A Flying Drum Machine

Xingbo Wang, Natasha Dalal, Tristan Laidlow, and Angela P. Schoellig

***Abstract*— This paper proposes the use of a quadrotor aerial vehicle as a musical instrument. Using the idea of interactions based on physical contact, a system is developed that enables humans to engage in artistic expression with a flying robot and produce music. A robotic user interface that uses physical interactions was created for a quadcopter. The interactive quadcopter was then programmed to drive playback of drum sounds in real-time in response to physical interaction. An intuitive mapping was developed between machine movement and art/creative composition. Challenges arose in meeting real-time latency requirements mainly due to delays in input detection. They were overcome through the development of a quick input detection method, which relies on accurate yet fast digital filtering. Successful experiments were conducted with a professional musician who used the interface to compose complex rhythm patterns. A video accompanying this paper demonstrates his performance.**

I. INTRODUCTION

New designs for flying robotic vehicles have been in constant development in recent years. The advancements in sensors and actuators have allowed us to produce flying robots that are ever smaller and safer. The once deadly machines can now become consumer-friendly products, causing the trend to shift from purely industrial or military applications to civilian and recreational settings. Examples include Spiri, a small programmable quadrotor created by the Canadian robotics company Pleiades [1], the BionicOpter dragonfly robot made by Festo [2], as well as the entire drone product line by Parrot [3].

These new developments pave the way for novel applications. We are inspired to create flying robots capable of interacting with humans in a very intuitive manner. Researchers in the recent years have been investigating novel methods of interaction. The most prominent examples are interactions through gesture [4]–[6], speech [7], face/eye tracking [7], and even thought-based control using brain-computer interfaces [8]. However, these advanced methods require complicated setups and specialized equipment, which may not be found in a consumer setting. Furthermore, all of these methods focus on one-sided operation. While that is sufficient for traditional teleoperation usages, it does not provide the level of interactivity usually required of a social, domestic robot.

We propose using physical contact as a new method of human-robot interaction with flying robots. Physical interactions have been shown to be an effective method for ground robots [9]; however, it has only rarely been used in the context of flying robots [10], [11]. While the short range requirement of physical interactions may be a disadvantage, there are many benefits in using this method, such as improved interactivity, more intuitive interface, and requiring no special equipment. This concept can lead to novel applications wherever humans and flying robots are collocated. Examples include flying robotic companions, robots in zero gravity environments, and robots in performance arts.

The authors are with the Dynamic Systems Lab (www.dynsyslab.org) at the University of Toronto Institute for Aerospace Studies (UTIAS), Canada. {xingbo.wang, natasha.dalal, tristan.laidlow}@mail.utoronto.ca, schoellig@utias.utoronto.ca

This work was supported by the Natural Sciences and Engineering Research Council of Canada under the grant RGPIN-2014-04634 and USRA ref. no. 352885 for Xingbo Wang.

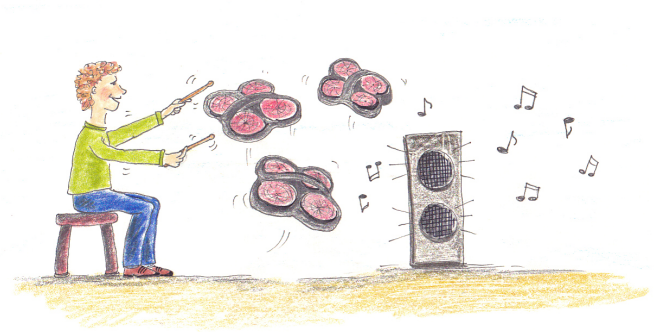

Fig. 1. Through physical interaction with a quadcopter a rhythm pattern is generated. A quadcopter serves as a novel artistic input device. (Picture: Kathrin Kleinbach)

In this project, we explore the benefits of physical interactions using a quadcopter as a platform. We programmed a quadcopter to accept physical interactions as inputs. Subsequently, we demonstrate one unique application of this system by using physical contact on the quadcopter to drive the playback and recording of percussion instrument sounds in real-time, effectively transforming the quadcopter into a flying drum machine (see Fig. 1).

