Real Time Embedded Image Processing System for Points of Interest Detection for Autonomous Unmanned Aerial Vehicles

E. P. Freitas*, I. A. Wieczorek*, C.E. Pereira*, and A. Vinel**

*Informatics Institute, Federal University of Rio Grande do Sul, Porto Alegre, RS, 91501-970 Brazil.
(E-mail: *epfreitas@inf.ufrgs.br; italoaw@gmail.com; cpereira@ece.ufrgs.br*)
** School of Information Technology, Halmstad University, Halmstad, 301 18, Sweden.
(E-mail: alexey.vinel@hh.se)

INTRODUCTION

The use of Unmanned Aerial Vehicles (UAVs) has grown in different fields in recent years. This increasing interest for these systems motivated the research community to further develop the technologies applied on UAVs. Due to its mobility of flight and access to isolated places or of difficult access, UAVs offer a great opportunity of solution to a variety of applications. Many researches are being developed with the use of UAVs equipped with sensors like cameras and lasers, which allow extracting information from the environment.

The energy consumption control of an UAV requires a lot of attention. This aspect is particularly important for small UAV platforms, which are the focus of this paper (from now on, the term UAV refers to small UAV platforms only, unless specifically explained otherwise). Then, depending on the application, different kinds of UAVs may be used to ensure a trade-off between energy consumption and application. To cover large areas, for example, fixed-wing UAVs have better results than multi rotors because the energy consumption and time required to complete the mission are reduced. On the other hand, to obtain images with higher resolutions and closer to a target, multi rotors are more suitable because their mobility enables rapid and drastic changes in its flight plan, besides their ability to rover around a given location.

Besides the flight mobility, UAVs can be equipped with embedded computing boards which can improve application's functionality and performance. Since the additional

payload of these little boards does not compromise the flight time, the UAV can be modified to meet the needs of a specific application. As an example, the UAV can be equipped with LEDs for signaling or speakers for emitting some emergency signals based on the processing of images acquired by an embedded camera.

Despite of the currently embedded computing boards processing power and size, a few years ago this was an impracticable idea due to the impact in the flight performance of mini UAVs. There were few options that allow embedded processing on UAVs. The use of embedded computing boards enables the development of real-time applications that use data gathered by the UAV during its mission to interfere in the mission itself. Boards like Field Programmable Gate Arrays (FPGA) (Altera 2015) or Raspberry Pi 2 (RBP) (Raspberry Pi 2015), Figure 1, are tools that help in data processing for this kind of application. For example, the detection of people or animals in large areas, fire or water locations, both using images collected by the UAV during the mission can be used to determine the next steps it should take, i.e. which areas it should go next.

Figure 1: Raspberry Pi 2.

In a single board is possible to perform more than one task, which allows the execution of simultaneous applications during the flight. The problem addressed in this paper is the possibility to perform multiple applications running, simultaneously, on a RBP board embedded on a UAV. The RBP board presents itself as a cheap and easy access alternative. It accepts multiple languages and different operating systems. However, RBP board is a general purpose hardware, which can prevent the performance of applications that require a flexible and specific architecture.

In order to carry on a performance study of the feasibility of using a Commercial Off-The-Shelf (COTS) platform such as the RBP, the requirements for the algorithm chosen to run on this board were based on an image processing algorithm. This algorithm was chosen because this kind of application generally requires a lot of processing power. The algorithm utilized in this work detects Points of Interest (POIs) by recognizing predetermined patterns, according to the mission specification. Through this algorithm, the images captured by the camera will be analyzed and, if a POI is found, the GPS coordinates of that point will be recorded. In order to achieve the objective of simultaneously running more than one application, a change was made in the POI detection algorithm, generating two instances of the same algorithm. The difference between them is that one has real-time requirements, and the other does not. Therefore, it is possible to prioritize the tasks of the algorithm which has real-time requirements in respect to others. So if these two instances are replicated on the board, it will be

simulating multiple applications running simultaneously. With a modified operating system to meet the real-time requirements, the tests will show the performance achieved by the image processing algorithm with real-time requirements, which has preference in respect to the others non-real-time tasks.

