OpenDroneControl [ODC]: An Open-Source Platform for Creative Use of Aerial Robotics

Tim Wood, RJ Duran, Sterling Crispin
Media Arts and Technology Program
University of California, Santa Barbara
fishuyo@mat.ucsb.edu | rjduran@mat.ucsb.edu | sterling@mat.ucsb.edu

Abstract

We've developed an open source software platform for developing interactive artworks and research projects with aerial robotics, called OpenDroneControl (ODC) [1]. It aims to be a community supported framework developed to be compatible with creative coding software such as Processing [2], Max [3], and open Frameworks (oF) [4]. ODC attempts to address the need for a common interface to a diverse range of available platforms as the field of aerial robotics grows.

ODC is designed with the intention of being an abstract interface connecting an application to any commercially available drone hardware platform and optionally providing additional functionality such as navigation and tracking. Application developers can utilize this common interface to easily target multiple drone platforms without redesigning their code which allows for rapid project development. This lets developers focus more on content and less on hardware troubleshooting. ODC provides access to platform specific sensors and functionality, and is also flexible enough to support platforms with additional sensors and third party peripherals.

1. Introduction

1.1 Origins

The development of this project was initiated by the Transvergent Research Group [5], including Media Art and Technology [6] graduate student researchers Tim Wood, Sterling Crispin, and RJ Duran. The technology was developed under the direction of Professor Marcos Novak, at the transLAB, housed within the California NanoSystems Institute (CNSI) at the University of California Santa Barbara.

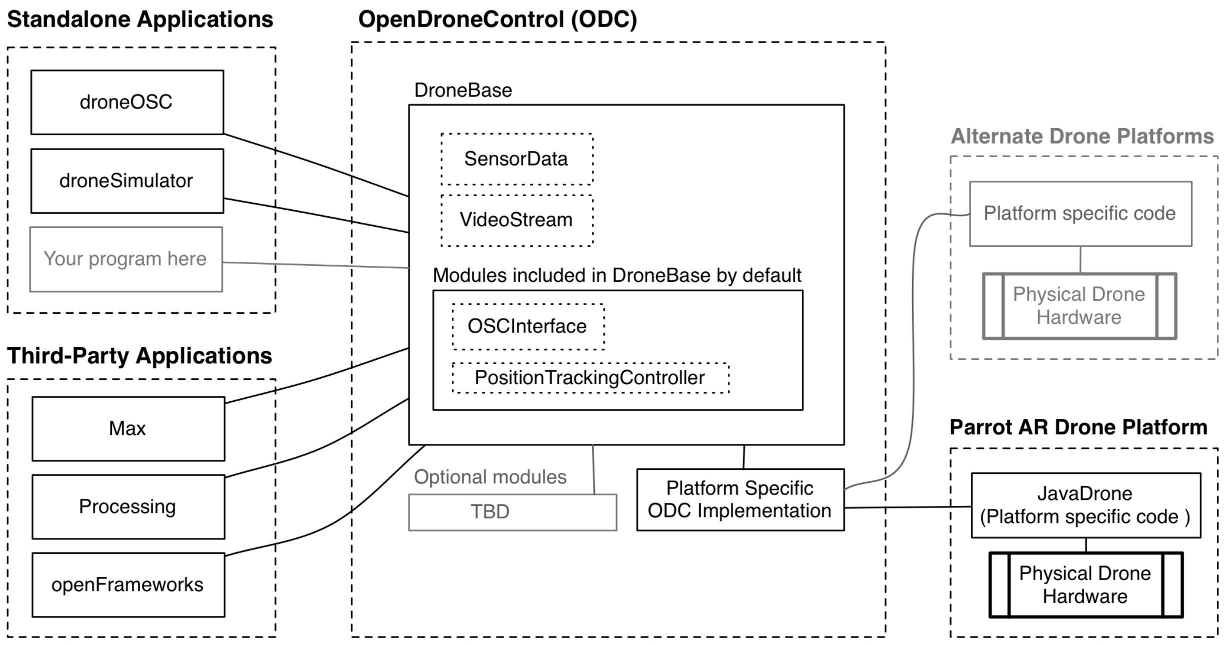
ODC originated from experiments within the transLAB's motion tracking environment using a Parrot AR.Drone [7] at the beginning of 2012. Initially the focus was to create an external Max object which could algorithmically control the drone using available spatial information provided by the OptiTrack [8] system. The production of several multimedia experiments and artworks required us to contemplate the role drones could play in society, and watched as a wider range of commercial platforms began appearing. ODC was created to encapsulate the work done into a framework that could support new hardware platforms and collaborators. We hope that this effort expands the accessibility of this often controversial technology to those who may discover new and positive ways of utilizing drone technology.