II. INTERACTION WITH THE QUADROTOR

Physical contact with a human is determined by estimating the external force and torque applied to the quadcopter within a certain range of frequencies. In this project, we are mostly interested in detecting forces that result from pushes or taps performed by the user. We want to avoid detecting any other source of external forces, such as high frequency vibrations or constant offsets. Filters are designed to remove those parts of the signal unrelated to a human interaction. Once the external forces and torques are known, we attempt to categorize them based on the type of interaction the user would have done to result in such net force and torque. This information is then used to map the user's input to an associated signal response. In the case of this project, we use the type of interaction to determine what sound to play.

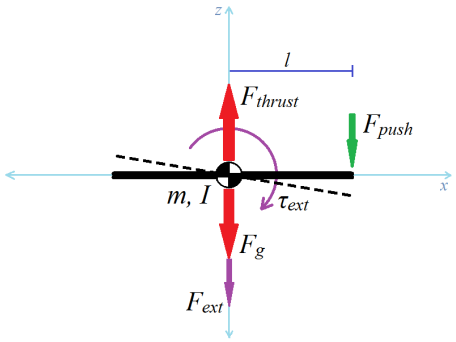

Fig. 2. Simplified diagram of a quadcopter (with mass m, arm length l, and rotational inertia I) showing the forces when an external force is applied during hovering, where the magnitude of the gravitational force ,F_g, is equal to the magnitude of the force produced by the motors, F_{thrust}. External pushes, $\mathbf{F}_{push}$, result in linear forces, $\mathbf{F}_{ext}$, and torques, $\boldsymbol{\tau}_{ext}$.

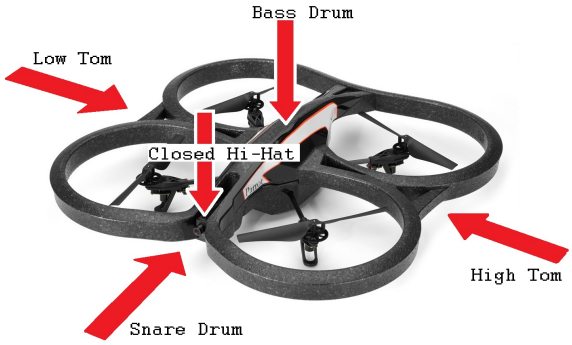

Fig. 3. Diagram of a quadcopter where five different interactions are labeled along with the corresponding drum sound response.

A. Assumptions

We assume the quadcopter is hovering with stabilizing controls. This guarantees that the robot stays in close proximity to the user. In turn, it simplifies the force and torque estimations in the subsequent sections.

B. Force and Torque Estimation

The first step is to estimate the externally applied forces and torques. Under the assumption that the quadcopter is hovering, we can say that the net force acting on the quadcopter is equal to the externally applied force. If the force is not applied directly through the center of mass, it also results in a torque $\boldsymbol{\tau}_{ext}$ about some axis through the center of mass. Referring to Fig. 2, we can write that (cf. [12])

$$\mathbf{F}_{ext} = \mathbf{F}_{push} = m\mathbf{a}$$

$$\boldsymbol{\tau}_{ext} = \frac{\mathbf{F}_{push} \times \mathbf{l}}{2} = I\boldsymbol{\alpha}$$

where m is the mass of the vehicle, I is the moment of inertia about the axis of rotation, $\mathbf{l}$ is a vector pointing to the center of mass, and $\mathbf{a}$ and $\boldsymbol{\alpha}$ are the linear and angular accelerations. The inertial parameters of the quadcopter do not need to be known for our purposes (and are not easy to measure), therefore the mass-normalized force and torque, viz. accelerations, are used. These can be calculated using data from the onboard inertia measurement units. The accelerometer provides the acceleration vector, which is directly proportional to the applied force. The gyroscope provides angular rates for roll, pitch, and yaw, from which one can numerically differentiate to find the angular acceleration. Numerical differentiation is done using the backward difference method because of real-time requirements.

