This is a repository copy of *Symbiotic relationship between robots - a ROS ARDrone/YouBot library*.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/104332/

Version: Accepted Version

Proceedings Paper:
Aitken, J. orcid.org/0000-0003-4204-4020, McAree, O. and Veres, S. (2016) Symbiotic relationship between robots - a ROS ARDrone/YouBot library. In: Control (CONTROL), 2016 UKACC 11th International Conference on. 2016 UKACC 11th International Conference on Control, 31 Aug - 02 Sep 2016, Belfast, UK. IEEE . ISBN 978-1-4673-9891-6

https://doi.org/10.1109/CONTROL.2016.7737516

© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works.

Reuse
Unless indicated otherwise, fulltext items are protected by copyright with all rights reserved. The copyright exception in section 29 of the Copyright, Designs and Patents Act 1988 allows the making of a single copy solely for the purpose of non-commercial research or private study within the limits of fair dealing. The publisher or other rights-holder may allow further reproduction and re-use of this version - refer to the White Rose Research Online record for this item. Where records identify the publisher as the copyright holder, users can verify any specific terms of use on the publisher's website.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

Symbiotic relationship between robots - a ROS ARDrone/YouBot library

Jonathan M. Aitken, Owen McAree, Sandor M. Veres
Department of Automatic Control and Systems Engineering
The University of Sheffield
Sheffield, S1 3JD
United Kingdom
Email: {jonathan.aitken, o.mcaree, s.veres}@sheffield.ac.uk

***Abstract*— A Symbiotic relationship between robots is theoretically developed. It is characterised by sharing sensory information and tightly coordinating operational logic by taking care of each other's needs during missions. The system is characterised by an intertwined reasoning system while having separate conditioning and execution of plans to achieve subgoals to support each other. The results are illustrated on strong operational inter-dependence of a rover and a drone through shared logical inference. The drone uses the rover as a landing pad and the rover uses the drone to complements its sensor system by information gathering. There is a GitHub library provided in association with the demonstration for generic use of adding cameras and cooperation logic to a AR.Drone 2.0 and a KUKA youBot system. The benefits of symbiotic relationship are quantitatively evaluated on the demonstration example.**

I. Introduction

A. Background

Ambient Intelligence [1] provides a model in a robot-human interaction, where the robot pro-actively, intelligently and unobtrusively aids a human in performing a task. The Symbiotic relationships [2] which exist between a human and a robot provides very natural operation. For example in the scenario discussed by Coradeschi and Saffiotti [2], a robot safely navigates a kitchen to take some milk from the fridge to help someone make breakfast. This performs a part of the task freeing the human to carry on with their everyday work, but providing an essential service making the overall task easier. This paper extends human-robot *symbiotic relationships* to explore them between robots, a robot-robot interaction, where one robot aids another in performing a task. The assisting robot provides information rapidly, pro-actively and as unobtrusively as possible to assist in performing a task.

The symbiotic relationship presented in this paper arose from examples of rover-drone collaboration. The drone provides a rapid inspection of the environment and can offer an efficient method to find the best possible routes to take, Goodrich et al. [3] present the clear advantage of using a Unmanned Aerial Vehicle (UAV) to locate objects quickly within an area. By collaborating with ground vehicles it is then possible to expand the tasks, as a more detailed search can be conducted. Parker et al. and Hseih et al. [4], [5] use a UAV to aid ground rovers in an urban environment. UAV navigation can be achieved through a Global Positioning System (GPS), however, in environments such as for a Mars Rover or when operating indoors this is an asset which is unavailable. Therefore a UAV must be capable of localising with reference solely to a rover using no external influences. This can then serve a useful asset that can provide a wider field of view, and more environmental information to guide the rover. For example Mueggler et al. [6] perform aerial navigation of a youBot using an aerial quadcopter to help route plan through a moveable series of obstacles. The example presented in this paper differs as the ground rover itself becomes a landing site for the quadcopter. This provides a resource that is more firmly fixed to the ground vehicle, that can be deployed autonomously, as required. This enables more efficient collaboration between ground and air vehicles and maximises the use of onboard power resources only to situations where they are required. It allows for more autonomy in the spontaneous collaboration of a UAV and rover, as has been observed when humans complete a search and rescue task in the field [7].

