Up and Away: A Visually-Controlled Easy-to-Deploy Wireless UAV Cyber-Physical Testbed

Ahmed Saeed[†‡], Azin Neishaboori[†‡], Amr Mohamed[‡], Khaled A. Harras[†]
[†]School of Computer Science, Carnegie Mellon University Qatar
[‡]Department of Computer Science and Engineering, College of Engineering, Qatar University
Email: cse.saeed@gmail.com, azin.neishaboori@gmail.com, amrm@ieee.org, kharras@cs.cmu.edu

***Abstract*—Cyber-Physical Systems (CPS) have the promise of presenting the next evolution in computing with potential applications that include aerospace, transportation, and various automation systems. These applications motivate advances in the different sub-fields of CPS such as mobile computing, context awareness, and computer vision. However, deploying and testing complete CPSs is known to be a complex and expensive task. In this paper, we present the design, implementation, and evaluation of *Up and Away (UnA)*: a testbed for Cyber-Physical Systems that use Unmanned Aerial Vehicles (UAVs) as their main physical component. *UnA* aims to abstract the control of physical system components to reduce the complexity of UAV oriented CPS experiments. *UnA* provides APIs to allow for converting CPS algorithm implementations, developed typically for simulations, into physical experiments using a few simple steps. We present two scenarios of using *UnA*'s API to bring mobile-camera-based surveillance algorithms to life, thus exhibiting the ease of use and flexibility of *UnA*.**

I. Introduction

Cyber-Physical Systems (CPS) have the promise of presenting the next evolution in computing by bridging the gap between the virtual world and the physical world. CPS applications include aerospace, transportation, factory automation, and medical systems [1]. This new paradigm is created based on new ubiquitous computing systems and communication services available all the time, everywhere. These systems and services forming networks of users, sensors, devices, and applications may seamlessly interact with each other and their environment in unprecedented ways [2]. CPS's are supported by developments in other research fields including mobile computing, embedded systems, computer vision, control, and communication. While these developments ultimately contribute to advancing different CPS components, most of the work is evaluated via simulations or experiments that focus on a specific sub-problem. Furthermore, the complex nature of full scale CPS experiments makes the task even harder, especially when it comes to testing individual "cyber" or "physical" components. From a different perspective, current generic CPS testbeds [3] are typically complex and incur large deployment costs.

In this paper, we present Up and Away (UnA), a testbed that aims at providing a low-cost, easy to use and deploy, physical system composed of multiple remotely controlled quadcopters. *UnA* provides an API that allows for integration of simulation code that will obtain input from the physical world (e.g. number of targets to track in a surveillance scenario) then provide control parameters (i.e. number of quadcopters and destinations coordinates) to the UAVs. Thus, *UnA* allows for rapid development of cyber-physical systems experiments that use quadcopter UAVs. *UnA*'s design choice of using UAVs is due to the their deployment flexibility and maneuverability. UAVs are suitable for many applications such as urban visual sensing in disaster response scenarios, crowd monitoring on demand, transportation, and fixing broken communication lines, thus allowing for cyber-physical experimentation with realistic objectives. *UnA* sets itself apart from earlier work by providing a complete, low-cost, and generic CPS testbed; such testbed is different from earlier work which either focuses on UAV control or application specific deployments.

Our main contribution with *UnA* is providing a complete and generic wireless testbed that allows for the detection and recognition of real objects, which are treated as input to the CPS. The system then uses the quadcopters to carry out CPS specific tasks, depending on the application superimposed on the system, and provides the sensory information from the UAV's array of sensors as the CPS output. Additionally, we provide a vision-based localization solution that uses color tags to identify different objects in varying light intensity environments. This localization technique is suitable for many deployments, especially with indoor scenarios or deployments. We use that technique to control the UAVs within the specified area of interest. Finally, the *UnA* architecture is designed in a modular fashion so that different parts (e.g. localization and control) can be replaced or improved depending on the deployment environment (e.g. using GPS for outdoor deployments).

To demonstrate the ease of use and potential behind *UnA*, we deploy several cluster-based target coverage algorithms [4], [5] on it. We share our experience implementing these algorithms on the system in two different scenarios running in varying environmental contexts.