C. Signal Processing

The force estimate signal calculated in the first step cannot be directly used for two reasons: the presence of noise resulting from high frequency vibrations and amplified through numerical differentiation, and the presence of a constant bias. A digital bandpass filter removes the unwanted parts of the signal. The design of the filter must also take into consideration the stringent latency requirements of real-time signal generation; in particular, when dealing with musical performances as it is the goal of this project. While there is no standardized latency requirement for digital/MIDI instruments, the digital music community agrees that the latency between triggering a note on the instrument and sound playback should not exceed 10 ms [13]; for instruments without tactile feedback, such as a theremin, latency tolerance can be as high as 30 ms [14]. Finally, we also require the filter to preserve the wave shape of the signal, so that distortion is kept to a minimum. The goal is then to design the filter to achieve the following criteria:

- eliminate constant biases and high frequency noise,
- preserve the shape of the signal in the passband, lest the interaction be misinterpreted,
- operate in real-time with minimal time delay.

This proves to be challenging, mainly due to the low sampling frequency of 200Hz. At that rate high-order recursive filters produce too much time delay to be usable: we saw delays as high as 50 ms with as low as 8th order filters. After several trials and fine-tuning, the following two filters are used:

- 4th order Bessel lowpass filter centered at 11 Hz,
- 2nd order Butterworth highpass filter centered at 0.7 Hz.

Bessel filters are designed to have maximally flat group delay in the passband, which means the signal distortion is low [15]. They also perform decently in terms of time delay compared to other filters. As a side note, elliptic filters perform better in terms of minimizing time delay [16], but their highly variable group delay results in heavy signal distortion around the cutoff frequency. The filter's performance will be discussed in Sec. V.

D. Characterization of Interaction

The next step is to detect and characterize when a user has interacted with the quadcopter. One easy way of detection is to set a threshold on the force estimate signal, say F_{thres}, and perform peak detection whenever the magnitude of the signal rises above the threshold, $\|\mathbf{F}_{ext}(t)\| > F_{thres}$

(similarly for $\boldsymbol{\tau}_{ext}(t)$ and τ_{thres}). The amplitude of the peak $\|\mathbf{F}_{ext}(t_{peak})\|$ is taken to represent the magnitude of the push, and a response is triggered when a peak is found. In our case the magnitude is mapped to the velocity (loudness) of the note to be played. This method is feasible, since the signal is already smoothed by the lowpass filter. However, early testing revealed a crucial shortcoming of this method: the time delay is too long. The peak of the external force happens shortly before the end of physical contact. Since the peak is delayed through filtering, the response is triggered moments after the user's hand has left the quadcopter. This is extremely undesirable, as the response of any musical instrument should be near immediate. Instead, we have developed a "quick detection" method, whereby the response is triggered as soon as the signal rises above the threshold, and we use the slope of the signal $||\dot{\mathbf{F}}_{ext}(t_{peak})||$ at that instant to estimate the magnitude. Using this method, the response can be triggered before the peak of the force happens. The performance of this method is discussed in Sec. V.

Upon detection, the force and torque vectors are captured and stored. By examining the various interactive zones shown in Fig. 3, and with the help of experimental data, one can determine the expected force vectors for each interaction. We categorize the detected force vector by figuring out which of the acceptable interactions it most closely resembles. We do this by comparing the relative magnitudes of each force and torque component against what they should be for a particular interaction. For example, pushing straight down the center results in a large force vector pointing towards the ground and very little torque while a downward push on the front edge produces a similar force vector but also a large torque with respect to the y-axis. Then, each type of interaction is mapped to a certain MIDI (Musical Instrument Digital Interface, see Sec. III) note number, corresponding to a specific note or sound.

Whenever an external force is applied, the quadcopter is made to move, but unlike a free floating mass the control systems will work to bring it back to its original state. During this "correction" step the quadcopter experiences additional inertial forces, which are unaccounted for in the force estimation method. If left alone these inertial forces will be registered as inputs and will trigger unintended responses. To negate these effects we have implemented a simple finite state machine for the quadcopter with two states, a HOVER state and a HIT state (see Fig. 4). In the HOVER state the quadcopter is able to accept user input in the form of physical interactions. Immediately after an input is registered, the state machine waits for the quadcopter to recover from the hit before it accepts another input. Once the program enters the HIT state, it continuously checks the external force signals and returns to HOVER state only when the magnitudes of the signals have fallen below and remain below the threshold F_{thres} for a period of time T_{steady}. We set $T_{steady} = 0.2\,\text{s}$ for a balance of accuracy and usability.

