Personal and Ubiquitous Computing
https://doi.org/10.1007/s00779-020-01379-2

ORIGINAL ARTICLE

SwarmCity project: monitoring traffic, pedestrians, climate, and pollution with an aerial robotic swarm

Data collection and fusion in a smart city, and its representation using virtual reality

Juan Jesús Roldán-Gómez [1,2] · Pablo Garcia-Aunon [1] · Pablo Mazariegos [1] · Antonio Barrientos [1]

Received: 5 May 2019 / Accepted: 6 February 2020

Abstract
Smart cities have emerged as a strategy to solve problems that current cities face, such as traffic, security, resource management, waste, and pollution. Most of the current approaches are based on deploying large numbers of sensors throughout the city and have some limitations to get relevant and updated data. In this paper, as an extension of our previous investigations, we propose a robotic swarm to collect the data of traffic, pedestrians, climate, and pollution. This data is sent to a base station, where it is treated to generate maps and presented in an immersive interface. To validate these developments, we use a virtual city called SwarmCity with models of traffic, pedestrians, climate, and pollution based on real data. The whole system has been tested with several subjects to assess whether the information collected by the drones, processed in the base station, and represented in the virtual reality interface is appropriate. Results show that the complete solution, i.e., fleet control, data fusion, and operator interface, allows monitoring the relevant variables in the simulated city.

Keywords Smart city · Robot swarm · Swarm intelligence · Data fusion · Immersive interface · Virtual reality

1 Introduction

The growth of cities is a fact worldwide: the population living in urban areas will increase from 54 to 66% in the next 30 years (United Nations report [1]). This phenomenon can lead to improvements in the management of resources, distribution of goods, transportation, and utilization of energy. Nevertheless, it can cause drawbacks such as traffic jams, pollution, noise, and security issues. The acquisition, processing, and visualization of urban data will be relevant to achieve higher levels of efficiency and safety.

Smart cities were conceived to deal with these challenges [2]. This approach applies information and communication technologies to ensure the sustainable development of urban areas, optimize the management of resources, improve the quality of life, and promote the participation of citizens. A prototypical smart city is focused on monitoring the city, understanding its state, and predicting its evolution [3].

The concept of smart cities is closely related to the concept of Internet of Things (IoT). In fact, most of the proposals involve the deployment of sensor networks throughout the cities [4]. Normally, these sensors are fixed in defined places and, therefore, only can collect data in these locations. The consequences are the appearance of biases in data and the reduction of the robustness of the whole system. An approach to address this issue is the integration of sensors in the public transportation system [5], which allows taking measurements in more locations, but not to control the sensor's positions.

✉ Juan Jesús Roldán-Gómez
jj.roldan@upm.es; juan.roldan@uam.es

Pablo Garcia-Aunon
pablo.garcia.aunon@upm.es

Pablo Mazariegos
p.mazariegos@alumnos.upm.es

Antonio Barrientos
antonio.barrientos@upm.es

[1] Centre for Automation and Robotics (UPM-CSIC), Technical University of Madrid, José Gutiérrez Abascal, 2, 28006 Madrid, Spain

[2] Present address: Department of Computer Engineering, Autonomous University of Madrid, Francisco Tomás y Valiente, 11, 28049 Madrid, Spain

Published online: 10 March 2020

The SwarmCity project[1] proposes the use of a swarm of aerial robots to monitor the state of a city, collecting relevant data about traffic, pedestrians, climate, and pollution, among others. The ultimate goal is to allow an operator, who can be the city government or any citizen, to know the state of the city, identifying relevant events and predicting future scenarios. In the case of government, this knowledge is useful to plan spaces, manage public services, and improve life quality. In the case of citizens, it is useful to plan their daily activities, such as traveling by private or public transport or avoiding activities in extreme weather conditions.

Why aerial robots? Because they are fast, agile, and practical and have a limited impact on the daily life of citizens. The use of drones in future smart cities, besides these clear advantages, will also present some issues that must be properly addressed [6]. Why a swarm? Because a group of simpler and lighter robots can perform different tasks in a parallel and robust way.

This paper presents the first integration of the SwarmCity project including a city simulator with traffic, pedestrians, climate, and pollution models; a simulated aerial swarm controlled by a behavior-based algorithm; and an immersive interface to allow operators to monitor the city. This system is better described in Section 3, explaining the elements taken from previous works and the main contributions of the present one.

The paper aims at studying the feasibility of the SwarmCity project concerning swarm intelligence, data fusion, and operator interface. Other aspects relevant for the implementation of this system in the real-world, e.g., the deployment and maintenance costs, drone load capacity and battery life, and flight regulations, are out of the scope of this work. Specifically, two relevant questions are answered: "Is the robot swarm able to measure the state of the city?" and "Are the operators able to understand what is happening in the city?"

2 Related works

As mentioned above, smart cities collect data about traffic, parking spaces, crowds, environmental conditions, and pollution, which can help to improve the management of public services, urban policies, sustainability, resource and waste management, and citizen participation. Most of the current smart cities are based on networks of static sensors and cameras, which are distributed in points of interest throughout the city [12]. However, some recent projects integrate sensors in public transportation systems, overcoming some of the issues derived from fixed sensors [13]. Moreover, there are research works that go further proposing crowdsourcing schemes, in which the sensors are installed in private vehicles and mobile devices [14]. The main contribution of the SwarmCity project is the possibility to move the sensors through the city, which allows measuring in the requested locations and times using fewer sensors.

The system proposed in this paper has some precedents in the literature. A complete survey on drones for improving smartness of smart cities can be found in [15], which classifies the applications of drones in seven categories: object detection and tracking, distributed processing, surveillance, data collection, path planning and navigation, traffic monitoring, and emergency services. Some of the tasks considered in this work are traffic monitoring, pollution monitoring, accident detection, fire detection, and network connectivity. This demonstrates that the variables considered in our work are relevant for the management of cities, as well as that drones are promising to measure them.

Another survey focused on drones for consumer applications can be accessed in [16], including applications of data relaying, monitoring and data collection, and product delivery, as well as challenges related to location accuracy, power consumption, communications issues, and public safety. This reveals a reality gap that must be considered by the SwarmCity project, to develop algorithms in simulated scenarios that in the future can be adapted to real ones. This reality gap encompasses not only some aspects related to the robotic platforms, such as their autonomy in terms of flight capability and battery life, but also some considerations in the algorithms of the swarm intelligence, data collection and fusion, and operator interface. In the present work, these latter challenges have been taken into account, whereas the former ones are considered out of the scope of the project.

A final survey on drones integrated with IoT can be found in [17], which considers multi-robot systems and utilization of technologies such as 4G/5G networks, cameras, sensors, and GNSS receivers. This work supports some of the proposals of the SwarmCity project, such as the communications of the fleet (e.g., drone-drone, station-drone, and drone-station), and the sensors integrated with the robots (e.g., cameras and environmental sensors).

Moreover, the literature contains some proposals about monitoring cities using aerial robots, each one focusing on a certain part of the system: drones, sensors, infrastructures, and communications. A framework for monitoring smart cities with drones is presented in [18], using urban infrastructure and public transportation for charging batteries, allowing communications and supporting tasks. Although the study of [18] thoroughly analyzes urban scenarios to find the real tasks, resources, and challenges, it does not address the development and implementation of the proposed system in simulated or real scenarios. The work described in [19] is

[1] The SwarmCity project is being developed by the Robotics and Cybernetics Research Group (RobCib) of the Centre for Automation and Robotics (CAR), which is formed by the Technical University of Madrid (UPM) and Spanish National Research Council (CSIC).

focused on planning the routes of aerial robots to optimize traffic monitoring, understanding this task as the periodic observation of traffic over the different streets of the city. This work considers a centralized architecture in which the base station represents the city as a graph with nodes and arcs, planning the paths through these arcs. Conversely, our work considers a distributed system in which the agents have to plan their trajectories having limited knowledge of the whole mission. Another application of drones for traffic monitoring is reported in [20], but the experiments are focused on the communications and video streaming between the Unmanned Aerial Vehicles (UAVs) and the Ground Control Station (GCS). Finally, the work described in [21] considers an aerial swarm for monitoring multiple variables of cities, focusing on the communications between agents and the data processing algorithms.

Three missions for drones in urban scenarios are considered in [22]: power-line inspection, civil-defense protection, and traffic monitoring. In this last case, drones are a promising alternative to conventional tools for measuring space-time variables, such as the traffic flow, speed, and density (also used in our work). Some requirements for drone operations in cities are posed in [23]: cameras with sufficient range, satellite positioning, hovering capability, automatic return in emergencies, high autonomy, and long-range communications (considered in our simulations). The accuracy of these platforms measuring microscopic traffic data is analyzed in [24], where some experiments detecting vehicles and pedestrians in intersections reveal video stabilization and geographic registration as the two key factors. While geographic referencing is considered in our simulations, video stabilization is not addressed, since it is important for the successful implementation on real platforms [25].

In addition to traffic and pedestrians monitoring, drones are widely used for remote sensing applications [26], such as measuring meteorological [27] and air quality parameters [28]. While measuring meteorological variables (e.g., temperature, relative humidity, pressure, solar radiation...) from a drone does not imply relevant complications, measuring the concentration of gases requires proper placement of the sensors in the drone [29] or the use of a capsule to take samples [30]. Given that these systems have been validated in real applications, we have integrated them into our simulations.