The rest of the paper is organized as follows, Section II presents earlier work on UAV testbeds and CPS testbeds. Then, we present the *UnA* architecture in Section III. Our experience with running *UnA* in two different scenarios is presented in Section IV. Finally, we conclude the paper and discuss future work in Section V.

II. Related Work

Several research groups have recently developed quadcoptor testbeds [6]–[9]. The GRASP testbed [9] uses off-the-shelf high-end Ascending Technologies Hummingbird quadcopters to demonstrate multirobot control algorithms. The ETH Flying Machine Arena [7] uses modified Hummingbird quadcopters to demonstrate several acrobatic and athletics control maneuvers. A testbed was developed by University of Colorado, Boulder for testing ad-hoc networking scenarios [10]. This testbed uses UAVs which were deployed in a 7 km^2 area to test different operation conditions of the DSR protocol. The work in [6] shows the development of a low-cost UAV testbed using Goldberg Decathlon ARF model airplane. The work aims at showing the design and development of hardware and software components to produce low-cost UAVs. The work in [11] presents an approach for controlling a UAV through its front camera.

Unlike other UAV-based testbeds, *UnA*'s architecture is more concerned with abstracting the control of UAVs rather than improving it. *UnA* presents an architecture that provides default components and facilitates replacing any of them to test its impact on the overall performance of the system. To that end, AR Drones were selected as the UAVs to be used. We prefer using quadcopters to fixed-winged airplanes due to their maneuverability especially in highly constrained spaces (e.g. indoors) in addition to exploiting their ability to hover. We specifically adopt the AR Drones, mainly built as a gaming platform, because they are extremely cheap relative to other potential devices [1] In addition, these UAVs come with a large array of sensors and a widely supported open source API to control the drone and obtain sensory information for its sensors (the specifications of the AR Drone are better described in Section III).

From a different perspective, CPS testbeds presented in [3], [12], like *UnA*, introduce Unmanned Ground Vehicle (UGV) as affordable solutions to allow CPS researchers to perform physical experiments. However, both testbeds rely on UGVs which typically have limited deployment options due to the ground surface nature that's required for UGVs to move on. *UnA* relies on UAVs which have fewer constraints regarding deployment areas and are suitable for a wider range of applications.

III. Up and Away

In this section, we introduce the design and implementation of UnA. We start with our design goals followed by a description of our architecture. We then discuss the details of how *UnA* handles targets and UAV localization, UAV control, CPS processing and communication.

A. Design Goals

The development of *UnA* is motivated by the need for low-cost, easy to deploy experiments for various CPS applications.

[1]An AR Drone 2.0 costs $300 while an AscTec Hummingbird costs around $4500 .

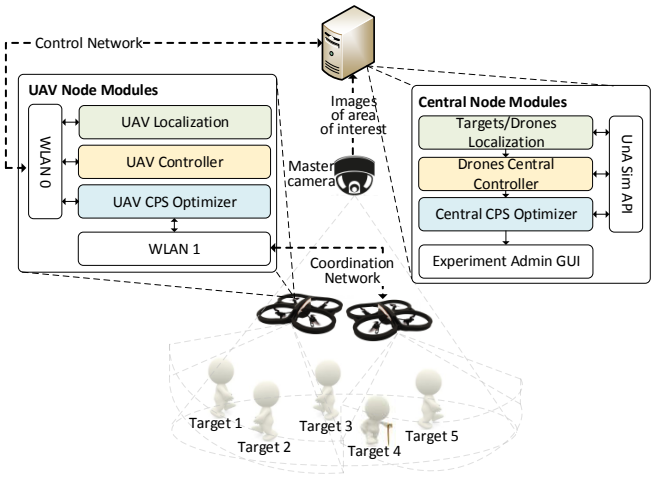


Fig. 1. UnA Architecture

Thus, *UnA*'s architecture is designed with the following goals in mind:

1. Multidisciplinary nature of CPS experiments: The development of CPS experiments requires awareness of several fields including hardware design, control systems, and software systems development [1]. This poses significant challenges for developing realistic experiments for such systems. *UnA* was designed with this unique feature of CPS in mind and therefore the *UnA* modules are extensible to support the development and replacement of any of the CPS's components.

