

Decentralized collaborative transport of fabrics using micro-UAVs

Ryan Cotsakis, David St-Onge and Giovanni Beltrame

Abstract—Small unmanned aerial vehicles (UAVs) have generally little capacity to carry payloads. Through collaboration, the UAVs can increase their joint payload capacity and carry more significant loads. For maximum flexibility to dynamic and unstructured environments and task demands, we propose a fully decentralized control infrastructure based on a swarm-specific scripting language, Buzz. In this paper, we describe the control infrastructure and use it to compare two algorithms for collaborative transport: field potentials and spring-damper. We test the performance of our approach with a fleet of micro-UAVs, demonstrating the potential of decentralized control for collaborative transport.

I. INTRODUCTION

Transport tasks for industrial, commercial and medical applications are now considering small-scale aerial technologies ready for deployment. New use cases are emerging, but the available payload remains the main limitation for their implementation. Quite a novel use case was presented to our team from a fashion designer: to involve micro-UAVs in the domestic task of getting dressed. This proposal is challenging at many levels, namely: interaction, clothing design, and UAV control. In this paper, we address the first piece of the puzzle: how to use very small and therefore safe devices to transport light pieces of clothing. Indeed, the proximity of the user and the narrow space of a dressing room prevent the use of large devices. However, a decrease in size tends to considerably decrease the payload. We propose to base our solution on the use of multiple micro-UAVs collaborating as a whole for the transport task, as shown in Fig. 1. The concepts required to control the fleet are taken from swarm theory and its latest robotic implementation, such as scalable behaviours based on local interactions.

The nature of the payload is itself a challenge for aerial transport. A flexible material supported from many points creates a complex dynamic between the carriers. Solving this problem has many other useful applications: supporting a net to catch other UAVs, transporting hoses to drop water over fires, or slings for rescue missions on mountains have comparable dynamics with bigger payload requirements. Nets, fabrics, hoses and slings, can all be approximated with one or many Deformable Linear Objects (DLOs).

Since suspended fabrics are heavily influenced by the motion of air, the forces on the UAVs are largely unpredictable, and can be quite large in magnitude. In this case, the UAVs that share the payload must compensate for any



Fig. 1: Four Crazyflie micro-UAVs collaborating to transport a white fabric without requiring any central controller.

disturbances in real time. This can be achieved by having the UAVs cooperate, and adjust their movement according to the positions and dynamics (acceleration, torque, etc.) of their neighbours.

This paper is structured as follows: we discuss inspiring works in Sec. II, then we detail our decentralized architecture in Sec. III and we derive the control algorithms in Sec. IV. Finally, the results of a set of experiments conducted with Crazyflie micro-UAVs are presented and discussed in Sec. V.

II. RELATED WORKS

The transport of DLOs has been recently studied [1]. The feasibility of this task, with three UAVs, was demonstrated in simulation using a particle swarm optimization to get the proper set of PID gains before flight. This work, as with most of the literature of flexible payload transport with UAVs, adopted a centralized controller. Similarly, the impressive body of work of D’Andrea’s group at ETH Zurich includes works such three UAVs throwing and catching a ball in a net [2] and the transport of a flexible ring with six UAVs [3]; but they rely on a fully centralized and latency-free state estimator and controller.

Distributed or decentralized approaches to the problem are seen less frequently. Distributed collaborative transport was achieved for wheeled robots, pushing a rigid geometric object [4] with a combination of potential field forces. More recently, two UAVs were designed to cooperate in a fully decentralized configuration for the transport of a rigid body using only inertial measurement and vision [5]. Both are inspiring works, but are still far from the target of this work.

The same type of micro-UAVs used in our experiments, the Crazyflie, was subject to a detailed swarm controller

M. Cotsakis is an undergraduate student from the University of British Columbia, 2329 West Mall, Vancouver, BC. Dr. St-Onge and Prof. Beltrame are with the Department of Computer and Software Engineering, École Polytechnique de Montréal, 2900 Boul Édouard-Montpetit, Québec CA. e-mail: (name.surname@polymtl.ca).