Regarding the control of aerial swarm and visualization of urban data, most of the works in the literature use conventional interfaces, but some recent works propose the application of immersive techniques. The work in [31] shows that VR is an appropriate technology to show spatial information in the context of smart cities, but their authors use a conventional screen to recreate virtual environments instead of an immersive device. A higher degree of immersion is reached in [32] employing a projector and 3D vision glasses. A comparison among multiple immersive technologies [33] reveals that head-mounted displays provide the highest degree of immersion, action presence, environmental presence, social presence, and engagement. Our work goes further taking advantage of this device to create a complete VR-based interface.

3 Developments

As introduced previously and shown in Fig. 1, the system developed in this work consists of four modules with different purposes:

- A city simulator that integrates models of traffic, pedestrians, climate, and pollution, described in Section 3.1.
- A drone swarm controlled by a behavior-based algorithm, which has been optimized to monitor cars, pedestrians, and environmental variables, as explained in Section 3.2.
- A base station that receives the measurements of drones and builds maps of traffic, pedestrians, temperature, and pollution, presented in Section 3.3.
- An operator interface that uses virtual reality (VR) to visualize the information, and voice and gestures to interact with the system, described in Section 3.4.

The four modules have been developed in different environments: the city simulator and operator interface using the Unity game engine, the swarm intelligence in MATLAB and the data fusion algorithms in Python. Three computers are used to simulate the whole system: one with Windows to simulate the city, another with Linux for controlling the swarm and treating the data, and another with Windows to run the interface. The four modules in the three computers are communicated through the Robot Operating System (ROS) [7], which manages the exchange of information between them, e.g., the urban data from the city to the interface, and the drones' actions from the swarm to the city.

In previous works, we developed the city simulator that integrates models of traffic, pedestrians, climate, and pollution [8]; the behavior-based algorithm that allows the aerial swarm to carry out search and surveillance tasks [9, 10]; and the data fusion algorithm to build traffic maps from car detections [11].

This work is an extension of [11] with the following contributions: generalization from traffic data to complete urban data; adaptation and optimization of the fleet control and data fusion algorithms to work with pedestrians, climate and pollution; and development and validation of an immersive interface to allow an operator to monitor the city.

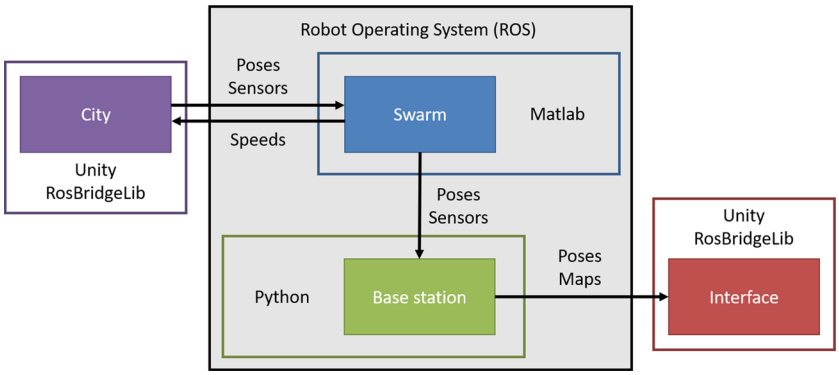

Fig. 1 Overview of the system, modules, and communications

3.1 City simulator

SwarmCity is a city simulator that reproduces a small European city.[2] As shown in Fig. 2, this city has a central district, two residential neighborhoods, an industrial area, a park and some public facilities, such as an airport, a train station, and a stadium. Additionally, this simulator includes models of traffic, population, climate, and pollution with behaviors that are inspired in real cities. For instance, the temperature and humidity depend on the date and time, the levels of pollution are higher in industrial areas than in residential neighborhoods, and there are agglomerations of people in workplaces in weekdays and leisure places during the weekends.

SwarmCity has been developed using the Unity game engine and the City Adventure, Road & Traffic System and Population Engine assets. Some research works use "Cities: Skylines" to generate complete cities with thousands of agents (people, vehicles, buildings...), such as [34] that simulates real urban processes and [35] that studies real estate and city planning. Other ones use "SUMO: Simulation of Urban MObility" simulator [36], which provides microscopic and macroscopic realistic data of traffic in a wide variety of road systems. We considered the use of these existing resources but finally decided to develop our city, using only open-source assets to build a light and fast prototype.

Although we have created a scene with a specific city, the project can be used to expand this city or create a new one from scratch. For this purpose, roads, facilities, buildings, and other urban elements must be designed, as well as traffic, pedestrians, climate, and pollution behaviors must be defined. The first task can be done intuitively using the unity interface, whereas the second one requires changing parameters or editing scripts. Nevertheless, the required effort is not great because some modules can be reused and others easily adapted. The main modules developed for this work are described with more details below.

[2] The city simulator can be downloaded from https://bitbucket.org/account/user/robcib/projects/SC

Currently, the city consists of 20 streets with different lengths (between 50 and 800 m) and shapes (straight and curves), and 22 intersections of various types (Ts, crosses, and roundabouts) with different rules (semaphores and priority). At the beginning of the simulation, a number of cars set by the user are spawned randomly on the roads. During the simulation, every car moves throughout the city at a certain speed that depends on the situation of traffic (light, heavy or jam), the type of section (straight, curve, or intersection), and the speed limit of the road (set by the user). When a car arrives at an intersection, it decides randomly the path to follow, and, depending on the situation, it moves immediately or waits until it has priority. As in real cities, intersections tend to accumulate cars, which can lead to traffic jams. Also, the user can generate traffic jams and select their locations and times.

The traffic of SwarmCity was analyzed to check if the model is realistic enough and, therefore, allows drawing relevant conclusions. Traffic models study the relationships between moving vehicles and infrastructures, to understand transport systems and optimize their designs. These models combine theoretical and empirical techniques and normally manage three variables: density, speed, and flow [37]. To analyze the traffic of SwarmCity, the roads have been discretized in square cells of $S \cdot S$ m^2. In this way, every detected car will be assigned to any of these cells, so that all the estimated traffic variables will be referred to these cells.

Traffic density is the number of vehicles per length unit that occupies a road section at a given moment. As described in Eq. 1, this variable (k) can be computed in our scenario as the number of cars detected in a cell (N) divided by the length of the side of the cell (S) and the number of lanes in the road (L).

$$k = \frac{N}{L \cdot S} \tag{1}$$

Traffic speed represents the distance covered per unit of time. There are two approaches to obtain this variable from the instantaneous speeds of the cars. On the one hand, *time mean speed* takes into account the cars that go through a certain point on the road during a given period. On the other

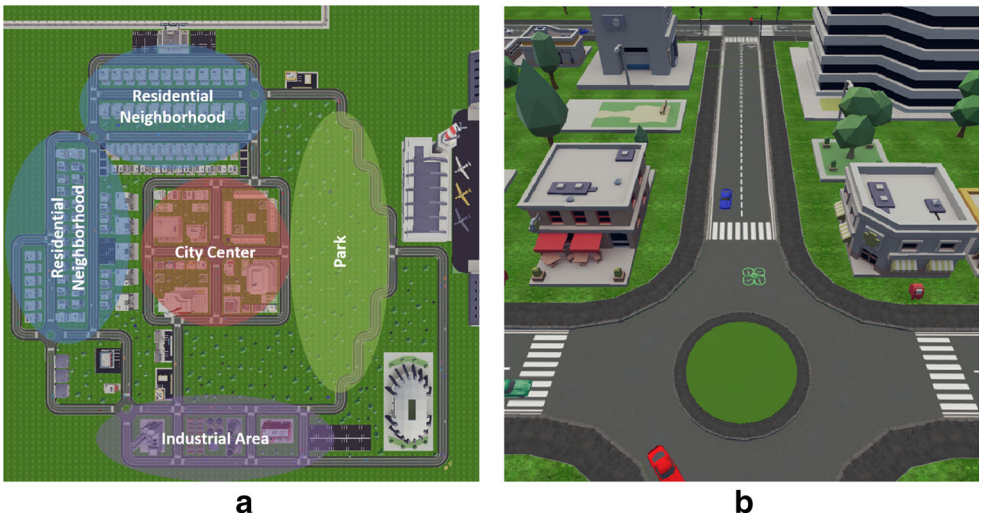

Fig. 2 SwarmCity: **a** bird's eye view of the city with its different areas and **b** a drone monitoring traffic over a roundabout

hand, *space mean speed* takes into account the cars that are located in a certain section of the road at a given moment. In this scenario, space mean speed is easier to implement because the drones are continuously covering sections of roads. Equation 2 shows the computation of traffic speed (V) as the average value of the individual speeds (v_i) of the vehicles detected in a cell (N).

$$v = \frac{1}{N} \cdot \sum_{i=1}^{N} v_i \tag{2}$$

Traffic flow is the number of vehicles that crosses a section of road per unit of time. As represented in Eq. 3, this variable (q) can be calculated as the product of traffic density (k) and speed (v).

$$q = k \cdot v \tag{3}$$

The measurements of traffic variables after simulating in SwarmCity for 8 h are collected in Fig. 3, which shows the pairs of flow and density values of multiple cars in different places and moments, together with the curves that represent the maximum and average values of flow for every density. This data is coherent with previous traffic modeling works [38] because it includes multiple roads with different features. For instance, the flow in the intersections will be lower than in the streets, since the vehicles have to reduce their speeds in these types of sections. In the same way, the flow in the curved sections will be lower than in the straight ones. Therefore, we can consider that the traffic of SwarmCity is enough realistic to use the generated data for future investigations.

