COMPRESSION ALGORITHM FOR PROCESS OPTIMIZATION IN PRECISION LIVESTOCK

Juan Esteban Castro
Universidad Eafit
Colombia
jjcastrog@eafit.edu.co

Juan José Moreno
Universidad Eafit
Colombia
jjmorenog2@eafit.edu.co

Simón Marín
Universidad Eafit
Colombia
smaring1@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

Black text = Simón and Mauricio's contribution

Green text = To complete for the 1st deliverable

Blue text = To complete for the 2nd deliverable

Violet text = To complete for the 3rd deliverable

ABSTRACT

Precision livestock allows ranchers to greatly improve and optimize the processes used in livestock farming, however this has also brought several challenges to overcome. Among them, we can highlight how to optimize the way of obtaining data to maximize efficiency in the use of energy and resources. This can result in higher profits for the ranchers since the different processes and challenges can be met with great effectiveness as they are finished in lower times thus resulting in the increase of productivity levels. This problem can be solved through automation and digitization in data collection, however it could happen that the information is optimized so much to the point where it no longer works properly with the algorithm that interprets the information, for this reason it is necessary to find the right balance to achieve good performance and efficiency.

la eficiencia de uso de energía y recursos a través de la automatización Which is the algorithm you proposed? What results did you achieve? What are the conclusions of this work? The abstract should have **at most 200 words**. (*In this semester, you should summarize here execution times, memory consumption, compression ratio and accurracy*).

Keywords
Compression algorithms, machine learning, deep learning, precision livestock farming, animal health.

1. INTRODUCTION

Precision farming is a new way of operating farms that allows greater control and organization of processes in livestock, which results in increased productivity and sustainability levels for those ranchers who make use of it. Being more specific, at this moment the motivation in the real world is the collection and compression of images of cattle to know their general state of health, so that with this information ranchers can take actions and measures to benefit the preservation and health of cattle.

1.1. Problem

Precision livestock aims to optimize as much as it can the use of energy and resources in the processes that ranchers use in order to increase other aspects of the business such as productivity and profits. Currently we can find a problem in this model, and it is the compression of images, because we need that these do not take up much space in order to be able to use the other remaining space to store other types of data that can be useful in the future. Nowadays, in precision livestock people make use of several video cameras which, with the help of software, develop an image processing algorithm that allows the diagnosis of cattle's health and other aspects. Our task is to find a way to compress the data obtained by the cameras to the maximum without making software labor harder or impossible to do. This is very important to society since big part of the population consumes livestock products every single day, and the health of these animals directly affect the health of people who consume it, so it is very important to know this information as ranchers can take actions to improve cattle's health and don't affect the final consumer.

1.2 Solution

In this work, we used a convolutional neural network to classify animal health, in cattle, in the context of precision livestock farming (PLF). A common problem in PLF is that networking infraestructure is very limited, thus data compression is required.

In this project we will develop a compression algorithm for images to allow a more efficient selection in precision livestock. The algorithm we chose in order to provide a better service is the Huffman coding algorithm, which is lossless. This algorithm works by assigning the repetition number of values and then creating a binary tree ordered by the repetition. Then, with the use of the tree we assign a value to each element.

1.3 Article structure

In what follows, in Section 2, we present related work to the problem. Later, in Section 3, we present the data sets and methods used in this research. In Section 4, we present the algorithm design. After, in Section 5, we present the results. Finally, in Section 6, we discuss the results, and we propose some future work directions.

2. RELATED WORK

In what follows, we explain four related works on the domain of animal-health classification and image compression in the context of PLF.

2.1 Automatic cough detection for bovine respiratory disease in a calf house

Bovine respiratory disease (BRD) is definitely a big challenge farmers must face as it could bring big economic losses in case of it going out of control. They would need to spend more money to treat animals quickly since they can infect other animals. Coughing is a common symptom in BRD so by using precision livestock farmers can detect really fast this symptom by using an algorithm that detects coughing sounds and allows them to take actions fast. The algorithm obtained data from four adjacent compartments during two periods of almost three months each and analyzed it by using 445 minutes of sound and 664 cough references. During the first period it showed great results for all the compartments, but this was not the case for the second period as only 2 compartments showed a precision higher than 80%. Still, it was a considered an experiment with good results since detected coughs match the number of animals with BRD. [1]