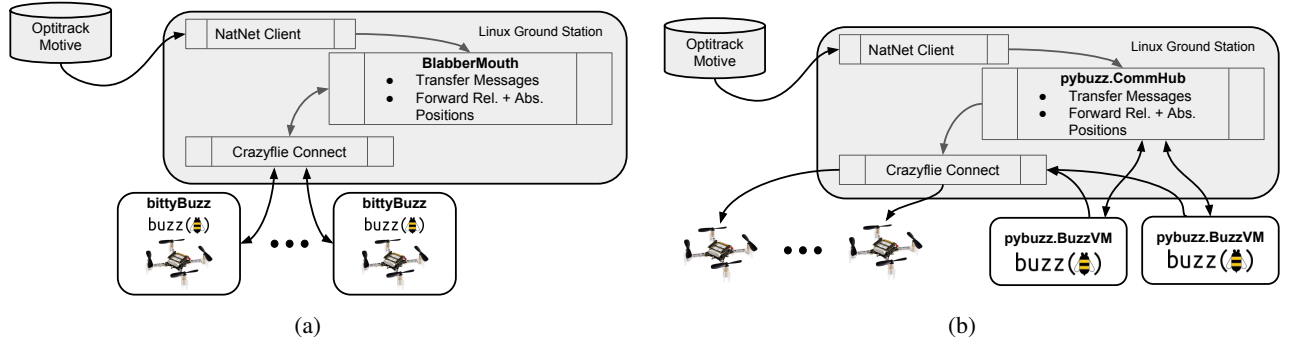


Fig. 2: Buzz deployment architecture for the Crazyflie: **(a)** Buzz running on-board. **(b)** Buzz emulated using PyBuzz.

design [6]. We leverage the stability of their on-board control and the scalability of their configuration to deploy our solution. Nevertheless, collaborative payload transport has never been attempted with such a small and sensitive device. Other solutions for controlling a swarm of Crazyflies has been considered [7], [8], but they are centralized. As with Preiss et al., their scripts are specific to their implementation for this platform, whereas we leveraged a domain-specific programming language for swarms [9] to guarantee portability to other platforms.

III. DECENTRALIZED ARCHITECTURE

The aim of this project is not only to achieve a fleet of Crazyflies collaborating on a transport task, but to study a behaviour that can be ported to other robotic platforms. The Buzz domain-specific language [9] and its associated virtual machine was successfully deployed on 3DR Solos, DJI Matrice 100, Intel Aero, and Clearpath Husky through a ROS connector as well as on K-Team Khepera, Kilobots and Zooids, natively.

A. Buzz

To accelerate the implementation of swarm behaviours, Buzz provides a set of special primitives from which we leverage two essential concepts: a) swarm aggregation and b) neighbour operations. We detail here these concepts to ease the comprehension of the following implementation, but all information can be found in previous publications specific to Buzz language [9] and example scripts that are available online¹.

Swarm Aggregation is a primitive which allows for grouping of robots into sub-swarms, through the principle of dynamic labelling [9]. The `swarm` construct is used to create a group of robots which can be attributed with a specific behaviour that differs from the other robots, based either on the task or robot abilities.

Neighbour Operations in Buzz refer to a rich set of functions which can be performed with or on neighbouring robots through situated communication [10]. Neighbours are defined from a network perspective as robots which have a direct communication link with each other. With situated

communication, whenever a robot receives a message, the origin position of the message is also known to the receiver.

Finally, any Buzz script is compiled into an optimized, memory-efficient, and platform-agnostic bytecode to be executed on the Buzz Virtual Machine (BVM). To interface the BVM with the robots' actuators and sensors, the integrator needs to write his own C hooks that are callable from a Buzz script.

B. PyBuzz

Rather than embedding the BVM into the firmware of each robot, we decided to accelerate the development of the control algorithms by creating a centralized emulator for our decentralized configuration. Emulating a distributed software architecture requires a wrapper that can be instantiated to connect to each hardware node. To solve this problem, we created a Python module for wrapping the BVM called PyBuzz, such that in Python, one can construct a BVM as a Python object, and link Python functions as callable functions in Buzz. When interpreting the Buzz object code, the BVM performs calls to these Python functions.

