Department of Mechanical Engineering, Imperial College London

Progress Report

Human Machine Interface - Architecture for Control of Drone Swarms

Liu, Yuxuan

Abstract

The goal of this project is to implement a complex gesture recognition system with inputs from the IMUs and MMG sensor signals. This project also includes the construction of a simulation of a swarm of quadcopters, with which the capabilities of the gesture recognition system can be added. In addition, to further improve the robustness, aesthetic appeal, and compatibility the HRI system, the interface casing will be redesigned and manufactured. Finally, a real quadcopter or a swarm will be controlled using the developed software and hardware.

We are currently halfway through this project and have made progress in all tasks. After many data-collection sessions a successful neural network has been created, with very impressive confusion matrix scores for gesture recognition. A simulation which allows a user to control a virtual swarm of drones has been created, but it is yet to be linked to the neural network. Detailed SolidWorks designs have been proposed for a new wristband and electronics box, these are almost ready to be manufactured.

We did not meet all of the targets set by the Gantt Chart, the simulation and neural network had a target to be completed by the end of 2016, yet have only recently in February has this been done. In some areas, we are ahead of schedule, work on the camera recording software has already commenced, 2 weeks ahead of the planned time frame. Through pushing ourselves in January we have caught up on any missed targets and are currently on track to finishing the project in perfect time.

Table of Contents

Introduction

With robotic assistance becoming increasingly important in virtually every field, it is now more crucial than ever to have human-robotic interfaces (HRI) in order to reliably link a robotic system to a designated human operator. HRIs enable teleoperation of robots with enhanced precision and with more comfort. This makes them ideal for uses in a wide range of applications in fields including surgery, human augmentation and handling hazardous substances.

Since November we have been working on our project of creating a HRI which controls a drone via a wristband. The main tasks in this project are to develop a neural network with movement & gesture recognition, redesign the existing hardware, develop a drone swarm simulation and to ultimately control an actual quadcopter. If time permits we will advance this further and control multiple quadcopters with camera control. Despite unexpected challenges and complications, we have made significant progress on this project.

The gesture recognition part of the codes and data in this project are managed with Git on GitHub. Our GitHub repository is at https://github.com/articuno144/DMT2017.

This report will detail our progress so far and describe any challenges we have faced. We will compare what we have achieved to our plan and determine if where we need to focus on in order to finish all parts of the project in the designated time. This section also includes some basic backgrounds of the neural network tools we used in this project.

Supervised Learning with Neural Networks

The neural network is a universal approximator for functions that have many inputs and outputs, and is an active research field in machine learning. Supervised learning is the most common type of machine learning problems, where the desired output is fed along with the inputs into the machine learning algorithms (neural network in this case), so that the underlying function that maps the outputs to the inputs can be learned. Training of a neural network includes three steps. Firstly, the input is fed forward in the neural network, and an output as a function of the input and the current network parameters is obtained. Secondly, the network prediction, or the output, is compared with the desired output and evaluated with a pre-defined cost function. Lastly, the neural network parameters are trained with optimisation algorithms with the aim of minimising the cost function. Only the first step, or forward feeding, is required when using a trained neural network. In this project, the inputs are the MMG and IMU signals, and the outputs are the gestures that we would like the neural network to recognise.

Convolutional Layers

Convolutional neural networks are widely used in modern computer vision tasks due to its desired property of being able to extract local features effectively. It is also used in our project as some of the fluctuations in the MMG signals can be viewed as local patterned and processed as a whole, as opposed to being broke down to small data points, which may lead to reduction in accuracy as shown in (Ma, 2016). In convolutional layers, the neurons are arranged in 3 dimensions: width, height, depth (Karpathy, 2016). These neurons are arranged as sets of small groups, and each group of neurons act as a learnable filter. For instance, in our project, a filter of neurons may refer to a group of neurons that only looks at a small window of 3 time steps for the 3 MMG signal streams (receptive field of 3*3*1). Each of the convolutional layers we use have 32 or 64 of such filters, each specialised at recognising a distinct feature in the MMG signal. An intuition of this is that in the first layer, some of the filters are highly active when observing consecutive rising of the signal at some gradient intervals, and some other filters are only active when seeing a valley in the MMG signal, as shown in Figure 1. Each of the filter sweeps across the signal matrix (300*3*1 for MMG), producing a 2-dimensional (296*1*32) feature map. The second convolutional layer further abstracts the feature map. Max-pooling layers are inserted after the convolutional layers to reduce the effect of shifting on the neural network output, also reducing the size of the feature map by the factor of 2.

