Simon Fraser University

IARC Competition Journal Paper

31st May 2015

Undergraduate Student Contributors

Christopher Nsimbe, Alireza Alidousti, Luke Mulder, Dev Bhullar, Refayet Siam, Kush Chhatbar, Kathleen Moriarty, Chander Siddarth, Abiman Mahendra, Evgeny Kuznetsov

Abstract

We present a vision-based autonomous solution to this year's International Aerial Robotics Competition (IARC) challenge. Data from our vision system is filtered for different kinds of information: static objects of the environment, and moving targets. The major static object of the environment is the grid on the arena. The features of static objects from the filtered vision data are fed into our local position estimate algorithm that relies of optical flow of these static features. We apply a Monte Carlo algorithm that gives us a rough estimate of where we are in the global (arena) grid. Information about our moving targets is fed into the target identification loop that characterizes the state of the targets. This information is used by the decision making program to plan and execute actions to affect the movement of the targets.

Introduction

We identified two major challenges in this year's competition:

1- Determining our local translations and global position (relative to the arena).

2- Determining the state and motion of the multiple targets.

The difficulty in determining our local translations is that in a GPS and SLAM free environment, we rely on optical flow algorithms that assume the vision sensors are getting data from static objects in an environment. The optical flow algorithms become unreliable when there are moving objects in the environment.

The targets have a short span of deterministic motion (5 seconds) and the near future position of the targets is dependent on the position and direction of other surrounding targets. Coupled with the fact that some targets get out of our field of view as the aerial vehicle moves, and that we can only see a limited patch of the arena at a given time, determining and keeping track of the state and motion of the ground targets is a major challenge. Knowing what area of the arena the aerial vehicle should inspect at a given time and what it should do once it identifies this region requires a heuristic solution heavily based on probabilistic robotics.

Obtaining reliable data for the optical flow algorithm can be resolved by identifying static objects of the environment, such as the grid of white lines. We use the Monte Carlo algorithm to estimate our global position [1].

For the ground targets, we characterize an area of the grid based on the number, direction and proximity of the ground robots to each other. We avoid areas we determine to have a "cluster of activity". We instead focus on less clustered areas, where the state of the target(s) is easily determined and more certain.

System Overview and Architecture

Our hardware system features a co-axial octocopter, integrated with the pixhawk flight controller.

The on-board computer is the Odroid U3, an embedded linux computer [2]. The vision sensor (camera) is attached on a Tarot T 2D gimbal under the octocopter. A radio link connected to the pixhawk is used as a safety switch override in case the system malfunctions. The diagram below further elaborates and details the hardware system.