By importing `pybuzz` in a Python script, one can construct any number of BVM objects. The translation between C and Python was achieved using Cython [11], a tool that is used for writing C extensions for Python.

On a ground station, we have information about the locations of all of the UAVs. These locations are fed into a communication hub, `pybuzz.CommHub`, as shown in Fig. 2-b. The communication hub emulates the robots' communication range, giving each BVM its current location, and the relative locations of its neighbours. While each BVM continuously steps through the Buzz script, the communication hub automatically manages the transmission of all messages sent between each BVM. The only information sent to the Crazyflie's hardware is its current location (received from `pybuzz.CommHub`) and instructions for course adjustments (received from `pybuzz.BuzzVM`).

In the end, a Buzz script tested with the PyBuzz emulator (Fig. 2-b) can work unmodified on board a Crazyflie (Fig. 2-a).

¹<http://the.swarming.buzz>

IV. DECENTRALIZED CONTROL

The accurate control of a fleet carrying a shared payload is a problem usually approached from a robust low-level control perspective [3]. However, modelling the payload between the robots as a set of local interactions can bring it up to the fleet coordination level, for which decentralized solutions are widely studied in swarm behaviours.

A. Flexible payload model

We take inspiration from the modelling of hoses, approximated for UAVs collaborative payload with discrete linear objects [12]. We also postulate that the transport is achieved only with UAVs at cruise heights, i.e. with the fabric freely hanging. However, our experiments show that this approximation can hold for stable take-off and landing while holding the fabric in certain conditions (see Sec. V).

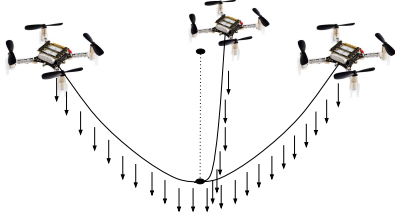


Fig. 3: Modelling fabric transport by connected DLOs. Each of the three Crazyflies carries half of a catenary curve with its vertex at a point directly below the centre of mass of the swarm

The payload can be approximated by cables with uniformly distributed weight creating catenary curves that hang from each robot to a point directly below the centre of mass of the swarm, which we will refer to as the *connecting point*. The vertex of each catenary curve coincides with the connecting point. This model is depicted in Fig. 3. With n robots all at the same height sharing a payload of mass m , the vertical tensile force T_z is approximately

$$T_z = \frac{mg}{n} \quad (1)$$

The horizontal force T_x is a much more involved calculation. We recall the equation of a catenary curve to be

$$z(x) = a \cosh \frac{x}{a} \quad (2)$$

where a is a parameter that describes how taught the approximating cable is. Remaining under the assumption that the robots have the same altitude, a can be solved for numerically in the transcendental equation

$$L = a \sinh \frac{x_0}{a} \quad (3)$$

where L is the length of the approximating cable from the robot to the connecting point, and x_0 is the horizontal distance from the robot to the centre of mass of the swarm. If we take the derivative of Eqn. 2, we see that

$$\frac{T_z}{T_x} = z'(x_0) = \sinh \frac{x_0}{a} = \frac{L}{a} \quad (4)$$

If we observe through Eqn. 3 that a is a function of x_0 , we can see that T_x is a function of x_0 and other constants:

$$T_x(x_0) = \frac{mg}{nL} a \quad (5)$$

B. Spring-Damper analogy

An intuitive method of having the UAVs maintain a stable formation is to simulate the forces due to springs and dampers emplaced between each robot. Spring and damper control was shown to work well in a simulated swarm [13]. The repulsive force between two robots as a function of the distance d between them would be

$$F_{rep} = -k(d - l_0) - B\dot{d} \quad (6)$$

where k and l_0 are respectively the spring constant and unstretched length of the spring, and B is the damping constant of the damper.

To tune the parameters k and B , recall the horizontal force on each robot due to the payload from Eqn. 5. Let us define k_p to be the value of T'_x at the equilibrium position of x_0 . Then for small perturbations from the equilibrium position, the horizontal force due to the payload can be approximated by a spring with spring constant k_p connecting the centre of mass of the swarm to the robot. By relating the payload to a spring force, we can interpret the entire system as a sum of forces from springs and dampers to simplify the analysis.