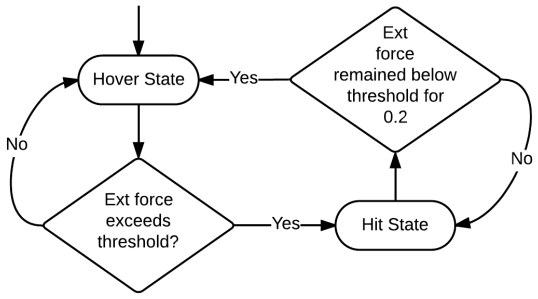

Fig. 4. State transition diagram for the finite state machine used to detect interaction between a human and the quadcopter by detecting external forces.

III. SOUND GENERATION

The interactive quadcopter program generates musical messages in the form of MIDI notes. Each MIDI note message contains the following information:

- MIDI note number: indicates the pitch of the note, or in the case of a drum kit it indicates the instrument to be played, e.g. snare, crash cymbal, vibraslap, etc.,
- duration of the note: a floating point number indicating how long the note should be held,
- velocity of the note: terminology used by the digital music community to mean the loudness of the note.

The MIDI note number is determined by the type of interaction, as shown in Fig. 3. The velocity of the note is calculated based on the magnitude of the interaction. The duration of the note is set to a constant value of 0.25 s, corresponding to a sixteenth note, for the sake of simplicity.

Our method only enables one note to be played at a time. In order to add musical complexity, we employ a recording technique called loop overdub. In this technique, a few bars of music are continually played on a loop; new notes are recorded into the position in the bar when they are played, on top of any existing notes. This allows users to create complex tracks by building up one note at a time. Furthermore, the recorded notes are programmatically quantized into a sixteenth-note grid. Recording quantization is a common tool in digital music, which can correct a user's imperfect timing in real-time. When a note is cued the sound is played by an audio workstation software (see Sec. IV) using a library of sampled drum sounds.

IV. EXPERIMENTAL SETUP

A. Quadrotor Vehicle

The quadrotor vehicle used in this project is the Parrot AR.Drone 2.0 quadcopter [17]. This is a low-cost vehicle designed to be used by the general public and includes a wide assortment of onboard sensors. It comes with features that enhance user safety, such as soft plastic propellers and an outer protective shell. Also, it has an onboard stabilization controller allowing it to hover based on onboard sensors, making it ideal for this project. We use the Robot Operating

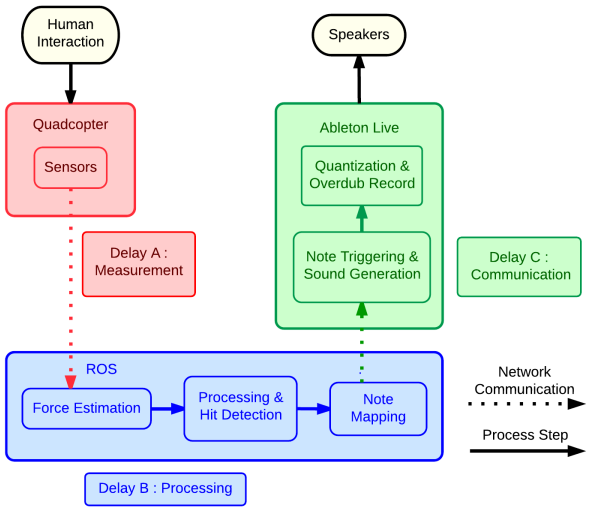

Fig. 5. The high-level system architecture showing how the different components work together. The quadcopter sends its sensor data to a computer running ROS. The data is processed in real-time. Whenever an interaction is identified, it is categorized, mapped to a note/instrument, and sent to another computer running Ableton Live. The note is then played, quantized, and recorded.

System (ROS) framework to interface with the quadcopter. ROS Hydro is installed on a laptop running Ubuntu 12.04. We use the AR.Drone ROS driver package available publicly online [18].