The main goal of this work is to develop and design a specific system for mini UAVs based on cheap and easy to handle COTS computing platforms, as the RBP board. This paper intends to show that it is possible to use the RBP board to run more than one CPU intense application simultaneously, typically required in UAV missions, and presents the performance analysis while running those applications, as well as its limitations for this kind of usage.

RELATED WORKS

A large number of algorithms are needed to make a UAV autonomous. In general, two approaches are used to process these algorithms: one proposes perform a great part of them in a ground station, while others propose embed all algorithms in the UAV itself (Hulens 2015).

When the processing depends on the ground station, there are drawbacks that can prevent a good application performance. For example, the range of the wireless communication signal is limited and the UAV cannot go too far from this station. Furthermore, for the image processing, the cost of compressing the image to send it from the UAV to the ground station can take time and waste processing time. Several studies have been developed based on ground stations.

As can be seen in (Ehsan 2009), in the cases in which image processing is performed in a ground station, the images captured by the cameras are compressed, and then are transmitted via wireless link to a ground station. In addition to the communication delay, the images can be received with noise caused by lost information during the transmission or the compression.

In the case of (Mondragón 2007), image processing is performed in a VIA mini-ITX 1:25 GHz board with 512 MB Ram, but the UAV is dependent on a laptop in the ground station to receive commands, such as selecting the object that the UAV should track. The application of this paper is to track an object-based features using the extrador SIFT. The tests were performed with a UAV equipped with a camera, showing that the proposed algorithm works well to track a user-selected object.

Similar to the previous work, in (Azrad 2010) the tracking of an object is also accomplished. However, in this paper, the image is not processed in the UAV itself, it is sent to a ground station and processed on a laptop. Once processed, the laptop sends to the UAV movement commands that it must make to continue tracking the object.

Although performing another type of application, (Chiu 2011) also depends on a ground station to perform the image processing. This paper proposes a flight control system that uses skyline detection algorithm. Images are captured by the UAV and sent to the ground station. After processed, the ground station sends the commands to the UAV. Different from these papers presented so far, the work that is being proposed here is not dependent on a ground station to perform image processing or trajectory calculations. The purpose is to make the UAV autonomous to perform a given mission with a fully embedded processing unit.

As stated, in addition to the ground station, the algorithms can be run on the UAV itself through embedded hardware. The drawbacks presented above do not happen in this case, since the images or information needed do not depend on a ground station. In (Edwards 2007) is presented a system that locates and tracks landing points for the UAV. After the point is identified, the system will drive the UAV to carry out the landing on its point. The advantage of this work is that the whole system is embedded in UAVs. The image processing algorithm is performed in a Field Programmable Gate Array card (FPGA). During testing the UAV was able to land at a fixed point and in a dynamic point.

Similarly to (Edwards 2007), (Krajník 2012) also uses a FPGA-based module for the image processing. This paper presents a simple and robust navigation based on images for a UAV. In other words, through the camera images the system determines the yaw quadrotor and vertical speed. In addition, (Krajník 2014) also uses an FPGA for processing images. In this case, the focus is the most popular algorithm features extraction, speeded Up Robust Features (SURF). This paper developed a hardware module based on FPGA, for the best performance of the SURF algorithm.

Despite all the researches using the FPGA board for image processing, it is possible to use others hardwares much more simple and easier to handle. In (Faigl 2013), instead of a FPGA, the core of the system is a Gumstix Overo board. This paper presents a location system based on images for a robot swarm with detection plates with black and white geometric pattern. Each robot is equipped with a black and white pattern card that allows the location by the other robots.

The FPGA board, despite a good performance, it is not very friendly to the development and requires specific technical knowledge of programming and hardware. On the other hand, the RBP board supports several programming languages and does not require much knowledge of hardware, besides being cheaper and easily accessible. Considering all the benefits and drawbacks of the related works discussed in this section, this work presents the RBP board as a COTS alternative to perform the image processing and path planning for UAV respecting the real-time requirements of these applications. The RBP is a low-cost board that can be used in several kinds of UAVs, besides allowing running

two or more applications at the same time. It is clear that any other similar board could also be used, such as BeagleBoard. The key idea is to achieve the desired results using a COTS board such as RBP.