We can calculate the net force on a particular robot in the x direction, the direction towards the centre of mass of the swarm, for a small change Δx from the equilibrium position. We denote this force as F_x and express it as a sum over the robot's N neighbours:

$$F_x = -\left(k_p + k \sum_{i=1}^N \cos^2 \theta_i\right) \Delta x \quad (7)$$

where θ_i is the angle from the centre of mass of the swarm to neighbour i with respect to the robot in question. In the same way we can calculate the force in the y direction resulting from a small perturbation Δy to be

$$F_y = -\left(k \sum_{i=1}^N \sin^2 \theta_i\right) \Delta y \quad (8)$$

By substituting equations 7 and 8 with the values

$$k_x = k_p + k \sum_{i=1}^N \cos^2 \theta_i, \quad k_y = k \sum_{i=1}^N \sin^2 \theta_i \quad (9)$$

we are able to simplify the system of springs down to two springs. One parallel to the x -axis with spring constant k_x , and one parallel to the y -axis with spring constant k_y . What remains is to simplify the system of dampers.

Similarly to how we defined k_x and k_y in Eqn. 9, we define

$$B_x = B \sum_{i=1}^N \cos^2 \theta_i, \quad B_y = B \sum_{i=1}^N \sin^2 \theta_i \quad (10)$$

and interpret our system separately in dimensions x and y when choosing parameters B and k . In the x direction

we have, attached to the Crazyflie's mass, a spring and a damper in parallel with parameters k_x and B_x respectively. In the y direction, we have a spring and damper in parallel with parameters k_y and B_y respectively. We tune B and k such that both of these systems in each dimension are nearly critically damped.

The calculation of \dot{d} in Eqn. 6 is not a straight forward task when d is being sampled discretely in time and space. To mitigate the errors induced by the lack of continuity, we use all n previously measured values of d , and weigh each measurement's importance with a coefficient that's magnitude decays exponentially with time. The simple implementation of this algorithm is to initialize another distance variable \tilde{d} that acts as a representative of all previous values of d .

Given that at time t_n we measure the distance between robots to be d_n , we define

$$\tilde{d}_n = w_n \tilde{d}_{n-1} + (1 - w_n) d_n, \text{ with } \tilde{d}_0 = d_0 \quad (11)$$

and

$$w_n = \exp\left(-\frac{t_n - t_{n-1}}{\tau}\right) \quad (12)$$

where τ is a time constant that governs how quickly information from previous measurements is lost. We aim to show that we can approximate the velocity as follows:

$$\dot{d} = \frac{d_n - \tilde{d}_n}{\tau} \quad (13)$$

In the case where d is measured at regular time intervals Δt , then $w_n = w = \exp(-\Delta t/\tau)$, $\forall n$ which gives a better approximation:

$$\dot{d} = \frac{(1 - w)(d_n - \tilde{d}_n)}{w \Delta t} \quad (14)$$

To show that Eqn. 14 is a reasonable approximation, let us consider the explicit form of \tilde{d}_n :

$$\tilde{d}_n = (1 - w) \sum_{i=0}^{\infty} d_{n-i} w^i \quad (15)$$

Since we aim to obtain an expression for \dot{d} , let us assume that it is relatively constant (at least for a short interval of time). This allows us to make the approximation

$$d_{n-i} \approx d_n - i \Delta t \dot{d} \quad (16)$$

With this approximation, Eqn. 15 can be expressed as

$$\tilde{d}_n = (1 - w) \sum_{i=0}^{\infty} (d_n - i \Delta t \dot{d}) w^i \quad (17)$$

which can be evaluated exactly:

$$\tilde{d}_n = d_n - \frac{w \Delta t \dot{d}}{1 - w} \quad (18)$$

Finally, solving for \dot{d} yields Eqn. 14.