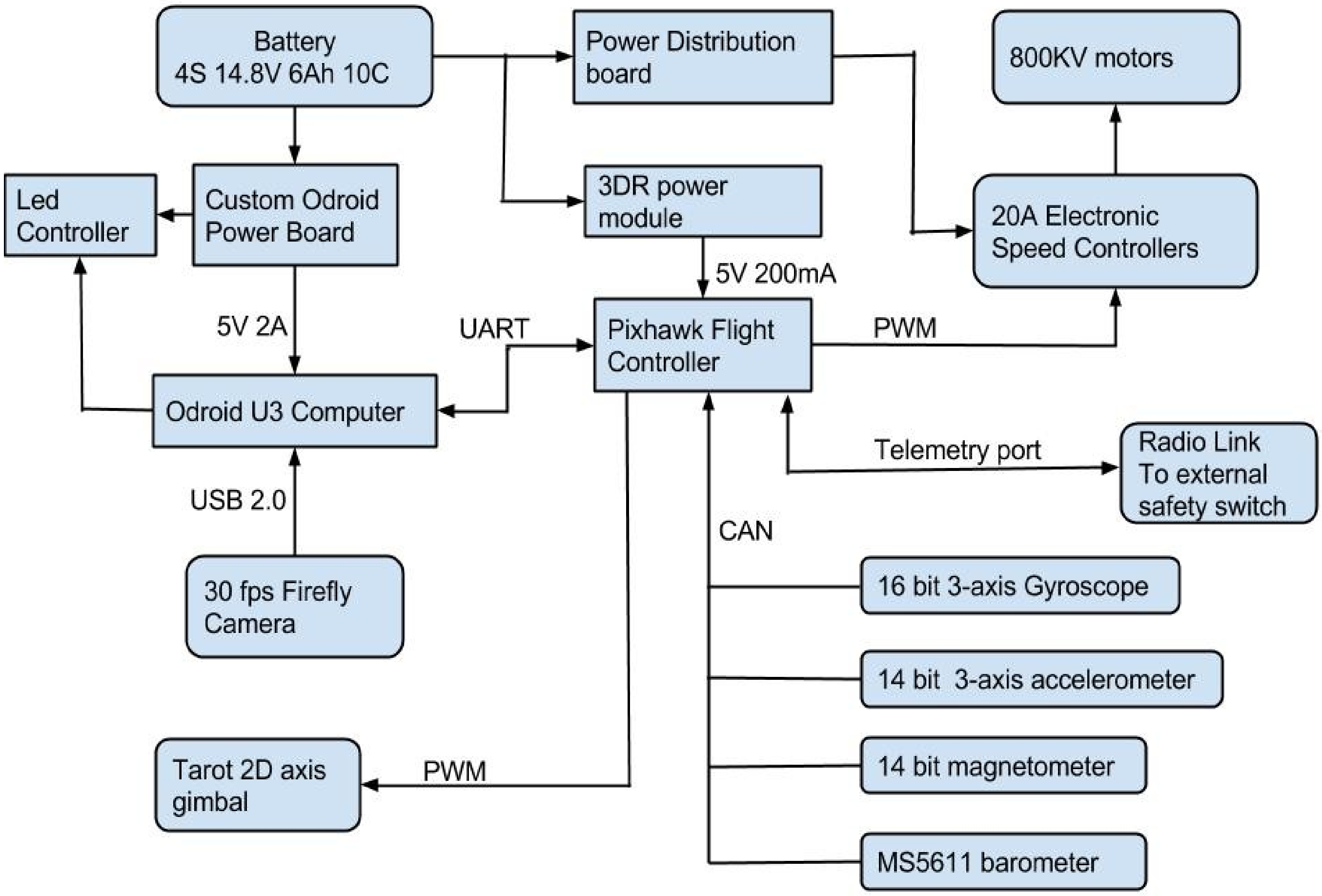


Figure 1. Hardware System Architecture

Since we joined the competition earlier this year, we have set weekly and bi-weekly targets and milestones which we have diligently worked towards.

The table below summarizes the our monthly targets and milestones

Team	Task	Deadline	Progress
Flight Control team	Flying in a square, using camera	9^{th} April	100%
	Flying to a point in air, - Flying to a target close on ground	16^{th} April	70%
	Landing in front of a target	23^{rd} May	50%
	Safety control with software	30^{th} May	80%
Image processing team	Detecting a simple grid	9^{th} April	100%
	Detect a grid in difficult backgrounds	16^{th} May	100%
	Detect a create in with a simple grid image	23^{rd} May	100%
	Detect a create in with a difficult grid image	30^{th} May	50%
AI team	Incorporating aerial vehicle into simulation with an initial simple behavior	9^{th} April	80%
	Set up simulation to test various strategies	16^{th} May – 23^{rd} May	50%
	Use a live data run with the simulation with a simple setup	30^{th} May	50%
Hardware team	Come up with robust and safe design – 3D printed	16^{th} April	100%
	Magnetic influence sensor to tapping creates.	23^{rd} May	50%
Financial team	Forward funding proposals to companies, Update website	16^{th} May	50%

Table 1. Monthly targets and milestones

Air Vehicle propulsion, and control

We use 8 motors, each rating 800KV to provide thrust with 10x47 propellers. There are 4 coaxial rotors, each with two opposite facing motors. The angular speed of the motors are controlled by 20A Electronic Speed Controllers (ESC's)

The table below shows the trust output of the motors depending on the propeller specification and power[3].

Motor:V2216-12 KV:800

Technical Datas	
KV	800
Configu-ration	12N14P
Stator Diameter	22mm
STator Length	16m
Shaft Diameter	3mm
Motor Dimension(Dia.*Len)	Φ27.8×34mm
Weight(g)	75
Idle Current(10)@10v(A)	0.3
No.of Cells(Lipo)	2-4S
Max Continuous current(A)180S	17A
Max Continuous Power(W)180S	180W
Max. efficiency current	(5-15A)>80%
internal resistance	175mΩ

Recommended Prop(inch)			
Standard	3s-1045/1150	Max thrust	3S-1150
	4S-8040/8050		4S-9047/9050

