how fast and by how much targets can change their speed. There are two possible scenarios for this case, both considering that the UAV has a circular field of view of 20m of radius:

a) Considering the worst situation, in which a still target starts to walk as soon as a UAV starts observing it, and that an average pedestrian walks, in a normal crowded situation, at 1.25m/s[28], it would take the pedestrian 16 seconds in the worst case to move far enough so that the UAV would not be able to find the target again in the next visit. The same value is valid for a target that is first walking at 1.25m/s and suddenly stops.
b) Considering a target that is already moving, it is a realistic assumption that a pedestrian usually will tend to keep its pace fluctuating around his current speed instead of suddenly stopping or going into a sprint. Considering a maximum deviation of $\pm$0.5m/s each 20s, it would take the pedestrian 30 seconds in the worst case to move far enough so that the UAV would not be able to find the target again in the next visit.

Analyzing the results from the previous section, most tests in which both the handover mechanism and the movement prediction algorithm are enabled have achieved average inter-visit times smaller than the 30 seconds calculated in case b, meaning that, in all situations, both considering open fields and limited urban-like environments (scenario 20), the system would be able, on average, to observe moving targets without losing them. For the open-field scenarios, however, the system cannot deal with targets that present a "sudden start-stop" behavior.

TABLE VII
SPAWNING TARGETS SIMULATION SETUP

Scenario	Number of Initial Targets	Number of Spawning Targets	Spawning Technique	Handover	Movement Prediction	Target Speed
Scenario 14	20	5	1 by 1	enabled	enabled	constant
Scenario 15	20	5	pack	enabled	enabled	constant
Scenario 16	25	-	none	enabled	enabled	constant
Scenario 17	20	10	1 by 1	enabled	enabled	constant
Scenario 18	20	10	pack	enabled	enabled	constant
Scenario 19	30	-	none	enabled	enabled	constant

TABLE VIII
SPAWNING TARGETS SIMULATION RESULTS

Number of Targets	25			30		
Target Spawn Technique	None	1 by 1	Pack	None	1 by 1	pack
Average Visit Count	11.2	10.4	11.4	10.2	9.6	10.6
Average Time Between Visits (Overall)(s)	26.7	26.4	24.6	29.4	27.3	25.38
Average Time Between Visits (After Spawn) (s)	-	26.3	28.9	-	29.1	29.8

TABLE IX
SCENARIO 20 SIMULATION SETUP PARAMETERS

Scenario 20	
Simulation Area	1500m x 50m
Number of UAVs	5
Number of Targets	25
Target Initial Speed	random from 0.5m/s to 2.0m/s
Target Moving Pattern	Human Mobility Behavior
UAV Speed	20m/s
Handover	Enabled
Movement Prediciton	Enabled

TABLE X
SCENARIO 20 SIMULATION RESULTS

Average Visit Count	76.50
Average Time Between Visits (s)	11.76
Target Observation Coverage	100%

Scenario 20, in turn, presents an inter-visit time of 11.76s, well below the 16s and 30s calculated on cases a and b respectively, highlighting the adequacy of the system to be used in urban environments. The results from this scenario show that the system is not only able to observe targets that do not change their speed abruptly (case b), but also able to observe targets that present a more erratic behavior, stopping and starting to walk again along the way (case a above).

Besides that, by calculating the intersection of the maximum motion range of the target and the UAVs field of view, it can be shown that, for scenario 20, the proposed algorithm can support speed variation steps of up to ± 1.5 m/s, combined with changes in direction of up to ± 68 degrees. This means that even targets that behave somewhat erratically, starting and stopping suddenly, and abruptly changing their movement direction between observations can still be monitored as long as they keep walking in the same general pattern.

In addition to the aforementioned results, scenarios 1 and 2 back up the extension to the genetic algorithm claimed as one of the contributions of this work, showing that the modified genetic algorithm is able to handle the TDTSP much better than a genetic approach to the common TSP. Once the movement prediction mechanism is activated, the average number of visits per target is increased by 29% and the average inter-visit time reduced by 17%. In this situation, scenario 1, not featuring the movement prediction mechanism, uses a solution to the common TSP approach, in which the targets do not move, to calculate its visiting path. Scenario 2, on the other hand, enables the movement prediction extension to the genetic solver, allowing it to solve the TDTSP considering the movement of the targets. This result reflects the fact that naive solutions to the general TSP only try to search and observe targets on their last known locations, being able to accurately observe only stationary targets, or those moving to slow to leave the field of vision of a UAV between two visits.