With w being an exponential function of $\frac{\Delta t}{\tau}$, it can easily be shown that as $\frac{\Delta t}{\tau} \rightarrow 0$,

$$\frac{1 - w}{w \Delta t} = \frac{1}{\tau} \quad (19)$$

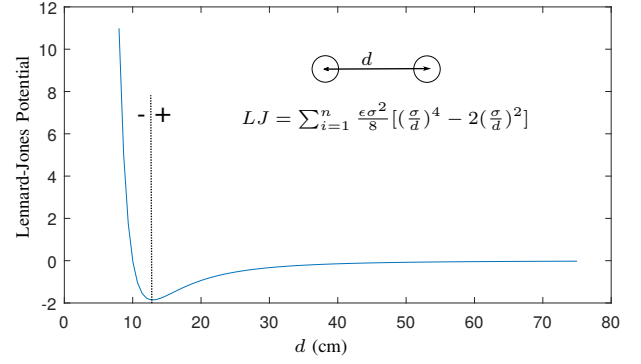


Fig. 4: Lennard-Jones potential adapted for Crazyflies. The '-' and '+' domains are respectively the repulsive and attractive parts, for which the pivot point is set with parameter σ . d is the distance between two robots and ϵ is a parameter acting as a control gain on the potential.

which when substituting in Eqn. 14 implies Eqn. 13. One downside to using Eqn. 13 is that we require $\tau \gg \Delta t$, and since \dot{d} is an approximation of the velocity $\frac{\tau}{2}$ seconds in the past, our measurement of \dot{d} is delayed. The lack of w and Δt dependence in Eqn. 13 implies that it can be used over Eqn. 14 whenever d is sampled irregularly, as long as τ is sufficiently large.

C. Bacteria interaction analogy

Among the most popular formalizations of biological swarm behaviours, potential functions are a simple, yet flexible control approach. Artificial potential functions have been used extensively for robot navigation and control [14], [15]. Based on their knowledge of their neighbours' positions, each robot computes a virtual force vector:

$$F = \sum_{i=1}^N f(d_i) e^{j\theta_i} \quad (20)$$

where θ_i and d_i are the direction and the distance to the i th perceived obstacle or robot, and the function $f(d_i)$ is the negative gradient of an artificial potential function. One of the most commonly used artificial potentials is the Lennard-Jones potential, adapted for our physical system as shown in Fig. 4.

The two parts of the potential equation represent the attractor and repulsor effect. This potential is driven by two parameters: the target distance between robots σ and the strength ϵ of the potential.

Based on its popularity, we selected Lennard-Jones potential to control the behaviour of our robotic swarm:

$$LJ = \frac{\epsilon \sigma^2}{8} \left[\left(\frac{\sigma}{d} \right)^4 - 2 \left(\frac{\sigma}{d} \right)^2 \right] \quad (21)$$

We can compute the resulting force exerted as the negative gradient of Eqn. 21:

$$F_{rep} = \frac{\epsilon \sigma}{2} \left[\left(\frac{\sigma}{d} \right)^5 - \left(\frac{\sigma}{d} \right)^3 \right] \quad (22)$$

This representation of the Lennard-Jones repulsive force has a very elegant Taylor expansion about $d = \sigma$, that is

$$F_{rep} = -\epsilon(d - \sigma) + O((d - \sigma)^2) \quad (23)$$

The first term in Eqn. 23 has the same form as the first term in Eqn. 6. With $\epsilon = k$ and $\sigma = l_0$, the Lennard-Jones potential is, to second order, identical to a spring system at the equilibrium position and can be approximated by a harmonic oscillator. By representing the potential in this form, we are able to compare Lennard-Jones experiments with spring-damper experiments by using the same parameters derived in Sec. IV-B.

D. Fleet translation

With both control approaches, a goal (target location) is represented as an attractor influencing the whole group. The final displacement vector at each step is then computed from a weighted sum of this attraction force and the forces resulting from the formation algorithm. Under ideal circumstances, the robots move towards their final position at constant velocity.