The pedestrians of SwarmCity are created in 12 areas of the city and can follow hundreds of paths around them. Each of these spawn points shown in Fig. 4 (a) is activated at a given moment: e.g., the station and industry points at the beginning and end of the working days, the park and stadium points during the weekends, and the city center points both in working days and weekends. The number of people generated in each of these spawn points is random within a range that can be defined by the user.

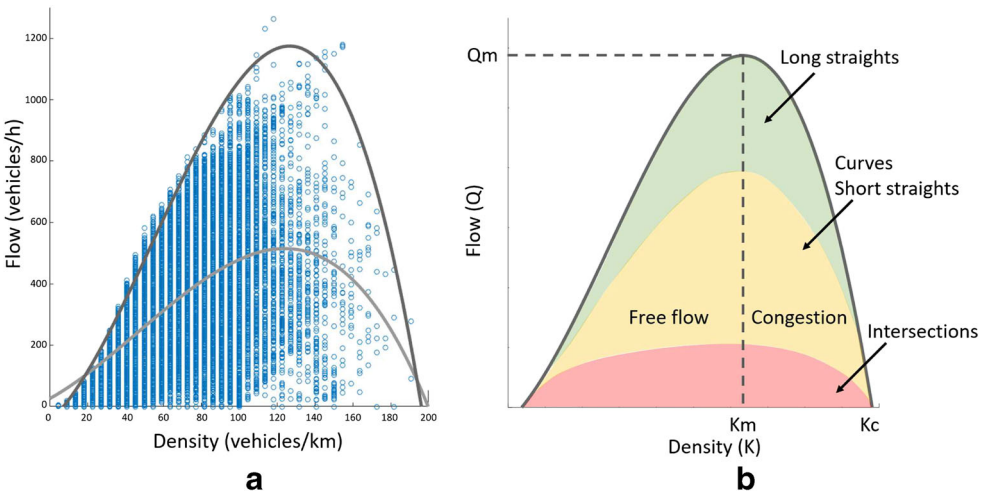

Fig. 3 Flow-density relationship: **a** data obtained from simulations (blue circles), maximum (dark gray) and mean curve (light gray) and **b** explanation of the different areas below the curve

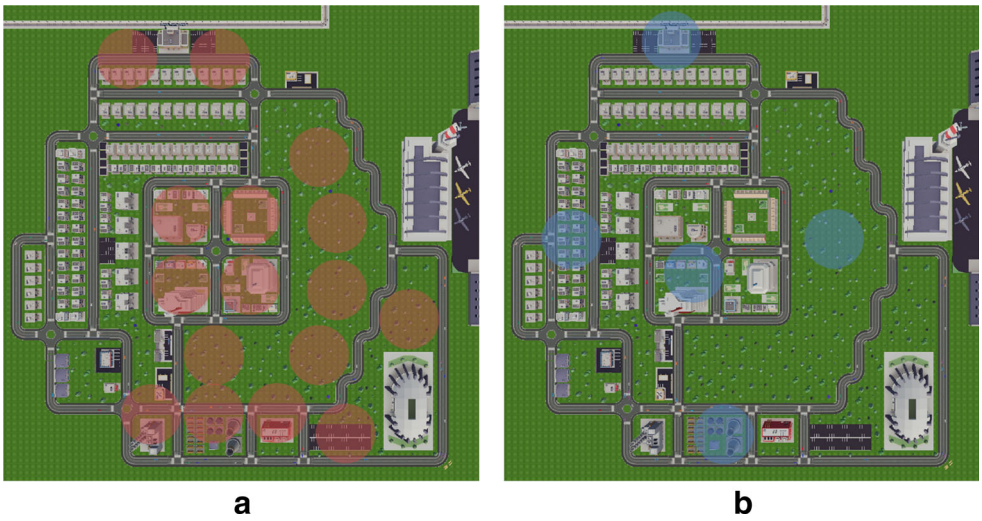

Fig. 4 Models: **a** spawn points for pedestrians and **b** reference points for the climate

Finally, the climate model of SwarmCity is based on the real data provided by the city of Madrid, Spain. This model integrates the following variables: temperature, humidity, rain, carbon oxides, sulfur oxides, nitrogen oxides, and solid particles. Each of the reference points shown in Fig. 4 (b) corresponds to a real weather station placed in a similar point of Madrid. As described in Eq. 4, the value of a climate variable V in a certain point p depends on the values of that variable in the reference points p_s with $s = \{1, 2, 3\ldots\}$ and the distances between the desired and reference points $d(p, p_s)$.

$$V(p) = \frac{\sum_{s=1}^{N} \frac{V(p_s)}{d(p,p_s)}}{\sum_{s=1}^{N} \frac{1}{d(p,p_s)}} \tag{4}$$

A video of the city simulator with traffic, pedestrians, climate, and pollution can be found in this link: https://youtu.be/YQ7wutNaqak

3.2 Swarm intelligence

The drone swarm is controlled by an algorithm originally developed for a search task in an open environment [9] and later adapted to perform traffic monitoring in SwarmCity [8]. Similarly to that works, the drones are considered to fly at a constant height of 20 m, so they can detect cars and people within a circle with a radius of 10 m. To organize the surveillance, the area is split into cells of 14.1 × 14.1 m and the drones move between their centers. Given that the amount of energy available in the batteries of the drones is limited, there are five recharging bases throughout the city. The agents visit these bases in periods of 5–10 min, depending on the energy consumption, and charge their batteries for 60 s, and then continue with the task.

The behavior-based algorithm is distributed, so every drone shares specific information with the others and individually decides the next cell to visit. As can be seen in Fig. 5, the algorithm is based on a network of seven behaviors (more detailed information can be found in [8, 10]). The six first behaviors (keep distance, keep velocity, energy saving, diagonal movement, collision avoidance, and surveillance) assign scores to the cells surrounding the current one, whereas the last one (final decision) computes the general score through these partial scores. The seven behaviors are explained below:

- Keep distance: Similarly to birds and fishes in nature, every agent tries to keep stable distances with the surrounding ones. For this purpose, repulsive forces act when the agents are closer than the equilibrium distance, whereas attractive forces act when the agents are further than that distance. See Fig. 6 (a).
- Keep velocity: In the same way, every agent tries to adopt the velocities of the surrounding ones. In this case, the influence of the other agents on a chosen one depends on their distances: the lower distance, the higher influence. See Fig. 6 (b).
- Energy saving: Every agent tries to consume the least possible energy in every movement. This energy consumption depends on the robot trajectory: keeping a

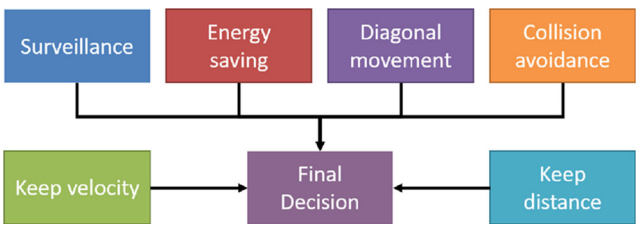

Fig. 5 Behavior-based algorithm adapted to surveillance tasks in the context of smart cities

direction saves energy, whereas turning right or left increases the consumption. See Fig. 6 (c).
- Diagonal movement: Diagonal movements reduce the overlapping between sensor footprints in comparison to the rest of possible movements. For this reason, the agents prefer these movements against vertical and horizontal ones. See Fig. 6 (d).
- Collision avoidance: Every agent gives priority to avoid collisions against the others over any other consideration. For this purpose, a strong penalty is used to avoid that two agents move to the same cell. See Fig. 6 (e).
- Surveillance: This behavior allows to explore the city monitoring the desired variables. It uses a map of pheromones to decide which direction to take next. Initially, the concentration of pheromones is uniform through the whole map. Then, every time an agent visits a cell, it removes some quantity of pheromones, reducing the interest to revisit that cell. However, the pheromones appear again as time goes by, given that cells should be revisited to get updated data. This behavior is slightly modified depending on the type of data to be monitored:

- Traffic: Two layers of pheromones are used: a first one (L1) for flying only over the roads, where the cars can be detected, and a second one (L2) for visiting more frequently the areas where a higher number of cars have been observed.
- Pedestrians: Similarly to the traffic task, two layers of pheromones are used: a first one (L1) for flying only over the city blocks, and a second one (L2) for visiting more frequently the areas with a higher density of pedestrians.
- Climate: In this case, a unique layer of pheromones is used to uniformly explore the whole map measuring the target variables (i.e., temperature and particles).

- Final decision: The previous behaviors generate scores for the surrounding cells according to their criteria. The final decision behavior computes a weighted addition of these scores and selects the cell with the maximum one.

The seven behaviors above described and shown in Fig. 6 have 23 parameters to be tuned for the cars and pedestrians observation, and 14 for the climate variables monitoring. The keep distance behavior has three parameters, the keep velocity one is influenced by a couple, and the surveillance one requires a total of 14. As an example, the keep distance behavior depends on the distance at which forces are not attractive nor repulsive ($\widetilde{d}_0$), the distance at which attractive and repulsive forces are maximum ($\widetilde{d}_{\max}$), and a distance coefficient that defines the desired distance (β_d). Moreover, the final decision behavior is different from the rest, since it is a weighted sum between the results of them, whose weights must be optimized too. More details can be found in [9]. Those parameters must be selected to maximize the efficiency of the algorithm. Therefore, three different configurations will be obtained: one to monitor cars, a second one to observe pedestrians, and a third one to measure climate variables.