Research in part supported by the EPSRC, grant numbers EP/L024942/1 and EP/J011843/1

B. Contribution

This paper makes several key contributions to the design of systems using the Robot Operating System (ROS), the AR.Drone 2.0 platform by Parrot and KUKA youBots.

- Developing techniques for showing efficient autonomous collaboration between a ground and air vehicle equipped with simple sensors. This takes place without advanced positioning systems such as GPS or VICON - being performed indoors using simple rover-UAV localisation. Providing information on the environment that can be used for navigation from an easily deployable on-rover UAV platform.
- Development, addition and release of a low-cost, high-quality, extensible, automatic at start-up additional camera for the AR.Drone 2.0. The libraries and installer files have been made available through GitHub.
- Extension of control and development and integration to Matlab, allowing quick, efficient development.

II. Scenario Description

The experimental scenario involves the exploration of an unknown environment by a ground rover. The environment contains a single target point of interest which is initially unknown and hidden from the rover, which has limited sensing range and movement speed. After operating the robot in isolation to gather baseline data, an autonomous UAV is introduced to assist the rover. This scenario is analogous to fields such as search and rescue operations [8] or planetary exploration [9].

The rover is initially located some distance from a wall (obstacle) which it is currently unable to detect, as shown in Figure 1. The target location is behind either the left or right hand side of the wall and can only be observed from that side. As can be seen in Figure 3 the obstacle is so large to prevent the rover using its on-board arm to view over it. As the rover approaches the wall it is unable to observe its extents so must traverse it in one direction. If the chosen direction does not lead to the target, the rover must retrace its steps and explore the other side of the wall.

When the UAV is provided to assist the rover, it will deploy and fly over the wall when it is first encountered. The UAV can then detect the target location and prevent the possibility of the rover taking the wrong route.

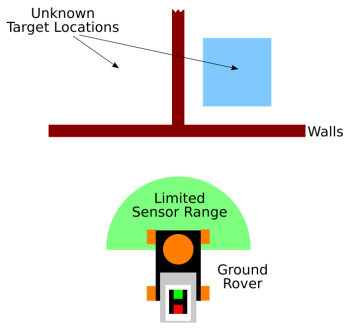


Fig. 1. Ground rover exploration scenario

III. Experimental Setup

A. Hardware - AR.Drone 2.0

The AR.Drone 2.0 is a commercial Unmanned Aerial Vehicle (UAV) manufactured by Parrot. It is a popular platform used for both indoor and outdoor tasks, especially surrounding surveillance [10] and navigation [11], [12], [13], [14] through a forward mounted camera.

The Robot Operating System (ROS) [15] is a widely-supported piece of middleware that provides a collection of useful software tools for supporting development of robot-led applications. To ease the use of the AR.Drone 2.0, a driver has been produced linking its Application Programming Interface (API) to ROS[1]. This enables easy and convenient use from any computing platform compatible with ROS.

1) Expanding the Platform: The AR.Drone 2.0 provides a good basic platform. However, it only provides a high-quality forward-facing camera. The existing downward-camera only provides video of lower quality used to perform optical flow measurements, also using the ROS driver it can not be used at the same time as the front camera; both are streamed using Bitmap images which consume large amounts of bandwidth. By adding an additional high-quality camera, navigation then can be performed using forward-facing and additional camera simultaneously. This additional camera can then placed anywhere deemed necessary on the platform, for example used as a high-quality downward-camera.

The AR.Drone 2.0 platform does not natively support the addition of cameras and therefore requires modification to provide this functionality. Previous projects have investigated the modification of onboard software [16]. We build upon this work to provide all the necessary software to add an additional camera.