1.2 Audience and Use Cases

ODC is intended for use by, but not limited to, artists, programmers, hackers, developers, makers, robotics experts, scientists, designers, researchers, educators, students, and hobbyists.

Potential single-agent applications include aerial photography, interactive performance, environmental sensing and data collection. Potential multi-agent applications could include, synchronized flight formations, physical or wireless drone-drone interactions and other swarm robotics applications.

2. Framework Design



The framework design consists of three components:

- ODC Core – generalized drone API
- ODC Modules – higher level components that can be used with any drone platform
- ODC Platforms – hardware specific implementation of the core interface

2.1 Core

The framework at its core is a generalized drone API with a few additional functionalities added.

ODC Core consists of three components:

- DroneBase – the abstract base class that must be implemented to create an ODC Platform
- SensorData – optional collection of current sensors and their values
- VideoStream – optional object for retrieving current frame from a video stream

DroneBase is an abstract interface to what a basic drone hardware platform needs to provide. These key functions include:

Description:	**Example:**
Drone initialization and deinitialization:	`connect, reset, disconnect`
Drone flight and movement:	`takeOff, land, move(x,y,z,w)`
Drone configuration and custom commands:	`config(key,value), command(com,args*)`

The ‘command’ function allows access to any platform specific functionality by passing it the name of the function and any collection of arguments. For example, to trigger an led animation pattern on the AR-Drone:

```
drone.command(“led”, pattern, frequency, duration)
```

SensorData is an object that contains a map of all the current sensors and their values. Every platform should create a corresponding SensorData object on initialization and update the sensor values periodically. We consider a sensor to be any piece of data that changes over time. This includes data such as: accelerometer values, gyroscope values, altitude, battery status, distance sensors, temperature sensor, flying status, and gps coordinates. It is easy to get sensor data and query your current platform to see if a specific sensor is available. This way applications can provide selective functionality based on the current platform’s available sensors. It is also possible to bind a callback function which is called every time a sensor is updated.

```
drone.sensors.hasSensor(“gyroscope”)
drone.sensors.get(“gyroscope”).value
drone.sensors.bind((sensor) => { println( sensor.name + “ : “ + sensor.value ) })
```

VideoStream is an object for handling access to drone video if it is available. Each platform with an available video stream should implement and set the VideoStream object on initialization. The VideoStream object simply provides the most recent frame from the stream.

```
if(drone.hasVideo()) drone.video.getFrame()
```

2.2 Modules

ODC can also provide additional functionality through higher level modules. These modules could cover a wide range of functionalities whether it is providing greater navigational control, autonomous behavior, face tracking or interfacing with custom equipped sensors. The list of possible extensions will surely grow. ODC currently provides two additional modules:

- OSCInterface -- Open Sound Control [9] server listening for control commands from the network
- PositionTrackingController -- used with an absolute positioning system such as an external infrared tracking system, to add precise spatial navigation possibilities

OSCInterface allows for multiple OSC enabled applications or devices to control a single drone from anywhere on the network. The OSC interface exposes all the functionality of the ODC platform even platform specific commands can be triggered through the network. It is also easy to start streaming sensor data over the network.

```
	drone.osc.start(8000)                    // starts OSC interface on port 8000
	drone.osc.sendSensors(“192.168.1.255”, 8001)    // start broadcasting sensor data
on port 8001
```

After the OSC interface is started, ODC will receive commands from the network such as:

```
	/connect
	/takeOff
	/move 0. 0. 0. 1.
	/config ‘maxEulerAngle’ 0.3
	/led 0 1. 1
```