Safety is an obvious concern in this setup. Physical interactions with a quadrotor means that the user must put his/her hand in close proximity with fast spinning rotor blades. Work can be done to mitigate these risks, such as adding a more protective shell or using a drum stick instead of hands for interaction. More importantly, however, the concepts presented can be applied to any flying vehicle, including a much safer lighter-than-air balloon robot without spinning blades.

B. Sound Software

For sound generation and music control we use the software Ableton Live 9 Suite. Ableton Live is an industry standard digital audio workstation (DAW) with many useful features for music composition and performance (such as loop overdub and quantization). Ableton's software was chosen for its popularity, multitude of functionalities, and extendibility with Max for Live. Max, or MaxMSP, is a visual programming language mainly for developing multimedia apps and programs. Ableton Live is only supported on Microsoft Windows and Apple MacOS, and ROS is only supported on Ubuntu Linux. The digital audio industry has almost no presence on the Linux OS. These programs therefore are run on separate computers and communicate over a network.

C. Communication

The two computers are setup to communicate information through UDP networking. MaxMSP has built-in support for Open Sound Control (OSC) [19], a protocol used by many modern electronic musical instruments and synthesizers. We make use of this feature and opt to serialize messages sent by ROS in accordance with the OSC protocol. This is made easy using a C++ library called oscpack [20].

The quadcopter's IMU sensor information is sent from the quadcopter to the ROS computer through WiFi and is received by the ROS driver. The ROS driver relays this information to the filter program, and the output is sent to the finite-state machine. Once processed, a message containing note playback information is passed to an OSC relay program. It packages the messages for OSC and sends them to the Ableton computer over the network. On the Ableton side, the message is received by a MaxMSP program we wrote, which then uses Max for Live bindings to control Ableton Live for note playback and recording.

V. RESULTS

This section validates the performance of the aforementioned setup. Referring to Fig. 5, we want to know whether or not the latency of the system is acceptable in terms of representing a musical instrument. The system latency can be split into three major sources of delays: the measurement delay D_A, the processing-detection delay D_B, and the communication-playback delay D_C. Before starting the analysis, it is worth mentioning that there is quite a bit of leeway in identifying when exactly the interaction is "triggered". Unlike a piano key or a guitar string in which the action of triggering a note is distinct in timing, the act of pushing on a flying vehicle takes place over a period of time. The exact time when an interaction is said to have taken place is not always clear. One could say it is the moment the hand touches the quadcopter, or when it breaks contact, or half way through the push, etc. Therefore, measuring the delay between the interaction and the response is not easily done, since the timing of the interaction is not entirely well-defined. We attempt to quantify the time delay between the start of the interaction to the triggering of the sound. However, keep in mind that ultimately the best judge for whether or not the latency is tolerable is the musician, not the technician.

A. Communication Delay

The communication delay between ROS and Ableton Live can be measured easily. We send a ping message from ROS to Ableton for which Ableton goes about its usual note-processing/adding routine. Upon completion the routine is repeated, and when it finishes for the second time a pong message is sent back to ROS. Half the time delay between the ping and pong messages is taken to be the communication delay between the two computers. This test is run on both wireless and wired connections, and the results are shown in Fig. 6.

Wired connections perform faster consistently and are more reliable, whereas the wireless connection experiences long delays at times, possibly due to lost UDP packets; because of this, wired connections are always preferred. These results show that the average communication delay

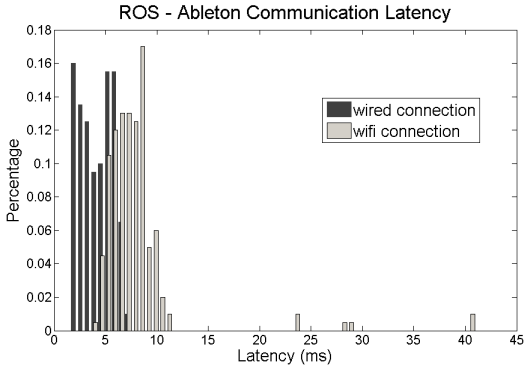

Fig. 6. Histogram of ROS-Ableton communication latency measurements, with $n = 200$ samples. For the wired connection, the average is $\mu = 4.2$ ms, the standard deviation $\sigma = 3ms$. For wireless connections, $\mu = 16.0ms$ and $\sigma = 9.0ms$.

