Parallel Programming to Analyze Crop-Health in Real-Time

Angie Natalia Molina Muñoz, Ansgar Brunn, Javier Chaparro and Alexander Pérez

Summary
Smart farming takes place by using recent UAV-technology for agricultural profitability. Multiple companies and research centers are working on realizing real scenarious. But there are only restricted experiences about how to process the data faster and better in real-time. Autonomous systems that can decide instanteously after receiving sensor information could increase the farming efficiency. The agroindustry research group from the Escuela Colombiana de Ingeniería and Universidad de Ciencias Aplicadas y Ambientales en Bogota-Colombia in alliance with the University of Applied Sciences Wuerzburg-Schweinfurt are committed to that idea.
This project uses an NVIDIA Jetson embedded board to process agricultural information in order to find abnormalities. In this article we prove that this board is able to make the calculations for a vegetation index in real-time from multispectral images.

Zusammenfassung
Die UAV-Technologie wird zunehmend zur Gewinnoptimierung in der Landwirtschaft im Rahmen des Smart Farmings eingesetzt. Eine Vielzahl von Unternehmen und Forschungseinrichtungen arbeiten an der Realisierung an aktuellen Fragestellungen. Nur begrenzte Kenntnisse sind jedoch zu der Frage vorhanden, wie die Daten schneller und in Echtzeit verarbeitet werden können. Autonome Systeme, die im Moment der Sensoraufnahme Entscheidungen treffen können, bieten ein großes Potenzial der Effizienzsteigerung für die Landwirtschaft. Die Forschungsgruppe »Agroindustrie« an der Escuela Colombiana de ingenieria und die Hochschule für angewandte Wissenschaften und Umwelt Bogota-Colombia arbeiten in Zusammenarbeit mit der Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt an dieser Idee.
In diesem Projekt wird eine NVIDIA Jetson Grafikkarte zur Verarbeitung von für die Landwirtschaft relevanter Information eingesetzt, um Störungen in der Vegetation zu finden. Dieser Beitrag zeigt, dass die Berechnung eines Vegetationsindexes aus Multispektraldaten in Echtzeit auf der Grafikkarte möglich ist.

Keywords: Parallel programming, plants health, NDVI, OpenCV, CUDA, NVIDIA Jetson.

1 Introduction

The world changes fast, the population grows, the water scarcity increases, the climate warms up and the farming conditions are in danger. In 2050 the world population will be about 9.7 billion (European Commission 2018). This implies a giant challenge for organizations (including the European Commission) committed to guarantee food production using sustainable resources and promoting research activities that respond to the farmers' needs.

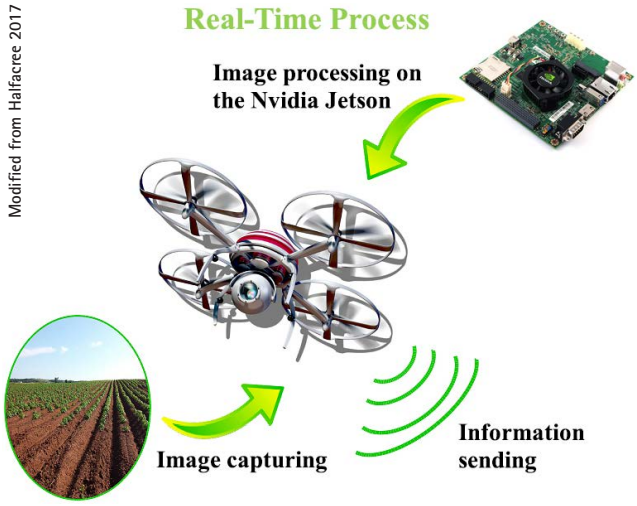

Fig. 1: General structure of the real-time analyses process

A common approach to work on that is the use of remote sensing techniques to determine whether the crop is healthy. This discipline has been working along decades to create methods to acquire information from images taken from satellites or airbornes. Right now, this kind of information is easily obtained from images taken from UAV because this technology has become popular. Aasen et al. (2018) gives an overview on recent developments. Specialized software is able to stitch a large amount of photographs to obtain a high resolution image and, if that image comes from a multi-spectral camera, it could create maps with a specific index, like the normalized differences vegetation index (NDVI).

