An Embedded System for Aerial Image Processing from Unmanned Aerial Vehicles

Jonas Fernandes da Silva, Alisson V. Brito, José Antônio Gomes de Lima
Centro de Informática
Universidade Federal da Paraíba (UFPB)
Cidade Universitária - João Pessoa - PB - Brasil
Email: jonas@ccae.ufpb.br, {alisson, jose}@ci.ufpb.br

Helder Nogueira de Moura
REINTEGRIS
Cidade Universitária - João Pessoa - PB - Brasil
helder@reintegris.com.br

Abstract- **This paper proposes solution capable to process aerial images from UAVs to identify failures in plantations and makes a comparison of the system running on light sized computers and low power computing platforms. An algorithm was developed based on *watersheds* using OpenCV library. The solution was embedded on X86 architecture *(AlteraDE2i-150)* and Intel Edison boards as well as on ARM architecture (Raspberry Pi 2). The results show that the proposed system is a cost-effective solution for the problem of fault identification in plantations, and can be embedded in UAVs for processing images in real time.**

Keywords*- *Embedded systems; UAV, Watersheds;

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) were initially developed only for military applications, but now are increasingly being used in various tasks, especially in environments that offer a risk to life or where human intervention is dangerous. Despite the reduction of investment in UAV projects applied to national defense in leading countries such as Germany, France and the USA, resulting in a negative impact on the growth of the UAV industry [1], the market is promising and fragmented throughout the world with a significant number of small and medium-sized companies.

In large areas of plantations, UAVs assist in monitoring using aerial imaging and its usage is more and more often due to savings generated by these devices, which eliminate displacement of experts and reduces the response time to identify problems. Its main advantage is the ability to travel a considerable distance with safety, reduced time and at lower cost compared, for example, using a helicopter. Some environmental applications are described in [2], which uses UAVs to detect infestation of *Huanglongbingv* (HLB), a destructive disease found in citrus agriculture in Brazil; and in [3] where UAV is used to monitor an agricultural region to identify invasive weeds and abnormalities in irrigation. Other studies report the use of UAVs for collecting digital aerial imagery to detect unauthorized changes to the land, such as deforestation [4], [5]. In most of these cases, the identification does not occur automatically during the flight, requiring human intervention for later analysis of the image, either by an expert or software, as in [6], where digital aerial images are obtained from a camera attached to a UAV in order to create topographic mapping, in which aerial images are available only after a few hours for analysis.

Thus, this work aims to develop and embed an algorithm for image processing system capable of automatically identifying failures in plantations, represented by high-contrast areas on the image, and calculate an approximate percentage of these areas in various types of vegetation. For that it uses light-sized and low-power computers based on x86 and ARM architectures. This work is relevant to present a solution that perform dedicated tasks quickly, consuming low power and with low cost. At same time, we aim to contribute to the scientific community through the performance comparison between computing platforms presenting different options for other interested research groups.

The proposed solution can be used to identify pests, problems on the ground or boundaries in plantations. The embedded system chosen must be coupled to an UAV (developed in future works) where the agricultural monitoring should be performed in real time during flight, eliminating the need to download the images or landing, enabling automatic diagnostics by UAV itself.

The Section II presents image-processing techniques. Section III deals with the embedded systems used in this study and describe software and hardware involved. Section IV presents the experiments performed in the laboratory, describing the used methods and achieved results. Finally, section V brings final considerations.

II. MATERIALS AND METHODS

For aerial image processing, some techniques were used. The thresholding technique consists in grouping the gray levels and is described in [7], so that the image formed is composed of light objects on a dark background so that the pixels of the object and background are grouped into two dominant groups.

The image expansion step is to remove small white noises, filling holes in the image. The removal of residual noise is based on the relative position between neighboring pixels, and aims to unconnected pixels union of the same intensity in the nearby areas, eliminating noise near the regions of interest. The image segmentation step is the separation of an input image into regions, in general, the more accurate targeting, the greater the chances of success in identifying regions of interest of the image [8]. Over the years, various image segmentation algorithms were developed. The technique used in this study is based on *watersheds,* which is a popular image segmentation algorithm and is based on the view of an image in three dimensions, similar to a topography where there are two spatial coordinates x and y, and the height is equal to the gray level of the corresponding pixel [9]. In this topographic interpretation, a watershed can be imagined, where light and dark spots represent hills and valleys in a landscape [10]. Water virtually covers the lower areas and the highest ones are highlighted. These limits correspond to the dividing lines of the dam, which are boundaries extracted by the segmentation algorithm.

