

# Compression algorithms to optimize battery consumption in precision livestock farming

Hernan Moreno  
Universidad Eafit  
Colombia

hmorenom@eafit.edu.co

Mariana Yepes  
Universidad Eafit  
Colombia

myepesa@eafit.edu.co

Simón Marín  
Universidad Eafit  
Colombia

smaring1@eafit.edu.co

Mauricio Toro  
Universidad Eafit  
Colombia

mtorobe@eafit.edu.co

## Abstract

Most of the protein consumed by mankind comes from livestock, and its demand is increasing day by day, despite efforts to develop artificial proteins to replace it in the future. Energy consumption on livestock farms is an obstacle to the use of technological tools to optimize processes. That is why it is important to facilitate the use of image compression algorithms for use in livestock farms to detect animal health, minimizing the use of computational and energy resources, thus reducing costs and optimizing the cattle raising process.

## Key words

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

## 1. Introduction

Precision Livestock farming (PLF) refers to the application of technology to a livestock farm in order to optimize herd management and minimize operating costs. A system of digital cameras at certain points where cattle pass through the herd, allows an analysis of the health of each animal based on its physical appearance. However, the images generate a high volume of data that consumes valuable resources (disk storage, energy, time) and therefore it is necessary to develop a compression algorithm without loss of image quality

### 1.1. Problem

Most published forecasts predict that global demand for meat will increase by at least 40% in the next 15 years [1]. At the same time, there is great concern about the transfer of diseases from livestock to humans, which makes animal health a high priority. Furthermore, there is great room for improvement in the treatment of animal diseases. For example, the use of anti-biotics is too high and needs to be reduced. The problem we face is to design an algorithm to compress the Cattle images taken in the field to determine through their analysis the health of the animals. The algorithm must consume as little energy and computational power as possible to optimize the available resources. In addition, it should be possible to revert to the original image when necessary.

### 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 infrastructure is very limited, thus data compression is required.

Essentially, in the lossy compression we implement the seam carving algorithm[6]. The main reason why we decided to implement it is because this algorithm can preserve the proportions of the objects while changing the image proportions. This algorithm was introduced by Shai Avidan and Ariel Shamir[21]. The idea of the Seam Carving algorithm is to find the seam (continuous sequence of pixels) with the smallest contribution to the image content and then carve it (remove it). This process is repeated several times until we obtain the required image width or height.

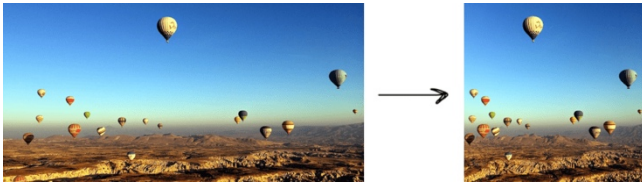


Figure 1. Image scaling using seam carving

### 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.

### 3.1 Abundance of unstructured information, and therefore difficult to apply. [20]

A clear example is given in Australia, and it is a model based on the Hazard Analysis Critical Control Point (HACCP) method, developed for cattle grazing companies, and which can be applied to other industries in the animal sector. It is based on the following:

- Identifying those aspects that have a major effect on productivity, profitability and/or sustainability: actions that, if not carried out correctly, will substantially reduce the viability of the enterprise. In the Australian case, 29 aspects were identified that met this condition.
- Identify for each of the above processes the variables to be measured to ensure that the process is carried out correctly.
- Apply the most cost-effective corrective action whenever the measurements are outside the predetermined limits.
- Establish Standard Operating Procedures (SOP) for specific companies and for each essential procedure to ensure that, under normal circumstances, the critical values measured will be within the predetermined limits.
- Provide the necessary tools to perform the measurements, interpret them, and decide on corrective actions. This is the "package" of the solution and farm personnel should be trained in its use.

### 3.2 Wireless network for livestock health monitoring. [19]

The agriculture and livestock industry are of vital importance in the UK, with an annual output in the order of £5.8 billion. About  $\frac{3}{4}$  of the kingdom's land is devoted to agriculture and livestock farming and employs about 500,000 people. In recent years there have been serious diseases. The two most recent were BSE (Bovine Spongiform Encephalopathy) and FMD (Foot and mouth disease) in 1986 and 2001 respectively. 4.5 million cattle were incinerated following the identification of BSE and nearly 4 million for FMD, at a total cost to the British economy of some £13 billion.