This kind of software works offline, it means the user needs to wait until the photographs were taken and the computer has processed the huge amount of photographs. The delay between the moment the photographs are captured and that the user really has the results to be able to make a decision about the crops' health could be more than two days.

The research group of Agroindustry at the Escuela Colombiana de Ingeniería Julio Garavito and Universidad de Ciencias Aplicadas y Ambientales en Bogotá-Colombia in alliance with the University of Applied Sciences Wuerzburg-Schweinfurt are aware of that and work on using drones for abnormalities detection in potato crops.

  DOI 10.12902/zfv-0249-2019

The objective is that a drone can identify in the air if an area is sick by using information from multi-spectral cameras, as farmers would do using their eyes but with the extra information provided by the spectral waves and a NVIDIA Jetson TK1 as a brain to determine the status of the plants as you can see at Fig. 1. The main idea is using the NVIDIA on-board to process farming information obtained through sensors, such as multi-spectral cameras, IMU, GNSS among others, sending that information or taking autonomous decisions based on the data.

That goal requires cooperation and a break down of the general idea into work pieces. Hence, a first step is to test the feasibility of using the GPU-NVIDIA embedded board for processing real-time images obtained by a multi-spectral camera for the detection of abnormalities in potatoes crops in cooperation. It focuses on first developments with the NVIDIA functionalities, which allow to create further developments, taking advantage of machine learning, and figuring out the speed of the image processing on the NVIDIA Jetson TK1 for the detection of abnormalities and whether it is as fast as to be considered real time.

2 Methods

2.1 Key devices

2.1.1 The multi-spectral sensor

The parrot sequoia is a multi-spectral sensor (Parrot Sequoia Team 2018). Its low weight enables it to integrate with a large variety of drones. The multi-spectral camera captures the amount of light reflected by the plants in four bandwidth waves: green, red, red edge and near infrared, which are invisible to the human eye. It also contains an RGB camera, global navigation satellite system (GNSS), inertial measuring unit (IMU), internal storage and wireless local area network connection (WLAN). The capture features can be configured using an app on a computer, tablet or smartphone joined to the WLAN.

On the app, it is possible to calibrate the camera and set the triggering mode: auto trigger captures images based on a defined overlap, distance trigger take the pictures every distance predefined between 5 m and 1 km, and time-lapse triggers as fast as 0.5 frames per second on the RGB images and 1 frame per second for the multi-spectral channels (Pix4D Team 2018) (cf. Fig. 2 and Fig. 3).

In this work, we focus on the images of the RED and the NIR channel.

2.1.2 The processing board

The NVIDIA Jetson is an embedded board that offers computational capabilities integrating a central processing unit (CPU), graphic processing unit (GPU) and sockets

Fig. 2: RGB image of the scene of crop agriculture

Fig. 3: Multi-spectral datasets of the scene. The inputs displayed in the windows are the NIR and the RED channel.

to connect a large number of different peripheral devices such as a camera, a keyboard, a mouse, a Kinect, digital and analog outputs and inputs, supporting communication protocols as Ethernet, serial RS232 and I2C among others. On their web page NVIDIA states: “NVIDIA Jetson is the world’s principal AI computing platform for GPU-accelerated parallel processing based on CUDA in mobile embedded systems. Ideal for deep learning and computer vision embedded projects.” (NVIDIA Corporation 2018). In spite of its capabilities it has a low power consumption (from 1.5 W to 45 W), which varies depending on the connected peripherals. These features and its low weight enable the board to be easily mounted on a UAV. Massive parallel programming of the GPU provides its processing speed.