The process for adding the camera can be summarised in a series of steps:

1) Cross-compile the AR.Drone 2.0 kernel source for version 2.6.32.9-gbb4d210 to include support for UVC Video drivers.
2) Cross-compile suitable video streaming software for the AR.Drone 2.0. For this project we chose MJPEG-Stream[2] as this provides a Motion JPEG stream along with an HTTP interface to provide easy access through either ROS or Matlab. This also provides full access to camera setup, so frame rate and image size can be specified. Using Motion JPEG is highly recommended as it provides a good level of compression to the video stream, this avoids bandwidth restrictions but produces a low-latency high-quality stream without compromising the native camera.
3) Modify the startup script of the AR.Drone 2.0 so that when it boots it will use bi-directional USB Support for the webcam. This is achieved by changing gpio 181 to be bi-directional, loading the UVC driver and starting MJPEG-Stream.
4) Plug the Webcam into the AR.Drone 2.0 USB port. As part of this project we use an e-con Sytems 5MP USB Webcam which is convenient for its low cost, small footprint and light weight.

Pleban et al. [16] and Daugaard and Thyregod [17] detail the process for installing and configuring the cross-compile environment. This cross-compile step is essential, as the AR.Drone 2.0 does not have an onboard compiler. Therefore it mus be install separately to a generic laptop which can then be used to compile code for the ARM Cortex A8 processor. Such a compiler is readily available and can easily be configured to be used within any environment[3],[4].

This allows the kernel to be recompiled producing a Universal Video device driver, "uvcvideo.ko", which can be inserted autonmatically on boot to provide USB Video

[1]https://github.com/AutonomyLab/ardroneautonomy.

[2]http://sourceforge.net/projects/mjpg-streamer/

[3]http://taghof.github.io/Navigation-for-Robots-with-WIFI-and-CV/blog/2012/01/13/Compiling-Code-For-The-ARDrone/

[4]http://www.nas-central.org/wiki/Setting_up_the_codesourcery_toolchain_for_X86_to_ARM9_cross_compiling

support[5]. Whilst the kernel contains the UVC Video support, it must be selected as part of the build, as by default it is turned off as it has been deemed non-essential. This cross-compiler can also be used to produce “mjpg-streamer”, a version of the MJPEG-Streamer compatible with the Arm Cortex A8.

2) Installing and Operating on the AR.Drone 2.0: In order to facilitate these tasks for future research, the requisite packages for modifying the AR.Drone 2.0 are available from the Author’s page on GitHub[6]. This set of packages provides all of the functionality detailed within this section, natively on startup. The project contains a collection of files sorted into a collection of sub-folders.

- File “uvcvideo.ko” - The custom kernel module, the Universal Video device driver, compiled for the AR.Drone 2.0. This must sit within a folder named “/custom_modules” on the AR.Drone 2.0.
- Sub-Folder “mjpg-streamer” - MJPEG-Streamer, compiled for the AR.Drone 2.0. This must sit as a sub-folder within a folder named “/custom_modules” on the AR.Drone 2.0.
- File “check_updates.sh” - A modified startup script for the AR.Drone 2.0, that will enable the kernel modules for the camera, and automatically start them on boot. This replaces the normal “check_updates.sh” file within the /bin folder on the AR.Drone 2.0.
- File “launch_stream.sh” - A script that is called within check_updates to launch MJPEG-Streamer, within which the resolution and frame rate of the camera can be defined. This must sit within a folder named “/custom_modules” on the AR.Drone 2.0.

B. *Hardware - KUKA youBot*

A standard KUKA youBot with a Hokuyu laser scanner[7] is used as the ground rover [18], which provides a 240° laser arc in front of the vehicle that can be used for locating objects. This loading platform of the vehicle is fitted with a helipad target to provide access for the AR.Drone 2.0.

IV. Control in Matlab and Simulink

Both the rover and the UAV are commanded through ROS[8] by setting the corresponding body frame velocity demands $(\dot{x}_d, \dot{y}_d, \dot{\psi}_d)$ with the addition of $\dot{z}_d$ for the UAV. The control system is developed in Simulink[9], and the Robotics System Toolbox[10] to communicate with ROS nodes.

The control systems assume full autonomous control of both the Rover and UAV. Once the experiment is started, no further human intervention is necessary in order for the rover to find the target, whether assisted by the UAV or not. The UAV is piloted by a fully autonomous controller which localises it to the rover as described in the following sections.