The algorithm for monitoring traffic was configured in a previous work [8], optimizing the parameters in a simplified simulator and validating the configurations in SwarmCity. Although it was optimized to work with 10 drones and 150 cars, it achieves a good performance in a wide range of drones and cars. Similarly, the algorithms for monitoring pedestrians and climate have been optimized in the same lighter and faster simulator, and then the five best configurations have been tested in SwarmCity, selecting the best one. For this purpose, two performance metrics are established—one for traffic and people detection, and another for climate monitoring—and a genetic algorithm is used to optimize them. More details of these optimizations are presented in Section 4.1.

3.3 Data fusion

The problem of data fusion considers N_D drones that are moving through the city while measuring the traffic, pedestrians, temperature, and pollution. Every drone takes measurements of these variables (m_d) every time t at its location (x, y). The objective of the proposed algorithm is to fuse these measurements ($m_d(x, y, t)$) obtained by the drones ($d = \{1, \ldots, N_D\}$) and build maps ($M(t)$) as similar as possible to the ground truth measurements ($M_{ref}(t)$). For this purpose, we developed three basic methods to create a map M at the time T from a set of measurements m_d obtained in previous instants $t < T$, where M and m_d can represent traffic, pedestrians, temperature, pollution, or any other variable of interest.

- Method 1: This method builds the map taking into account the most recent measurements of the drones. Initially, the map is a matrix with all the elements equal to -1. Then, when a drone obtains a measurement at a certain location, it is added to the map replacing the previous value. The result is a map that contains the most recent data for every cell, but may be vulnerable to the noise produced by traffic disturbances and sensing errors.
- Method 2: This method builds the map M computing the mean of the measurements M_t in a certain time window before the current time $t = T - W, \ldots, T$, as defined by:

$$M = \frac{1}{W} \cdot \sum_{t=T-W}^{T} M_t \tag{5}$$

- If the drones take several measurements at a point and do not come back during the time window, the mean of these

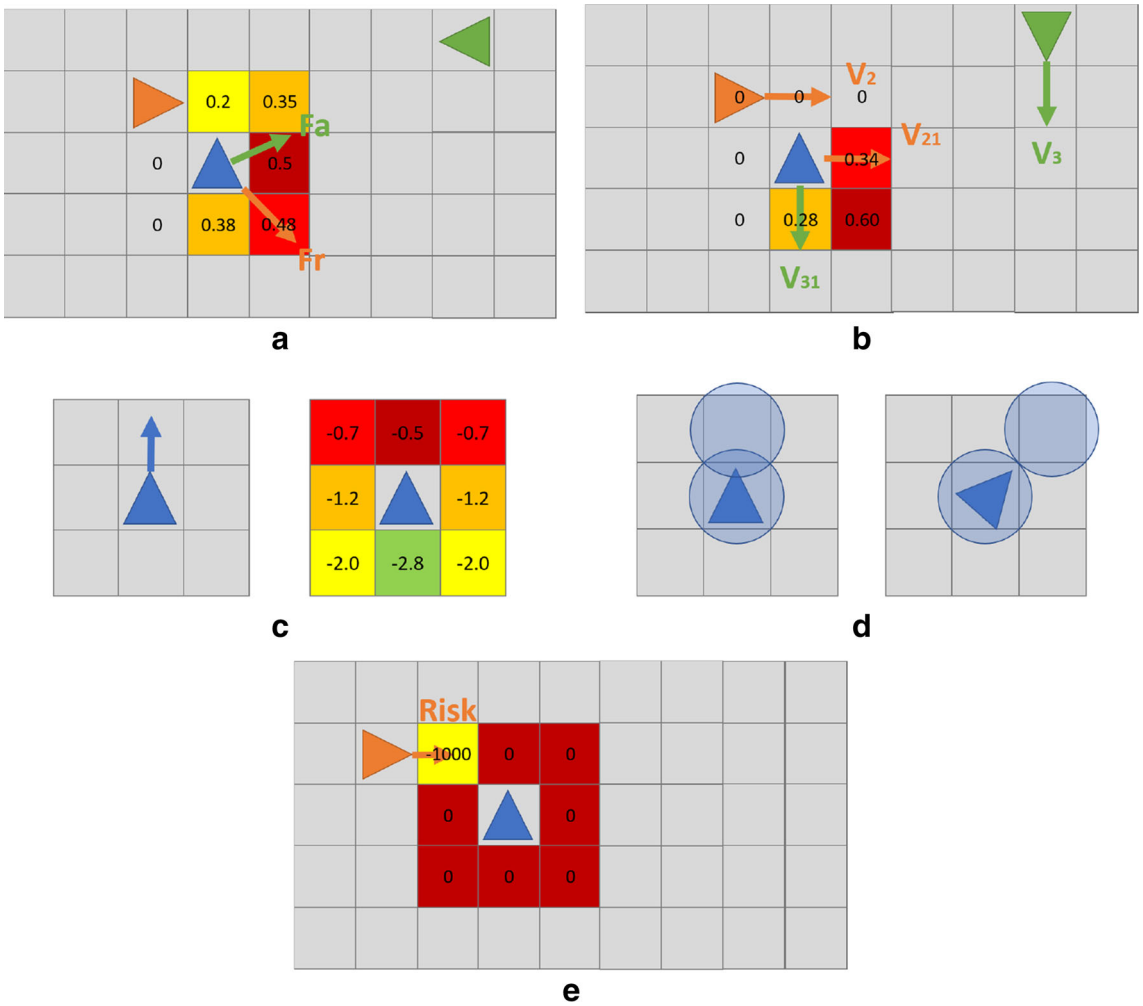

Fig. 6 Behaviors: **a** keep distance, **b** keep velocity, **c** energy saving, **d** diagonal movement, **e** collision avoidance

measurements remains on the map. The time window W can be tuned to minimize the error between the estimated and real maps. This method provides more stability and robustness but may take a long time to detect changes in measured variables. As can be checked, the method 1 is a particular case of the method 2 when $W = 1$.

– Method 3: Like the previous one, this method builds the map M from the measurements M_t in a certain time window W, keeping the most recent measurements when there are no more in this period. Moreover, this method applies a weighted mean to give more importance to recent measurements. As shown by Eq. 6, the weights are generated through a negative exponential function whose shape depends on a time constant T_c:

$$M = \frac{\sum_{t=T-W}^{T} e^{-\frac{T-t}{T_c}} \cdot M_t}{\sum_{i=T-W}^{T} e^{-\frac{T-t}{T_c}}} \tag{6}$$

– In this way, two parameters (W and T_c) must be tuned to minimize the error between the estimated and real maps. This method is still robust against noise and may adapt better to changes. Method 2 is a particular case of method 3 with infinite T_c.

The three methods were optimized for traffic data in a previous work [11] and method 3 with $W > 150$ and $T_c = 10$ provided the best results. In this work, the three methods are optimized for pedestrians and temperature/pollution data and their results are compared in Section 4.2. The results for traffic, pedestrians, and temperature/pollution are expected to be different, since the distribution of these variables and the fleet control are not the same. If these results are not satisfactory, other models will be studied, such as Kalman or particle filters.

3.4 Operator interface

An interface is required to show the data obtained by the drones in the simulated city and processed by the data fusion

algorithm in the base station. As previously explained, this data is related to traffic, pedestrians, temperature, and pollution, and must be shown in maps with different ranges of colors.

The interface has been developed using immersive technologies to enhance the perception and reduce the workload of operators. Previous studies support this hypothesis in different contexts, such as the training of industrial operators for assembly tasks [39] and the control and monitoring of multi-robot missions [40, 41]. In this case, immersion can improve the understanding of the information, as well as allow to intuitively command the swarm.

Among the different immersive technologies, virtual reality has been chosen against augmented reality (AR), since it is more suitable for the exploration of extensive areas with large amounts of data [42]. VR requires the integration of different elements to deal with data description and management, display or rendering techniques and integration of users in simulation loops [43]. In addition to virtual reality, the interface provides operators with natural language and gestures commands to configure the point of view and the shown information.

The main purpose is to improve the capacity of operators to monitor high numbers of robots and understand large amounts of information, considering that they have not been extensively trained and have short periods. In this particular case, the operators need to know what is happening in the city, where are the drones, and where it is necessary to explore, whereas the interface must achieve an understandable way to show them all this information. Immersive interfaces must turn the user subjective impressions into comprehensive and realistic experiences. As a way of immersion, VR is called to be applied for supporting many tasks soon, such as education [44], medical interventions, and industrial design [45].

The VR interfaces use specialized hardware to generate the immersive experience, increasing the fidelity in the reconstruction of scenarios and providing the operators with tools for interaction. In this work, we used the HTC Vive head-mounted device (HMD), as well as the Leap Motion hand detector. The combination of both devices makes possible the integration of the hands of operators in the VR scenario, as well as the implementation of natural language to control the interface.

The HTC Vive HMD is used to visualize the virtual city with the relevant information and to manage the interface voice control. It consists of a head-mounted display with two screens and headphones, two base stations that allow determining its position and orientation in the room, and two controllers that can be used for interacting in the VR environment. Moreover, the Leap Motion is a small gadget formed by two cameras and three LEDs, which can detect and render both hands in VR environments. This device allows operators to use their own hands instead of the HTC Vive controllers, which gives way to a much more natural and intuitive interaction. Figure 7 shows the Leap Motion installed on the front side of the HTC Vive glasses.

The VR environment was created by using Unity, a game engine that is widely used in the videogame industry, and a set of assets, packages produced by ourselves or third parties. The Unity project reproduces the city with its buildings, roads, and other equipment in the same locations. Apart from this main structure, the interface integrates multiple maps (traffic, pedestrians, temperature, and pollution), so the operator can change among the variables to evaluate the state of the city.