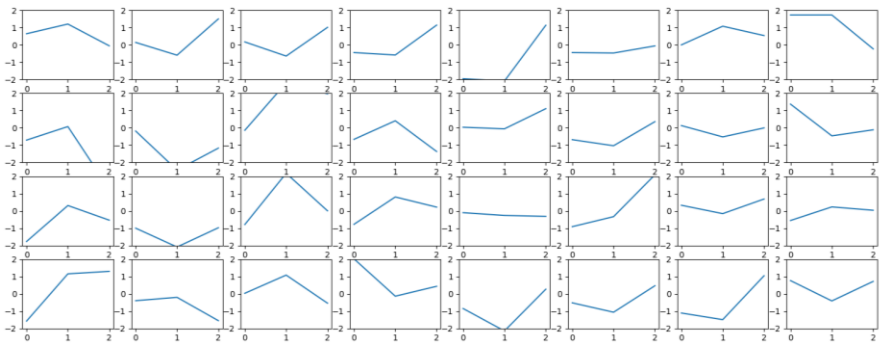


Figure 1 Sample weights in the first convolutional layer for the MMG signals

Fully Connected Layers

After the convolutional layers and the max-pooling layers are applied to the input signals, ideally the local features are extracted into the feature map, and the activations in these features can be fed into the fully connected layer to classify the final output. Dropout is applied in the fully connected layers to prevent overfitting (Srivastava N, 2014).

Rectified Linear Units (ReLU)

The ReLU is a popular non-linear activation function that thresholds the inputs at zero. It is used opposed to sigmoid and tanh because it is computationally much cheaper, and is found to greatly accelerate the convergence of stochastic gradient descent process. (Krizhevsky A, 2012)

Adam Optimiser

Adam optimiser is an adaptation of the gradient descent method by combining the advantage of momentum updates and the RMSProp method. It is shown that this method is robust and well-suited to many non-convex optimization problems (Kingma D, 2014). Furthermore, this method is suitable for training neural network with non-stationary objectives, which is crucial for our calibration process. In this project, we used Adam optimiser to train the neural network.

Due to the large amount of data in the training set, full batch training is not feasible. Hence, we use a mini-batch stochastic training instead.

Main Achievements

Gesture Recognition

In this project, a convolutional neural network is trained to recognise various gesture inputs from the user. The network is first trained with a large set of gesture data collected from 27 participants, and calibrated for each user with a small number of their gesture trials before use.

Data Collection

When using the armband, the accelerometer, gyroscope, magnetometer and MMGs signals are sent to the NU interface 100 times per second. The NU interface keeps the latest 300 sets of data, so that effectively a running window of data from 300 time steps is maintained. In gesture recognition, we use only the signals from the three accelerometers and three MMG sensors as these signals are more indicative. The running window of data is then formatted into matrices of 300-by-6 and standardised before feeding into the neural network. A sample of standardised MMG signal is shown in Figure 2. During the data collection process, every participant is asked to perform each gesture 15 times. Their MMG and IMU signals are saved as .csv files, and processed in *Matlab*. After that, we handpicked the window of 300 time steps based on where the peaks are located in the MMG signal graph shown in Figure 2, and each of the gesture performed produces 8 consecutive training pairs (input-output). As for noise, the entire duration of the collected signal should be classified as noise, and every snapshot of the running window is labelled as noise data and fed to the neural network. We approximate that when using the gesture recognition algorithm to control the quadcopters, approximately 95% of all data would be noise, and we used this percentage to distribute the training data for isometric training.

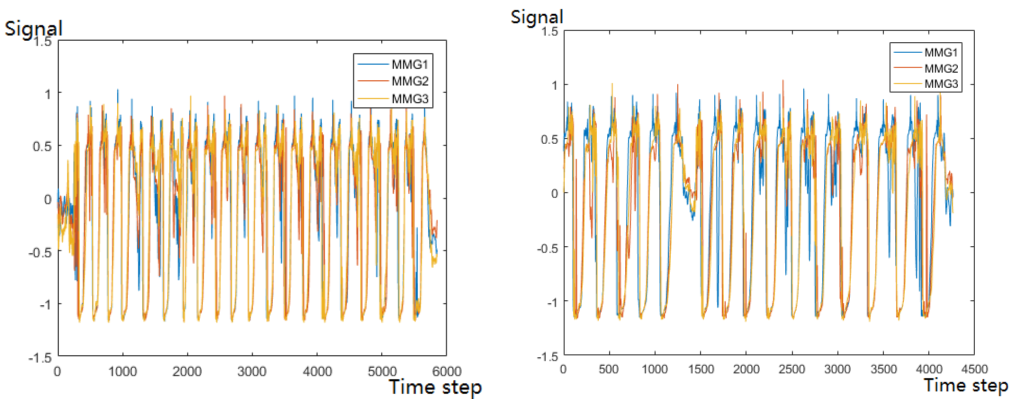


Figure 2 Samples of MMG signal of muscle tensing from two different participants, each of the peaks corresponds to tensing the muscle once