2.2 Image analysis to refine measurements of dairy cow behavior from a real-time location system

By monitoring animal activity, farmers can have more data to research and develop better precision livestock tools. A group of engineers researched about this topic by monitoring some delimited areas using a real-time location system in which they obtained the animals positions and then an algorithm interpreted them as their behavior. The system has a precision of 16 cm and after lots of data recollected, they could notice behaviors such as the use of brushes and licking mineral blocks. At the end of the experiment the engineers could stablish a sensibility of about 80%, so they see this experiment as an important tool to help design and create new devices useful for farmers and vets. [2]

2.3 A software tool for the automatic and real-time analysis of cow velocity data in free-stall barns: The case study of oestrus detection from Ultra-Wide-Band data

In nowadays precision livestock huge amounts of data are collected and some of it must be processed and interpreted in real time. For this reason, the processes by which the information is analyzed must be improved and it is mandatory to design new and better tools. For this reason, a group of researchers developed a software capable of doing real-time analysis of cow velocity data acquired by an ultra-wide band real-time location system. This gives farmers the possibility to work in real time with the information by acquiring the RTLS data updated at short time intervals and shows them graphs they can use to detect patterns and take actions in base of the information they are given. [3]

2.4 A computer vision approach based on deep learning for the detection of dairy cows in free stall barn

It is very important to get precise and useful information about animals when we talk about precision livestock, this is why technological tools are so useful. They allow us for example to detect and track the position of every single cow at every moment and by using a deep learning algorithm we can notice some patterns that will give us hints about animal's health. More specifically, some scientists trained a neural network that with the help of some video cameras recognizes every cow based on their morphological appearance with a precision ranging between 0.64 and 0.66. The results showed this is a clear step towards achieving a very reliable source of information since they saw the algorithm will get a higher precision after receiving more sources of reference and will be helpful so solve challenges involving animals. [4]

3. MATERIALS AND METHODS

In this section, we explain how the data was collected and processed and, after different image-compression algorithm alternatives to solve improve animal-health classification.

3.1 Data Collection and Processing

We collected data from Google Images and Bing Images divided into two groups: healthy cattle and sick cattle. For healthy cattle, the search string was “cow”. For sick cattle, the search string was “cow + sick”.

In the next step, both groups of images were transformed into grayscale using Python OpenCV and they were transformed into Comma Separated Values (CSV) files. It was found out that the datasets were balanced.

The dataset was divided into 70% for training and 30% for testing. Datasets are available at https://github.com/mauriciotoro/ST0245-Eafit/tree/master/proyecto/datasets .

Finally, using the training data set, we trained a convolutional neural network for binary image-classification using Google Teachable Machine available at https://teachablemachine.withgoogle.com/train/image.

3.2 Lossy Image-compression alternatives

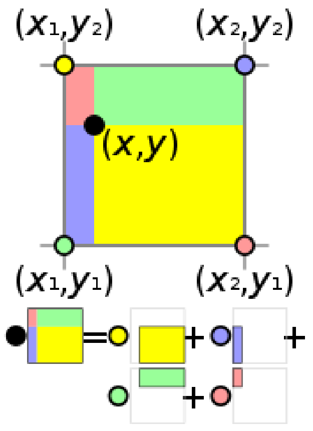
In what follows, we present different algorithms used to compress images.

3.2.1 Seam carving

Seam carving is an algorithm for image resizing. This algorithm consists in establishing several seams, which are paths of least importance in an image, and removing them to reduce the image size or even inserting them to increase it. It also allows defining areas to avoid pixel modification and includes the ability of removing objects. However, the thing that makes this algorithm lossy is that by changing the image, some parts are cut or even modified, causing the image to be distorted. The complexity of this algorithm is O(w*h+w+h), being “w” the with in pixels and “h” the height.

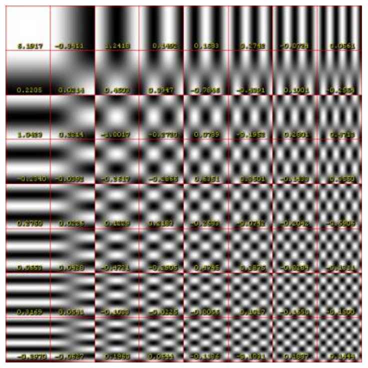
3.2.2 Image scaling

Image scaling itself refers to the resizing of a digital image. When you scale a vector image, the graphic primitives that make up the image can be scaled using geometric transformations. The Bilinear interpolation algorithm are one of the most used for image scaling. This algorithm consists of interpolating the pixel color values, introducing continuous transition into the output. Although this algorithm can be useful in continuous-tone images it reduces contrast which makes it lossy. The complexity of this algorithm is O(n).