2. Support of large scale experiments: CPS experiments with a large number of physical nodes requires the testbed to support different communication models between the nodes. In addition, the processing power required to optimize the overall state of the system can grow exponentially with the number of nodes. The modularity in *UnA* enables executing different component implementations on different machines to scale with the application needs and testbed size.

3. Seamless control of the UAVs: UnA aims at reducing the complexities in CPS experiments design incurred by the automatically operated mechanical components of the system. We accomplish this by abstracting the control algorithms and hardware design of those physical components from the CPS application developer, which enables people to focus more on the "cyber" related challenges. In addition, autonomous control of off-the-shelf UAVs is a difficult task [7], [9]. Thus, *UnA* relies on the easy to use AR Drone API and extends it to use way-point navigation instead of controlling the angular rotations (i.e. roll, yaw and gaz) of the UAV.

B. UnA Architecture

Figure 1 depicts the basic *UnA* architecture. *UnA* allows UAVs to communicate with each other, process sensory information locally and act accordingly (i.e. control itself). It also provides a base node that monitors the behavior of all UAVs. Moreover the central node can handle processing intensive

tasks that UAVs off-load to it. The system is comprised of the two following major modules.

1. UAV Node Modules are responsible for obtaining and processing information about the drone and the environment's state, calculating the drone's corresponding objectives accordingly, as well as controlling the drone's motion to achieve these objectives. *UnA* is built on top of the Parrot AR Drone 2.0 [13]. The AR Drone is a quadrotor helicopter that is electrically powered with two cameras: a front 720p camera with a 93° lens and a vertical QVGA camera with a 64° lens. These drones are controlled by an external computer through WiFi, and are equipped with an onboard ARM Cortex A8 processor with 1 Gbit RAM that runs a Linux 2.6.32 Busybox. The onboard computing machine is responsible for collecting and reporting the state of the drone to an external computer that controls the drone. The AR Drones are relatively cheap, easy to program and comes with a firmware that provides assisted maneuvers that facilitates motion control tasks such as hovering.

2. Central Node Modules are responsible for monitoring the state of each drone, manually controlling the drone, and providing part of the environment state information to the drones. Due to the limited processing capacity of current off-the-shelf UAVs, UnA's architecture enables off-loading some of the processing tasks from the UAVs to the central node to enable real time responses to changes in the environment. The *Experiment Admin GUI* displays the state information of the different UAVs. It also supports the manual control of the UAVs for emergency cases. The main purpose of UnA's architecture is to support the rapid development of CPS experiments that use UAVs as their Physical part. While *UnA* can be extended and customized for different CPS applications, we assume that simulation code was already developed by the experiment administrator. Thus, the central node supports communication between its different modules with other applications through sockets. The *UnA Sim API* provides the appropriate wrappers to allow the simulation code to interface with *UnA* to bring the simulations to life by off-loading all the *UAV Node Modules* functionalities to the simulation engine through the central node.

For the remainder of this section, we explain the different layers of the *UnA* architecture according to the color coding in Figure 1.

C. UAV and Target Localization

The UnA architecture supports different approaches for localization: 1) A Parrot Flight Recorder GPS module and 2) Using a master camera mounted at a high point to track the location of the drone. While the first approach is straight forward, it doesn't work in indoor environments. For our experiments in Section IV, we focus on indoor environments, hence, vision-based localization will be used.

Multiple UAV tracking using computer vision with multiple cameras is a challenging task to implement [14]. For deployment flexibility, we propose a simpler approach that uses a single camera, mounted at a high point in the center of the area of interest, to track the drones by tagging each drone with a distinct color tag. Tracking objects of a certain color could be implemented using OpenCV [15]. This approach accommodates tracking more targets by adding more colors, if target identity matters, or having one color for all targets, if identification is not an objective.

(a) Image filtered using red filter.

(b) Centroid of red objects marked with blue dots.

Fig. 2. The process of detecting red colored objects in master camera footage.

While earlier work in UAVs control [7], [9] relies on Motion Capturing Systems (MCS) for tracking UAVs, we chose the single camera approach as a much cheaper alternative that provides reasonable accuracy in the 2D plane and relies on the drone's altitude sensor to obtain input on the third dimension[2]. We further study persistent color tags identification in varying color intensity conditions.