Tested with SunnySky motor 20A ESC

Prop	Volts (V)	Amps (A)	Watts (W)	Thrust (g)	Thrust (oz)	Efficiency (g/W)	Efficiency (oz/W)
1047	7.4	5.8	42.92	510	17.99	11.88	0.42
	10	9.2	92	800	28.22	8.70	0.31
	11.1	10.5	116.55	960	33.86	8.24	0.29
11X7	7.4	5.8	42.92	510	17.99	11.88	0.42
	10	9.5	95	850	29.98	8.95	0.32
	11.1	10.9	120.99	1020	35.98	8.43	0.30
12X6	7.4	7.7	56.98	680	23.99	11.93	0.42
	10	12.1	121	1020	35.98	8.43	0.30
	11.1	13.8	153.18	1130	39.86	7.38	0.26

Table 2. Thrust, pitch and power information

The aerial-vehicle currently allows for additional payload of about 800g and flight time of about 12 minutes. Any additional payload reduces flight time.

The software and control system is built on the Robotic Operating System(ROS) and is based on the Micro Aerial Vehicle Link (MAVLINK) specification.

The flight control system uses dynamic local position for navigation, and the global position estimate is used to check whether we are in the right bounds of the arena, ie not above 3 meters for the allowed time limit, or not out of the arena.

The Artificial loop identifies a local way-point to fly to. It then sends this target way-point and the speed at which the vehicle should travel and an acceptance radius around the targeted way-point to the flight control loop. The flight control loop begins monitoring its translation from time, t=0 (when the way-point command was sent). The flight control loop sets the linear acceleration required to reach the desired way-point at the request speed, ie