Among the assets used in this project, the Leap Rig package is especially relevant, since it includes the Hands Models and Event Systems, which make possible the hand detection and gesture control. First, the Hands Models asset detects all the phalanges of the hands, allowing to know whether the hands are opened or closed, as well as any other interesting position. Second, Event System asset manages everything related to the representation of hands in the scene.

The interface hosts five maps: traffic, which represents the traffic density in vehicles per kilometer; pedestrians, which represents the number of people per 100 square meters; temperature, which shows the temperature in Celsius degrees; pollution, which shows the pollution particles in parts per million; and maximum values, which points out the locations with the highest values of traffic, pedestrians, temperature, and pollution. All the maps instead the latter one use the same range of colors and albedo: the lower values are represented in green tones and more transparent, whereas the higher values are shown in red tones and less transparent. Figure 8 shows the interface with maps of different variables.

As previously stated, the interaction of operators with the interface is based on natural language and gestures. On the one hand, the voice commands collected in Table 1 allow users to interact with the interface, changing the active map and modifying the city. On the other hand, the gestures shown in Table 2 and Fig. 9 are used to move on the scene and search

Fig. 7 Leap Motion hand detector with HTC Vive HMD. Source: Leap Motion

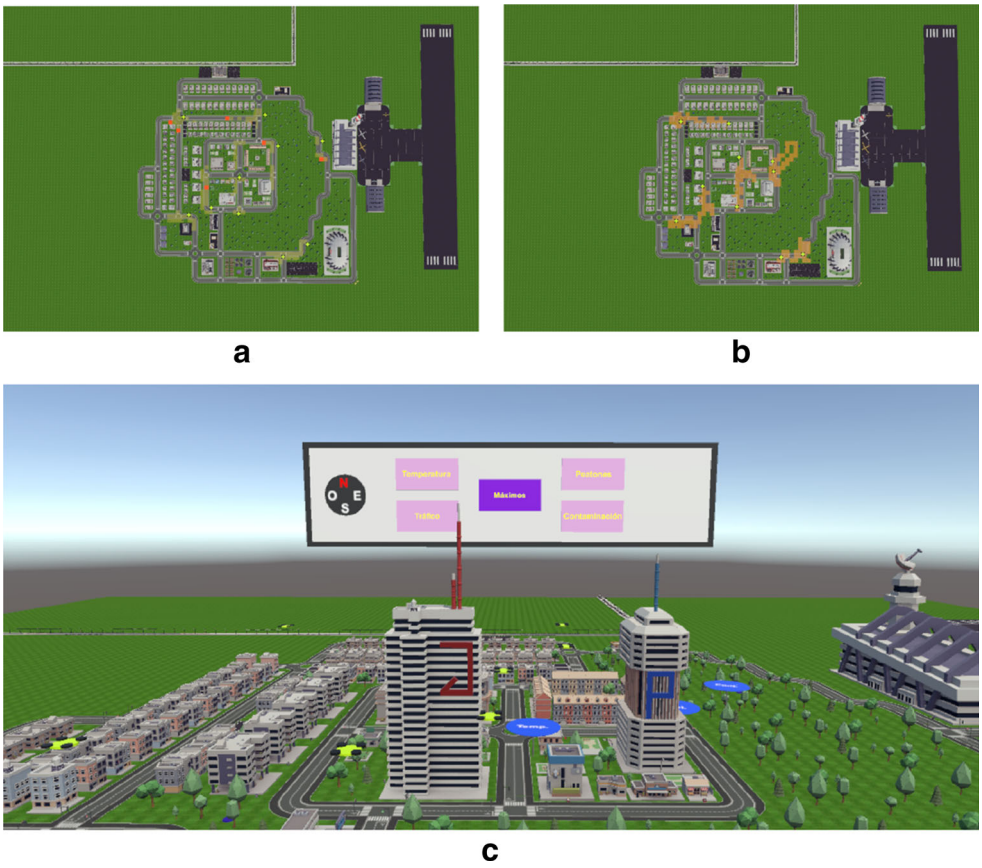

Fig. 8 Maps: **a** traffic (top view), **b** temperature (top view), and **c** maximum values (user view)

the best point of view. Finally, as shown in Fig. 10, the operators can teleport to a location in the city pointing with the index finger at this location and saying the command "jump."

A video of an operator using the interface to monitor the city can be seen in this link: https://youtu.be/jWhyRjXLVEs.

4 Experiments

The experiments were performed to validate the whole system: swarm control, data fusion, and operator interface. As introduced above, we wanted to answer two questions: "Is the robot swarm able to measure the state of the city?" and "Are the operators able to understand what is happening in the city?"

To answer the first question, the real and measured maps of traffic, pedestrians, climate, and pollution are compared. For this purpose, the behavior-based algorithm has been optimized to search pedestrians and measure temperature/pollution (see Section 4.1), as well as the data fusion algorithm to build maps from the measurements of these variables (see Section 4.2).

To answer the second question, a set of experiments with operators using the interface to monitor the city has been performed. In these experiments, we have checked if the operators understand the most relevant events that happen in the city, as reported in Section 4.3.

Table 1 Voice commands

Voice command	Interface action
Traffic	Shows the map of traffic.
Pedestrians	Shows the map of pedestrians.
Temperature	Shows the map of temperature.
Pollution	Shows the map of pollution.
Maxima	Show the map of maximum values.
Clean	Remove the buildings to make the maps clearer.
Restart	Recovers the buildings after they were cleaned.

4.1 Swarm intelligence

As previously mentioned, the drones are controlled by a behavior-based algorithm with 7 behaviors and 23/14

Table 2 Gesture commands

Gesture command	Interface action
Thumbs up/down	The player ascends/descends in the scene.
Fists	The player moves in the direction of his/her fists.

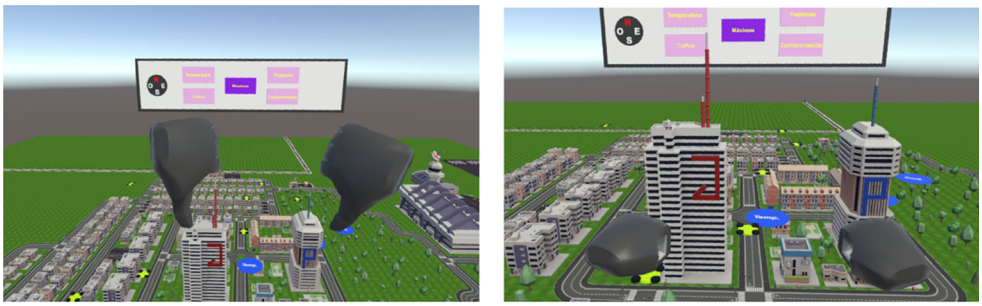

Fig. 9 Down movement executed by the user

parameters. Three variants of this algorithm are taken into account: the first one for traffic monitoring, the second one for pedestrians' observation, and the final one for climate/pollution monitoring. The three algorithms have been optimized by using simplified simulators. The first one creates cars that appear in certain points inside the roads, stay during a time window in those positions and then disappear. Afterward, new cars are created in different positions. The second simulator is similar to the first one, but creating pedestrians outside the roads. In the last simulator, the swarm only covers the whole city equally, without distinguishing between areas with or without roads.

The algorithm has been optimized to maximize the efficiency in every simulator:

- For cars/pedestrians monitoring, the efficiency is defined as the number of cars/pedestrians observed during each time window, divided by the total number of cars/pedestrians.
- For climate monitoring, the efficiency is defined as the minimum achievable mean age, divided by the mean age of the information in the map. The age of the information associated with a cell is defined as the time elapsed since the last information was obtained. Given that the amount of drones available is limited, the minimum achievable mean age is simply estimated by:

$$a_i = \frac{1}{2}\frac{A_{SC}}{N_a \cdot v_n \cdot 2 \cdot R_f} \tag{7}$$

where a_i is the ideal age, A_{SC} is the area of the city, v_n is the nominal flying velocity, and R_f is the radius of the sensor footprint.

The optimization of the algorithms has been performed using a genetic algorithm, which is similar to the used in [8] and has the following features:

- Individuals: The 23/14 parameters normalized between 0 and 1 (see Table 3).
- Population: 20 members initialized randomly.
- Fitness: Efficiency in the simplified simulator.
- Crossover: Roulette wheel technique with probabilities proportional to fitness values.
- Mutation: Random with a probability dependent on the diversity of the member compared to the population.
- Selection: the best 20 members from the 40 available.
- Termination: Maximum number of generations (20), generations without an improvement of 10% in the best member (3) or generations without an improvement of 10% in the mean of the population (2).