The main issues for a Wireless Monitoring System are:

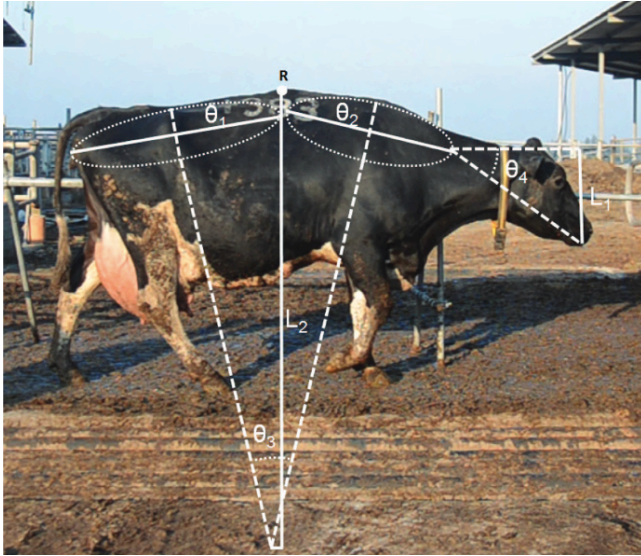
- Signal attenuation due to the animal's body: this is due to the absorption of the radio signal by the bodies of the animals in the herd.
- Low memory capacity in the nodes: this makes it not possible to store the information to send it later. It is therefore necessary to develop a routing protocol to send the information in real time to the base station via multi-hop.
- Mobility: Wireless monitoring networks are normally static. In a cattle herd the cattle are in continuous movement and therefore the sensors are in continuous movement. Therefore, the network topology and routing configuration must be dynamic to adapt to changes in the position of the cattle and sensors.

### 3.3 Analysis of individual classification of lameness using automatic measurement of back posture in dairy cattle

This paper presents an algorithm for automatic camera analysis of dairy cows as they approach a milking robot. By performing image analysis and calculating the model parameters from the image information, it was possible to develop this algorithm for automatic detection of lameness problems in dairy cows. These techniques allow a frequent and fully automatic follow-up of each cow, a process that can no longer be easily performed by the farmer. As soon as the calculated individual performance parameters change, a warning alerts the farmer. This can have important benefits, because lameness is a major welfare problem in modern dairy cows, where up to 25% can be severely affected.

The algorithm seeks to calculate the BMP or body movement pattern. The highest point (R) of the total

curvature of the animal's back was used as a starting point to calculate the BMP. Two ellipses were fitted to the left and right sides of the point R and their orientation,  $\theta_1$  and  $\theta_2$ , was calculated. The intersection between the two lateral axes of both ellipses was determined and the resulting angle,  $\theta_3$ , was calculated;  $L_2$  is the vertical distance between this intersection point and R. A line was drawn joining the position of the cow's snout to the longitudinal axis of the front ellipse;  $L_1$  is the vertical distance between the cow's snout and the longitudinal axis of the front ellipse. The angle  $\theta_4$ , between the horizontal line joining  $L_1$  with the longitudinal axis of the frontal ellipse and the line joining the cow's snout with the longitudinal axis of the frontal ellipse, was calculated.



$$\text{BMP} = w_1 \times \frac{\theta_2}{\theta_1} + w_2 \times \frac{\theta_4}{\theta_3} + w_3 \times \frac{L_1}{L_2},$$

Figure1. Equation to estimate the cows BMP

The linear combination of these 4 angles ( $\theta_1$ ,  $\theta_2$ ,  $\theta_3$ ,  $\theta_4$ ) and 2 distances ( $L_1$ ,  $L_2$ ) was used to calculate the BMP: where (weight)  $w_1$  describes the relationship between the front and back ellipse,  $w_2$  includes the relationship between both ellipses and head position, and  $w_3$  again relates head position to back curvature. To calculate a BMP value for each cow, the mean of the BMPs calculated every 5 frames was used.

In total, 169 of 223 cases were correctly classified, which is 76% accurate. The results indicate that accuracy is only high when non-lame cow cases are detected. The accuracy of cows classified as non-lame was 83%, while it was only 61% for lame and 64% for severe lame [4].

### 3.4 Livestock Detection Based on Convolutional Neural Network

To maintain the ecological balance of the grassland and obtain the number of grazing animals, an improved livestock detection algorithm based on a full convolutional neural network is proposed.

If the density of grazing livestock in the grassland is too large and exceeds the regulatory capacity of the ecosystem, it will cause overgrazing and lead to the degradation of the grassland. Reducing grazing density in grasslands has become a method to effectively prevent overgrazing and realize natural grassland protection. Therefore, it is of great importance to use livestock detection technology to obtain the grazing amount of grassland accurately and efficiently.

The paper proposes image detection algorithms. Based on the livestock detection algorithm mentioned in the literature, MultiResUNet is proposed to obtain different scales of the image, obtain the layered image patches, and take the candidate regions by morphological operation. An accuracy of 95.37% is obtained.[5].