Finally, the scalability results complement and support the choice for a distributed approach in detriment of a centralized one. A distributed system increases the robustness of the solution once, in many cases, obstacles such as building and trees can momentarily interfere with the communication between the UAVs themselves and/or the UAVs and a ground control station, rendering a centralized algorithm unusable for some time. On a distributed scenario, on the other hand, UAVs can always communicate to their nearest neighbors to cooperate and coordinate the mission and the target distribution, even if some UAV, or a subgroup of them, are not able to communicate with the rest of the fleet. Furthermore, a centralized approach implies that a given node of the system, be it a ground control station or a specific UAV, is responsible for the coordination of the whole mission. This means that if this coordinator node is for some reason compromised or is unable to communicate, the whole mission can be compromised or even taken down. The results gathered from scenarios 14 to 19 enrich the discussion about decentralization showing that, even in cases in which some node is lost, other nodes may be able to take the responsibility on the targets of the missing node without a significant degradation on the overall mission performance.

VIII. CONCLUSIONS AND FURTHER WORK

This paper presents a proposal that enables UAVs to periodically monitor a group of moving targets that simulate the movement of walking individuals in crowded environments. The goal on developing this work was investigating how well

a group of UAVs can continuously monitor a large group of individuals (targets) in a crowd, alternatingly visiting each of them at a time while trying to not lose sight of any of these targets. A system equipped with a group of UAVs running this proposal can be used for law-enforcement applications, assisting authorities to monitor crowds in order to identify and follow suspicious individuals that can have attitudes that could be classified as vandalism or linked to terrorist attack attempts.

Simulations performed to test the developed approach have shown that it is indeed able to perform such monitoring task, visiting all the targets during the simulations, while still minimizing the time between visits made to these targets. The genetic algorithm has shown a great potential to solve this kind of moving target monitoring and TDTSP problems, and the implemented handover mechanism was able to handle target redistribution assertively.

Proposed future works on this project comprehend the analysis and study of the genetic algorithm and its parameters, such as the number of generations, genome and population size, fitness function and fitness analysis to try to enhance the system performance. These enhancements could be focused on augmenting the number of visits to the targets and reducing the time between these visits. Another important addition to the work would be the adaptation of the auction and handover mechanism. This adaptation would be mainly about tweaking the bid rules, to reduce the number of handovers performed during the system execution and the best allocation of the targets. Additionally, better simulation scenarios can be analyzed, such as those in which targets follow different patterns of restrict movement model, studying a larger variety of movement patterns of crowds taking part in demonstrations or riots.

ACKNOWLEDGMENTS

R. S. de Moraes is supported by the Sweden's Innovation Agency - Vinnova, as part of the national projects on aeronautics, NFFP7, project CLASSICS (NFFP7-04890). E.P. de Freitas thanks the Brazilian research funding agencies CNPq and CAPES for partially financing the developed project.

REFERENCES

[1] J. Keller. (2015, Aug.) Wrong again: Uav market heading nowhere but up over the next decade. [Online]. Available: http://www.militaryaerospace.com/articles/2015/08/uav-market-blog.html

[2] T. Bäck, U. Hammel, and H. paul Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Transactions on Evolutionary Computation*, pp. 3–17, 1997.

[3] E. Adamey and U. Ozguner, "A decentralized approach for multi-uav multitarget tracking and surveillance," vol. 8389, 2012, pp. 838 915–838 915–6. [Online]. Available: http://dx.doi.org/10.1117/12.918975

[4] A. Khan, B. Rinner, and A. Cavallaro, "Cooperative robots to observe moving targets: Review," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 187–198, Jan 2018.

[5] J. Kim and J. L. Crassidis, "Uav path planning for maximum visibility of ground targets in an urban area," in *2010 13th International Conference on Information Fusion*, July 2010, pp. 1–7.

[6] H. Oh, S. Kim, H. s. Shin, and A. Tsourdos, "Coordinated standoff tracking of moving target groups using multiple uavs," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 2, pp. 1501–1514, April 2015.

[7] L. E. Parker and B. A. Emmons, "Cooperative multi-robot observation of multiple moving targets," in *Proceedings of International Conference on Robotics and Automation*, vol. 3, Apr 1997, pp. 2082–2089 vol.3.

[8] Z.-S. Cai, L.-N. Sun, H.-B. Gao, P.-C. Zhou, S.-H. Piao, and Q.-C. Huang, "Multi-robot cooperative pursuit based on task bundle auctions," in *Intelligent Robotics and Applications*, C. Xiong, Y. Huang, Y. Xiong, and H. Liu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 235–244.

[9] A. M. Elmogy and F. O.Karray, "Cooperative multi target tracking using multi sensor network," *Int. J. Smart Sens. Intell. Syst*, vol. 1, no. 3, pp. 716–734, 2008.

[10] P. B. Sujit and R. Beard, "Distributed sequential auctions for multiple uav task allocation," in *2007 American Control Conference*, July 2007, pp. 3955–3960.

[11] P. K. Yadav and W. Meng, "Mobile targets localization in a field area using moving gaussian peaks and probability map," in *11th IEEE International Conference on Control Automation (ICCA)*, June 2014, pp. 416–421.