Convolutional Neural Network

Gesture recognition in this project is carried out through a deep convolutional neural network (CNN). The neural network receives the 300-by-6 data matrix and returns a 9 dimensional one-hot vector that indicates the recognised gesture, with dimension 1 to 8 corresponding to the gesture and the last dimension corresponding the noise (no gesture). For instance, the neural network returns [0,0,1,0,0,0,0,0,0]' if it classifies the data matrix as gesture 3 (thumb up), and returns [0,0,0,0,0,0,0,0,1]' if it classifies the data as noise.

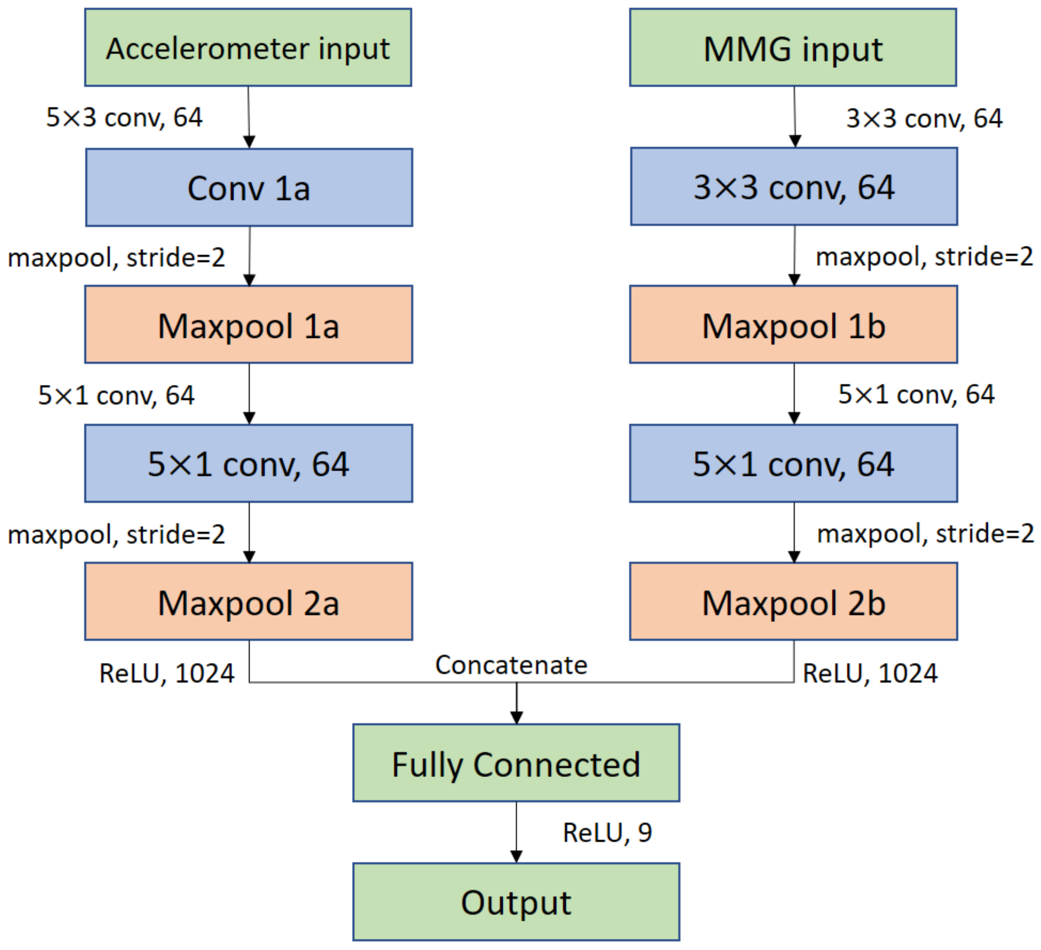


Figure 3 Convolutional neural network structure

A schematic of the CNN structure is shown in Figure 3. A convolutional neural network is used as it can see a region of data as a whole, and decide whether the region can be recognised as a known feature. For instance, in Figure 4, the circled region contains signals that are highly fluctuating. For simple neural network structures like multilayer perceptron, such a region may be rather confusing, as the value in each specific time step may be drastically different between trials. However, CNNs can recognise the entire region as a whole, extracting local features much more effectively. The maxpool layer reduces the effect of signal shifting on the network prediction and reduces the subsequent data size. ReLU is a computationally efficient non-linearity unit, and is used instead of sigmoid or tanh functions to increase training and computation speed.