METHODS

Points of Interest Application

As introduced in the last section, there are two kinds of image processing instances running in the RPB. One of them has the real-time requirements and, to ensure the execution performance of this algorithm, it has the higher priority level attributed. To make this possible, a real time operating system was installed on the RBP. The used operating system was the ArchLinux (ArchLinux 2015), which is a general purpose GNU/Linux distribution with its development focused on simplicity and minimalism. It is installed as a minimal base system, configured by the user upon which their own ideal environment is assembled by installing only what is required or desired for their unique purpose. To meet the project requirements, the operating system kernel was modified to a real time kernel using the linux-libre-rt package (Linux-libre-rt 2015), available to download in ArchLinux repositories.

The objective of this algorithm is the detection of POI, where these points are defined as square objects of size equal to 40 x 40 cm. Figure 2 shows an overview of the algorithm. The algorithm starts with the conversion of color space from RGB to CIE-Lab, this conversion is based on the work presented in (Ooi 2009) and it shows a better color image segmentation performance than HSV, RGB, I1I2I3 and XYZ spaces.

In the second step of the algorithm, the segmentation was done through the three video channels (L, a and b) in parallel to find all the others pixels that could belong to any POI. The result is three binary images that are compared through a binary mask operation implemented with operation logic AND between them, pixel by pixel.

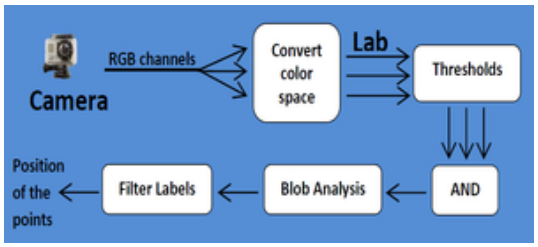

Figure 2: Image processing algorithm schematic.

Then this image is a binary image that contains objects that maybe are POIs. Using the concept of 8-neighborhood, these objects that can be POIs are labeled, and for each is calculated its properties (area and eccentricity).

To choose the objects that are of interest, the algorithm make two verifications: firstly it verifies the size of the object, searching for similarities with the size of markers. In the image, the area of the marker is 100 square pixels. The condition to identify the objects

is based on its dimensions. If the objects has side 50% greater or lesser than the average size, they are eliminated. The second verification is the eccentricity evaluated. If the A-DIMENSIONAL parameter, which varies between zero and one, is close to one, this means that is a line, and otherwise, this means that is a circle.

Power Line Detection Application

An alternative application that was not analyzed in this paper, but is under study by this research group and has similar real-time aspects as the one described above, is a UAV autonomous navigation algorithm using the output of a power line detection by an image processing algorithm. This autonomous navigation algorithm is used to autonomously guide UAVs through power lines, allowing real time inspections during the flight.

This application, which is under development, segments the power lines from the rest of the image through a series of image filters. Then, analyzing the image result identifies line segments which are used as reference to the autonomous navigation.

Figure 3 shows an image from a preliminary version of the application in execution. It is important to observe the complexity of the background in this frame. It highlights the robustness of the algorithm. The challenge is to keep processing the frame in a rate that enables the usage of the lines' detection to interfere in the UAV movement control. The tasks executing the filters and the movement control have to share the same hardware resources and obey strict timing constraints to work correctly.

Figure 3: Power line detection application.

Experiment

In order to perform the evaluation of the hardware performance of the RBP board and the designed system, four test scenarios running both instances of the POI detection algorithm have been proposed, as shown in Table I.

Table 1. Scenarios.