[12] D. J. Pack, P. DeLima, G. J. Toussaint, and G. York, "Cooperative control of uavs for localization of intermittently emitting mobile targets," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 4, pp. 959–970, Aug 2009.

[13] A. Kolling and S. Carpin, "Cooperative observation of multiple moving targets: an algorithm and its formalization," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 935–953, 2007. [Online]. Available: https://doi.org/10.1177/0278364907080424

[14] H. Chen, K. Chang, and C. S. Agate, "Uav path planning with tangent-plus-lyapunov vector field guidance and obstacle avoidance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 2, pp. 840–856, April 2013.

[15] F. Burkert and F. Fraundorfer, "Uav-Based Monitoring of Pedestrian Groups," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, no. 2, pp. 67–72, Aug. 2013.

[16] H. Abeledo, R. Fukasawa, A. Pessoa, and E. Uchoa, "The time dependent traveling salesman problem: polyhedra and algorithm," *Mathematical Programming Computation*, vol. 5, no. 1, pp. 27–55, 2013. [Online]. Available: http://dx.doi.org/10.1007/s12532-012-0047-y

[17] C. Helvig, G. Robins, and A. Zelikovsky, "The moving-target traveling salesman problem," *Journal of Algorithms*, vol. 49, no. 1, pp. 153 – 174, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0196677403000750

[18] B. Englot, T. Sahai, and I. Cohen, "Efficient tracking and pursuit of moving targets by heuristic solution of the traveling salesman problem," in *52nd IEEE Conference on Decision and Control*, Dec 2013, pp. 3433–3438.

[19] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973. [Online]. Available: http://www.jstor.org/stable/169020

[20] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton, NJ, USA: Princeton University Press, 2007.

[21] J.-D. Wei, *Approaches to the travelling salesman problem using evolutionary computing algorithms*. INTECH Open Access Publisher, 2008.

[22] J. H. Holland, *Adaptation in Natural and Artificial Systems*, 2nd ed. Cambridge, MA: MIT Press, 1992.

[23] R. L. Galvez, E. P. Dadios, and A. A. Bandala, "Path planning for quadrotor uav using genetic algorithm," in *2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, Nov 2014, pp. 1–6.

[24] A. Sonmez, E. Kocyigit, and E. Kugu, "Optimal path planning for uavs using genetic algorithm," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2015, pp. 50–55.

[25] H. Ergezer and K. Leblebicioglu, "Path planning for uavs for maximum information collection," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 1, pp. 502–520, Jan 2013.

[26] O. Abdoun, J. Abouchabaka, and C. Tajani, "Analyzing the performance of mutation operators to solve the travelling salesman problem," *IJES, International Journal of Emerging Sciences*, vol. abs/1203.3099, pp. 61–67, March 2012.

[27] 3DRobotics, "Iris plus quadcopter," 2014. [Online]. Available: http://3dr.com/support/articles/207358106/iris/

[28] Z. Fang, J. Yuan, Y. Wang, and S. Lo, "Survey of pedestrian movement and development of a crowd dynamics model," *Fire Safety Journal*, vol. 43, no. 6, pp. 459 – 465, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0379711207001270

[29] K. Maeda, K. Sato, K. Konishi, A. Yamasaki, A. Uchiyama, H. Yamaguchi, K. Yasumoto, and T. Higashino, "Getting urban pedestrian flow from simple observation: Realistic mobility generation in wireless network simulation," in *Proceedings of the 8th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '05. New York, NY, USA: ACM, 2005, pp. 151–158. [Online]. Available: http://doi.acm.org/10.1145/1089444.1089471

Rodrigo Saar de Moraes has received his MSc in Electric Engineering from the Federal University of Rio Grande do Sul in 2018, and his Bachelor degree in Computer Engineering from the the same university in 2015. He is currently a doctoral in the Graduate Program in Computer Science at Linköping University, in Sweden, working at the Real Time Systems Lab. In the past his studies focuses on Unmanned Aerial Vehicles (UAVs), especially related to the development of intelligent approaches to coordinate groups of UAVs to perform missions in challenging application scenarios. His current work is focused on high level architecture and network verification.

Edison Piganton de Freitas has a Ph.D. in Computer Science and Engineering from Halmstad University in Sweden, 2011, in the area of wireless sensor networks, MSc in Computer Science by Federal University of Rio Grande do Sul (UFRGS), Brazil, in 2007 and Bachelor degree in Computer Engineering by Military Institute of Engineering, Brazil, 2003. Currently, he holds an associate professor position at UFRGS, affiliated to the Graduate Programs in Electrical Engineering and to the Graduate Program in Computer Science, acting as a member of Research Group in Control, Automation and Robotics – GCAR and the Embedded Systems Lab. He works in several research areas of computer science and engineering, with a special focus to Real-Time and Embedded Systems particularly applied to Wireless Sensor Networks, Unmanned Aerial Vehicles, and Avionics Systems