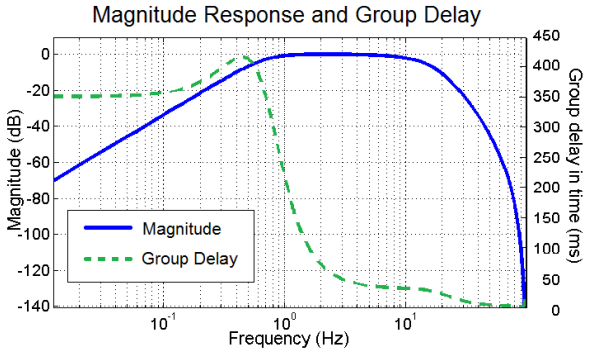

Fig. 7. Magnitude and group delay response of the filter cascade used for processing the accelerometer signals.

D_C between ROS and Ableton, including music processing time, is 4.2 ms on a wired connection.

B. Measurement Delay

The measurement delay D_A is not easy to calculate. For this project we assume the sensor's delay is much smaller than the network latency. The latency value depends on the quality of the hardware, i.e. the router and the computer's wireless adapter. It also depends on the distance between the devices; however the hovering requirement ensures that the quadcopter is never far away from the router. In our setup, we measured the ping from the ROS computer to the quadcopter to be about 2 ms, and therefore the one-way measurement delay is estimated to be about 1 ms.

C. Estimation and Detection Delay

Fig. 7 shows the group delay of the filters used for signal processing. We see that at around 5 Hz the group delay is about 30 ms, and decreases as we go up in frequency. Compared to this value, the remaining processing times are negligible, and therefore at first glance the value of D_B is around the same, 30 ms.

Fig. 8 shows the effect of this filter on the torque estimate signal. High frequency noise in the signal is greatly reduced, which is crucial for input detection. The overall shape of the signal is preserved, owing to the properties of the Bessel

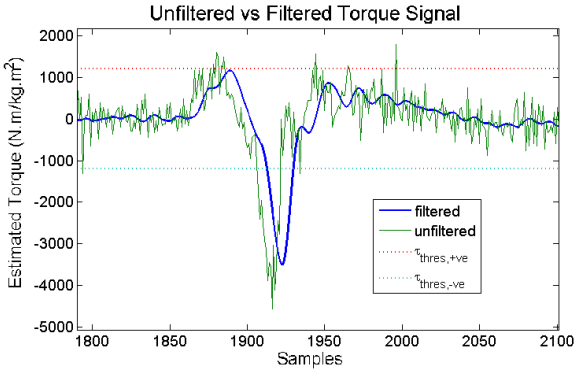

Fig. 8. Example of filtered and unfiltered signals with the chosen threshold value marked. A hit is detected whenever the magnitude of the filtered signal rises above the threshold.

filter. However, even with this carefully chosen filter it is clear that the output signal is delayed by a non-trivial amount. In this example, the peak is detected in the torque estimate roughly 30 ms after it appeared in the angular acceleration measurements. Peak detection is not the optimal approach. Instead, the quick detection method mentioned in the previous section works well here. Using the quick detection method and a threshold of 1200 N m/ (kg m^2), the interaction is detected 20 ms *before* the peak happens. The value of D_B is effectively reduced to 0, since the detection does not rely on waiting to find the peak. But again it is hard to give a numerical result for the delay, since it is unclear whether or not the peak of the external force can be considered the point of trigger.

D. Overall System Performance and User Experience

Putting the whole system together, we want to see how it performs as a real-time sound generation device. Fig. 9 tells the whole story. It shows the chain of signals generated when a user interacts with the quadcopter. Looking at the accelerometer readings we see that the interaction begins shortly after $t = 78.42$ s and ends at around $t = 78.55$ s. The peak is approximately at $t = 78.5$ s. The force estimate signal is delayed by about 30 ms. Using the quick detection method the note is triggered at around $t = 78.46$ s and played 5-7 ms later. This means that the note is triggered 30-40 ms after the start of the push, as expected, but about 20 ms before the acceleration peaks, and well before the end of the push. Although we only present this data for one type of interaction, results like these are typical of all other types.