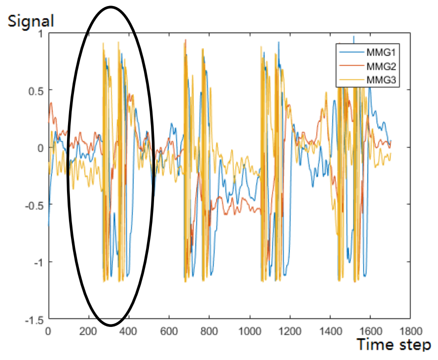


Figure 4 Four trials of tapping the forearm twice

The accelerometer and MMG inputs are fed separately into two sub networks, as they may have different optimal feature lengths and hence convolution size. By trial-and-error, the convolution layer setup that produces the highest accuracy is indicated in Figure 4.

Network Training and User Calibration

Pretraining

The CNN is first trained with a large dataset with gesture signals from 27 participants. The purpose of this step is to train the convolution layers as feature extractors, so that they can recognise the local patterns in the gesture signals.

Calibration

Figure 2 shows the MMG signals of the same gesture collected from different participants. The signal shapes are rather different. Hence it is necessary for each user to calibrate the gesture recognition network before using it. During the calibration process, the users are asked to perform each of the gesture 8 times when prompted, and these gesture signals will be used train the neural network with a much smaller learning rate than the pretraining process. At first, freezing of the convolution layers were tried, as ideally these layers only act as feature extractors and it is the last two layers that determines the network outputs. However, the results were slightly worse than training weights in all layers, so the latter method is used. Small learning rate is used for the calibration process to prevent the network from being overfitted on the small user-specific dataset.

Table 1 Confusion matrix, gestures: 1. Tensing muscle 2. Snapping 3. Thumb up 4. Thumb down 5. Double tapping forearm 6. Palm up 7. Palm down 8. Tensing muscle twice (like double click) 9. Noise

	1	2	3	4	5	6	7	8	9	Sum
1	0.854	0.047	0.000	0.099	0.000	0.000	0.000	0.000	0.000	1.000
2	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000
3	0.016	0.000	0.406	0.563	0.000	0.000	0.000	0.000	0.016	1.000
4	0.099	0.000	0.000	0.901	0.000	0.000	0.000	0.000	0.000	1.000
5	0.000	0.000	0.000	0.000	0.964	0.000	0.000	0.000	0.036	1.000
6	0.000	0.000	0.000	0.000	0.000	0.495	0.000	0.000	0.505	1.000
7	0.068	0.010	0.000	0.120	0.000	0.005	0.729	0.000	0.068	1.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	1.000
9	0.000	0.001	0.004	0.004	0.000	0.000	0.001	0.001	0.988	1.000

The confusion matrix of the eight gestures and noise in the neural network at this stage is shown in Table 1. As can be seen from the table, the neural network does a good job recognising snapping, tapping arm, and noise, but the accuracy for recognising thumb up and palm up is low. These two gestures will not be used in the simulation or the drone control.

A testing procedure will be created and added after the calibration to make sure that the neural network indeed recognises the gestures performed by the users.

Visualising CNN

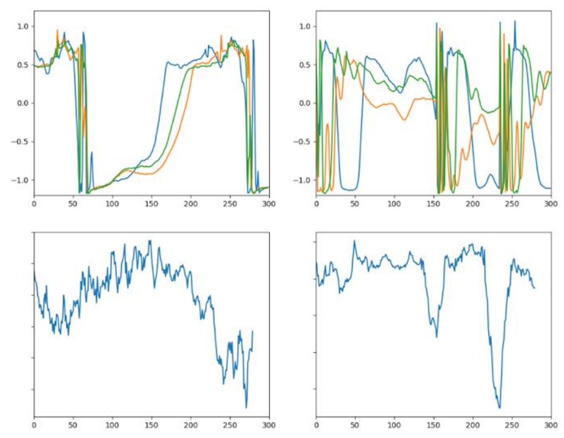


Figure 5 Three input signals (top) (snapping, double tap, and double tensing) and their respective activation after a part of the signal is covered. For instance, in the middle image, the probability of the network recognising the double tapping gesture dropped significantly after covering up either of the tapping signal. Conversely, occluding other regions have much smaller impacts.