2.2 Normalized Difference Vegetation Index

Ending the 1970s, scientists found a relationship between the plants productivity and the radiation absorbed by them. That association resulted in the Normalized Difference Vegetation Index (NDVI) (cf. Albertz 2001 or Weier and Herring 2018). Researchers at NASA have collected satellite NDVI information to measure the productivity status from different regions and compare them yearly. Both healthy and unhealthy plants reflect the near infrared light (NIR) but just healthy plants absorb a high

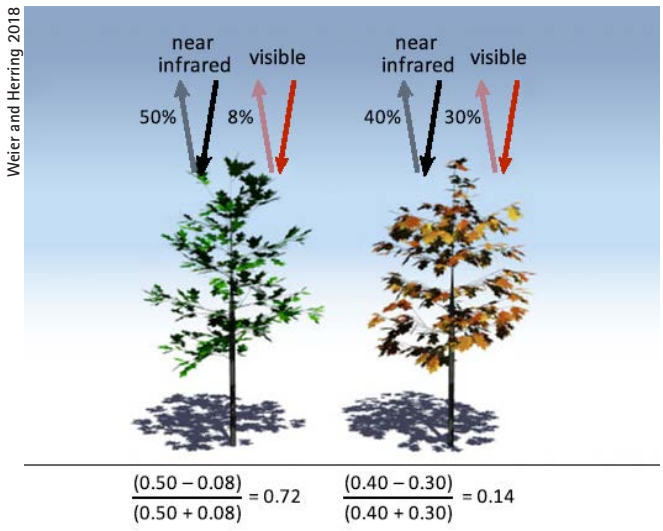

Fig. 4: Both healthy and unhealthy plants reflect the near infrared waves. With the NDVI we can identify a vegetation element because just unhealthy reflect a high amount of red light.

amount of red light (RED) (Fig. 4). The index is calculated using the equation:

$$NDVI = \frac{NIR - RED}{NIR + RED} \quad (1)$$

If the near infrared reflectance (NIR) increases as it does with vegetation and the red light (RED) reflectance rises as well, that means they are not using the red lights for the photosynthesis process, they are less healthy (Fig. 5). If we subtract the red light from the near infrared, the numerator will be significantly less than the denominator and we will obtain values close to 0. Typically values are between 0 and a threshold t_1, which is season dependent. In this scenario, a threshold $t_1 = 0.33$ has been found to be appropriate. If the near infrared reflectance increases, as it does with vegetation, and the red light is almost not reflected, similar values in the numerator and denominator are obtained. The result is an index close to 1. The values from t_1 to $t_2 = 2t_1$ are considered as moderately healthy plants and values from t_2 to 1 as very healthy plants. Finally, if the near infrared reflectance is less than red light, it is not vegetation at all and it could be in most of the cases considered as soil.

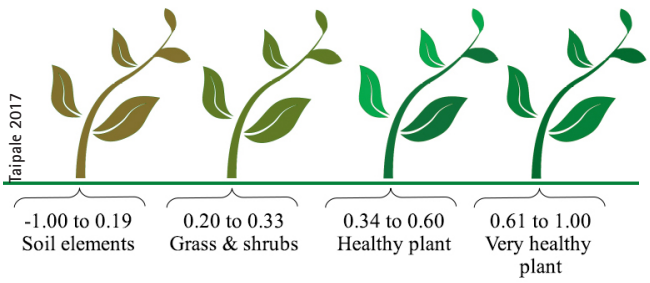

Fig. 5: Four levels of health and their NDVI range

2.3 The general process structure

The images were captured by members of the Agroindustry research group from the Escuela Colombiana de Ingeniería Julio Garavito and the Universidad the Ciencias Aplicadas in Bogotá and Boyacá, Colombia. For the first tests, we read the input images from USB memory, which could be replaced by the USB camera cable. Then, we calculated the NDVI index on the Jetson. We implemented two programs, one just using OpenCV library functions, that are executed just on the CPU, and a second version adding CUDA functionalities to take advantage from the GPU speed processing, both of them are written in C++.