The application was developed using the *OpenCV (Open Source Computer Vision*[1]) library, which is an open-source library for computing vision. It is free and widely used by companies, research groups and government agencies, with support for various operating systems, for example, Windows, Linux, MacOS and Android. Its primary focus is the processing of real-time images [11], i.e., cites security monitoring, analysis of biomedical images and even the use by UAVs [12].

In this work, the developed application is tested in embedded platforms evaluate the more suitable one. The tested platforms are Altera *DE2i-150,* Intel Edison and Raspberry Pi 2. These platforms promise high performance and low power consumption.

The *Altera DE2i-150* embedded platform is manufactured by Terasic®. It is 250x170mm with 800g, consisting of two main blocks: the first for general processing through the Intel® Atom N2600 1.6GHz based on x86 architecture. The second block ensures high flexibility for projects through an *Altera Cyclone IV* FPGA. It has 2GB DDR3 main memory, integrated GPU, 64GB SSD storage, a VGA interface, HDMI, Wireless, Bluetooth, audio and Ethernet.

Intel Edison platform main characteristic is its small size, with 60x28mm, including the communication interface, and only 16g weight. It features an Intel Atom Dual Core 500MHz processor based on the x86 architecture, 1GB DDR3 main memory and internal storage of 4GB. The Intel Edison Breakout Board has Bluetooth communication interfaces, wi-fi and USB.

Raspberry Pi 2 is based on ARM architecture and is developed by Raspberry Pi Foundation. Its small size with dimensions of 85x56mm and 42g weight allows use in various applications. It has ARM7 900MHz Quad Core processor, 1GB of main memory, and data storage support via a micro SD slot. The platform includes USB communication interfaces, HDMI and Ethernet, in addition to connections for camera couplings and display. It also has General Purpose Input / Output (GPIO) ports responsible for input and output of digital communication through a set of 40 pins, allowing connection of sensors or other devices.

III. EXPERIMENTS

The images were acquired by an UAV, but processing was further conducted in the laboratory by the embedded systems. In future work, the system will be embedded in an UAV for direct image capture and processing in real time.

The images were obtained by a remotely controlled quadcopter model DJI Phatom FC-40 over an area of environmental conservation and over an agricultural area. The quadcopter has a camera attached with 1280x720 pixels resolution. A group of fifteen images were selected and added to the pictures database for power and performance analysis.

The *watershed* technique presented in section II was used for image processing. It is divided into five steps: image acquisition, conversion to gray scale, thresholding, image dilation, distance transformation and a second thresholding step.

The second step is the conversion of the original image (Fig. 3a) to grayscale (Fig. 3b). This facilitates the thresholding executed in the next step. The grayscale operation is done by *cvtColor* from *OpenCV*. Following, the global thresholding technique is applied.

Proposed algorithm

```
InputImage <- "inputImage.jpg"
OutputImage<- "outputImage.jpg"
FUNCTION BORDER (InputImage)
  Border<- FIND-BORDER (InputImage)
  FOR (k = 0; k < Border.Size; k ++)
     borderImage<- DrawContour (Border)
  FOR (x = 0; x < Border.Size; x ++)
    FOR (y = 0; y <Border.Size; y ++)
         Totalx<- Totalx + Border (XY).x
         Totaly <- Totaly + Border (XY).y
    END FOR
  AvgX <- Totalx / x
  AvgY <- Totaly / y
END FUNCTION
FUNCTION AREA(InputImage )
  FOR (y = 0; y<InputImage.Lines; y++)
    FOR (x = 0; x <InputImage.Columns; x++)
         IF ImagemOrigem (y, x) == 0
         IncBlack
 ELSE
         IncWhite
 END FOR
END FUNCTION
MAIN
   Image1 <- INPUT (InputImage)
   Image2 <- GRAY_SCALE (Image1)
   Image3 <- thresholding(Image2)
   Image4 <- DILATE (Image3)
   Image4 <- NORMALIZE (Image4)
   Image5 <- DISTANCE_TRANSFORM (Image4)
   Image6 <- thresholding(Image5 )
   BorderImg<- BORDER(Image6)
   AREA (Imagem6)
SAVE_FILE (OutputImage, Image6)
END
```