The optimization of the algorithm for traffic was performed in the previous work [8], providing the parameters shown in

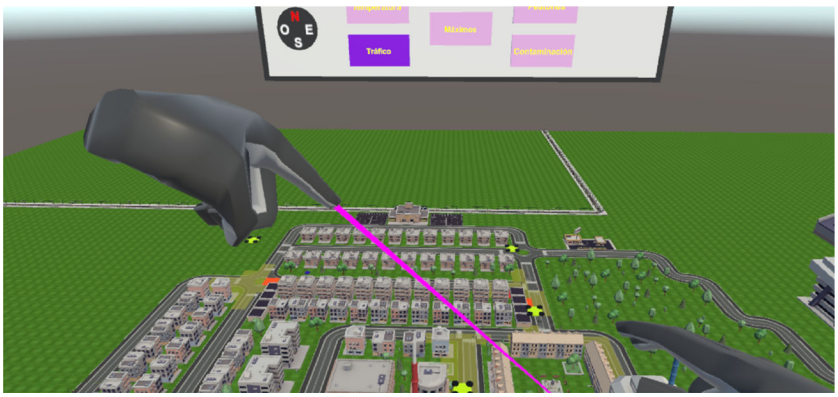

Fig. 10 User shooting a ray for teleport in the city

Table 3 Configurations of the algorithm for traffic, pedestrians, and temperature/pollution monitoring

Parameter	Description	Traffic	Pedestrians	Temp/ Pol
Surveillance				
ϕ_0^{L1}	Init. phe. value, L1	187.3	105.2	123.6
ϕ_0^{L2}	Init. phe. value, L2	66.6	48.5	–
S_{roads}^{L1}	Source of phe., roads	27.0	26.4	18.5
$S_{\text{no-roads}}^{L1}$	Source of phe., no-roads	4.80	7.06	(18.5)
S^{L2}	Source of phe., targets	1.6	6.07	–
D_{roads}^{L1}	Diff. coeff., no-roads	0.64	0.17	0.40
$D_{\text{no-roads}}^{L1}$	Diff. coeff., roads	0.49	0.87	(0.40)
D_{roads}^{L2}	Diff. coeff., no-roads	2.92	1.56	–
$D_{\text{no-roads}}^{L2}$	Diff. coeff., roads	1.07	1.58	–
δ^{L1}	Drop of phe., L1	−0.09	−0.60	−0.85
δ^{L2}	Drop of phe., L2	−0.44	−0.31	–
β_Φ	Evaluating mode	0.16	0.50	0.64
$\Delta t_{\text{targets}}$	Time window	212.2	134.8	–
R_{targets}	Inhibit radius	16.7	57.13	–
Keep distance				
$\widetilde{d}_0$	$\widetilde{d} \mid \boldsymbol{F}_{t_j}^i = 0$	13.6	11.8	24.2
$\widetilde{d}_{\max}$	$\widetilde{d} \mid \boldsymbol{F}_{t_j}^i$ is maximum	49.7	28.0	50.0
β_d	Distance coeff.	0.92	0.50	0.40
Keep velocity				
α_v	Distance coeff.	4.96	−0.22	−5.1
Velocity control				
v_n	Nominal velocity	7.6	10	4.0
Final decision				
α_E	Energy cost coeff.	157.8	94.3	134.2
α_{DM}	Diagonal movement coeff.	105.2	18.4	112.6
α_D	Keep distance coeff.	85.4	75.0	66.0
α_V	Keep velocity coeff.	74.6	50.5	105.0

Table 3 and efficiencies of 26% in the simplified simulator and 25% in SwarmCity. The optimization for pedestrians finished after 11 generations because there were no improvements in the best member and mean of the population, reaching the efficiencies shown in Table 4 (41% in the simplified simulator and 11% in SwarmCity) with the configuration collected in Table 3. Finally, the optimization for temperature/pollution finished after 13 generations because there were no improvements in the best member and mean of the population, reaching the efficiencies shown in Table 4 (59% in the simplified simulator and 48% in SwarmCity) with the configuration collected in Table 3. In the case of pedestrians' search, the difference between the efficiencies in simplified and SwarmCity simulators is remarkable and can be attributed to the differences in the number of pedestrians (lower in SwarmCity) and their distribution in the map (more heterogeneous in SwarmCity). However, we have observed that the best members of the simplified simulator also achieve the best scores in SwarmCity.

4.2 Data fusion

As already mentioned, three methods are proposed to fuse the data provided by the drones and build maps of the target variables. Method 1 takes the last measurements at every location of the city to build the maps, whereas methods 2 and 3 respectively use means and weighted means in certain time windows. In a previous work [11], the algorithm was optimized for traffic data and method 3 with $W > 150$ and $T_c = 10$ provided the best results. In this section, the algorithm is optimized for pedestrians and temperature/pollution data.

Four 1-h-long simulations with 10 drones monitoring pedestrians and temperature/pollution were performed to generate data and test multiple configurations of the algorithm. As

Table 4 Efficiencies of the five best members in pedestrians and temperature/pollution

Configuration	Simulator	SwarmCity
Pedestrians		
1	0.41	0.11
2	0.41	0.08
3	0.41	0.07
4	0.41	0.05
5	0.41	0.05
Temp/Pol		
1	0.59	0.48
2	0.58	0.48
3	0.58	0.46
4	0.58	0.45
5	0.58	0.44

can be seen in Fig. 11, four maps are obtained every second: two that show the real pedestrian concentration and temperatures in the city, and two that collect and process the values measured by the drones. As in the previous work, the real maps have been averaged in a time window of 30 s, to determine permanent situations instead of instantaneous phenomena. As shown in Eq. 8, we can define the error e as the difference between real and estimated maps (M_{ref} and M):

$$e = \sqrt{\frac{1}{N_P} \cdot \sum_{(i,j):M(P)\geq 0} \left(M_{ref}(P) - M(P)\right)^2} \tag{8}$$

Only the N_P points with measurements $(P \mid M(P) \geq 0)$ are taken into account. This variable is useful to measure the performance of data processing algorithms but may produce disturbances when evaluating swarm control algorithms. For instance, a configuration that leads the drones to be static at certain points could produce less error than another that allows a wider exploration of the map, if the second one does not visit periodically all the points and keeps non-updated data.

We applied the three methods with different parameters: method 2 with $W = \{10, 20, 30, 40, 50, 60, 90, 120, 150, 180, 210, 240, 300, 360, 420, 480, 540, 600\}s$ and method 3 with the same time windows W and $T_c = \{10, 50, 100, 500, 1000\}s$. The errors for pedestrians and temperatures obtained with the proposed methods and their dependence on the time windows are shown in Fig. 12. As can be seen, the best results for pedestrians are provided by the method 2 with $W \in [120,180]$, whereas in the case of temperature are provided by the method 2 with $W < 40$. Therefore, method 3 with $W = 150$ and $T_c = 10$ is used for traffic monitoring and method 2 with $W = 120$ and $W = 30$ for pedestrians and temperature monitoring hereinafter.

4.3 Operator interface

A set of experiments were performed to validate the immersive interface developed to monitor the state of SwarmCity. For this purpose, 33 volunteers were recruited from the university, including BSc, MSc and PhD students with ages between 20 and 32. In these experiments, each

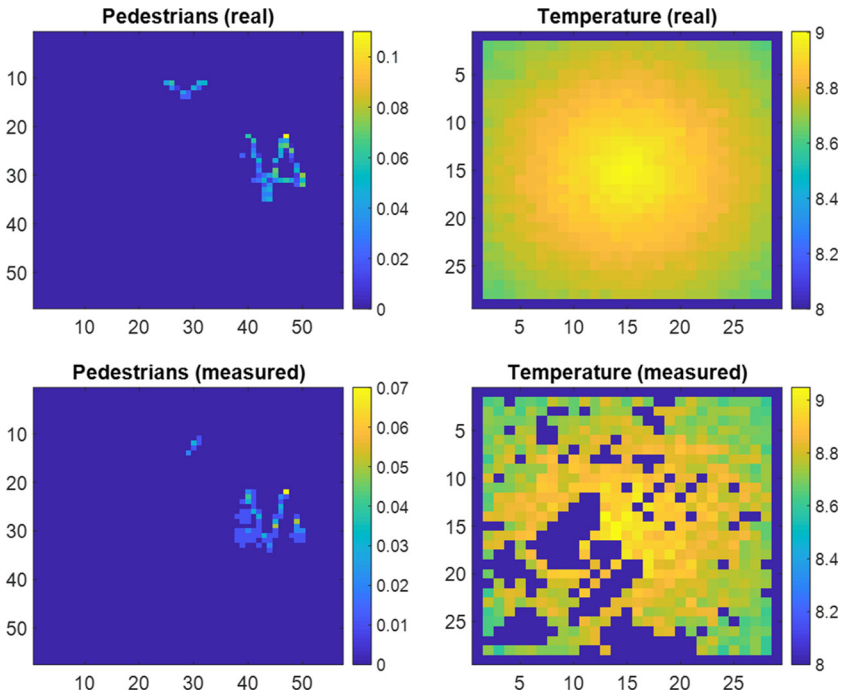

Fig. 11 Real and measured maps of pedestrians and temperatures during one of the simulations