Providing motion tracking enhanced navigation was one of the initial goals of building ODC for use in the transLAB at UCSB. The PositionTrackingController Module adds the ability move the drone to a point in space and create more complicated flight patterns and waypoints. External tracking data can be fed into the PositionTrackingController, allowing it to calculate and automatically adjust the drone’s flight dynamics to move it to a designated destination in 3D space. The PositionTrackingController uses multiple PD (proportional-derivative) controllers to calculate appropriate movement commands to the drone. This tracking component adds the ability to send the drone moveTo commands given that you send a step command at a regular interval to update the drone’s current position:

```
	drone.tracking.moveTo( 0, 1.0, 0, 0 )
```

On receiving each frame of tracking data (x,y,z,yaw):

```
	drone.tracking.step( x, y ,z, yaw )
```

2.3 Platforms

ODC currently provides a platform implementation for the Parrot AR.Drone using the pre-existing JavaDrone API [10]]. ODC also includes a very rudimentary virtual drone simulation platform called SimDrone which can be used to test out ideas without physical hardware. Additional ODC platforms can be easily created by implementing the classes described above.

2.4 Language

ODC is currently implemented using Scala [11]. Our initial decision to use Scala was based on our goal to get started with the AR.Drone quickly while working toward providing higher level reusable modules. It was also important for us to provide ways of interfacing with rapid prototyping and creative coding platforms such as Max and Processing. Scala runs on the Java Virtual Machine (JVM) and is able to be used with other JVM languages and libraries. It is easy to build as an external object for Max or to use as a library in Processing.

Scala is a powerful hybrid object oriented / functional programming language with a robust set of features. Scala is also very fun to work in, providing very minimal syntax and type inference, it often feels like using a scripting language. Scala provides a powerful Actor system that allows for easy development of concurrent applications and distributed systems, which will surely come in handy and allow for communication between multiple ODC instances in the future.

A downside to using a JVM language as the basis for our ODC implementation is that native drone APIs will probably be provided primarily in C. This would simply require using a JNI bridge to implement the corresponding ODC platform. However, this may not be the case however and we will have to be flexible and work with new platforms as they come out.

2.5 Examples

2.5.1 Standalone

The standalone examples are designed to be run from inside the root directory (opendronecontrol/odc) using Scala and sbt [12]. To run, first connect to the AR.Drone 2.0 device over WiFi using the default IP address (192.168.1.1) for the hardware, then execute the following command:

```
./sbt "project examples" run
```

This executes a script that downloads a local copy of sbt for running the examples, changes the current project, then compiles and runs available classes in the directory. The user will be prompted to select the example to run from a list of options.

GetSensorData.scala	-- Read sensor data from the hardware.
TakeoffLand.scala	-- Tell the device to take off, hover for 10 seconds, then land.
TakeoffLand2.scala	-- Tell the device to take off, hover for 10 seconds, then land.
TakeoffMoveLand.scala	-- Tell the device to take off, oscillate the drone left/right, then land.
droneOSC.scala	-- Create an AR.Drone client and start listening for OSC messages.

2.5.2 Processing

DroneOscP5.pde	-- Control drone and receive sensor data using OscP5

2.5.3 Max

OpenDroneControl.maxhelp	-- Example patcher file

2.6 Applications

The apps directory is for holding included applications that use ODC.

- maxmsp-external - a Max mxj external for controlling the AR.Drone ODC Platform
- DroneSimulator - an example application visualizing the virtual SimDrone Platform
- LeapController - an example application using a Leap Motion [13] controller to fly an AR.Drone.

3. Projects Developed using ODC

ODC has been used at UCSB to develop two substantial artworks “Catch and Release” and “Charon” as well as a number of spatial, auditory and visual experiments and performances

“Catch & Release” [14] is a collaborative artwork by Tim Wood, Sterling Crispin and RJ Duran. It explores the connection between man and machine through structured and improvisational dance and virtual embodiment. The dancer, Tim Wood, leads a dramatic performance that plays on choreographed and improvisational dance techniques in relation to the acrobatic motion of a flying robot known as a drone or quadcopter. The piece begins softly as both man and machine awaken. They emerge into the space then progress to explore each other from what was previously foreign and unknown. As the piece progresses it unfolds in a tense and energetic exchange between Wood and the robot, at times in his control and at other moments, out of his control for the duration of the piece.