[1] http://opencv.org

Due to its simplicity, low computational power is demanded. The thresholding function used in our experiments value of $T = 200$. That means pixels with values lower than 200 are set to zero, otherwise to 255. That results into a process called *binarization* (Fig. *1c).* The Threshold parameter T should be defined for each type of vegetation, due to intrinsic characteristics of each group of images and to other factors such as light. Here, the parameter T was defined from tests and observations.

The next step refers to dilation of the image, through the use of function, *dilate* with parameter equal to three iterations (Fig. 1d), whose goal is to join disconnected pixels of same intensity in nearby areas. This is used for further processing of the residual noise generated from the thresholding step.

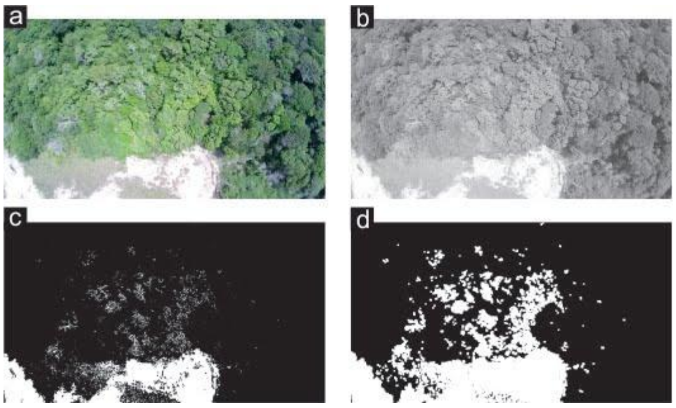

Fig. 1 - Original image (a),gray scale (b), thresholding (c) and dilation (d)

The fourth step utilizes the function *Distance Transform* of the dilated image. From a 3x3 matrix various points are distributed according to the calculation of the distance from the array center to the edge, resulting in a gradient image (Fig. 4a). Finally, a further step of thresholding is performed to filter the gradient levels generated from the previous step, highlighting only the regions of interest (Fig. 2b). For this step a threshold $T = 0.35$ was chosen since it brought satisfactory previous results.

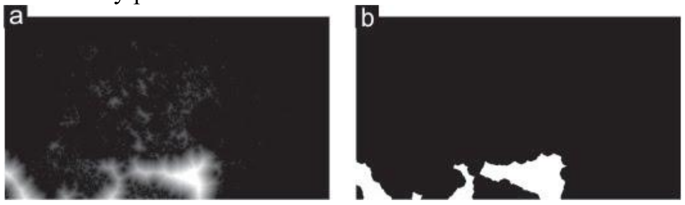

Fig. 2 – Images after application of distance transformation (a) and thresholding (b)

Further, two other functions are used, the first uses OpenCV *findBorders* that provides all the coordinates of the pixels of the image boundary region. From that, the average of the coordinates x and y are calculated in order to estimate a central position of the failure spot on plantation. Thus, the function *drawBorders* draws the boundary of the spot, as shown in Fig. 3. A second function is intended to calculate the total area of spots. As the image is binarized, with pixel values of 0 or 255, the failure percentage is found by calculating the frequency of occurrence of these values in the image.

For example, the image shown in Fig. 2b, has 921,600 pixels (considering 1280x720 resolution), from which 53,062 pixels have value 255. This means a failure of 5.76% of the image area. Furthermore, the system grouped the spots into four areas presented in Fig. 3, marked as A1, A2, A3 and A4.