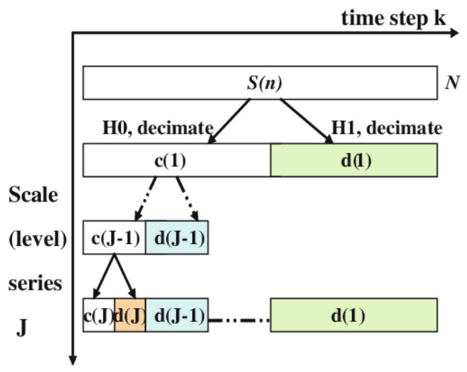
3.2.3 Discrete cosine transform

Discrete cosine transform is an algorithm that is used usually to compress JPEG images, and it is based in Fourier’s discrete transform. This algorithm decomposes the images in cosines sums and is expressed in several mathematical formulas that make it somehow easy to implement. Its complexity is O(n^2).



3.2.4 Wavelet compression

Wavelet compression has the objective of storing image data in as little space as possible in a file, it also has the capability of being lossless. This algorithm is a type of discrete cosine transform that uses wavelets. It consists of a series wavelet that represent a square-integrable function by orthonormal series generated by a wavelet.

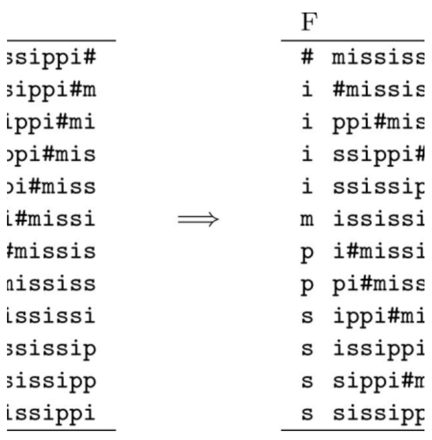


3.3 Lossless Image-compression alternatives

In what follows, we present different algorithms used to compress images. *(In this semester, examples of such algorithms are Borrows & Wheeler Transform, LZ77, LZ78, Huffman coding and LZS).*

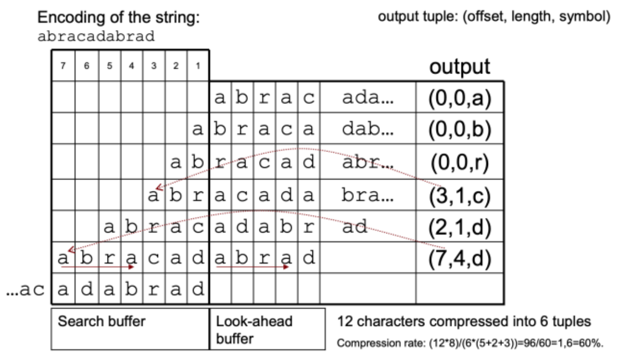
3.3.1 Borrows & Wheeler Transform

The Borrows & Wheeler Transform is an algorithm used for data compression techniques. This algorithm reorganizes a string of characters into a simpler one. It is widely used because it compresses easily the repeated characters and does not require additional information to revert it. This algorithm takes the string of characters and transforms it by classifying all the text displacement int lexicographic order, taking out the last column and the index of the original string in the group of permutations made. It has a complexity of O(n).



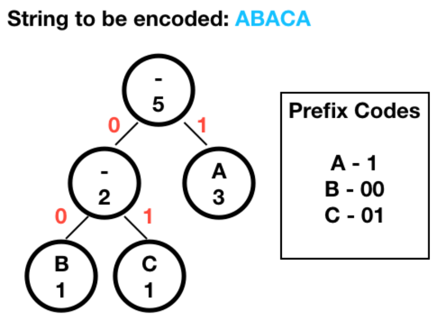
3.3.2 LZ77 & LZ78

These two algorithms were made in 1977 and 1978. They are very important because they are the base of plenty of variations like LZW and more. Furthermore, they created the base of several files like GIF and algorithms for PNG and ZIP. LZ77 iterates sequentially an input string and saves a match in the search buffer. Then using other parameters, it starts to code between the given sequence. The difference between LZ77 and LZ78 is that LZ78 does not use a sliding window and already has something similar built inside. Making these two algorithms complex. They have a complexity of O(n^2).