Verification and understanding of the CNN is done through a neural network visualisation technique and the drone control simulation.

An approach that investigates which part of the signal that the neural network classification prediction comes from is by occluding parts of the image (Matthew D Zeiler, 2013) (Karpathy, 2016). In this approach, a patch of the signal is set to zero, and the patch is iterated across all time steps in the signal, so effectively a covered-up signal matrix is fed into the neural network. The neural network activation of all the covered signals are compared, so that it can be told which part of the signal is most responsible for the neural network classification. Figure 5 shows the activations obtained from this approach, and it can be clearly seen that the regions that are most responsible for the network classification corresponds well to our intuition.

Compared to the Gantt Chart drafted in the quality plan, the neural network was done behind the schedule. It was planned to be finished by the end of last term, but it was only finished in late January. This is because of the delay in the data collection, and that we modified the number of MMGs for each armband so that the data need to be collected again.

Processing Speed

The forward time for the neural network on Xeon E3 1230 v5 CPU is tested to be approximately 0.0014s averaged over 500 trials, much faster than the data reception rate of once every 0.01 seconds, hence real time processing is feasible.

Data Flow

The neural network is built with *TensorFlow* on *Python*. However, the NU interface from which the data are collected is written in *C#*. Named pipes are used to pass data between the two programs. A prototype of the pipes which can pass byte arrays between the two programs is uploaded onto the project *GitHub* repository under /pipe_test/.

Location Feedback

In our project, if time permits, we are also aiming to control an actual quadcopter swarm. The control of the quadcopter swarm requires knowledge of the spatial location and the roll, pitch, and yaw angles of the quadcopter. We plan to acquire the spatial location with webcams. A python program that reads from the webcam has been created and uploaded to the GitHub repository under /camera_test/. An obstacle that we are currently facing is that the webcam (Logitech HD Webcam C310) feedback has a ~0.4s delay from real time. To tackle this issue, we proposed two solution. Firstly, a webcam with smaller delay can be purchased. According to (MHadmin, 2016), Sony PS3 Eye has a much lower delay of about ~0.075s. Secondly, the drones that we plan to purchase has on-board IMUs that provide the roll, pitch and yaw angles from which we can roughly predict the actual location of the quadcopters from its delayed location. These solutions hopefully will reduce the delay problem and will be tested by March. However, from the design review, we learned that control of the actual quadcopters would be a challenging task. It will be the focus of our project if the all other parts of the project are finished as planned by March.

The prototype of the named pipe and the camera stream was done ahead of the schedule we set in the Gantt Chart, and took much shorter time than we expected.

Hardware

Hardware design aims to use a minimal amount of material to assemble all components together wisely in the form of an armband. We believe the quality of the hardware is as critical as the software performance. The physical armband is not only what gives consumer first impression when he/she sees the product for his/her first time, but it is also what users will touch and wear every time they use it. Thoughtful hardware design is fundamental for great product experience, so it is critical to us.

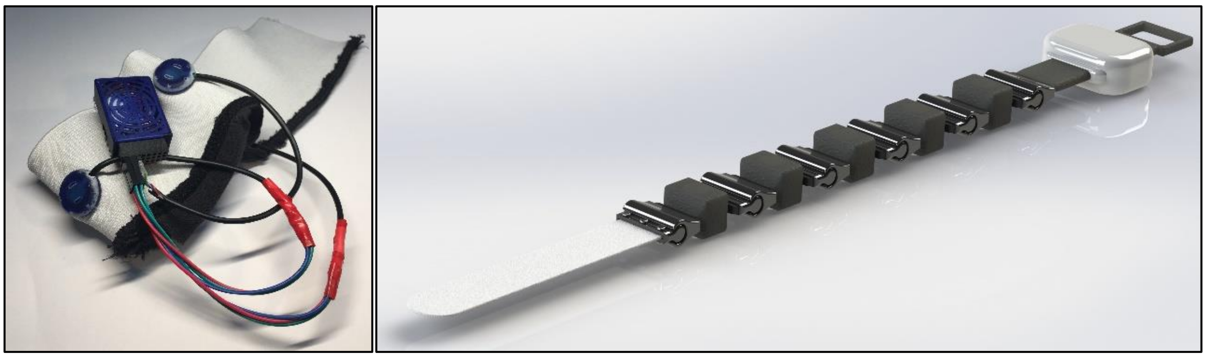


Figure 6 Armband Design

Hardware development runs alongside the fine tuning of gesture recognition algorithm. The current testing armband setup is mainly using elastic ribbon (Figure 6 left), where all MMG sensors are inset in the ribbon's pre-cut holes. The wiring connection between MMGs and the IMU upgrades from pin headers soldered on wires to Dupont wire (i.e. wire that has built-in pin header). The current testing unit is fully functional, and the fine-tuned algorithm based on existing unit should be working similarly in the new design that we are currently designing.