Number	# tasks	Description
1	2	Two tasks Real Time
2	4	Two tasks Real Time and two simple tasks
3	8	Two tasks Real Time and six simple tasks
4	12	Two tasks Real Time and ten simple tasks

Scenario 1 was fitted to evaluate performance and behavior of two real-time tasks running simultaneously. Scenario 2 was designed aiming to show the behavior of the two real time tasks and the other two non-real time tasks. In this scenario is interesting to observe the behaviors difference between the two task types while the four available cores in RBP are occupied. Scenario 3 was thought to increase the simultaneous processing load in RBP board. Finally, scenario 4 is intended to simulate a bottleneck for the system, trying to find the limit of tasks that can be simultaneously performed. In other words, the purpose of this series of tests is to analyze the behavior of the RBP while increasing the processing load and how it impacts the real-time algorithms deadline fulfillment. Information such as CPU and memory usage, algorithms processing time, deadline fulfillment, and real time and non-real time tasks performance will be presented in next section.

RESULTS AND DISCUSSION

A – Core Usage

The RBP features a quad-core processor ARM Cortex-A7 900MHz and the first analysis shows the task scheduling on the four cores with scenarios 1 and 4 running.

Figure 4 shows the behavior of the four cores running scenario 1. In this scenario the two real time tasks were scheduled over core 1 and 4, leaving cores 2 and 3 virtually idle running the essential tasks of the operating system. In this situation the system shows clearly underutilized, showing no processing feature that point to an overload.

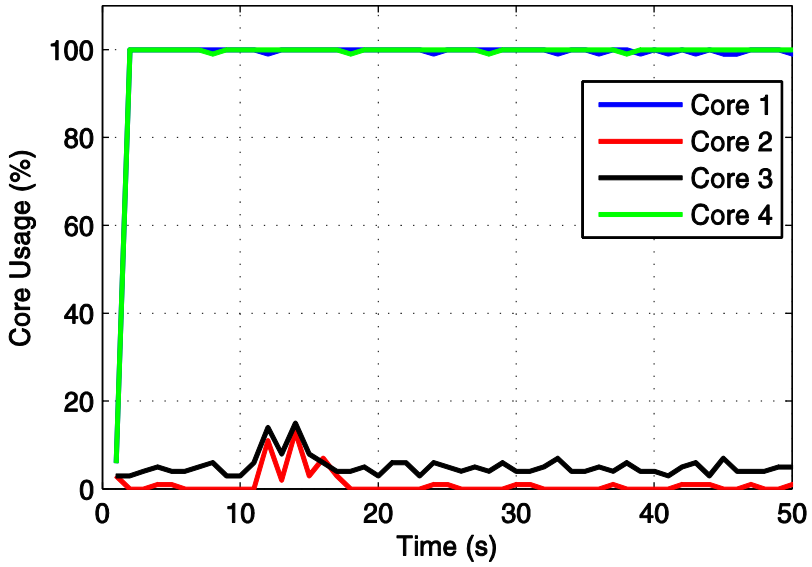

Figure 4: Behavior of the RBP cores running the tasks described in scenario 1.

Table 2. Number of missed deadlines.

Number	# tasks	# Deadline not respected
1	2	0
2	4	737
3	8	896
4	12	977

Figure 5 shows the cores usage on the scenario 4, in which 12 tasks are running simultaneously. Unlike the previous scenarios, now all the four cores work with almost 100% of utilization throughout the trial period, due to the significant increase in the number of tasks running in parallel. This hardware usage level is prejudicial to the system performance, as well as the performance of specific applications running, as it is possible to be seen in Table II, which presents information about the deadlines fulfillment.