The basic algorithm for both programs is the same: load the input images, create the output and extra variables, calculate the NDVI, measure the process time and print the results. The specific details vary with the specific utilities used, the Fig. 6 presents the two code structures

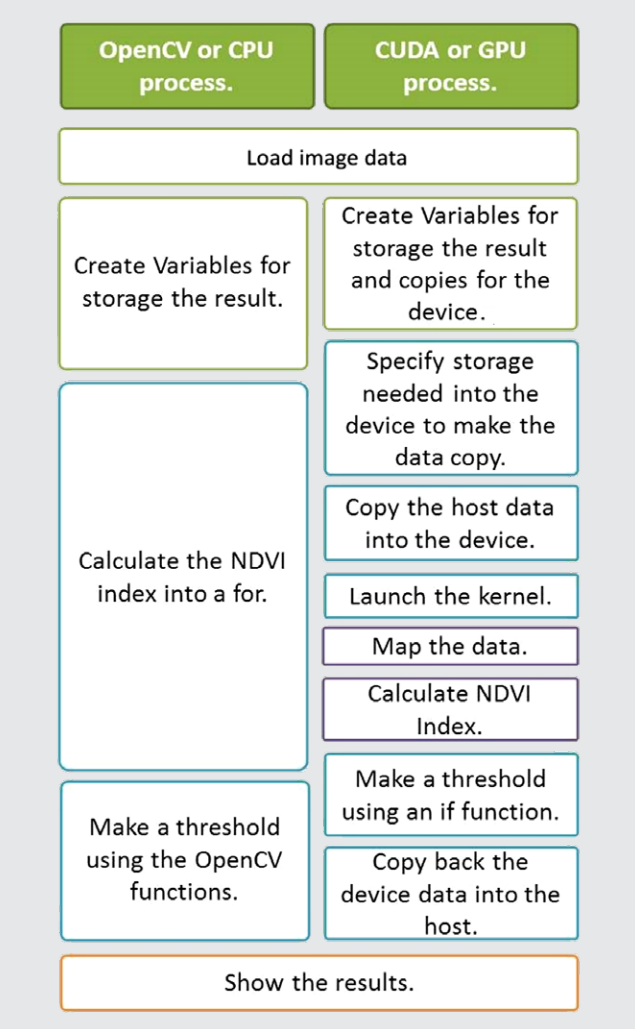

Fig. 6: Parallel CPU and GPU resume process description: The GPU process implies more complex structure and a harder work during the programming. The green frames correspond to the initialization steps, the blue frames contain the main CPU functions, the purple frames represent the functions launched on the kernel and finally the yellow frame symbolizes the output phase.

simultaneously. The GPU process implies a more complex structure and a harder work during the programming stage, because of additional necessary commands for the memory managment.

2.4 The OpenCV or CPU process structure

"OpenCV is an open source computer vision and machine learning software library" (OpenCV 2018). It provides functions for computer vision, augmented reality, gesture recognition developments, "more than 2500 algorithms" (OpenCV 2018). You can create from simplest to most complex applications. Due to its free license, it is currently used by companies, research groups and governmental institutions. A special OpenCV version is available on the Jetson board within the JetPack, the software installation package for the operating system, OpenCV4Tegra, which is a optimized version for GPU uses.

The first step in the OpenCV or CPU process is to initialize the inputs, outputs and auxiliary variables: we use the getTickCount() function to measure the time, read the NIR, RED and RGB images from the USB folder and create the result images with the same size as the input images, using one channel and defining a float data type since we will store the NDVI index, which is a decimal value. The second step is to calculate the NDVI index: we walk through the image using two for functions calculating pixel per pixel the NDVI value accessing the pixel value with the .ptr function, specifying the data type, the position and the channel; the third step is to classify healthy and unhealthy values: we use the threshold function to categorize the pixels as it is showing in the following equation establishing as threshold value t_1 (cf. sec. 2.2). Values lower than the limit are determined as unhealthy vegetation and soil material. This result will be displayed with a 0 intensity (black), and values bigger than the limit are settled as healthy plants (binarization). Those will be displayed with a 255 intensity (white).