[5]The kernel is available from `https://devzone.parrot.com/`

[6]`https://github.com/jonaitken/ARDroneCamera`

[7]`https://www.hokuyo-aut.jp/02sensor/07scanner/urg_04lx_ug01.html`

[8]`http://www.ros.org/`

[9]`http://uk.mathworks.com/products/simulink/`

[10]`http://uk.mathworks.com/products/robotics/`

A. *UAV Control*

The UAV is initially located on a landing pad mounted to the back of the rover. Once commanded to take-off, the downward camera is used to localise the drone over the landing pad, Figure 2. Position over the helipad is controlled with decoupled lateral and longitudinal PID controllers.

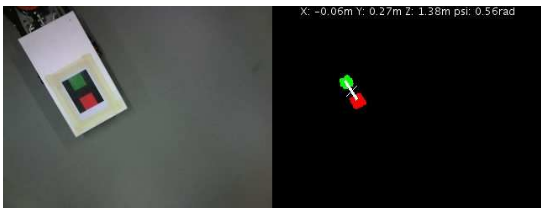


Fig. 2. Helipad localisation

The UAV is commanded to climb at a constant rate of $\dot{z}_d = 0.2ms^{-1}$, whilst maintaining position over the landing pad, until the forward camera detects that the way is clear. The UAV then flies forwards by dead-reckoning to pass over the wall and the downward camera tasked with target detection. Figure 3 illustrates the stages of the UAV assistance flight.

The control block diagram in Figure 4 shows the separate components used as part of the mission. These control the basic operations of the AR.Drone 2.0 through ROS, commanding landing, takeoff and flight control. The block diagram contains components to identify the landing symbol, and take flight-control decisions.

The flight controller is decomposed into a state flow shown in Figure 5. The stateflow for the AR.Drone 2.0 can be decomposed into a series of steps that are easily captured in a Simulink stateflow:

- Takeoff - Takeoff to a height of 1m is an automated process on the AR.Drone 2.0.
- HelipadTakeOff - Once above 0.5m, and as the aircraft is transitioning out of ground effect, allow the autonomous controller to take control and begin to centralise the vehicle over the target. Begin to climb to investigate surroundings, this climb takes place until the forward path in front of the vehicle is clear to ensure collision does not occur.
- Forward - This state maintaining a view on the helipad, to ensure that localisation on the KUKA youBot perseveres, and that the way forward is clear. Move forward at low speed for 5s or until a target is located.
- Backwards - After 5s or when a target is found reverse to localise over the helipad.
- HelipadLanding - Descend over the helipad maintaining localisation on the target to 0.6m.
- Landing - Initialise landing using inbuilt automated procedure on the AR.Drone 2.0.

B. *Rover Control*

The rover is initially tasked with driving forwards at a constant velocity of $\dot{x}_d = 0.1ms^{-1}$ until the wall is sensed at approximately $0.5m$. Upon encountering the wall, without UAV assistance, the rover then chooses a random

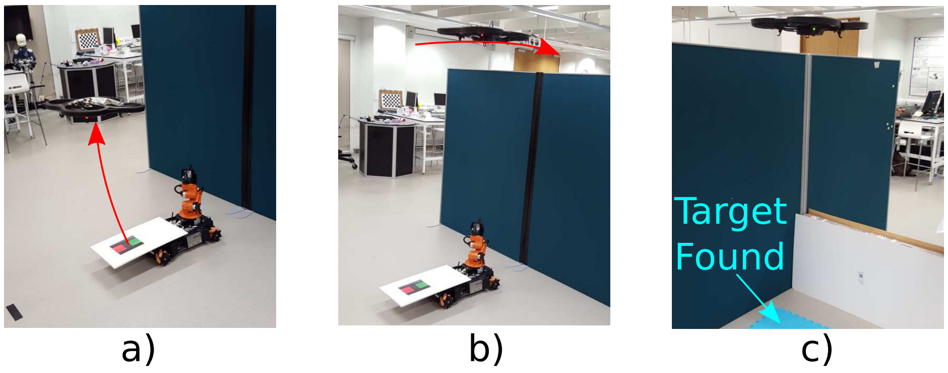


Fig. 3. UAV assistance flight profile. a) Take-off and climb over landing pad. b) Proceed over wall. c) Detect target