Key philosophies that we carry through entire hardware design development stages are modularity and freedom (Figure 6 right). In addition to implementing our design philosophies, we must ensure that the requirements set in the PDS are met, so a brief explanation on having such design approach is presented. Modularity focuses on robustness of the armband and ease of manufacturing:

- **Robust:** Each MMG or IMU module is manufactured and packaged independently. The malfunction of one MMG module will not damage the other units. The system can still perform tasks (max number of recognized gestures will be reduced) if one or more MMG modules are damaged.
- **Manufacturability:** The design is separated into independent tasks. The whole system is only made up of three components: IMU, MMG, and the flexible strap. Different components can be outsourced independently to different manufacturers.
- **Simple Repair:** The malfunction system can be repaired by only replaced the damaged module, and there is no need to alter the rest of the system.
- **Upgradability:** Existing users can experience cool features of future generation systems by simply replacing old modules with updated ones. Customers do not need to buy a complete new system for each product generation, and it is much environmental friendly.

Freedom focuses on giving users and developers maximum flexibility on their own armbands (e.g. armbands can have different number and combination of sensors). To illustrate, we have tested systems with two and three MMG sensors. Having more MMG sensors can increase the maximum number of gestures that the system can recognize, and locating sensor(s) on different combinations of muscle groups results in different gesture recognition accuracy.

There are a lot of different MMG sensor arrangements, and we want to try as many as we can, however changing the number or arrangement of MMG sensors usually require physical alterations on the hardware. Therefore, we design the product with modular foundation, and adding or removing MMG sensors is as simple as playing LEGO blocks. Having a modular design, we provide plenty of flexibility to our users and developers:

- Users can have different number (i.e. range from 1 to 7) of MMG(s) on their setups.
- Users can put MMG(s) on different combination of arm muscle groups.

Having the flexibility, users can optimize their systems for their unique tasks. For example, users' tasks only need two or three gestures. They can have a system with only one module. Users can optimize their systems' performances by trying different combinations of muscle groups to put MMGs.

Giving developers the flexibility not only benefit them but also us. Myogram is a still a developing technology, and muscular contraction is a complex biological phenomenon. We encourage developers to be "crazy" on their systems because their unique experiments will improve our understanding of this complex biological system. Therefore, we need to work closely with all developers, and their knowledge obtained from all different system setups are critical to optimizing our product.

Detailed Component Design

MMG and IMU

There are two main challenges on MMG and IMU module design. First, the thin metal film of the MMG sensor must touch the skin. Therefore, MMG sensor needs to sit on a flexible structure which can curve around the curvature of arm. Two different approaches are proposed to tackle this: silicone and flexible ribbon. The current design adapts the latter approach (Figure 6 right), where the main body of the module is hard plastic, and the extended ribbon is made up of flexible material (e.g. silicone, fiber, leather). Secondly, the thickness of all modules cannot exceed 9 mm, which is specified in PDS. The current IMU module design is 16 mm, and it is difficult to shrink the dimension without alterations on PCB design. Further considerations and discussions are needed on IMU module dimensions.