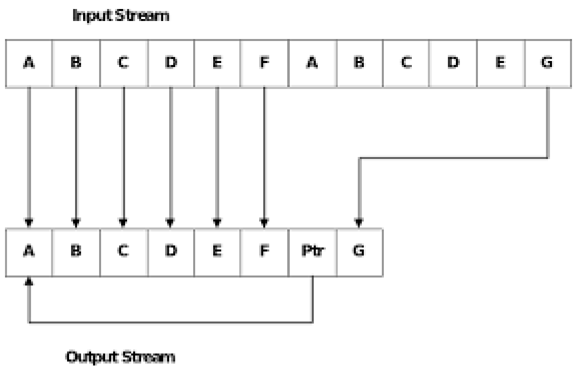
3.3.3 Huffman coding

The Huffman coding algorithm works by assigning the number of repetitions of each value of the string to code, then, it creates a binary tree with the data and assigns a specific number to each value. This algorithm is not that complex, making it one of the most used. It has a complexity of O(nL), being L the maximum length of a codeword.



3.3.4 LZS

This algorithm uses the sliding window of LZ77 and combines it with Huffman coding. This algorithm looks for similarities in between the data that will be printed and the last 2 kilobytes of the sliding window, if a coincidence is found, it creates an offset reference for the dictionary. Otherwise, it marks the next byte as a byte. It is not as complex as LZ77, but it is a simpler version of it. It has a complexity of O(n^2).



4. ALGORITHM DESIGN AND IMPLEMENTATION

In what follows, we explain the data structures and the algorithms used in this work. The implementations of the data structures and algorithms are available at Github[1].

4.1 Data Structures

We will use lists in python (dynamic vectors) as our data structure in order to create our algorithm. Dynamic vectors can be found in almost every modern programming language. For example, we can find them in Python as lists, in Java as arraylists and in C++ as vectors. Dynamic vectors

[1]http://www.github.com/ ????????? /proyecto/

are known for being simple but capable of storing a lot of information. It is very similar to arrays with the only difference that arrays have a defined length which cannot be modified at the time of execution, while dynamic vectors can expand while executing the program. The complexity of search, insertion and erase in dynamic vectors is O(n), if you only access the dynamic vector then the complexity is O(1).

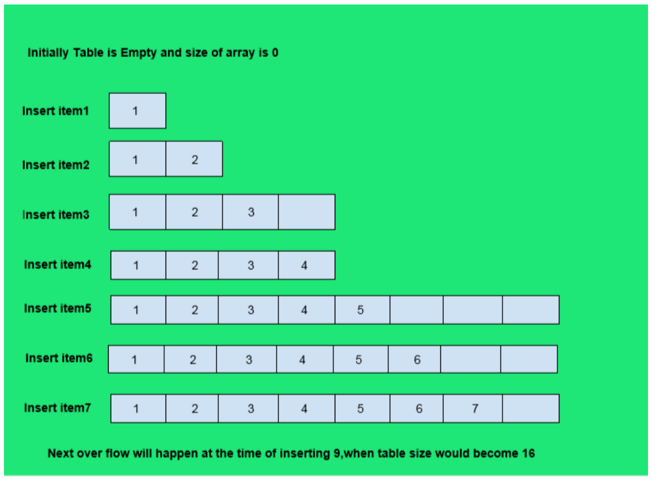


Figure 1: Insertion of element in dynamic vector.

- **4.2 Algorithms**

In this work, we propose a compression algorithm which is a combination of a lossy image-compression algorithm and a lossless image-compression algorithm. We also explain how decompression for the proposed algorithm works.

There is a tool in mathematics called linear interpolation which consists of using linear polynomials to construct new data points within the range of a discrete set of known data points, giving a new curve in which, all these data fit properly. This is why we will implement this mathematic tool to create new groups of data from the CSV file. First, the algorithm will read the data in form of a matrix and will take the first two numbers of a row and it will sum them with the two elements in the same position in the row below. Then it will divide this sum into 4 and give a new number that will replace these numbers in a new matrix.. Also, the algorithm will give the result of the averaging in Int numbers which means that if by any chance the algorithm receives a Float number it will convert it into an Int number. This process will repeat continuously until the process is completed in the hole data set, giving a much more simplified matrix.