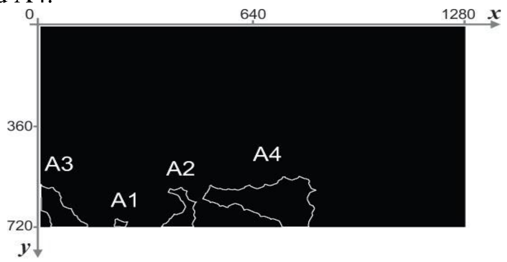

Fig. 3 -Detected failure areas

The performance of the system running in different architectures was evaluated using the execution time. The results are presented in Table I. The same input images were used for the three platforms and evaluated. As the used functions are exact, the same output images were obtained.

Table II – Execution time running in embedded platforms

	Embedded Systems		
	DE2i-150 (sec.)	***Intel Edison (sec.)***	***Raspberry Pi 2 (sec.)***
Image 1	0.484	1.203	0.899
Image 2	0.505	1.265	0.896
Image 3	0.508	1.301	0.917
Image 4	0.499	1.274	0.869
Average (15 images)	**0.507**	**1.278**	**0.925**

In order to reduce the table size, the results for only 4 images are presented in Table I. The average execution time for all 15 images are presented in the last line of the table. The results obtained were only based on code execution, disregarding the compilation and image presenting times. As expected, Altera *DE2i-150* platform achieved better performance than the other two platforms. Using it, the average execution time was about 500 milliseconds. Using Intel Edison images were processed on average in about 1.3 seconds, which is about 2.5 times slower than DE2i-150. The Raspberry Pi 2 need about 920 milliseconds to process the images on average, which is 1.8 times slower than DE2i-150. This result could be improved, as micro SD card number 4 was used, with throughput of 4 MB/sec, which is lower than the others that use SSD internal storage. A faster micro SD card might improve the results for Raspberry Pi 2. The total time to process all 15 images by platforms DE2i-150, Intel Edison, Raspberry Pi 2 was approximately 7.6, 20.7 e 16 seconds, respectively.

Regarding energy consumption, the current was measured and registered by an external Arduino-based circuit. A script was configured to collect the current at each

2 seconds. From the tests, it was observed that Intel Edison consumed lower power than the other two, consuming on average 120mA. This is about 8.2 times less than the highest consumption platform, *DE2i-150,* and 2.4 times less than Raspberry Pi 2. Altera DE2i-150 and Raspberry Pi 2 consumed on average about 990mA and 290mA, respectively. A chart of power consumption is shown in Fig 6.

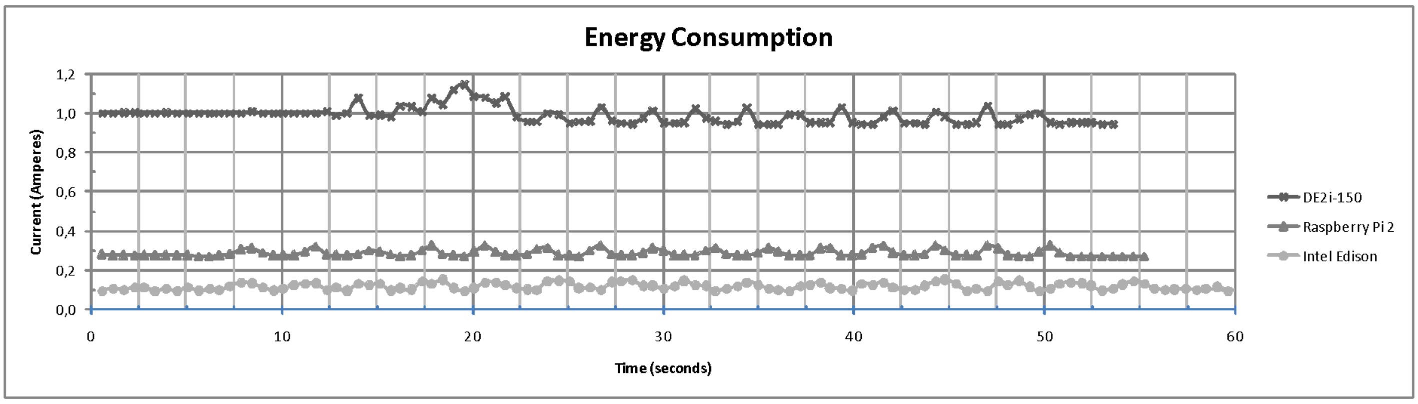

Fig. 6- Power consumption of embedded platforms during image processing

IV. Final considerations

In this work, an image processing system was proposed. The achieved results seem satisfactory, regarding the goal of automatic identification of failure areas in plantations by Unmanned Vehicles (UAVs). The system could be embedded in small size and lightweight platforms, with satisfactory execution time and lower power consumption.