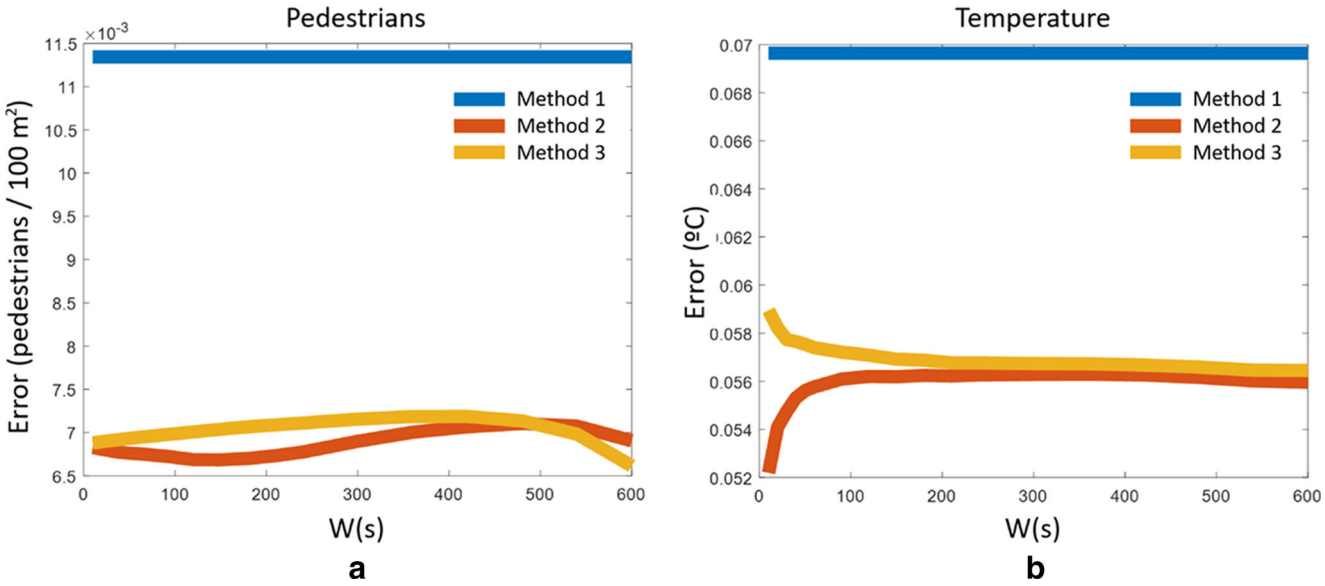

Fig. 12 Errors obtained between real and estimated maps using the three methods

volunteer had to perform one mission, monitoring the city for 4 min and answering a questionnaire about the observed phenomena.

Three missions were created for the experiments:

- Mission 1: A car breaks down in the city center, causing an important traffic jam. The temperatures are slightly higher in the south than in the rest of the city. The rest of the variables are not relevant.
- Mission 2: An accident in a factory causes a fire that increases the temperature and pollution in the industrial area. The traffic is higher in the city center, but there are no jams. The rest of the variables are not relevant.
- Mission 3: A festival takes place in the park, gathering hundreds of people in that place. The traffic is slightly denser in the city center, whereas the temperatures are a little bit higher in the industrial area. The rest of the variables are not relevant.

The questionnaires had the following structure:

- Personal information: Name, gender, age, studies, email, and experience with video-games and virtual reality (from 0 to 10).
- Performance in the mission:
  - Mission 1: Most relevant phenomenon (jam), area with higher traffic (center), map with more variability (traffic), and area with higher temperatures (south).
  - Mission 2: Most relevant phenomenon (fire), area with higher traffic (center), most similar maps (temperature and pollution), and area with higher temperatures (south).
  - Mission 3: Most relevant phenomenon (festival), area with higher traffic (center), area where more drones are needed (park), and area with more pedestrians (park).
- Evaluation of the interface:
  - Workload: Experienced mental demand, physical demand, temporal demand, effort, performance, and frustration (from 0 to 10).
  - Interface: Evaluation of visualization elements (city, maps, displays...) and interaction tools (voice, gestures, movements...) and proposal of changes for the next version of the interface.

As mentioned above, every volunteer performed one mission and answered four questions about it. The results are shown in Table 5. The participants obtained a mean score of 7.73 over 10 with a standard deviation of 2.53, which reveals that their understanding of the scenario was high. This can be considered a good result considering that the volunteers were performing this task for the first time and some of them had not got prior experience with virtual reality. This result validates not only the immersive interface but also the selection of information.

Table 5 Performance of the operators in the missions (from 0 to 10)

Mission	Mean	Std. Dev.
Mission 1	8.4091	1.6855
Mission 2	7.7273	3.0526
Mission 3	7.0455	2.6968
Total	7.7273	2.5282

Moreover, the volunteers declared the workloads collected in Fig. 13. The six variables related to workload have been taken from the NASA Task Load Index (NASA-TLX) [46], a widely used assessment tool for the workload perceived when performing a task. Mental demand quantifies the required mental and perceptual activity, whereas physical demand quantifies the required physical activity. Temporal demand measures the time pressure felt by the operator due to the pace of tasks. Performance represents how successful was the operator in performing the task, and effort indicates how hard had the operator to work to accomplish it. Finally, frustration determines if the operator felt irritated, stressed and annoyed, or content, relaxed and complacent during the task. As can be seen, all the variables that impact positively on workload are below 5/10, whereas the performance is slightly above this threshold. The total workload for the operators, obtained as the mean of these variables, has a mean of 3.56 with a standard deviation of 1.71.

Regarding the visualization of the information, seven operators described the interface as good, five as clear, and two as intuitive. Nevertheless, the operators also mentioned some drawbacks: the color code is complex (2), buildings hide some parts of maps (2), the menu is not enough visible (1), the location in the map is sometimes hard, and maps cannot be combined (1).

Regarding the interaction with the interface, four operators described the interface as good, four as intuitive, and one as effective. The voice commands were evaluated positively by two operators, whereas the gestures were mentioned positively by one. However, the operators also revealed some problems: the movements are slow (2), the flight can be improved (2), the jump does not work properly (2), the detection of hands sometimes fails (1), and ascend/descend movements can be improved (1).

Finally, the operators made some suggestions for the next version of the interface: changes in the color code (2), improvements in the laser pointer (2), faster (1) and agile (1) movements, combination of motions (1), possibility to see past states of the maps (1) and a mini-map with the location of the operator in the city (1). All these suggestions will be taken into account in future works.

4.4 Discussion

The experiments were performed to validate the complete system consisting of swarm intelligence, data fusion, and operator interface. These tests were used to optimize both the behavior-based algorithm that controls the swarm and the data fusion method that creates the maps. Two research questions were asked about the capability of the swarm to determine the state of the city and the ability of operators to know this state.

The behavior-based algorithm has shown remarkable flexibility since it can be adapted to diverse scenarios with rapid optimization. In previous work, the algorithm was optimized to search cars leading the drones to fly over the roads, whereas in this work, it has been optimized to search pedestrians allowing them to fly over the blocks, as well as to uniformly cover the city to measure environmental variables. The efficiencies obtained in both simulators, as well as the quality of the collected data, demonstrate the performance of this algorithm. A limitation of the optimization has been revealed in the pedestrians' search: if the optimization scenario is considerably different from the application one, the efficiency of the algorithm may decline. Therefore, accurate models of the

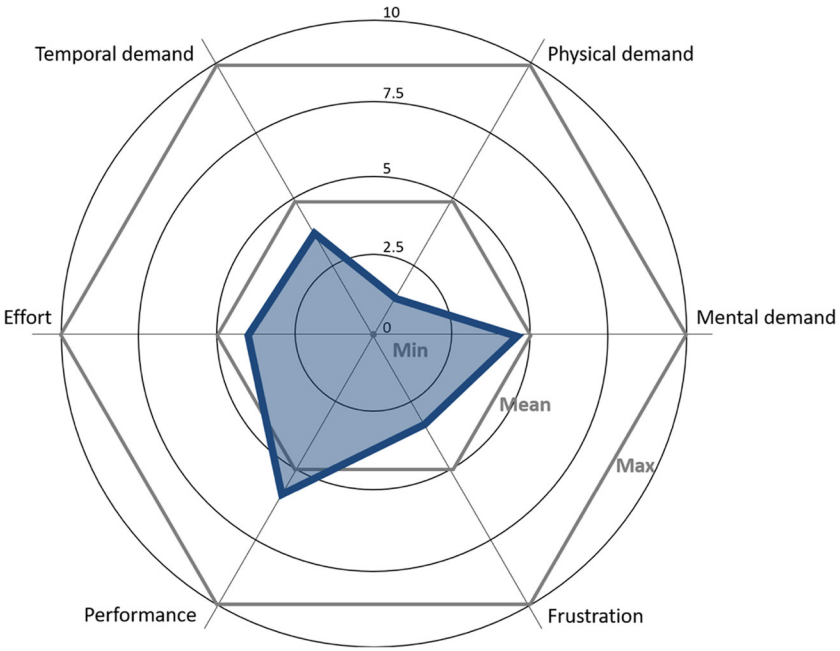

Fig. 13 Workload of the operators in the missions

different variables are required to adequately optimize the parameters of the algorithm.

Moreover, the data fusion algorithm used to build maps from drone measurements has shown similar flexibility. This method can be adapted to map discrete (e.g., detections of cars and pedestrians) and continuous variables (e.g., climate and pollution parameters) just by choosing a method and tuning its parameters properly. The high similarity between the real and measured maps of traffic, pedestrians, and temperature demonstrates the appropriateness of this algorithm.

Finally, the interface uses virtual reality to immerse the operators in the city and show them the maps, as well as voice and gesture commands to configure the displayed information. The tests performed with operators demonstrate that the interface allows them to understand the state of the city with an acceptable workload. The volunteers evaluated positively their experience and proposed some changes for the interface. These results validate not only the quality of the interface but also the performance of the swarm control and data fusion algorithms, given that the information obtained by these algorithms is enough for the operators to understand the state of the city.

These results reveal that the two research objectives have been addressed successfully. The system has been validated in a small simulated city using a fleet of ten drones, but it can be scaled to larger simulated cities just by adding more drones. Nevertheless, more work is required to overcome the reality gap and apply these developments to a real robot fleet in a real smart city.

5 Conclusions

In this paper, a prototype of a monitoring system for smart cities based on an aerial robotic swarm is proposed, developed, and validated in simulation. The main contributions of the work are applying a behavior-based algorithm optimized to monitor pedestrians, climate, and pollution in cities, developing a data fusion algorithm to build maps from these measurements taken by the drones, and using an immersive interface to allow an operator to monitor the state of the city. As far as we know, our interface is the first that has this degree of immersion and is devoted to monitoring a smart city.