This system performs well for users who prefer the note triggering in the middle of the push or later. The action is like that of a piano key, where the key needs to be depressed to a certain extent before the note is triggered. Similarly the quadcopter needs to be pushed by a certain amount before the note is played. Users who prefer the note be triggered right at the beginning of contact may not be satisfied with this performance. Those users might prefer snappy striking and hitting actions, like on percussion instruments, rather than full contact pushes.

Haig Beylerian, a professional music student currently working at WaveDNA Ltd., helped us do a user test of the

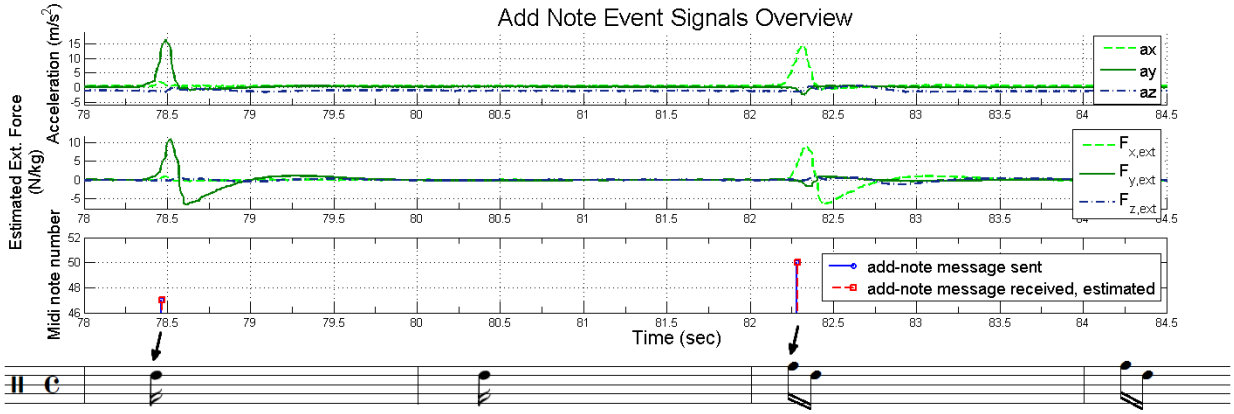

Fig. 9. Acceleration signals over time along with the force estimate signals and the generated notes. The notes are quantized to the nearest sixteenth position and recorded on a loop. The resulting score is shown in standard percussion notation with rests omitted for clarity. Arrows indicate added notes.

Fig. 10. Photo of Haig making music using the flying drum machine, taken from video.

system. After some initial learning curve, Haig was quickly able to adapt to this new instrument and create intricate drum beats. When asked about the latency he said it was unnoticeable and that the response was "excellent". Using this system, Haig created a complex drum track using loop overdub. A recording of his performance is available at http://tiny.cc/FlyingDrumMachine, and a photo taken from the recording is shown in Fig. 10. Despite the system having an inherent non-trivial latency, the quick detection method was able to compensate for the delay and reduce the apparent latency to a level that is acceptable to at least one user. More user tests should be performed to qualitatively verify this result.

VI. CONCLUSIONS

We have set out to develop a human-quadcopter interface that communicates through physical contact, and demonstrate its practicality by using it to create musical sounds as a drum machine. We used on-board sensors to estimate the externally applied forces and torques acting on a hovering quadcopter. A method was developed to quickly detect when an interaction has occurred. The interaction is then categorized as a type of input and mapped to individual responses. In this application, we map the inputs to certain notes in a drum kit. The information is communicated to a digital audio workstation software, which is programmed to play and record the appropriate notes. Analysis of the system shows that there is a significant amount of latency compared to other MIDI instruments. The delay between the interaction and the response is around 30-40 ms. However, due to the nature of the note triggering action, this may not be noticeable to most users. Preliminary user testing suggests that the responsiveness is good, but more thorough user surveys are needed to confirm this hypothesis. Additionally, the quality of the interaction itself must be evaluated. Much more can be done to optimize that aspect. We consider the project successful, showing that physical interaction is a unique and effective method for humans to truly and intuitively interact with flying robots.