CONNECTOR

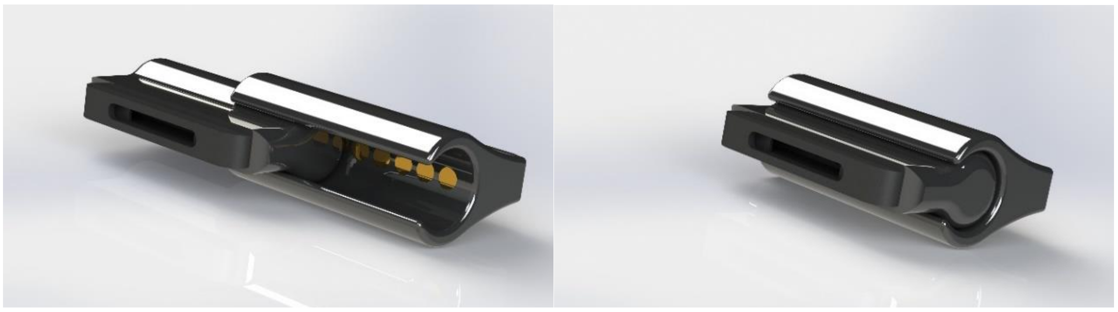


Figure 7 Connector Design

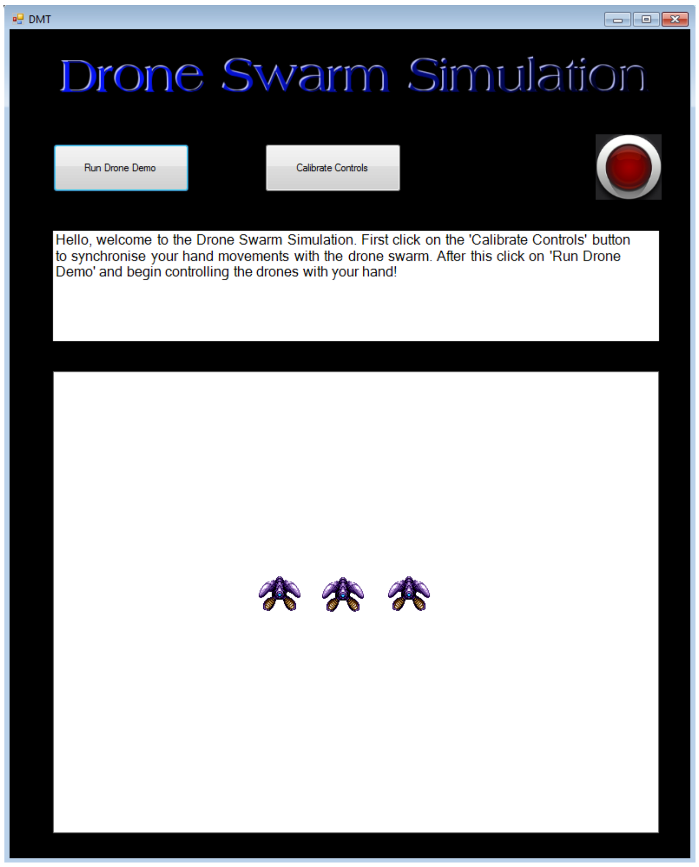
A robust connector guarantees a stable connection between independent modules. We prioritize connector design from all hardware designs not only due to its importance but also because the same connector design applies for any module designs.

The connection must be mechanically strong when connectors are connected. In the same time, users should be able to easily disconnect modules when they want to. Two main categories of connector design were proposed: button clip and sliding slot. We have chosen the second approach in the current design (Figure 7 left) because the sliding slot approach is geometrically stronger than button connection does. Moreover, it will be difficult to click the button in place when dust and dirt are accumulating inside female button clip over time.

The connectors provide not only mechanical connection but also electrical. Inside the male/female connectors, there are spring loaded pin (pogo pin) conductors. There are eight pins in each connector (Figure 7 right), which two are used for power delivery and the others for signals. When the male connector is slide into the female connector, the male and female pins will be connected. The pins are covered inside the connector bodies during connection, which is designed to be water spill proof (PDS requirement). Tests on spill proof and connectivity will be conducted for the manufactured prototypes.

Simulation Interface

Using *C#* windows form applications, a simple program has been created allowing the user to control a swarm of three drones. This simulation provides an opportunity to test and showcase the success of our convolutional neural network and gesture recognition before proceeding to an actual quadcopter.



Hello, welcome to the Drone Swarm Simulation. First click on the 'Calibrate Controls' button to synchronise your hand movements with the drone swarm. After this click on 'Run Drone Demo' and begin controlling the drones with your hand!

Figure 8 Screenshot showing current simulation interface model

The user is given step by step instructions on how to calibrate the software by completing a series of gestures as a light flashes. After this has been completed the user may begin controlling the drones, making the move in a box and giving them commands such as split, assemble or rotate.

Currently keyboard input is used as the program is yet to be synchronised with the developed convolutional neural network. Once this is done, the drones will be moved by movement detected by the accelerometer, gyroscope, and magnetometer. In addition, 5 gestures will be assigned to 5 commands.

Compared to the initial Gantt Chart in our Quality Plan, the simulation is behind schedule, as data is yet to be transferred from the HRI to the simulation. This step should have been completed by November. This delay is due to the required time to familiarize ourselves with *C#* being longer than anticipated as the language is very different to the Embedded C previously learnt.