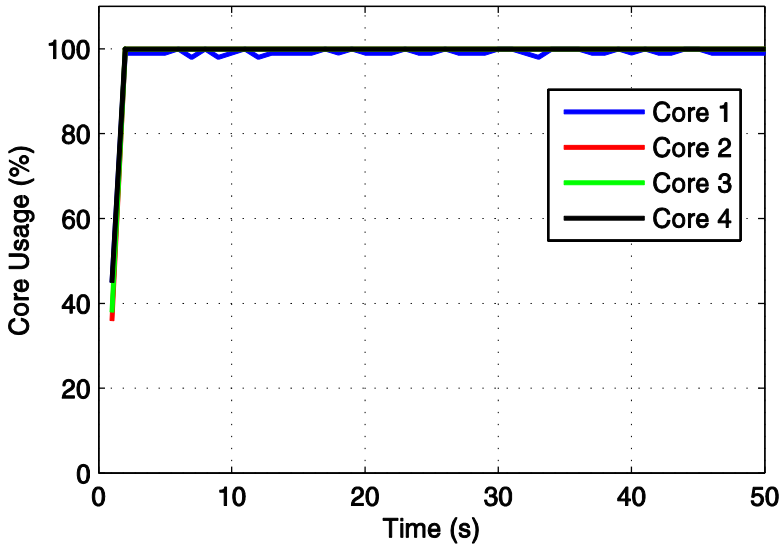

Figure 5: Behavior of the RBP cores running the tasks described in scenario 1.

The deadlines fulfillment (Table II) was achieved by analyzing a real-time task on each proposed test scenario. The time value used as threshold to determine whether the deadline was missed or not, was the maximum execution time (WCET) of the image processing algorithm on scenario 1, which is 0.3309s. This value was taken from a sample of 1000 executions of scenario 1, and also represents the number of samples of sets of runs 2, 3 and 4.

As shown in the Table II, as the processing load increases, the number of missed deadlines significantly increases. In scenario 2, 73.7% of the deadlines were missed. This value increases in scenario 3, to 89.6%. In the scenario 4, 97.7% of deadlines are not met. The number of deadlines not met from the scenario 2 already makes unfeasible the usage of this system for critical real time applications that require time synchronization, for instance. These results certainly present a better performance if the threshold value of time was increased. However, as the goal of the test is not present good results of meeting deadlines, but to evaluate the capacity of the board's hardware, the threshold value of time was kept.

B - Memory Usage

The RBP has 1GB of RAM, which is a fairly significant value to its proposed hardware. The analysis of memory usage occurred in the same way that the analysis of CPU utilization, i.e., the 4 different scenarios, with the processing load gradually increasing in each scenario.

The result of this analysis (Figure 6) is quite logical, since the increase of processing load requires a greater use of memory. The most relevant information is that while the system is overloaded at scenario 4, less than 80% of memory is used by the 12 simultaneously running applications. This shows that 1GB of RAM does not become a bottleneck of the system.

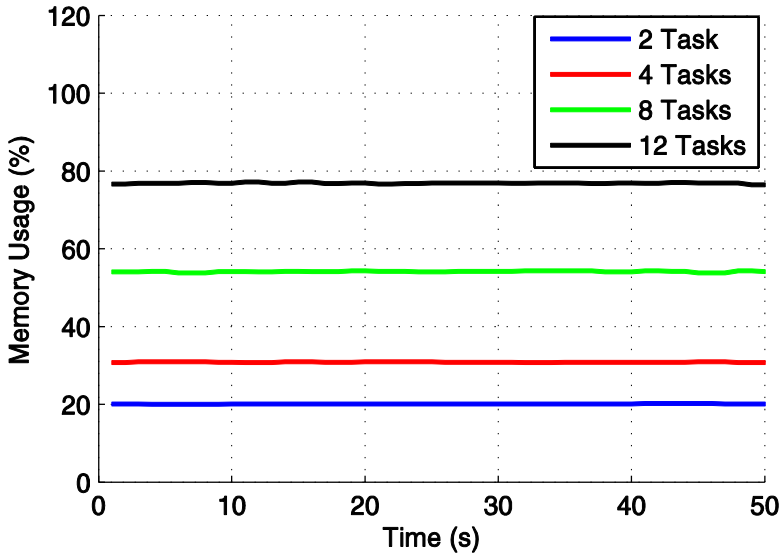

Figure 6: Behavior of memory usage for each scenario.

C – Real time vs Non-real time tasks

At this stage of testing the remarks was the behavior of real time tasks compared to the behavior of non-real-time tasks, as well as the behavior of real time tasks in different tests scenarios, in order to evaluate the CPU usage.