V. EXPERIMENTS

To start, we devised an unorthodox method to have a swarm of Crazyflies be seen and distinguished by our motion sensing platform, Optitrack, since the UAVs are too small to allow for many unique configurations of reflective markers. Our solution was to attach a single marker on each Crazyfly, which provides the position of the UAV in 3-D space, but not the orientation. We are able to control the attitude of the Crazyfly with the on-board controller using its Inertial Measurement Unit. To have the robots be uniquely identifiable, we tell the software approximately where each Crazyfly is expected to start, and conduct a grid search before takeoff. For the duration of the flight, we assume continuous motion of the Crazyfly. This way the software can identify which robot is which based on the previous frame.

Building on the infrastructure work described in Sec. III, we implemented Buzz scripts for both algorithms. We conducted eleven flight tests, each involving three Crazyflies. The first of these tests demonstrates the flight of the three Crazyflies without any payload, nor any control algorithm facilitating the distances between robots. Five of the flight tests demonstrate the transport of fabric using the spring-damper algorithm, and the last five demonstrate the transport of fabric using the Lennard-Jones algorithm. All other variables were kept constant. In each flight test, the Crazyflies were instructed to take off from slightly elevated platforms to 80 cm and maintain this altitude while travelling horizontally at constant velocity. For the tests involving the transport of fabric, one of the corners of the fabric was manually tied to each Crazyfly with fishing line before takeoff.

A. Weight limit

From BitCraze specifications, a Crazyfly can lift 42 g and its own weight is 27 g, leaving 15 g for a payload, from which we already use 0.1 g for an Optitrack marker.

In order to confirm and test these numbers, we measured the maximum lifting capability of the Crazyflies. Using a digital spring scale with a resolution of 1 g, we took 5 measurements of the maximum payload, and all measurements gave 18 g.

B. Spring-damper control versus Lennard-Jones potential

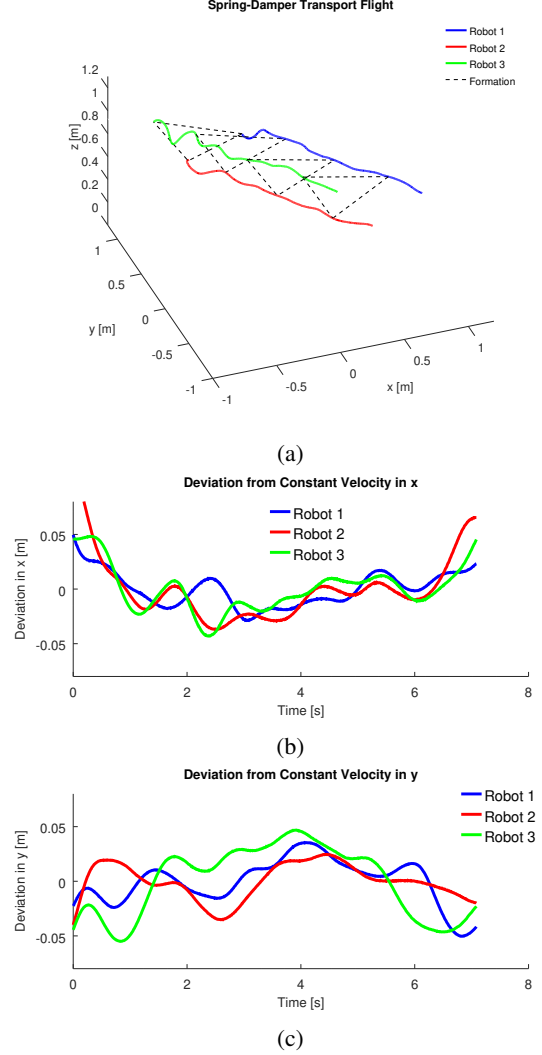


Fig. 5: **Spring-damper** flight test: (a) Overview of flight path of the three Crazyflies while cooperatively carrying fabric during one of the 5 flight tests of its kind. The vertices of the dotted triangles indicate where the Crazyflies are at 2 second intervals. (b) Deviation of each robot's x coordinate from the best approximating constant velocity trajectory for the curves in (a). Each curve is a plot of x vs t with the zeroth and first order terms subtracted, i.e. the average value and average slope of each curve is 0. (c) Exactly the same construction as (b), but taking the y vs t graph, rather than the x vs t .