“Charon” [15] is a work by Sterling Crispin which utilizes 3D printing in an attempt to physically embody the tension between humans, robotic autonomous agents, and the virtual models which these agents rely on to understand the world.

4. Related Work

4.1 Similar Frameworks

ODC is one of many open source frameworks for managing the flight control mechanisms of the AR.Drone 2.0 platform. It attempts to differentiate itself by being a community oriented project capable of controlling a variety of hardware platforms. This is possible by using a common programmatic interface and extensible instruction set. By developing and using a common syntax to operate a variety of aerial robotics platforms we can begin to design and build projects leveraging hardware configurations best suited for the application at hand.

Among the countless frameworks and open source initiatives for controlling aerial robotics in a variety of applications, Nodecopter [16] and OpenPilot [17] can be seen as stable platforms capable of providing similar control mechanisms. NodeCopter runs using Node.js [18], which is a Google Chrome based JavaScript platform for building scalable network applications. It is easy to begin using with basic JavaScript programming knowledge and has a healthy number of community developed modules to extend functionality. ODC is inspired by this design philosophy and aims to make it easier for artists and designers to develop their own libraries for extending functionality.

OpenPilot is a platform using open source software developed for applications in aerial photography and video. They maintain support for multicopters, helicopters, and fixed wing aircrafts. ODC aims to integrate a variety of platforms to allow users options for controlling vehicles based on their application or project goals.

4.2 Related Frameworks for Artistic Expression

ODC takes inspiration primarily from two open source projects: Processing and open Frameworks (oF). These frameworks are community supported and directed towards the advancement of artistic projects in new media. They are freely available for anyone to download and are typically used in interactive applications related to visual, musical, and spatial arts.

Processing is known for it's general appeal to artists and designers and it's easy learning curve. It builds on Java, which is a common programming language running on the JVM, meaning applications developed for one system follow the "write once, run anywhere" paradigm. This allows the same software to run on a variety of platforms, which is also the intention of ODC.

oF is a more advanced framework due to it being built on top of the C/C++ programming language. This is typically a steeper learning curve for non-programmers due to nuances in the language and abstract logic. Both Processing and oF are designed to be extendable through user libraries and grow stronger with community support. ODC is also built to allow users to integrate additional functionality through libraries.

5. Conclusions and Future Work

ODC is an open source platform for developing interactive artworks and research projects with aerial robotics. Through the process of developing the initial stages of the framework, we conclude there is still a large amount of work to be done in developing a stable usable platform. This is where we invite other developers to contribute to the project in any way they can. Future work involves extending functionality through the development of hardware and software modules to augment behavioral characteristics, sensing, and artistic aesthetics.

As aerial robotic platforms become more commonplace it is our collective duty to shape the roles they inhabit in our lives, and we hope that ODC can provide a platform for this exploration.

6. References

[1] OpenDroneControl, http://www.opendronecontrol.org/
[2] Processing, http://processing.org
[3] Cycling ‘74, Max, http://cycling74.com/
[4] openFrameworks http://www.openframeworks.cc
[5] Transvergent Research Group, http://translab.mat.ucsb.edu
[6] Media Arts and Technology, http://mat.ucsb.edu
[7] Parrot, AR.Drone http://ardrone2.parrot.com/
[8] Natural Point OptiTrack, http://www.naturalpoint.com/optitrack/
[9] Open Sound Control, http://opensoundcontrol.org/
[10] JavaDrone, https://code.google.com/p/javadrone/
[11] Scala Programming Language, http://www.scala-lang.org/
[12] Scala sbt, http://www.scala-sbt.org/
[13] Leap Motion, https://www.leapmotion.com/

[14] Tim Wood, Sterling Crispin, RJ Duran, Catch & Release, http://rjduran.net/projects/catch-release
[15] Sterling Crispin, Charon,http://www.sterlingcrispin.com/charon.html
[16] NodeCopter, http://nodecopter.com/
[17] OpenPilot, http://www.openpilot.org/
[18] Node.js, http://nodejs.org/