$$st(x,y) = \begin{cases} 255 & \text{if result } (x,y) > 0.33 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

2.5 The CUDA or GPU process structure

"CUDA is a parallel computing platform and programming model" (NVDIA Corporation 2018) that allows access to the GPU cores resources in order to accelerate deep learning, analytics, and engineering applications, e. g. image processing. Introductory examples can be found in Sanders and Kandrot (2015) and Storti and Yurtoglu (2015).

CUDA supports the most common languages C/C++, Fortran, Phyton, among others. The CPU, which is called the host, has a sequential code. When a part of the code consists in processing a huge amount of data, the host asks the device, the GPU, to process that data in its thousands of threads. The programming model allows to program both processors on the same block code. The NVIDIA compiler divides the work for each one depending on the instructions indicated by applying a few keywords on the host. Those keywords let the developer run one piece of the code in one thread. The amount of threads depends on the capabilities of the GPU. CUDA assumes the device is a co-processor to the host, each one having their own separate memories.

The GPU programming follows a strict structure, since the execution success depends on it. In short, it requires on the CPU side: allocating the enough storage on the GPU, moving the original data from the CPU memory to the GPU memory; launching the kernel – to ask the GPU to work; copying the result data on the CPU back from the GPU. CUDA GPUs have many parallel processors grouped into Streaming Multiprocessors, or SMs. Each SM can run multiple thread blocks. To take full advantage of all those threads, it is necessary to launch the kernel with multiple thread blocks. Within the kernel every thread is responsible for a small piece of data. It knows its own data using again keywords provided by CUDA: gridDim.x contains the number of blocks in the grid (blocks of parallel threads), blockIdx.x indicates the current thread block index in the grid, blockDim.x specifies the threads amount in the thread block, and threadIdx.x represents the thread index at the thread block (Harris 2017). The .x represents the x dimension, actually a grid is able to have 2Dims, a block of threads 3Dims and a thread 3Dims. You can imagine the data structured in a big cube, which is composed by "smaller" cubes, likewise they are composed by tiny cubes that contain single information with 3Dim data. The GPU programming has a strict structure for mapping the data to the GPU memory and vice versa, which is a process where each thread also obtains a single index based on its location in that big cube. The developer decides on the number of dimensions that are needed.

The NDVI calculation on the GPU process is briefly desribed in the following. We start on the CPU: reading the images from the USB folder and declaring the outputs for the host data, basically as we did during the OpenCV process, but this time creating a device copy of all of them. Then, the storage needed for using the data inside of the GPU is calculated. Afterwards, the space for the device image copies is allocated using the CUDA function cudaMalloc. It is necessary to specify the data type. The input data is copied from the host to the device using the cudaMemcpy function. In this step we use also the Mat.ptr() function to copy the images. To define how the work should be distributed in our big cube, we declare a 2Dim variable block indicating that we use 16 blocks. In the variable grid we store the quantity of threads needed to cover the complete image. We create two GPU process codings, they just differ on this step: changing the

amount of blocks from (16,16) to (1,1), which means just one block. The nvdi kernel is launced (we called our kernel function ndvi since it calculates the NDVI index) like a standard C++ function is called, indicating between brackets the variables given to the function, but additionally specifying inside of three angle brackets the information of the data distribution – the launch instruction of our kernel looks something like

```
ndvi<<<grid,block>>>(var1, var2, var3, ...);
```

advancing we jump inside our kernel function where we start: mapping the data from 2Dim to just one index, we calculate first the start position of every thread block (block index times the block size: blockIdx.*blockDim) and adding the thread's index within the block (threadIdx.), in resume

```
myIndex = blockIdx*blockDim. + threadIdx
```

(Harris 2017). In the next step, we calculate the NDVI as a single calculation of just one pixel and binarizing the result as we did in the OpenCV process (CUDA does the proliferation). Thereupon, we jump back to the host, copying the results back from the device to the host variables and, lastly displaying the final results.