Post-experiment, we analyzed the robustness and stability of the collaborative transport algorithms, and quantified the cooperation between robots. For each flight test, we took seven seconds of linear translation data, which is depicted in Fig. 5 and Fig. 6. Fig. 5-a is the flight trajectory of one of

the collaborative fabric transport tests using a spring-damper algorithm. Fig. 6-a is the flight trajectory of one of the collaborative fabric transport tests using a Lennard-Jones potential. In a decentralized system, the behaviour of one robot can have great impacts on all of its neighbors. In particular, with non-deterministic forces arising from the flexible payload, it is expected that each robot must dynamically accomodate for its neighbours when they undergo instability. Thus, in a statistical analysis we should be able to measure these adjustments.

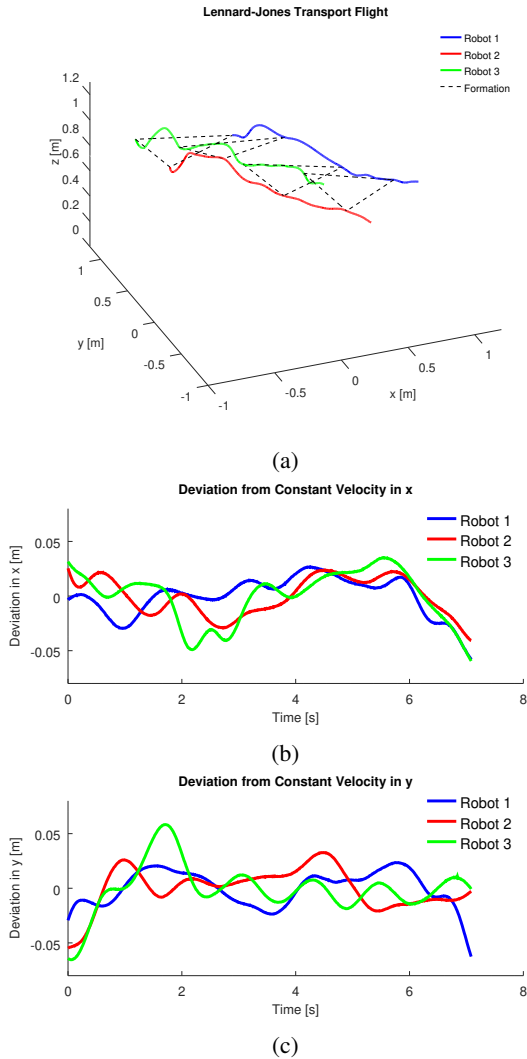


Fig. 6: **Lennard-Jones** flight test: (a), (b), (c) See Fig. 5 for description of these figures.

Parts (b) and (c) of figures 5 and 6 illustrate the instabilities of each robot in a single dimension with respect to time. Without a shared payload the movement is expected to be directed only towards the goal, so these plots shows the deviation from that constant velocity trajectory. It can be observed that there is a similarity between the curves of Fig. 5, and of Fig. 6. Synchronicity of UAV movements is ideal in this decentralized system, and it can be quantified by the average of the correlation coefficients of each pair of

TABLE I: Comparison of correlation coefficients for all flight tests.

	$\bar{\rho}_x$	$\bar{\rho}_y$	$\frac{\rho_x + \rho_y}{2}$
No Fabric	0.19	-0.17	0.01
Spring-Damper	0.74	0.44	0.59
	0.66	0.31	0.48
	0.60	-0.02	0.29
	0.31	0.03	0.17
	0.29	-0.06	0.12
Lennard-Jones	0.47	0.67	0.57
	0.43	-0.08	0.17
	0.68	0.20	0.44
	0.55	0.28	0.41
	0.26	-0.06	0.10

signals $\bar{\rho}$. This was done for each of the two dimensions x and y , and for each flight test. The results are presented in Tab. I.