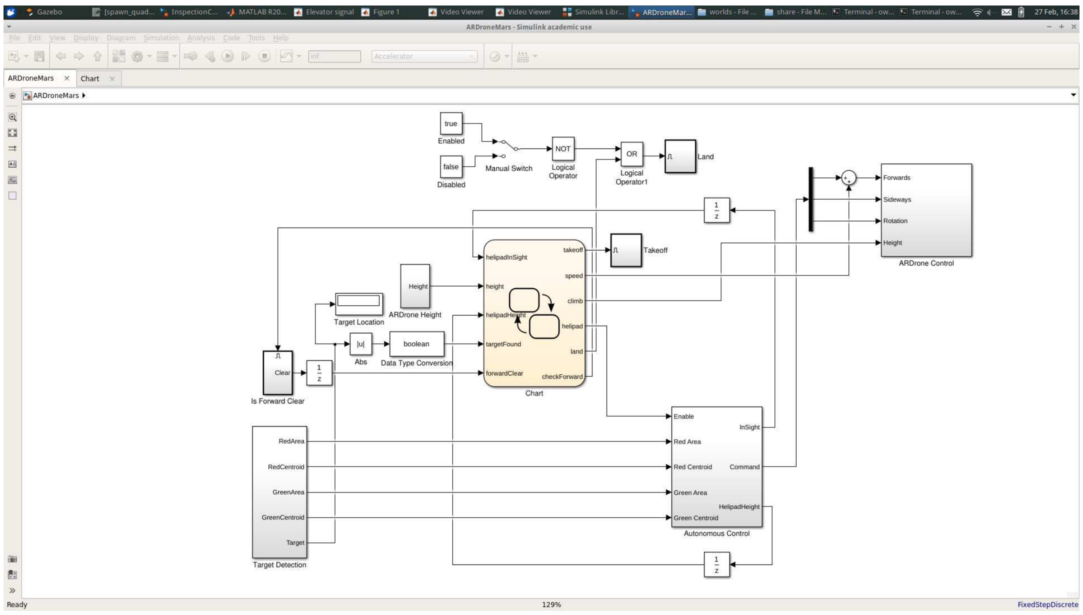
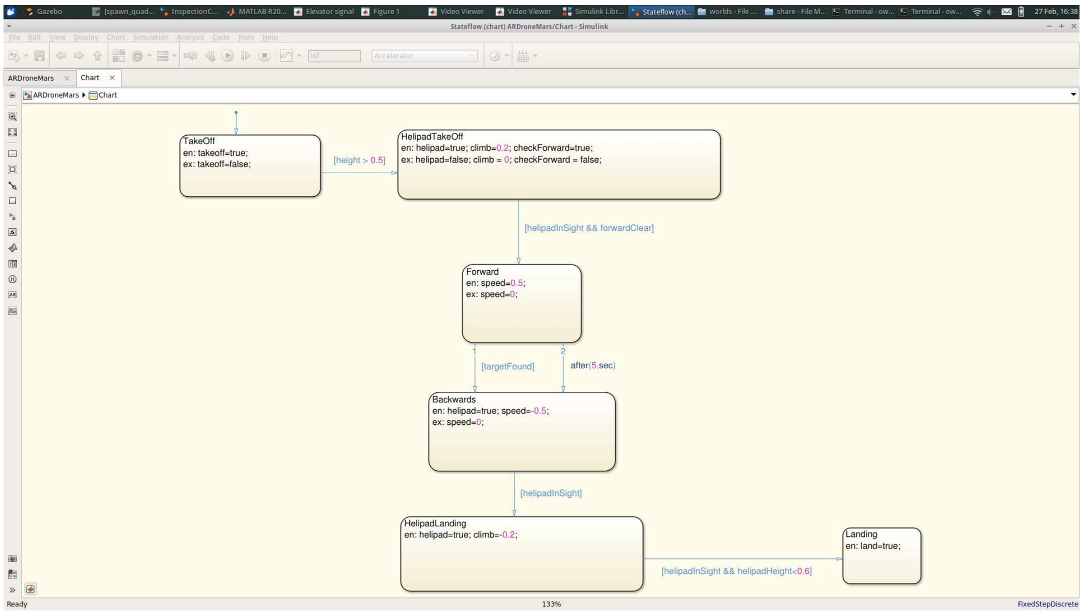