Before Algorithm

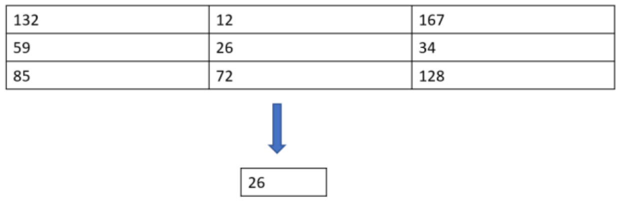
113	2	128	49	102	48	94
104	59	83	38	88	111	10
97	133	55	67	133	150	16
126	43	131	37	105	36	13
49	38	101	61	108	142	24
135	42	8	27	44	107	82
150	60	129	96	58	35	87
33	103	42	66	149	27	33

After Algorithm

169	147	193	130
153	133	168	116

4.2.1 Lossy image-compression algorithm

A common lossy image-compression algorithm is image scaling, and to do this "The nearest neighbor" is commonly used. This algorithm consists of choosing the middle pixel between a 3x3 matrix. This algorithm can be used but the losses in the image might affect it a lot, and lose more data than planned, that is why we used de linear interpolation that is a lossy image-compression algorithm but less drastic than the nearest neighbor one. To decode the nearest neighbor algorithm you just fill up the 9 pixels with the value of the pixel you selected, making the decompression not as effective as we want it to be.



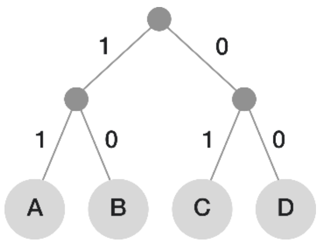
132	12	167
59	26	34
85	72	128

26

Figure 2: Image scaling using Nearest neighbor compression algorithm.

4.2.2 Lossless image-compression algorithm

Usually when we don't want to lose any information when compressing data, we result with a lot of size in our files. This is why it is way more effective to make use of algorithms such as Huffman Coding. Based on the frequency of the data we created an algorithm that does not classify the information equally. It makes easier to access the data with more frequency and "harder" to access the information with less frequency, which just means it will take more steps to access this information. This is very useful because this way we make use of our resources effectively, it is like having only 2 seats and having 2 elders and a young man. We give the seats to the elders because they need them more, in this case the elements with higher frequency need more resources and therefore we make it easier for the computer to access them. To decompress the information, we make use of 0 (left) and 1(right) in the tree where we put our data. So, to access each piece of information we use a specific code made of 0's and 1's. For example, let's say the word "Hello" is referred as 1101 as it appears less times. But let's say the word "He" is represented by the number 00 as it appears more. Then to access information we need this number, and in the tree, we made the words with the highest frequency to have shorter numbers and words with less frequency to have longer numbers. This is how we compressed and decompressed our data.



4.3 Complexity analysis of the algorithms

Explain, in your own words, the analysis for the worst case using O notation. How did you calculate such complexities. Please explain briefly.

Algorithm	**Time Complexity**
Compression	$O(N^2*M^2)$
Decompression	$O(N^3*M*2^N)$

Table 2: Time Complexity of the image-compression and image-decompression LZ77 algorithm in which N is the row of characters and S the size of buffer.

Algorithm	**Memory Complexity**
Compression	O(N*M)
Decompression	O(N*M)

Table 3: Memory Complexity of the image-compression and image-decompression algorithms. N is equal to the rows and M is equal to the columns that it modifies.

Algorithm	**Memory Complexity**
Compression	O(N*M)
Decompression	O(N*M)

4.4 Design criteria of the algorithm

We designed the algorithms trying to be as similar as possible to how they were originally designed, in other words, we focused a lot on the process of this and that each one had the same purpose and processes as they were originally stablished on their papers. In the lossy algorithm, in which we use linear interpolation, the purpose of the code is found in a very mathematical way, and it helps us to significantly reduce the size of the file, in the case of our code it reduces the file size in ½ or 1/3 depending on the CSV. However, for really big CSV probably it will take a lot of time and also it loses a lot of quality from the original image. On the other hand, in the LZ77 algorithm, since this algorithm is more related in computing, it is very easy and simple knowing how to implement it in programming language and does not lose any information which makes it very appropriate for compressing.

5. RESULTS

5.1 Model evaluation

In this section, we present some metrics to evaluate the model. Accuracy is the ratio of number of correct predictions to the total number of input samples. Precision. is the ratio of successful students identified correctly by the model to successful students identified by the model. Finally, Recall is the ratio of successful students identified correctly by the model to successful students in the data set.

5.1.1 Evaluation on training data set

In what follows, we present the evaluation metrics for the training data set in Table 3.

	Training data set
Accuracy	0.02
Precision	0.03
Recall	0.01

Table 3. Binary image-classification model evaluation on the training data set.