Despite the high performance observed by Altera *DE2i-150* and the variety of available communication interfaces, this is not suitable for the proposed system, due to its large size and high power consumption, impeding its use attached to the UAV used in this work. Intel Edison platforms (x86) and Raspberry Pi 2 (ARM) are two possible platforms to be embedded in an UAV, due to their small size, light weight and low power consumption. The first presented several advantages, such as smaller size, lighter weight and lower power consumption, as well as having wireless communication, being a strong candidate to be coupled to an UAV, despite having a slightly lower performance compared to the Raspberry Pi 2.

On the other hand, the Raspberry Pi 2 platform, in addition to superior performance has other interesting advantages, such as greater communication possibilities. It allows an easy configuration, allows GUI and flexibility to the inclusion of new devices via GPIO extension.

In this scenario, the most suitable platform to be used will depend on the requirements of each application. Considering this project, weight, size and power consumption are more important than all other features, as the goal is to embed in a UAV. Also, as the difference in performance was only 350 milliseconds, Intel Edison platform should be chosen used in this project.

References

[1] C. Drubin, "The Global UAV Market 2015-2025," *Microw.*J.,vol. 58, no. 3, pp. 53-54, Mar. 2015.

[2] LAC Jorge, ZN Brandão, and RY Inamasu, "Insights and recommendations of use of UAV platforms in precision agriculture in Brazil," *Remote Sens. Agric. Ecosyst. Hydrol.*XVI,vol. 9239, no. 2004, p. 923 911, 2014.

[3] S. . Herwitz, L.. Johnson, S.. Dunagan, R.. Higgins, D.. Sullivan, J. Zheng, B.. Lobitz, J.. Leung, B.. Gallmeyer, M. Aoyagi, R.. Slye, and J.. Brass, "Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support," *Comput. Electron.*Agric.,Vol. 44, no. 1, pp. 49-61, Jul. 2004.

[4] LF Felizardo, RL Mota, EH Shiguemori, MT Neves, AB Ramos and F. Mora-Camino, "Using ANN and terrain for UAV surveillance," in *13th International Conference on Intelligent Hybrid Systems (HIS*2013),2013 pp. 1-5.

[5] LF Felizardo, RL Mota, EH Shiguemori, MT Neves, AB Ramos and F. Mora-Camino, "Expanding Small UAV Capabilities with ANN," in *Second International Conference on Image Information Processing (ICIIP 2013)*,2013, pp. 516-520.

[6] A. Ahmad, KN Tahar, WS Udin, KA Hashim, N. Darwin, Hafis M., M. Room, NAM Hamid, Nurul Farhah Adul Hamid Azhar, and SM Azmi, "Digital Aerial Imagery of Unmanned Aerial Vehicle for Various Applications," in *IEEE International Conference on Control System, Computing and*Engineering,2013, pp. 535-540.

[7] PL Rosin and E. Ioannidis, "Evaluation of global image thresholding for change detection," *Pattern Recognit.*Lett.,Vol. 24, no. 14, pp. 2345-2356, Oct. 2003.

[8] RC RC Gonzalez and Woods, *Digital Image*Processing,3rd$^{ed.}$ São Paulo, 2010.

[9] A. A. Bieniek and Moga, "An effcient watershed algorithm based on connected components," *Patter*Recognit.,Vol. 33, pp. 907-916, 2000.

[10] A. Bleau and LJ Leon, "Watershed-Based Segmentation and Region Merging," *Comput. Vis. Image*underst.,Vol. 77, no. 3, pp. 317-370, Mar. 2000.

[11] TD Prasanthi, K. Rajasekhar, T. V Janardhana, and BV V Satyanarayana, "Design Of ARM based Face Recognition System using Open CV Library," vol. 1, no. 9, pp. 233-240, 2012.

[12] G. Bradski and A. Kaebler, *Learming*OpenCV,First Edit. 2008.