1) Implementation: The *UnA* localization system relies on color tags that are attached to each of the targets and drones. While differentiating between the identities of drones is important, differentiating between the identities of targets is not of the same significance for most of the target coverage experiments. Thus, each drone is tagged with a unique color tag and all targets are given the same color tag.

The localization algorithm relies on a color filtering algorithm that isolates spots with a certain color. Thus, the algorithm takes as an input the number of colors to search for, and the filter parameters for each color. The algorithm pulls images from the central PTZ camera and applies each of the filters. The output of the algorithm is the number of spots for each color and the centroid of each of the spots. Figure 2 shows the output of the filter for red parameters and the identified centroid of these red spots.

2) Accommodating Changes in Light Intensity: One of the constraints forced by the nature of our deployment area is the effect of having sun light as part of the illumination of the area. In particular, as the sun's position changes over the day, the intensity of colors changes as well. This makes persistence detection of colors using static filters much more challenging. Figure 3 shows the effect of changing light intensity on the color tags we picked, over different times of the day.

To determine the colors that can be detected consistently throughout the day using static filtering, several experiments were conducted with different color models (i.e. RGB and HSV), different colors, and different locations (as in Figure 3). Our results showed that Blue, Red, Green, and Yellow are the

[2]While PTZ cameras can cost several thousand dollars, MCS costs up to tens of thousands of dollars.

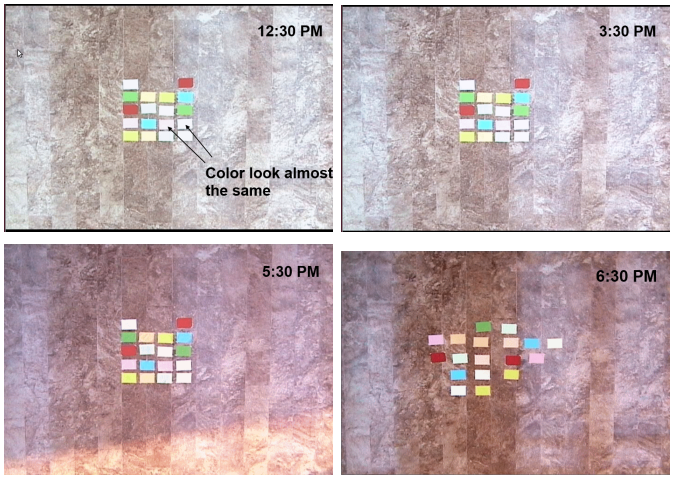


Fig. 3. The view from the master camera at different times of day showing gradual changes in light intensity.

Color	12:30 - 3:00 PM	3:00 - 4:30 PM	4:30 - 5:00 PM
Red	0%	0%	0.3%
Yellow	0%	0%	0.2%
Green	0%	0%	0%
Pink	9.5%	37.8%	10.14%
Blue	0%	0%	0%
White	0%	6.8%	48.3%

TABLE I
PERCENTAGE OF COLOR MISS DETECTION OF EACH COLOR USING A STATIC COLOR FILTER OVER THE PERIOD OF CHANGING LIGHT INTENSITY.

colors that were most persistently detected irrespective of the time of day as shown in Table I. These would represent the colors recommended to be used for tagging the drones and targets.

D. UAV Control

The *UAV Controller* and *Drones Central Controller* are responsible for moving the drone to a specific X, Y coordinates with a certain orientation within the area of interest. Due to our choice of AR Drones, the controller requires four different threads to control and query the drone's sensors: a) AT Commands thread, b) NAVDATA Thread, c) Video Client Thread, and d) Coordinate Controller Thread.

a) AT Commands Thread: AR Drones are controlled through the AT commands protocol which is sent to the drone over WiFi. The AT commands are encoded as 8-bit ASCII characters that start with "AT*" followed by the command name. AR Drones require receiving an average of 30 AT commands per second for proper control. Thus, a thread is forked to dispatch AT commands to the drone. This thread communicates with other threads to update the parameters it sends to the drone accordingly.

b) NAVDATA Thread: UnA relies on several of the sensory readings captured by the sensors on the drone (e.g. altitude sensor and compass). This information, called navdata, are sent to the controller from the drone approximatively 15 times per second in demo mode, and 200 times per second in full mode.