The experiments performed with operators monitoring the city show the performance of the proposed system. On the one hand, it has been demonstrated that the robot swarm controlled by a behavior-based algorithm is a suitable tool to find the relevant information of a city. On the other hand, it has been proven that the operators using an immersive interface can perceive the state of the city, discovering the behavior of different variables and detecting the most relevant events.

In future works, the behavior-based algorithm will be adapted to perform other tasks in the city: e.g., search for parking sites and assign them to vehicles, support the cleaning of streets, and transport packages among different locations. Besides, some well-known models for data fusion, such as Kalman and particle filters, will be tested to improve the quality of maps. Finally, a set of tools for commanding the fleet will be implemented in the interface, such as setting areas of interest and target variables to the swarm.

Acknowledgments This work is part of the "SwarmCity project: monitoring future cities with intelligent flying swarms," developed by the Robotics and Cybernetics Research Group of the Centre for Automation and Robotics (UPM-CSIC).

References

1. Desa U et al (2014) World urbanization prospects, the 2011 revision. Population Division, Department of Economic and Social Affairs, United Nations Secretariat
2. McLaren D, Agyeman J (2015) Sharing cities: a case for truly smart and sustainable cities. MIT Press
3. Anthopoulos L (2017) Smart utopia vs smart reality: learning by experience from 10 smart city cases. Cities 63:128–148
4. Petrolo R, Loscri V, Mitton N (2017) Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms. Trans Emerg Telecommun Technol 28(1):e2931
5. Jamil MS, Jamil MA, Mazhar A, Ikram A, Ahmed A, Munawar U (2015) Smart environment monitoring system by employing wireless sensor networks on vehicles for pollution free smart cities. Procedia Eng 107:480–484
6. Mohammed F, Idries A, Mohamed N, Al-Jaroodi J, Jawhar I (2014) UAVs for smart cities: opportunities and challenges. In: 2014 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, pp 267–273
7. Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) ROS: an open-source robot operating system. In: ICRA workshop on open source software, vol 3, Kobe, p 5
8. Garcia-Aunon P, Roldán JJ, Barrientos A (2018) Monitoring traffic in future cities with aerial swarms: developing and optimizing a behavior-based surveillance algorithm. Cognitive Systems Research
9. Garcia-Aunon P, Barrientos Cruz A (2018) Comparison of heuristic algorithms in discrete search and surveillance tasks using aerial swarms. Appl Sci 8(5):2076–3417
10. Garcia-Aunon P, Cruz AB (2018) Control optimization of an aerial robotic swarm in a search task and its adaptation to different scenarios. J Comput Sci 29:107–118
11. Roldán J, Garcia-Aunon P, Peña-Tapia E, Barrientos A (2019) Swarmcity project: can an aerial swarm monitor traffic in a smart city? In: 2019 IEEE International Conference on Pervasive Computing and Communications. IEEE, p 1720
12. Silva BN, Khan M, Han K (2018) Towards sustainable smart cities: a review of trends, architectures, components, and open challenges in smart cities. Sustain Cities Soc 38:697–713
13. Hashimoto K, Yamada K, Tabata K, Oda M, Suganuma T, Rahim A, Vlacheas P, Stavroulaki V, Kelaidonis D, Georgakopoulos A (2015) iKaas data modeling: a data model for community services and environment monitoring in smart city. In: 2015 IEEE International Conference on Autonomic Computing. IEEE, pp 301–306
14. Urra O, Ilarri S (2019) Spatial crowdsourcing with mobile agents in vehicular networks. Veh Commun 17:10–34

15. Alsamhi S, Ma O, Ansari M, Almalki F (2019) Survey on collaborative smart drones and internet of things for improving smartness of smart cities. IEEE Access
16. Vashist S, Jain S (2019) Location-aware network of drones for consumer applications: supporting efficient management between multiple drones. IEEE Consumer Electron Mag 8(3):68–73
17. Lagkas T, Argyriou V, Bibi S, Sarigiannidis P (2018) UAV IoT framework views and challenges: towards protecting drones as "things". Sensors 18(11):4015
18. Kim H, Mokdad L, Ben-Othman J (2018) Designing UAV surveillance frameworks for smart city and extensive ocean with differential perspectives. IEEE Commun Mag:99
19. Chow JY (2016) Dynamic UAV-based traffic monitoring under uncertainty as a stochastic arc-inventory routing policy. Int J Transp Sci Technol 5(3):167–185
20. Srinivasan S, Latchman H, Shea J, Wong T, McNair J (2004) Airborne traffic surveillance systems: video surveillance of highway traffic. In: Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks. ACM, pp 131–135
21. Wu D, Arkhipov DI, Kim M, Talcott CL, Regan AC, McCann JA, Venkatasubramanian N (2017) Addsen: adaptive data processing and dissemination for drone swarms in urban sensing. IEEE Trans Comput 66(2):183–198
22. Kyrkou C, Timotheou S, Kolios P, Theocharides T, Panayiotou C (2019) Drones: augmenting our quality of life. IEEE Potentials 38(1):30–36
23. I. Pasnak, "Justification possibility of using drones to study the parameters of traffic," 2017
24. Barmpounakis EN, Vlahogianni EI, Golias JC, Babinec A (2019) How accurate are small drones for measuring microscopic traffic parameters? Transp Lett 11(6):332–340
25. Li Z, Chen X, Ling L, Wu H, Zhou W, Qi C (2019) Accurate traffic parameter extraction from aerial videos with multi-dimensional camera movements. Tech Rep
26. Pajares G (2015) Overview and current status of remote sensing applications based on unmanned aerial vehicles (UAVs). Photogramm Eng Remote Sens 81(4):281–330
27. Sekula P, Zimnoch M, Bartyzel J, Kud M (2019) Application of airborne measurement system for monitoring vertical profiles of meteorological parameters, black carbon and particulate matter concentration for urban area of kraków. Geophys Res Abstr 21
28. Villa TF, Gonzalez F, Miljievic B, Ristovski ZD, Morawska L (2016) An overview of small unmanned aerial vehicles for air quality measurements: Present applications and future prospectives. Sensors 16(7):1072
29. Roldán JJ, Joossen G, Sanz D, Del Cerro J, Barrientos A (2015) Mini-UAV based sensory system for measuring environmental variables in greenhouses. Sensors 15(2):3334–3350
30. Xiaoyuan Y, Jiwei D, Tianjie Y, Qingfu Q (2016) A method for improving detection of gas concentrations using quadrotor. In: 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference. IEEE, pp 971–975
31. Lv Z, Yin T, Zhang X, Song H, Chen G (2016) Virtual reality smart city based on webVRGIS. IEEE Internet Things J 3(6):1015–1024
32. Paravati G, Lamberti F, Sanna A, Ramirez EH, Demartini C (2012) An immersive visualization framework for monitoring, simulating and controlling smart street lighting networks. In: Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques, pp 17–26
33. L. Dupont, L. Morel, and M. Pallot, "Exploring the appropriateness of different immersive environments in the context of an innovation process for smart cities," 2016
34. Bereitschaft B (2016) Gods of the city? Reflecting on city building games as an early introduction to urban systems. J Geogr 115(2): 51–60
35. P. Haahtela *et al.*, "Gamification of education: cities skylines as an educational tool for real estate and land use planning studies," 2015
36. Krajzewicz D, Erdmann J, Behrisch M, Bieker L (2012) Recent development and applications of SUMO-simulation of urban mobility. Int J Adv Syst Meas 5(3&4)
37. Aparicio F, Arenas B, Gómez A, Jiménez F, López J, Martínez L, Páez F (2008) Ingeniería del transporte. CIE Dossat, Madrid
38. van Wageningen-Kessels F, Van Lint H, Vuik K, Hoogendoorn S (2015) Genealogy of traffic flow models. EURO J Transp Logist 4(4):445–473
39. Roldán JJ, Crespo E, Martín-Barrio A, Peña-Tapia E, Barrientos A (2019) A training system for industry 4.0 operators in complex assemblies based on virtual reality and process mining. Robot Comput Integr Manuf 59:305–316
40. Roldán J, Peña-Tapia E, Martín-Barrio A, Olivares-Méndez M, Del Cerro J, Barrientos A (2017) Multi-robot interfaces and operator situational awareness: study of the impact of immersion and prediction. Sensors 17(8):1720
41. Roldán JJ, Peña-Tapia E, Garcia-Aunon P, Del Cerro J, Barrientos A (2019) Bringing adaptive & immersive interfaces to real-world multi-robot scenarios: application to surveillance and intervention in infrastructures. IEEE Access 7:2169–3536
42. Roldán JJ, Peña-Tapia E, Garzón-Ramos D, de León J, Garzón M, del Cerro J, Barrientos A (2019) Multi-robot systems, virtual reality and ROS: developing a new generation of operator interfaces. In: Robot Operating System (ROS). Springer, pp 29–64
43. Garcia-Alonso A, Gil J, Borro D (2005) Interfaces for VR applications development in design. In: Proceedings of the virtual concept, p 109
44. Dede C (2009) Immersive interfaces for engagement and learning. science 323(5910):66–69
45. Korves B, Loftus M (2000) Designing an immersive virtual reality interface for layout planning. J Mater Process Technol 107(1–3): 425–430
46. Hart SG, Staveland LE (1988) Development of NASA-TLX (task load index): results of empirical and theoretical research. Adv Psychol 52:139–183, Elsevier