Sr = requested travel speed

Vi = current 3D velocity vector

Pt = target 3D way-point vector

Pi = current 3D translation vector since t=0

Ra = accepted radius around Pt

ai= 3D acceleration vector to be set

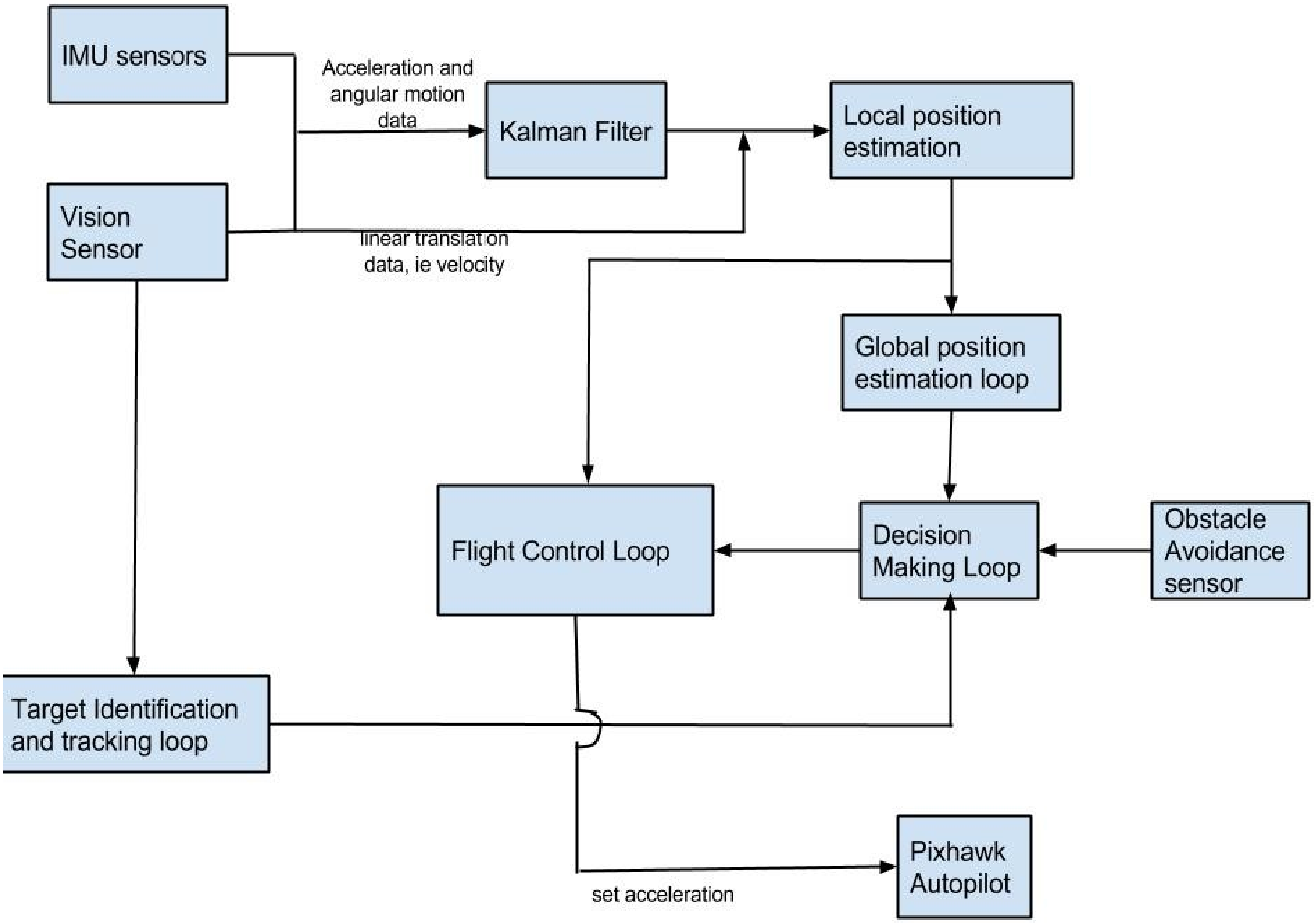
The acceleration is set according to

*ai = (V*V - Vi*Vi) / 2*(Pt-Pi)*

Where V is the desired velocity, such that |V| = Sr and the orientation of V is the same as the vector (Pt-Pi)

When the aerial vehicle is with the acceptance radius of the target way-point, the local origin is reset to (0,0,0)

The pixhawk flight controller features an autopilot features capable of setting linear accelerations, and angular translation of the aerial vehicle based on the fused computer vision and accelerometer motion data.



The diagram below shows the flight control system

Figure 2. Control Systems Diagram

The pixhawk connects to an external ground control RC transceiver that acts as a safety switch. The ground control RC transceiver is used to set different flight modes. The main flight modes are;

A) autonomous flight mode, in which the pixhawk autopilot is taking commands only from the flight control loop on the Odroid computer on-board

B) Land mode, the aerial vehicle performing a landing. During this time, it will not accept commands from the flight control loop on-board.

C) Stabilize mode, the aerial vehicle is only controlled externally using the RC transceiver. Switching to this mode, without pressing any of the control buttons on the RC transceiver, causes all the motors to stop spinning and the aerial vehicle falls. This can be used as an ultimate safety control feature in case the aerial vehicle is moving too fast to perform a soft landing, or any other unforeseen scenario.

Additionally the on-board flight control loop has triggers set to perform an autonomous flight termination; velocity, acceleration and angular limits are set.

Sensor Payload

The filtered optical flow data from the vision system is fused with the Inertial Measurement Unit (IMU) data from the pixhawk to determine the orientation, angular and linear accelerations of the aerial vehicle. A kalman filter is applied to filter the data from the IMU sensors and vision sensors to produce more reliable motion estimates of the aerial vehicle. The vision system independently provides and it is the only source of reliable linear velocities and translations.

The Guidance and Navigation Control (GNC) features a a 16 bit 3-axis gyroscope, a 14 bit 3-axis accelerometer, a 14 bit magnetometer, an altimeter and a firefly camera. The altitude calculated from the data obtained from the vision sensors is very reliable if we are flying above a certain certain, ie the aerial vehicle can see clearer grid squares and use that to determine a very accurate estimate of the relative height to the ground.

Below a certain height, ~ less than 0.5m, the vision based altitude becomes very inaccurate, and the vertical velocity from the altimeter is preferred and used to correct the vertical height. The altimeter based vertical altitude is also preferred in other cases where vision data is unreliable, ie unforeseen camera frame error, aerial-vehicle goes out of arena, etc

The targets are identified when the aerial vehicle holds its position and uses the images from the camera to identify parameterised (circle -like) moving features. The moving targets are also tracked when the aerial vehicle is performing a way-point navigation by removing identified static features (grid features) from the edge image fed into to the target identification and tracking loop. We also use the other known information about the moving target (speed of 0.33m/s) to filter and update the target identification and tracking loop.

The obstacle avoidance sensor s used are infrared sensors mounted on the safety enclosing of the aerial vehicle. The vision data is also used to identify the obstacle ground robots. Below a height of two meters, the infrared sensors are greatly relied on to determine position of obstacles relative to the aerial vehicle. This space containing the obstacles is considered “unsafe” and the target identification loop passes the information to the decision making process, that plans way-points to avoid this area. The aerial vehicle could also simply climb to a safe height.