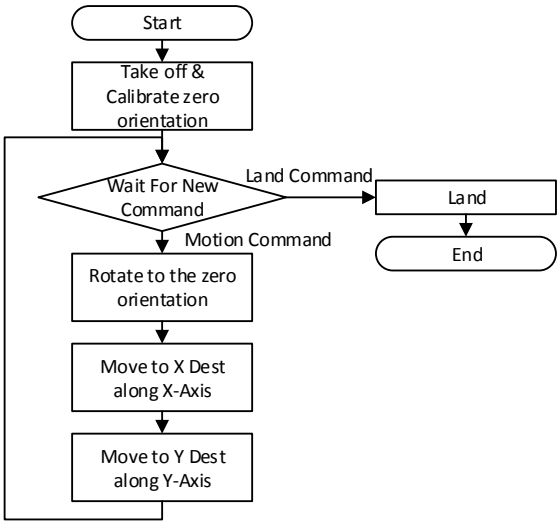


Fig. 4. A UAV's control loop.

c) Video Client Thread: Videos from the front and bottom cameras are the most important outputs for any CPS using the drones. Images from this video stream are sent to the controller via WiFi. Then, the received images are decoded and saved for further processing by the CPS.

It is important to note that all drones that are part of a *UnA* experiment stream their video feeds to the controller simultaneously. This introduces a processing bottleneck due to the decoding overhead of each of the received frames. This overhead is addressed by reducing the frame rate of the streamed videos from each drone. This can also be adapted dynamically according to the variation in the event being covered as well.

d) Coordinate Controller Thread: Quadcopters can move in any direction with any orientation which requires accurate settings of the yaw, roll, and pitch angular speeds. However, the simultaneous control of those angles is problematic due to faulty sensor readings and inaccurate rotors due to the cheap hardware used (we present more details on hardware limitations in Section IV-D). Hence, complex control algorithms are required to perform such complex motion patterns. Thus, we choose a simple controlling method to avoid setting all three angular speeds. Figure 4 summarizes the algorithm controlling the drone motion.

To reach a certain coordinate, the Coordinate Controller Thread first needs to determine in which quarter does the destination fall with respect to the drone's location. This allows for having a clue whether the drone will be moving to the left or right and whether it will be moving to the front or back. Then, the orientation of the drone is set to $90°w.r.t.$ the x-y plan of the area of interest. The change in orientation uses the

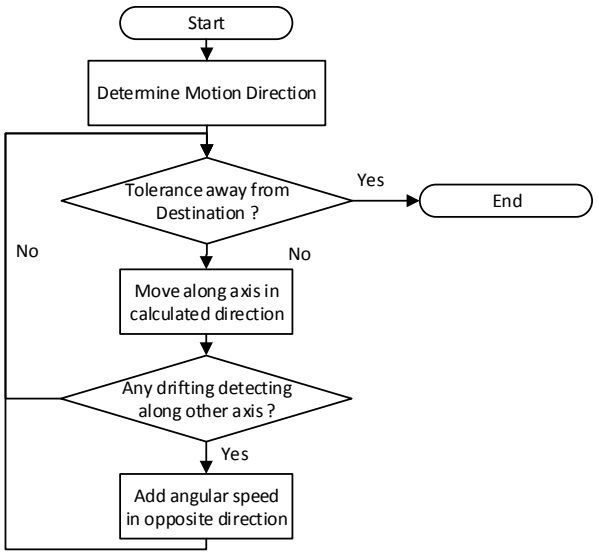


Fig. 5. Control loop for motion over an axis.