Figure 7 represents the execution of scenario 2, with two real time and two non-real time tasks. It is possible to observe that the blue and red lines have a higher priority than others, and do not have any standard that relates both, which are the non-real time tasks. This shows that the real time tasks have advantage against the non-real time tasks, which is an expected behavior.

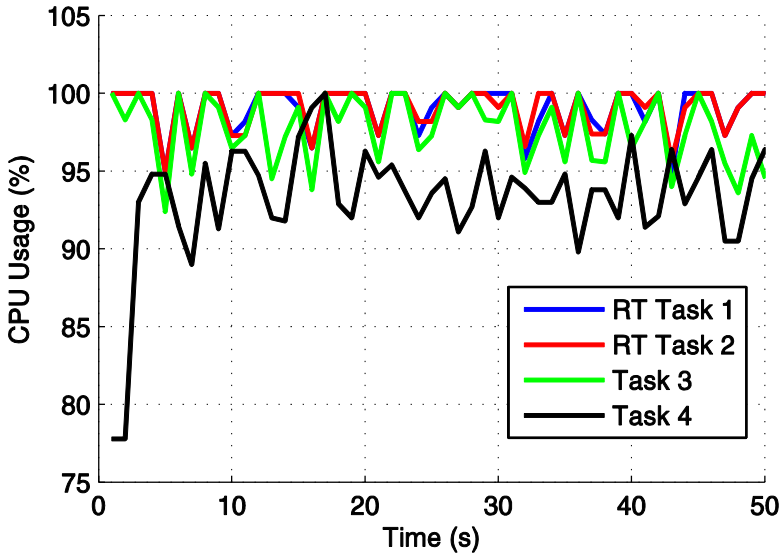

Figure 7: Behavior of real-time and non-real-time tasks running in scenario 4

The final analysis, and no less important, is the variation of the CPU usage of a real time task in each test scenario. This test aims to show the impact of system overhead while running a real time task. To this, a real time task was separately analyzed, with the processing load varying in each scenario.

Figure 8, at the topside, shows the CPU utilization of two different real time tasks, the blue line in scenario 1 and the red line in scenario 2. Both tasks are running on a non-overload environment and having a CPU usage profile quite stable.

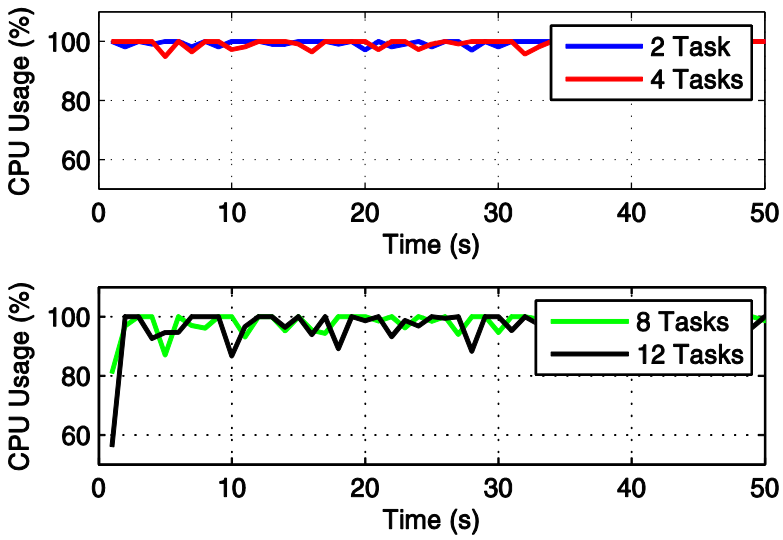

Figure 8: Behavior of a real-time task in each scenario.

Figure 8, at the downside, does the same analysis of the previous graph, but now for the scenario 3 and 4, through the green and black lines, respectively. Here the CPU already has overload affecting the CPU usage for the tasks and consequently impacting the performance of applications running, resulting in missed deadlines.

CONCLUSIONS

After the analysis highlighted by the graphs it is possible to see that the more the number of processes running concurrently with the real time process, the more deadlines were missed. The RBP supports more than one application running at the same time, as can be seen by the graphics. For better performance from the application, it is interesting to assign it to the real time requirements, thus the task will have advantage in relation to the others.