It is now planned to link the neural network with this simulation by 19th February, also the graphics and presentation of the simulation will be refined. We will then test and refine both the simulation and the neural network accordingly.

Design Review

To help improve and find any flaws with our project so far, we arranged a design review with a senior member of staff, Dr Nicolas Cinosi. He had no prior knowledge of the project, allowing him to give a fresh and unbiased view of our work and goals. In addition, he has expertise in control systems and mechatronics, making his knowledge very useful for us.

We initially briefed him on our overall project and then what we had achieved so far. Then we began discussing the next step of controlling actual quadcopters with the HRI. Our plan was to use a set of cameras in a room to determine the position of the quadcopters and by using a feedback control system ensure that they do not collide with each other and can carry out commands together. Dr Cinosi explained to us that this will be a very difficult challenge, due to the dynamic instability of the quadcopters that a control system would bring.

He advised not using a control system for a single quadcopter, only when it is necessary for them to interact with each other. It was suggested that we progress step by step, first perfecting a single quadcopter control with the HRI, then a single with camera control and if time permits undertake the challenge of multiple interacting quadcopters.

Next, we showed him the current CAD design of our revamped hardware. He was very impressed by the modular system we had opted for, where the number of IMUs can be chosen based on the given application, allowing the device to be used for other projects. He could not see any faults with the design, but as he does not specialise in design & manufacturing, he was unable to assure us of our plan to manufacture the design. In order to get more of an insight on our hardware we plan to have another design review with a professor specialising in design.

Overall the design review was very helpful, we will now rethink our use of control systems in the quadcopters and take a step-by-step approach, refining each stage before progressing to the next. Also, it was especially reassuring to have an expert approve of the work we have done so far.

Design Verification Plan

As we complete each of our tasks we must thoroughly test what we have done to ensure that it is fully functional before advancing to a more complex stage. This is especially important in our programming as minor adjustments can suddenly cause the entire system to stop working.

By the end of February, we will finish linking our drone swarm simulation with our developed neural network. This will allow us to practically test the reliability of our neural network to ensure that the gestures are recognised a high probability and that the motion sensors are smooth in detecting movement. Also, the simulation will be tested to guarantee it is glitch-free and presentable. The test will be simply using the simulation for a length of time and ensuring all the controls are adequate. It will be tested by our own team and peers to give a range of opinions.

In the middle of March the new hardware will have been manufactured. We will test this using the drone swarm simulation as the single quadcopter will not be ready yet. This test will allow us to assess if the control still runs smoothly with the new hardware, or if addition data collection is necessary to account for any changes the new wristband has brought. This will be tested on a range of arm sizes to ensure that our design fits the population described in our PDS. In addition, the tests allocated in our PDS will be carried out.

Our single quadcopter control is scheduled to be completed at the end of March. It will be tested to ensure that all the requirements of our PDS are met. The motion control will be thoroughly evaluated to confirm it is smooth. At this stage, we will test using a sample from the public, this will ensure that the product is easy to use for the whole population, not just Imperial students. The test will involve carrying out a series of objectives using the motion control, for example moving the drone from one fixed point to another.

Lastly, at the end of May our physical drone swarm with location control will be finished. This will require extensive testing to ensure that the drones interact with each other correctly and safely. The quadcopters will be tested in a room with positioned cameras. Similar to the previous test, we will carry out a series of tasks and ensure that they all run smoothly. Once we have verified the safety of the swarm we will test on the public to ensure that it is user-friendly for the general population.

Conclusion

In terms of the project progress, all parts of our project are on track despite the delay last year. The progresses on software will be further checked as we assemble the neural network with the simulation by the end of February. The hardware design is planned to be manufactured and verified by mid-March. If the assembly process goes well, our current progress will allow us to stick with our most ambitious plans – applying location control on one quadcopter and trying to control an actual swarm of quadcopters in May.

References

Karpathy, A. (2016). CS231n Convolutional neural networks for visual recognition.

Kingma D, B. J. (2014). Adam: A method for stochastic optimization.

Krizhevsky A, S. I. (2012). Imagenet classification with deep convolutional neural networks.

Ma, Y. (2016). *MMG Gesture Recognition with Neural Network.*

Matthew D Zeiler, R. F. (2013). Visualising and understanding convolutional networks. Springer International Publishing.

MHadmin. (2016, March 29). *Finding a low-latency webcam*. Retrieved 2017, from MakerHardward.com: https://www.makehardware.com/2016/03/29/finding-a-low-latency-webcam/

Srivastava N, H. G. (2014). Dropout: a simple way to prevent neural networks from overfitting.