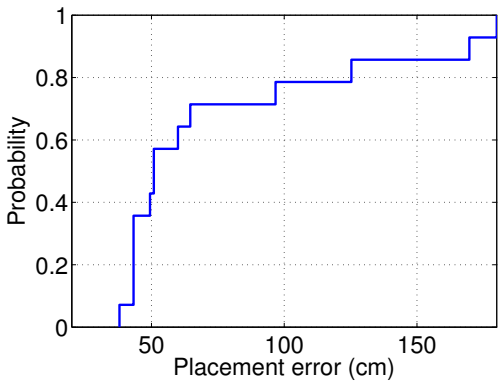


Fig. 6. CDF of distance error in drone placement.

compass for feedback. The controller then moves the drone along the x-axis to its specific coordinate then along the y-axis after which the drone is finally rotated to the specified orientation. This control algorithm obtains feedback from the UAV localization module and the UAV internal compass in order to actuate any errors in placing the drone due to its inertia. Figure 5 summarizes the algorithm controlling motion along each axis.

We benchmarked the *UnA*'s placement error by placing the drone arbitrarily and then moving it to a certain location over 14 times. Figure 6 shows the CDF of the error in placing the drone (i.e. the distance between the drone's actual location and its intended location). These inaccuracies could lead to an overall placement error with a median value of approximately $50cm$ in a 10m x 10m area. The effects of these placement inaccuracies are significantly increased due to the small size of the deployment area. Nonetheless, the current state of *UnA* is useful for many applications and particularly as the area of interest grows larger.

E. CPS Optimization

The *UAV CPS Optimizer* module is responsible for updating the objective (i.e. location and orientation) of the drone after obtaining updates on the state of the drone and the environment from the drone's sensors, other drones, and the central node. This module has three modes of operation: 1) distributed mode, 2) central mode and 3) emulation mode. In the distributed mode, a distributed objective function is used where each UAV calculates its own new objectives after each update, which is suitable for lightweight distributed algorithms. In the central mode, the *UAV CPS Optimizer* reports the updates to the *Central CPS Optimizer* which calculates the new objectives based upon updates from all UAVs. Finally, in the emulation mode, the UAVs implement a distributed objective function but off-load its calculations to the central node to meet the realtime responsiveness requirements.

F. Control and Coordination Networks

One of our design choices is to separate the intercommunication between the UAVs and the communication between each UAV and the central node into two different networks. This design choice is motivated by two reasons: 1) While communication between UAVs can be established in an ad-hoc, unreliable fashion, communication between the UAVs and the central nodes should be using an infrastructure that is connected to the UAVs at all times, 2) AR Drone 2.0 [13] provides an SDK for monitoring and controlling the UAVs that requires continuous communication between the UAVs and a central node. We rely on that SDK to facilitate the different tasks of the *Central Node Modules*.

For the implementation of the *Control Network*, we use the built-in WiFi card but changed its configuration to work as a client instead of as an access point, which is its default configuration. This way all nodes can connect to the central node through a WiFi infrastructure. As for the *Coordination Network*, we plug a WiFi dongle into the drone, compile the dongle's driver to work on the ARM-based Busybox, and install the driver on the drone. The dongle is then configured to work in Ad-Hoc mode and join the drone's SSID as soon as its visible.

IV. Experience using UnA

In this section, we demonstrate the steps of bringing a sample CPS simulation to life using *UnA* in two different scenarios within a mobile-camera-based surveillance system. The system's objective is to maximize the number of targets covered by a set of mobile cameras. For the rest of the section, we will show the different steps of implementing the proposed CPS using *UnA*.

A. Target Coverage System Description

The implemented CPS is a computationally efficient heuristic for mobile camera placement and orientation for the coverage of a set of localized targets. The proposed algorithm

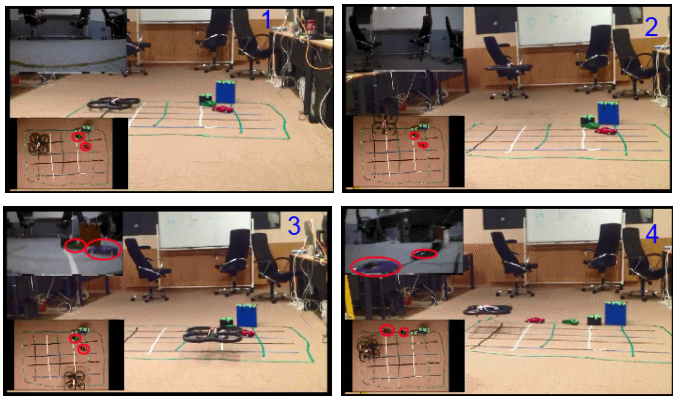


Fig. 7. The top left corner is the view from the UAV while the bottom left corner is the view from the master camera. Targets are circled in red when they appear in the view of either cameras.