3 Results

The output is shown in Fig. 7. The result of the NDVI calculation is on the left. Each pixel represents the index value in a gray scale equivalence. Dark pixels identify unhealthy plants or soil elements with an index close to 0, white pixels contain information from the healthiest plants with highest values of index calculation close to 1. Most of the pixels contain a gray value and represent index values somewhere in the middle. The figure on the right is the segmentation image, which is a binary image, deviding the vegetation, grouped into healthy and unhealthy plants. Black pixels mean unhealthy plants. White pixels represent the healthy plants.

Fig. 7: The RESULT image (left) shows the NDVI calculation and the SEGMENTATION image (right) the binary image after the thresholding function.

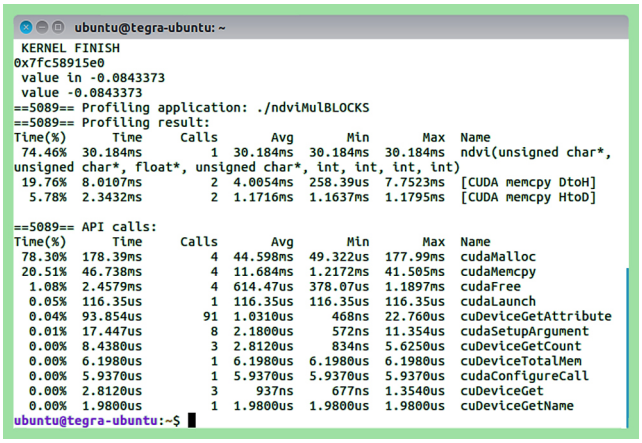

Fig. 8: That's almost a 3x speedup, from running multiple blocks (82.33 ms down to 30,184ms for the launch). 40.4 milliseconds for the whole CUDA processing.

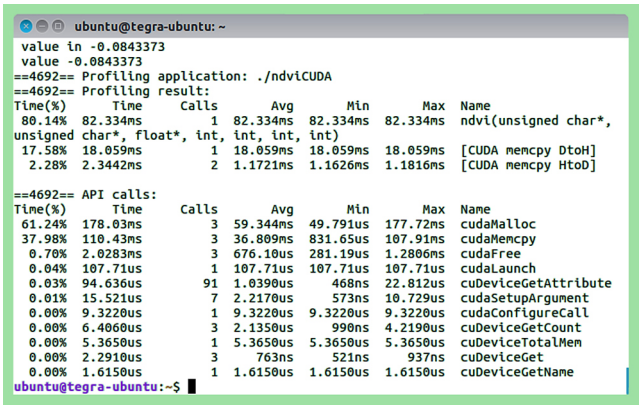

Fig. 9: The output from nvprof command, showing the time needed for the NVDIA call: It is shown as well the detailed time spend for other tasks associated with the process.

The processing times are shown in Fig. 8 and Fig. 9. The GPU grouping process dividing the data into blocks is almost 10 times faster than the CPU process. It took 0.1 seconds for the application without dividing the data, that is almost 3 times faster during the launching phase than running multiple blocks. The speed difference between video recording (typically 20 or 25 frames per second) and the NDVI calculation is about 0.4 milliseconds.

4 Discussion

We showed that the velocity of the process on the NVIDIA Jetson board can be considered real time. The speed is very close to the recording video speed and the outcome time is even faster than the acquiring time from the sequoia Parrot camera, which can just take one set of multi-spectral images per second. This provides enough time to do extra processes like sending new instructions to the drone pilot or perform additional calculations, in order to find extra details about the crops.

Up to now, the empirical resulting images give an unclear view of what is healthy or not after the NDVI calculation. For this reason, it was necessary to create a segmentation image that shows just two possible results more clearly, healthy or unhealthy. However, this output contains a considerable amount of noise. Following studies will focus on reducing that effect.