ACKNOWLEDGMENT

We thank the team at WaveDNA Ltd., viz. Adil Sardar, Glen Kappel, David Beckford, and Saro Migirdicyan. Special thanks to Haig Beylerian for the user test and video demonstration.

REFERENCES

[1] Pleiades Robotics Inc., "Pleiades — say hello to spiri." online, September 2014, http://pleiades.ca/.

[2] Festo AG, "Bionicopter — festo corporate," online, September 2014, http://www.festo.com/cms/en_corp/13165.htm.

[3] Parrot SA, "Parrot - products - civil drones, zik, automotive, asteroid, flower power, jumping sumo, minidrone," online, September 2014, http://www.parrot.com/usa/products/.

[4] T. Naseer, J. Sturm, and D. Cremers, "Followme: Person following and gesture recognition with a quadrocopter," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Nov 2013, pp. 624–630.

[5] K. Boudjit, C. Larbes, and M. Alouache, "Control of flight operation of a quad rotor ar. drone using depth map from microsoft kinect sensor," *International Journal of Engineering and Innovative Technology*, vol. 3, pp. 15–19, 2013.

[6] W. S. Ng and E. Sharlin, "Collocated interaction with flying robots," in *RO-MAN, 2011 IEEE*, July 2011, pp. 143–149.

[7] S. Pourmehr, V. M. Monajjemi, R. T. Vaughan, and G. Mori, ""you two! take off!": Creating, modifying and commanding groups of robots using face engagement and indirect speech in voice commands," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 137–142.

[8] K. LaFleur, K. Cassady, A. Doud, K. Shades, E. Rogin, and B. He, "Quadcopter control in three-dimensional space using a noninvasive motor imagery-based braincomputer interface," *Journal of Neural Engineering*, vol. 19, p. 046003, 2013.

[9] C. Guo and E. Sharlin, "Exploring the use of tangible user interfaces for human-robot interaction: A comparative study," in *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 121–130. [Online]. Available: http://doi.acm.org/10.1145/1357054.1357076

[10] F. Augugliaro and R. D'Andrea, "Admittance control for physical human-quadrocopter interaction," in *Proc. of the European Control Conference (ECC)*, 2013, pp. 1805–1810.

[11] Perspective Robotics, "Fotokite - aerial photography for everyone," online, February 2015, http://www.fotokite.com.

[12] A. J. D. Bois, *The mechanics of engineering*. Wiley, 1902, vol. 1, p. 186.

[13] T. Mki-Patola, "Musical effects of latency," in *Proc. of Suomen musiikintutkijoiden 9. valtakunnallinen symposium*, 2005, pp. 82–85.

[14] T. Mki-Patola and P. Hmlinen, "Latency tolerance for gesture controlled continuous sound instrument without tactile feedback," in *Proc. of the International Computer Music Conference*, 2004, pp. 11–16.

[15] J.-P. Thiran, "Recursive digital filters with maximally flat group delay," *IEEE Transaction on Circuit Theory*, vol. ct18, pp. 659–664, Nov. 1971.

[16] S. A. Skogstad, S. Holm, and M. Hovin, "Digital iir filters with minimal group delay for real-time applications," in *Proc. of the Int. Conf. on Engineering and Technology (ICET)*, Oct 2012, pp. 1–6.

[17] Parrot SA, "Ar.drone 2.0. parrot new wi-fi quadricopter," online, June 2014, http://ardrone2.parrot.com.

[18] M. Monajjemi and et. al., "Autonomylab/ardrone_autonomy," online, March 2014, https://github.com/AutonomyLab/ardrone_autonomy.

[19] A. Schmeder, "Everything you ever wanted to know about open sound control," Center for New Music and Audio Technologies, University of California, Berkeley, Tech. Rep., March 2008.

[20] R. Bencina, "oscpack, a simple c++ osc packet manipulation library," online, April 2013, http://www.rossbencina.com/code/oscpack.