considers the problem of finding the minimum number of cameras to cover a high fraction of a set of targets as a clustering problem. We propose several variations for the clustering approaches in [4], [5], [16]. The coverage algorithm is a central algorithm whose inputs are the location of all targets and the mobile cameras, and its output is a set of location and orientation parameters for all mobile cameras. It is important to note that for this application, target identity is not important. Thus, we choose the color green for all targets.

All required computations were offloaded to the central node because the Target Coverage System doesn't require any processing to be made on the drones. The MATLAB implementation of simulation algorithms are integrated into *UnA* through socket communication between the *UnA* process and the simulation MATLAB code, requiring minimal coding overhead.

B. Experimental Setup

For both scenarios, we use an Axis 213 PTZ Network Camera as the master camera. The master camera is connected through ethernet to the Central Node, a Lenovo ThinkPad T430. Two Parrot AR Drones 2.0 are also used. The drones are connected to the central node through WiFi. The drones' wireless configuration are adjusted to work in `managed mode` and connect to a central WiFi access point. Green tiles are used as static targets and remotely controlled toy cars are used as mobile targets.

C. Experimental Scenarios

Two scenarios are shown in this paper to show the versatility of scaling *UnA* deployments in terms of coverage area, number of drones, and nature of targets.

1) Scenario 1: Mobile Targets Coverage:

`http://youtu.be/OZhdTNw5eXM`

The area of deployment is 1.25 by 2.1 meters (4.1 by 6.8 feet). Figure 7 shows different snapshots of *UnA*. The scenario of this experiment is to first have the two targets (circled in red) positioned in a certain configuration and have the UAV visually cover them. Then, move the targets and make the UAV reposition itself to cover the targets.

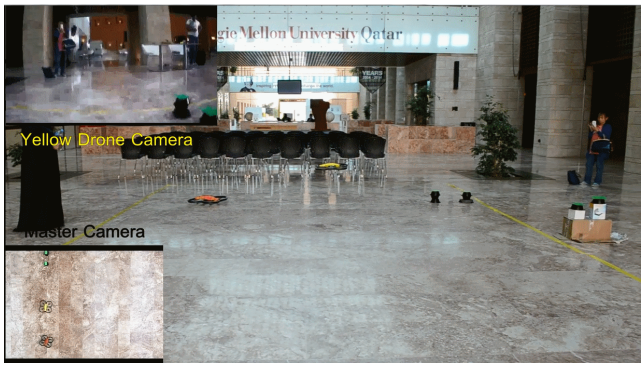
Fig. 8. Layout of the second deployment area showing drones in green circles and the master camera in the red circle.

The first two snapshots show the UAV moving itself to cover the targets in their initial configuration. The third snapshot shows the UAV covering both targets (i.e. both targets are within the view of the drone's camera). The fourth snapshot shows the UAV after moving to cover the drones in their second configuration.

2) Scenario 2: Target Coverage using Multiple Cameras:

`http://youtu.be/9pDZtdtGm2g`

The area of deployment is 5.3m by 4.7m (17.3ft by 15.4ft). Figure 8 shows the layout of the testbed. The scenario of the experiment is to first have two targets which will require one drone to be dispatched to cover them (Figure 9(a)). Then, two more targets are introduced and the second drone is dispatched to cover them (Figure 9(b)). The maximum coverage range of the drones was set to around 1 m (i.e. $R_{max} = 1$ [16]) so as to force the algorithm to dispatch two drones to cover the four targets. Note that the maximum range of the drone's camera is, typically, much larger than 1 m but this parameter was set to demonstrate a multiple-drone scenario.



(a) Step 1: two targets covered by one drone.

(b) Step 2: four targets covered by two drones.

Fig. 9. Screen shots of an experiment with two drones showing progress between the two steps of the experiment.