5.1.2 Evaluation on test data set

In what follows, we present the evaluation metrics for the testing dataset in Table 4 without compression and, in Table 5, with compression.

	Testing data set
Accuracy	0.01
Precision	0.012
Recall	0.013

Table 4. Binary image-classification model evaluation on the testing data set without image compression.

	Testing data set
Accuracy	0.001
Precision	0.0012
Recall	0.0013

Table 5. Model evaluation on the testing data set with image compression.

5.2 Execution times

In what follows we explain the relation of the average execution time and average file size of the images in the data set, in Table 6.

Compute execution time for each image in Github. Report average execution time Vs average file size.

	Average execution time (s)	***Average file size (MB)***
Compression	100.2 s	12.4 MB
Decompression	800.1 s	12.4 MB

Table 6: Execution time of the *(Please write the name of the algorithms, for instance, seam carving & LZ77)* algorithms for different images in the data set.

5.3 Memory consumption

We present memory consumption of the compression and decompression algorithms in Table 7.

	Average memory consumption (MB)	***Average file size (MB)***
Compression	634 MB	3.12 MB
Decompression	9 MB	878.12 MB

Table 7: Average Memory consumption of all the images in the data set for both compression and decompression.

To measure memory consumption, you should use a profiler. A very good one for Java is VisualVM, developed by Oracle, http://docs.oracle.com/javase/7/docs/technotes/guides/visualvm/profiler.html. For Python, use C Profiler.

5.3 Compression ratio

We present the average compression ratio of the compression algorithm in Table 8.

	Healthy Cattle	***Sick Cattle***
Average compression ratio	1:23	1:34

Table 8: Rounded Average Compression Ratio of all the images of Healthy Cattle and Sick Cattle.

6. DISCUSSION OF THE RESULTS

Explain the results obtained. Are precision, accuracy and sensibility appropriate for this problem? Is the model over-fitting? Is memory consumption and time consumption appropriate? Is compression ratio appropriate? Does compression changes significantly precision on the test data set? *(In this semester, according to the results, can this improve animal-health classification in the context of PLF?)*

6.1 Future work

In the future we would like to have a better and deeper understanding of the algorithms so that we can comprehend how they work from the deepest of the computer to improve it. We would like to understand how files are created in the folders and optimize them so that they are faster and implement new things in the algorithms so that they can tell us quickly if any folder or file can be compressed or not. In general, there are many ways in which this project can be improved and has a lot of potential to create really useful and cool things, but this may not be very easy since we found it somewhat hard and tedious at times to find information about these algorithms in the surface of the internet, and we really had to go down deep to understand things better.

ACKNOWLEDGEMENTS

Identify the kind of acknowledgment you want to write: for a person or for an institution. Consider the following guidelines: 1. Name of teacher is not mentioned because he is an author. 2. You should not mention websites of authors of articles that you have not contacted. 3. You should mention students, teachers from other courses that helped you.

As an example: This research was supported/partially supported by [Name of Foundation, Grant maker, Donor].

We thank for assistance with [particular technique, methodology] to [Name Surname, position, institution name] for comments that greatly improved the manuscript.

REFERENCES

1. Lenn Carpentier, Daniel Berckmans, Ali Youssef, Dries Berckmans, Toon van Waterschoot, Dayle Johnston, Natasha Ferguson, Bernadette Earley, Ilaria Fontana, Emanuela Tullo, Marcella Guarino, Erik Vranken, Tomas Norton, Biosystems Engineering, Volume 173, Pages 45-56, 2018
2. Bruno Meunier, Philippe Pradel, Karen H. Sloth, Carole Cirié, Eric Delval, Marie M. Mialon, Isabelle Veissier,Image analysis to refine measurements of dairy cow behaviour from a real-time location system, Biosystems Engineering, Volume 173, Pages 32 – 44, 2018)
3. Claudia Arcidiacono, Simona M.C. Porto, Massimo Mancino, Giovanni Cascone, A software tool for the automatic and real-time analysis of cow velocity data in free-stall barns: The case study of oestrus detection from Ultra-Wide-Band data, 2018
4. Patrizia Tassinari, Marco Bovo, Stefano Benni, Simone Franzoni, Matteo Poggi, Ludovica Maria Eugenia Mammi, Stefano Mattoccia, Luigi Di Stefano, Filippo Bonora, Alberto Barbaresi, Enrica Santolini, Daniele Torreggiani, Computers and Electronics in agriculture, volume 182, 2021