Fig. 4. Control Block Diagram for the AR.Drone 2.0


Stateflow (chart) ARDroneMars/Chart - Simulink
ARDroneMars
Chart
[height > 0.5]
[helipadInSight && forwardClear]
[targetFound]
after(5,sec)
[helipadInSight]
[helipadInSight && helipadHeight<0.6]
Ready
133%
FixedStepDiscrete

TakeOff
en: takeoff=true;
ex: takeoff=false;

HelipadTakeOff
en: helipad=true; climb=0.2; checkForward=true;
ex: helipad=false; climb = 0; checkForward = false;

Forward
en: speed=0.5;
ex: speed=0;

Backwards
en: helipad=true; speed=-0.5;
ex: speed=0;

HelipadLanding
en: helipad=true; climb=-0.2;

Landing
en: land=true;

Fig. 5. State Flow Diagram for the AR.Drone 2.0

direction to explore first and proceeds with $|\dot{y}_d| = 0.1ms^{-1}$ whilst maintaining $0.5m$ spacing to the wall with a simple proportional controller. Once the edge of the wall is passed the rover proceeds behind it by dead-reckoning and attempts to visually detect the target.

If no target is present the rover retraces its step via dead-reckoning until it is back in front of the wall, at which point it repeats its search in the opposite direction.

When assisted by the UAV, the rover stops when the wall is first encountered and launches the UAV to fly over the wall and detect the target. The rover must then wait for the UAV to return to its helipad and then proceed in the correct direction as detected by the UAV.

The process for the KUKA youBot can then also be decomposed into a Simulink stateflow:

- ApproachWall - Approach the wall at a constant, low speed. Once at a fixed distance of 0.5m, branch to a series of different options.
- goLeft - Traverse the wall, to find the edge, initially heading left.
- goRight -Traverse the wall, to find the edge, initially heading right.
- goBack - Once the enclosed space has been inspected for the object reverse back past the wall to commence investigation in the other direction.

Each of the movement-based states can be further decomposed into options, goLeft and goRight are identical actions composed of a series of states:

- Stop1, Stop2, Stop3, Stop4 - Cease movement, zeroing any active youBot movement.
- TraverseWall - Move either to the right or to the left, setting appropriate lateral velocities for the youBot.
- PassWall - Once the edge of the wall has been found, set longitudinal velocity for the youBot to travel forwards 1.5m to pass the wall.
- SearchPos - Move around the wall to reach the search position in the enclosed space.
- FindTarget - Visually inspect the enclosed space for the object of interest.

The goBack command resets the youBot position on the blind-side of the wall. It can be decomposed into states:

- SearchPos - Move back from the search position out of the enclosed area.
- PassWall - Move backwards to move back to the blind-side of the wall.
- TraverseWall - Move to the centre of the wall in the appropriate direction.
- Stop3, Stop4 - Cease movement, zeroing any active youBot movement.

C. Algorithmic Overview

In this experiment there are two specific cases that are investigated, lone operation of the rover and co-operation with a UAV. These steps can be summarised as a pair of simple processes that can be conducted fully autonomously:

- Proceed to wall.
- Once at the wall pick a random direction to proceed, proceed until corner found.
- Inspect whether object is in location. If it is then mission is completed.
- Perform reverse operations to return to start point
- Select other direction to proceed along the wall, proceed until corner found.
- Inspect whether object is in location. If the object is present the mission is a success, otherwise the object is not present.