D. Limitations

Several limitations were encountered while using the AR Drones including repeated failures in the built in sensors (e.g. compass and altitude sensors). These failures cause serious problems in controlling the rotation and altitude of the drone. Furthermore, errors in drone placement and maneuvering inherently appear due to the simplicity of the control algorithms we adopt as well as some recurring errors in assisted maneuvers provided by the AR Drone's firmware (e.g. spontaneous rotations, unstable hovering, and sudden changes in drone altitude). These errors affect the experiments reproducibility to some extent.

V. Conclusions and Future Work

In this paper, we have introduced *Up and Away*: a test bed that enables bringing individual CPS components together into full scale experiments allowing for the evaluation of those different components under realistic conditions with minimal cost and deployment overhead. We have shown how our system, *UnA*, reduces the cost and overhead of CPS experimental deployments, enables a large range of CPS experiments to be built on it, and is modular in a way that allows for the development and integration of new components. We have also shown how using off the shelve drones come with some limitations, along with the lack of accuracy in UAV placement. We plan to address these limitations by further adding a RaspberryPi board with several sensors mounted on it to provide more sensitive sensory readings that would allow for better control of the drones.

Our future plans for *UnA* include larger deployments with three drones or more. We also plan to deploy the Click Modular router on the UAVs to allow for a more extensive evaluation of ad-hoc communication protocols between the UAVs. Furthermore, we plan to extend the "physical" components abstraction to allow the seamless integration of several other mechanical systems (e.g. ground vehicles). We also plan to explore other approaches for UAV tracking (e.g. RFID tracking).

VI. Acknowledgement

This work was made possible by NPRP grant # 4-463-2-172 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

[1] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 731–736.

[2] L. Sha, S. Gopalakrishnan, X. Liu, and Q. Wang, "Cyber-physical systems: A new frontier," in *Machine Learning in Cyber Trust*. Springer, 2009, pp. 3–13.

[3] C. Fok, A. Petz, D. Stovall, N. Paine, C. Julien, and S. Vishwanath, "Pharos: A testbed for mobile cyber-physical systems," *Univ. of Texas at Austin, Tech. Rep. TR-ARiSE-2011-001*, 2011.

[4] A. Neishaboori, A. Saeed, K. Harras, and A. Mohamed, "On target coverage in mobile visual sensor networks," in *MobiWac*, 2014.

[5] ——, "Low complexity target coverage heuristics using mobile cameras," in *IEEE MASS*, 2014.

[6] D. Jung, E. Levy, D. Zhou, R. Fink, J. Moshe, A. Earl, and P. Tsiotras, "Design and development of a low-cost test-bed for undergraduate education in UAVs," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*. IEEE, 2005, pp. 2739–2744.

[7] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1642–1648.

[8] M. Valenti, B. Bethke, D. Dale, A. Frank, J. McGrew, S. Ahrens, J. P. How, and J. Vian, "The mit indoor multi-vehicle flight testbed," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 2758–2759.

[9] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-UAV testbed," *Robotics & Automation Magazine, IEEE*, vol. 17, no. 3, pp. 56–65, 2010.

[10] T. X. Brown, S. Doshi, S. Jadhav, and J. Himmelstein, "Test bed for a wireless network on small UAVs," in *Proc. AIAA 3rd "Unmanned Unlimited" Technical Conference, Chicago, IL*, 2004, pp. 20–23.

[11] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2815–2821.

[12] T. L. Crenshaw and S. Beyer, "Upbot: a testbed for cyber-physical systems," in *Proceedings of the 3rd international conference on Cyber security experimentation and test*. USENIX Association, 2010, pp. 1–8.

[13] T. Krajník, V. Vonásek, D. Fišer, and J. Faigl, "Ar-drone as a platform for robotic research and education," in *Research and Education in Robotics-EUROBOT 2011*. Springer, 2011, pp. 172–186.

[14] H. Oh, D.-Y. Won, S.-S. Huh, D. H. Shim, M.-J. Tahk, and A. Tsourdos, "Indoor UAV control using multi-camera visual feedback," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1-4, pp. 57–84, 2011.

[15] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O'reilly, 2008.

[16] A. Neishaboori, A. Saeed, A. Mohamed, and K. Harras, "Target coverage heuristics using mobile cameras," in *International Workshop on Robotic Sensor Networks*, 2014.