For the Spring-Damper algorithm, the mean μ and the standard deviation σ of the correlation coefficients from each flight test are

$$\mu_{SD} = 0.33, \quad \sigma_{SD} = 0.20 \quad (24)$$

while for the Lennard-Jones algorithm,

$$\mu_{LJ} = 0.34, \quad \sigma_{LJ} = 0.20 \quad (25)$$

The experiments show that the spring-damper and Lennard-Jones algorithms do in fact allow the robots to maintain safe, stable distances with similar efficacy. The non-linearity of the Lennard-Jones potential induces the same amount of cooperation between robots as the dampers that are present in the spring-damper control algorithm.

VI. FUTURE WORKS

As mentioned at the beginning, this work is a first step along the way to get a safe fleet of micro-UAVs to dress a human. The decentralized approach was demonstrated to work well for such a scenario. Leveraging PyBuzz, other decentralized behaviours will be tested to succeed in creating autonomous flying dressing-aid. We showed our result to the designer and we will now try various shapes of clothing, designed specifically to be transport with micro-UAVs. Of course, the robustness of the algorithm will be enhanced, and together with infrared proximity detection, the interaction with users can begin to be studied.

ACKNOWLEDGEMENT

This project is based on an original idea of the designer and researcher Ying Gao, professor at the Fashion School of the University of Quebec in Montreal.

REFERENCES

- [1] J. Estevez, J. M. Lopez-Guede, and M. Graña, "Particle Swarm Optimization Quadrotor Control for Cooperative Aerial Transportation of Deformable Linear Objects," *Cybernetics and Systems*, vol. 47, no. 1-2, pp. 4–16, 2016.
- [2] R. Ritz, M. W. Müller, M. Hehn, and R. D'Andrea, "Cooperative quadcopter ball throwing and catching," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4972–4978, 2012.

- [3] R. Ritz and R. D'Andrea, "Carrying a flexible payload with multiple flying vehicles," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3465–3471, 2013.
- [4] Y. Dai, Y. Kim, S. Wee, D. Lee, and S. Lee, "Symmetric caging formation for convex polygonal object transportation by multiple mobile robots based on fuzzy sliding mode control," *ISA Transactions*, vol. 60, pp. 321–332, 2016.
- [5] G. Loianno and V. Kumar, "Cooperative Transportation Using Small Quadrotors Using Monocular Vision and Inertial Sensing," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 680–687, 2018.
- [6] J. A. Preiss, W. Hönig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," *IEEE International Conference on Robotics and Automation ({ICRA})*, pp. 3299–3304, 2017.
- [7] M. Furci, G. Casadei, R. Naldi, R. G. Sanfelice, and L. Marconi, "An open-source architecture for control and coordination of a swarm of micro-quadrotors," *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, pp. 139–146, 2015.
- [8] W. Hönig and N. Ayanian, "Flying multiple UAVs using ROS," *Studies in Computational Intelligence*, vol. 707, pp. 83–118, 2017.
- [9] C. Pinciroli and G. Beltrame, "Swarm-Oriented Programming of Distributed Robot Networks," *Computer*, vol. 49, no. 12, pp. 32–41, 2016.
- [10] K. Støy, "Using situated communication in distributed autonomous mobile robots," *Proceedings of the 7th Scandinavian Conference on Artificial Intelligence*, pp. 44–52, 2001.
- [11] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, "Cython: The best of both worlds," *Computing in Science and Engineering*, vol. 13, no. 2, pp. 31–39, 2011.
- [12] J. Estévez, J. M. Lopez-Guede, and M. Graña, "Quasi-stationary state transportation of a hose with quadrotors," *Robotics and Autonomous Systems*, vol. 63, no. P2, pp. 187–194, 2015.
- [13] K. Belkacem and F. Cherif, "Swarm Robots Circle Formation via a Virtual Viscoelastic Control Model," in *International Conference on Modelling, Identification and Control*, Algiers, 2016, pp. 725–730.
- [14] E. Rimon and D. Koditschek, "Exact Robot Navigation Using Artificial Potential Functions," *Robotics and Automation, IEEE*, vol. 8, no. 5, pp. 501–518, 1992.
- [15] J. H. Reif and H. Wang, "Robotics and Autonomous Systems Social potential fields: A distributed behavioral control for autonomous robots*," *Robotics and Autonomous Systems*, vol. 27, pp. 171–194, 1999.