When assisted by the UAV, the rover stops when the wall is first encountered and launches the UAV to fly over the wall and detect the target. The rover must then wait for the UAV to return to its helipad and then proceed in the correct direction as detected by the UAV.

With UAV assistance the mission becomes less complex with the additional source of information:

- Proceed to wall.
- Once at the wall command UAV launch.
- UAV flies forward maintaining visual lock on the rover.
- UAV detects ground target and relative location with respect to rover.
- UAV returns to rover and lands reporting direction of target if found.
- If no target found the mission is complete otherwise rover proceed until corner found to reach location.

V. Results

The scenario described above was executed 10 times in each configuration, with and without UAV assistance. The total time taken to complete the task is illustrated in Figure 6.

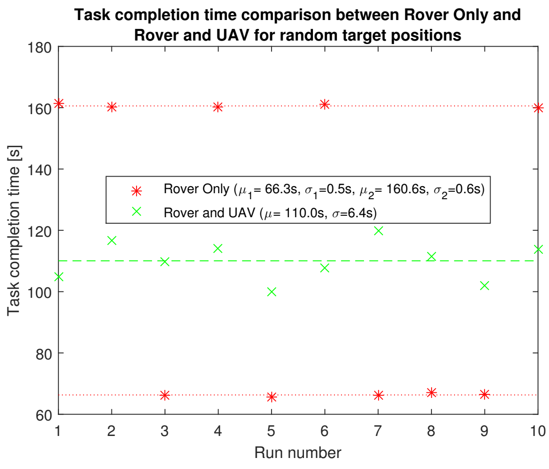


Fig. 6. Task completion time comparison between Rover only and Rover and UAV

It can be seen that the completion time for the rover alone exhibits a bimodal distribution, determined by whether the rovers initial (random) search direction is correct. When assisted by the UAV the rover always chooses the correct direction, resulting in a unimodal distribution.

The mean completion time of the UAV assisted results is approximately midway between the two means of the rover only results due to the time taken for the UAV to find the target. The variance of the UAV assisted results is an order of magnitude higher than that for the rover alone due to a higher variability in flight times; this is especially true when landing on the helipad where good visual lock must be acquired in order to execute the manoeuvre.

Whilst this behaviour can somewhat be anticipated, use of the UAV will always succeed more quickly. The nature of this interaction is key to the speed. In this case the larger, and more inaccessible the object, the larger the benefit of using the UAV. The rover always travels twice the distance required when it initially picks an incorrect direction, and any travel is redundant when there is no object. The UAV, providing information through a symbiotic relationship, always enables the rover to take the correct decision and travel the minimum distance, even with no object.

VI. Conclusions

The results presented in this paper illustrate one scenario in which a modified AR.Drone 2.0 UAV can be used to assist a robot exploration task. An additional speed improvement could be achieved by allowing the rover to begin moving before the UAV returns to land, having the UAV either land on a moving helipad [19] or wait for the rover to reach the target before landing. The techniques in this paper link with the Model-Based Framework for developing safe and verifiable algorithms [20], and are developed within the same Matlab framework.

This achievement has been assisted by adding an additional camera to the Parrot AR.Drone 2.0. In this case the camera is mounted to face downwards, although it can be mounted in any orientation desired. This provides a video stream capable of being used for vision-based control of the UAV. This provides a resources that can be used to plan and execute a mission, providing redundancy in cameras that can be exploited allowing vehicles to autonomously reconfigure [21], [22], in the case of failure or needing to perform different tasks; for example forward navigation, and hovering above a target as shown.

It has been demonstrated that a rover assisted by a UAV is able to complete the task in a consistent time, with significant improvements over the worst case rover-only results. The particular scenario used is a simplification of a real world scenario, which contains precisely one target in one of two reachable locations. The benefits of UAV assistance will become more significant in more complex scenarios. For example, if there is no target in either location the rover need not waste time exploring them. Alternatively, if the locations are not reachable by simply traversing the obstacle, the UAV can additionally provide course routing assistance.

References

[1] C. Ramos, J. C. Augusto, and D. Shapiro, "Ambient intelligence — the next step for artificial intelligence," *Intelligent Systems, IEEE*, vol. 23, no. 2, pp. 15–18, 2008.