This work is a first step of a bigger project. We started almost from scratch, including the tool's learning and a short time for the test part to analyze more inputs with other features. The next steps will involve the evaluation of the final output and restructure of the system changing variables in order to improve the final results, like capturing closer pictures to get improved classifications, varying the height of the flight and calculate the healthy vegetation index taking into account the radiometric calibration from the camera.

Nowadays, a giant amount of data is coming from uncountable sensors and tools. This data can be processed quickly using parallel programming solving specifically applied problems. The Jetson Board allows the implementation of those applications for remote usage bringing that technology in real time to new application fields.

Acknowledge

The authors thanks to Prof. Fernando Javier Peña at Universidad de Ciencias Aplicadas (UDCA) in Bogotá, Colombia by their collaboration to have access to the potato crops.

References

Aasen, H., Honkavaara, E., Lucieer, A., Zarco-Tejada, P. J. (2018): Quantitative remote sensing at ultra-high resolution with uav spectroscopy: A review of sensor technology, measurement procedures, and data correction workflows. Remote Sensing, 10 (7), 2018. ISSN 2072-4292. DOI: 10.3390/rs10071091. www.mdpi.com/2072-4292/10/7/1091.

Albertz, J. (2001): Einführung in die Fernerkundung: Grundlagen der Interpretation von Luft- und Satellitenbildern. Wiss. Buchges. ISBN 9783534146246. https://books.google.de/books?id=KdFAAAACAAJ.

European Commission (2018): La investigación y la innovación agraria. http://ec.europa.eu/agriculture/research-innovation/index_es.htm.

Halfacree, G. (2017): Nvidia Jetson TK1 – Full Board. https://www.flickr.com/photos/120586634@N05/14672953894.

Harris, M. (2017): An even easier introduction to CUDA. https://devblogs.nvidia.com/even-easier-introduction-cuda.

NVDIA Corporation (2018): CUDA zone, 2018. https://developer.nvidia.com/cuda-zone.

NVIDIA Corporation (2018): Embedded system – build something amazing. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems.

OpenCV (2018): OpenCV – About. https://opencv.org/about.html.

Parrot Sequoia Team (2018): Parrot Sequoia+. www.parrot.com/business-solutions-us/parrot-professional/parrot-sequoia#parrot-sequoia-.

Pix4D Team (2018): Pix4d. https://pix4d.com/sequoia-faq.

Sanders, J., Kandrot, E. (2015): CUDA by Example: An Introduction to General-Purpose GPU Programming. Addison-Wesley Professional, 6th edition, 2015. ISBN 0131387685, 9780131387683.

Storti, D., Yurtoglu, M. (2015): CUDA for Engineers: An Introduction to High-Performance Parallel Computing. Addison-Wesley Professional, 6th edition, 2015. ISBN 013417741X, 9780134177410.

Taipale, E. (2017): NDVI and your farm: understanding NDVI for plant health insights. https://sentera.com/understanding-ndvi-plant-health.

Weier, J., Herring, D. (2018): Measuring Vegetation (NDVI & EVI). https://earthobservatory.nasa.gov/Features/MeasuringVegetation.

Contact

Angie Natalia Molina Muñoz | Javier Chaparro | Alexander Pérez
Escuela Colombiana de Ingeniería Julio Garavito
AK. 45 No. 205-59 (Autopista Norte)
Bogotá, Colombia
angie.molina-m@mail.escuelaing.edu.co
javier.chaparro@escuelaing.edu.co
alexander.perez@escuelaing.edu.co

Ansgar Brunn
University of Applied Sciences Würzburg-Schweinfurt (FHWS),
Faculty of Plastics Engineering and Surveying, Geo group
Röntgenring 8, 97070 Würzburg, Germany
ansgar.brunn@fhws.de

This article also is digitally available under www.geodaesie.info.