The RBP board is an inexpensive hardware and easily accessible, and has met the requirements set by the application until a certain limit, as presented by the acquired results. The POI detection algorithm uses less resource than is offered by the hardware, which allows adding other applications on the same board, since it meets the bottleneck on the number of cases tested in this work. So is possible to conclude that is feasible to use a cheap COTS processing board such as RBP in a mini UAV to process CPU intense software such as the image processing one tested in this work. Directions for future work are manly in further analyzing the composition of the tasks able to run in this kind of COTS processing board, such as navigation planning algorithms and other image processing algorithms. Another important study to be done is the energy consumption analyzes, especially in overload system scenarios.

ACKNOWLEGMENT

The authors thank to the Swedish-Brazilian Research and Innovation Centre – CISB and to the Brazilian company SKYDRONES for the provided support to develop this research.

REFERENCES

[17] Altera, Available at: https://www.altera.com/products/fpga/overview.html, Accessed in 2015-10-20.

[14] ArchLinux, Available at: https://www.archlinux.org, Accessed in 2015-11-20.

[3] B. Edwards and J. Archibald and W. Fife and D. J. Lee, 2007, *A vision system for precision MAV targeted landing*, In Computational Intelligence in Robotics and Automation, CIRA. International Symposium on (pp. 125-130).

[2] C. C. Chiu and C. T. Lo, 2011, *Vision-only automatic flight control for small UAVs*, IEEE Transactions on Vehicular Technology, 60(6), 2425-2437.

[6] D. Hulens and T. Goedemé and J. Verbeke, 2015, *How to choose the best embedded processing platform for on-board UAV image processing?*, in Proceedings VISAPP on 1-10.

[9] I. F. Mondragón and P. Campoy and J. F. Correa and L. Mejias, 2007, *Visual model feature tracking for UAV control*, In Intelligent Signal Processing, WISP. IEEE International Symposium on (pp. 1-6).

[5] J. Faigl and T. Krajnik and J. Chudoba and L. Preucil and M. Saska, 2013, *Low-cost embedded system for relative localization in robotic swarms*, In Robotics and Automation, ICRA. IEEE International Conference on (pp. 993-998).

[11] L. P. Behnck, 2014, *Controle de missão de voo de veículo aéreo não-tripulado.* Avaliable in: http://www.lume.ufrgs.br/handle/10183/105055.

[15] Linux-libre-rt, Available at: https://aur.archlinux.org/packages/linux-libre-rt, Accessed in 2015-11-20.

[13] R. W. Sinnott, 1984, *Virtues of the Haversine skytel*, v. 68, p. 158.

[16]Rasberry Pi 2, Available at: https://www.raspberrypi.org/products/raspberry-pi-2-model-b, Accessed in: 2015-10-20.

[4] S. Ehsan and K. D. McDonald-Maier, 2009, *On-board vision processing for small UAVs: Time to rethink strategy*, In Adaptive Hardware and Systems, AHS. NASA/ESA Conference on (pp. 75-81).

[1] S. Azrad and F. Kendoul and K. Nonami, 2010, *Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm*, Journal of System Design and Dynamics, 4(2), 255-268.

[12] S. P. Brooks and B. J. Morgan, 1995, *Optimization using simulated annealing*. The Statistician, 241-257.

[7] T. Krajník and M. Nitsche and S. Pedre and L. Pˇreuˇcil and M. E. Mejail, 2012, *A simple visual navigation system for an UAV*, In Systems, Signals and Devices (SSD). 9th International Multi-Conference on (pp. 1-6).

[8] T. Krajník and J. Šváb and S. Pedre and P. Cížek and L. Preucil, 2014, *FPGA based module for SURF extraction*, In Machine vision and applications, 25(3), 787-800.

[10] W. S. Ooi and C. P. LIM, 2009, *Fusion of colour and texture features in image segmentation: an empirical study*, The Imaging Science Journal, v. 57, n. 1, p. 8-18.