[2] S. Coradeschi and A. Saffiotti, "Symbiotic robotic systems: Humans, robots, and smart environments," *Intelligent Systems, IEEE*, vol. 21, no. 3, pp. 82–84, 2006.

[3] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, "Supporting wilderness search and rescue using a camera-equipped mini uav," *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 89–110, 2008.

[4] L. Chaimowicz, A. Cowley, D. Gomez-Ibanez, B. Grocholsky, M. Hsieh, H. Hsu, J. Keller, V. Kumar, R. Swaminathan, and C. Taylor, "Deploying air-ground multi-robot teams in urban environments," in *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, 2005, pp. 223–234.

[5] M. A. Hsieh, A. Cowley, J. F. Keller, L. Chaimowicz, B. Grocholsky, V. Kumar, C. J. Taylor, Y. Endo, R. C. Arkin, B. Jung, D. F. Wolf, G. S. Sukhatme, and D. C. MacKenzie, "Adaptive teams of autonomous aerial and ground robots for situational awareness," *Journal of Field Robotics*, vol. 24, no. 11-12, pp. 991–1014, 2007.

[6] E. Mueggler, M. Faessler, F. Fontana, and D. Scaramuzza, "Aerial-guided navigation of a ground robot among movable obstacles," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2014, pp. 1–8.

[7] T. Perkins and R. R. Murphy, "Active and mediated opportunistic cooperation between an unmanned aerial vehicle and an unmanned ground vehicle," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2013, pp. 1–8.

[8] G.-J. M. Kruijff, F. Colas, T. Svoboda, J. Van Diggelen, P. Balmer, F. Pirri, and R. Worst, "Designing intelligent robots for human-robot teaming in urban search and rescue." in *AAAI Spring Symposium: Designing Intelligent Robots*, 2012.

[9] R. Volpe, "2014 Robotics Activities at JPL," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), Montreal, Canada*, vol. 17, 2014.

[10] 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.

[11] P.-J. Bristeau, F. Callou, D. Vissiere, and N. Petit, "The navigation and control technology inside the AR.Drone micro uav," in *18th IFAC world congress*, vol. 18, no. 1, 2011, pp. 1477–1484.

[12] J. J. Lugo and A. Zell, "Framework for autonomous on-board navigation with the AR.Drone," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1-4, pp. 401–412, 2014.

[13] M. Saska, T. Krajník, J. Faigl, V. Vonásek, and L. Přeučil, "Low cost mav platform AR.Drone in experimental verifications of methods for vision based autonomous navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[14] 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*, 2012, pp. 2815–2821.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[16] J.-S. Pleban, R. Band, and R. Creutzburg, "Hacking and securing the AR.Drone 2.0 quadcopter: investigations for improving the security of a toy," in *Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications*. International Society for Optics and Photonics, 2014.

[17] M. Daugaard and T. Thyregod, "Semi-autonomous indoor navigation for an airborne robot," Master's thesis, Aarhus University, 2012.

[18] R. Bischoff, U. Huggenberger, and E. Prassler, "KUKA youBot-a mobile manipulator for research and education," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1–4.

[19] O. McAree, J. Clarke, and W.-H. Chen, "Development of an autonomous control system for a small fixed pitch helicopter," in *2nd International Conference on Advanced Computer Control (ICACC)*, 2010.

[20] O. McAree, J. M. Aitken, and S. Veres, "A model based design framework for safety verification of a semi-autonomous inspection drone," in *Proceedings of the UKACC International Conference on Control (CONTROL)*, 2016.

[21] L. A. Dennis, M. Fisher, J. M. Aitken, S. M. Veres, Y. Gao, A. Shaukat, and G. Burroughes, "Reconfigurable autonomy," *KI-Künstliche Intelligenz*, vol. 28, no. 3, pp. 199–207, 2014.

[22] J. M. Aitken, S. M. Veres, and M. Judge, "Adaptation of system configuration under the robot operating system," *Proceedings of the 19th world congress of the international federation of automatic control (IFAC)*, 2014.