There is no off-board communication save for the external radio safety controller and debug and test data link for practice runs. All software control loops are run on-board the Odroid linux computer. The Odroid computer communicates to the pixhawk auto pilot through a UART serial connection. The communication protocol between the Odroid and the pixhawk is the MAVLINK protocol. The Odroid also has an unused wifi link, which we are still developing for man/machine interfacing.

A Lipo battery rated 14.8V 6Ah is used to power the whole system. We built a custom power board for the Odroid, that produces a steady 5V and maximum of 2A for the Odroid, while the pixhawk features its own power board that produces a steady 5V and a 200mA current limit.
Additionally there is a power distribution board for the ESC’s to each motor.

Flight preparations and Risk Reduction

We use the following check list for preflight, tests, preparation and risk reduction:

A) *Motor check*; We check whether each individual motor works as expected. This is done without propellers on.

B) *Motor + Propeller check*; The propellers are attached to the aerial vehicle and tightened. The motors are run to check whether the propellers are firmly held.

C) An *auto pilot preflight check* is perform to check for the correct calibration of GNC sensors. A live stream of data from the GNC sensors is observed when the aerial-vehicle is moved around by hand.

D) The *connection* between the *Odroid and pixhawk* is tested

E) The *pixhawk - gimbal control* is tested

F) The *obtaining data from the camera* is performed, and *local position estimation loop* is tested. (The aerial-vehicle is not flying at this point, it is instead hand-held)

G) The external radio *emergency landing switch* and the *kill switch* are tested.

H) The aerial-vehicle should perform *a simple takeoff, hold altitude and land maneuver*.

I) The aerial-vehicle should perform *a simple square way-point navigation* and then land.

The checklist above includes testing for software, hardware and firmware safety and reliability.

Part of gaining safety knowledge is through lab and outdoor experience with the aerial-vehicle. While we have not purposely crashed our vehicle to test its robustness, we have inadvertently tested this through failed practice runs. We discovered the most vulnerable parts to damage were the propellers. In this effect we have built rotors guards around the rotor guards to protect them and also to protect other people and equipment from them. The pixhawk is steadily mounted and protected in its enclosing, and so is the Odroid. The camera is protected in a safety enclosure which is then attached to the gimbal.

The Tarot T-2D brush-less gimbal has it’s own IMU sensor suit, and shock absorbers to counter vibrations and its adjusts its servo to keep the camera facing downward [4].

The aerial-vehicle also features status LED's to give visual feedback of the state of the aerial-vehicle. In addition to the RC transceiver that features a human pilot controller interface, the status LED’s form the man/machine interface. We are currently developing a web interface over wifi the Odroid’s wifi for a wider man/machine interaction and safety platform.

We developed our own simulator to visualize the movement of the ground robots, the aerial-vehicle and to test our decision-making loop [5].

Conclusion

This year’s competition presents a series of problems that integrate different spheres of knowledge. The solutions include hardware solutions that requires physical structure, ie safety rotor guard casing, vision data filtering to obtain static environment features and moving targets, probabilistic robotics in tracking and estimating the state of activity in clustered areas of the grid, and a heuristic algorithm to determine what areas on the grid to act on, and what kinds of actions to perform.

The vision system is one of the most elements of our solutions. Most of the key information about our environment is going to be obtained from the vision system, ie our linear velocity, state of the ground targets, obstacle robots, and the arena boundaries. We have therefore dedicated a lot of time and effort in ensuring we process and obtain reliable data from our vision sensor.

References

[1] Sebastian Thrun, Wolfram Burgard, Dieter Fox, "Probabilistic Robotics and autonomous agents" ,2005, pg 187-212

[2] Odroid-U3, *http://www.hardkernel.com/main/main.php*, Updated September 2014 , Accessed on April 2015

[3] Motor information, power rating and thrust, *http://www.foxtechfpv.com/sunnysky-221612-kv800-motor-p-383.html,* Accessed 15th May 2015

[4] Tarot T 2D Gimbal Information *https://store.3drobotics.com/products/tarot-t-2d-brushless-gimbal-kit,* Updated February 2015, Last Accessed 15th May 2015

[5] SFU IARC flight simulator, *https://bitbucket.org/skyx-squadron/iarc-mission-7/,* April 2015.