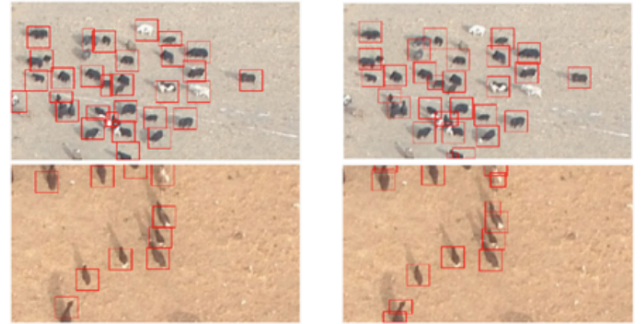


Figure2. Result of the image detection algorithm [5]

## 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 Stream carving

Seam carving is an intelligent image resizing algorithm to adapt the image content to the screen without distorting important objects. It is an efficient approach to image resizing by removing unimportant pixels using gradient information and dynamic programming [6] The algorithm receives an image, then, calculates the weight/density/energy of each pixel, this is done by an algorithm called gradient magnitude and from the energy, makes a list of seams.

The seams are sorted by energy, with low energy seams being of less importance to the image content. The algorithm removes the low energy seams as necessary and finally returns the image. Seam carving also allows manual definition of areas where pixels cannot be modified and can remove entire objects from photographs.

Stream carving, based on the seam carving algorithm, can now introduce larger seams into the resized image, i.e., seams with a width greater than one pixel, which they call streams.

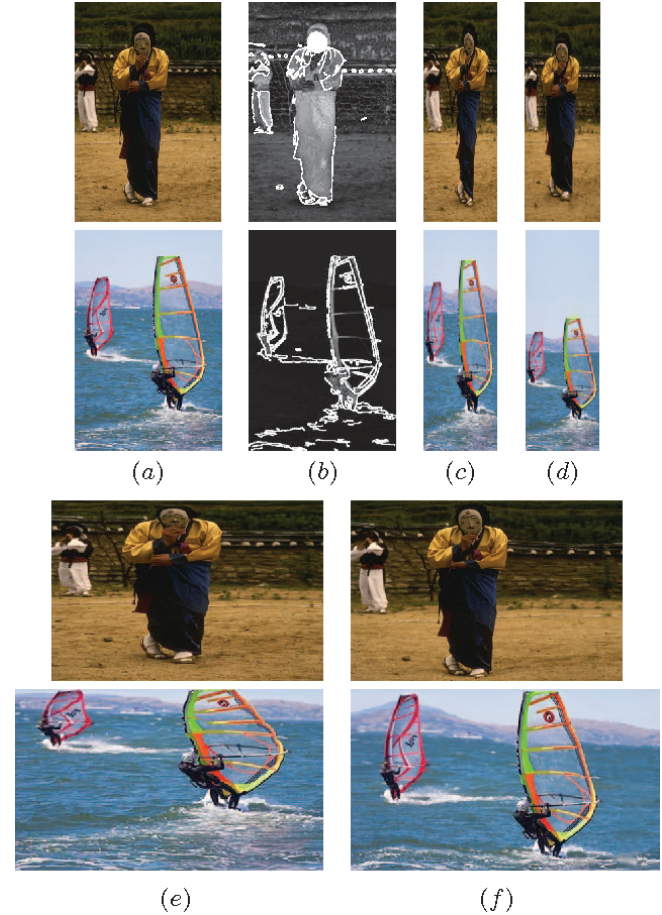


Figure 3. Stream carving algorithm process [7]

The resulting gaps are recovered with an inpainting method. The algorithm uses an adaptive importance map that combines several features, such as gradient magnitude, faces, edge and straight lines detection. This approach induces an increase in the quality of the retargeted image compared to the original seam carving method and provides similar or better results than other current image retargeting techniques. The results of this approach are similar to or better than other current image retargeting techniques. [7]

#### 3.2.2 Discrete cosine transform (Discrete cosine transform)



First proposed by Nasir Ahmed in 1972, it is a transformation technique widely used in signal processing and data compression in digital media, including images (such as JPEG and HEIF), digital video (such as MPEG and H.26x), Digital Audio (such as Dolby Digital, MP3 and AAC), Digital TV (SDTV, HDTV and VOD) and Digital Radio (AAC+ and DAB+), as well as in voice coding (AAC-LD, Siren and Opus).

Like any Fourier-based transform, DCTs express a function or a signal in terms of a sum of sinusoids with different frequencies and amplitudes. Like the Discrete Fourier Transform (DFT), a DCT operates on a function at a finite number of discrete data points. The main difference between DCT and DFT is that the former uses only cosine functions, while the latter uses both cosine and sine functions.

Formal definition: the discrete cosine transform is an invertible linear function, where are the real numbers, or equivalently, an invertible square matrix  $N \times N$ . In the DCT-I variant, the  $N$  real numbers  $X_0, \dots, X_{N-1}$  are transformed into  $N$  real numbers  $X_0, \dots, X_{N-1}$  according to the formula: [12]

$$X_k = \frac{1}{2}(x_0 + (-1)^k x_{N-1}) + \sum_{n=1}^{N-2} x_n \cos \left[ \frac{\pi}{N-1} nk \right]$$

Some authors then multiply the terms  $X_0$  and  $X_{N-1}$  by  $\sqrt{2}$ , and the terms from  $X_0$  and  $X_{N-1}$  by  $1/\sqrt{2}$  which makes the DCT-I matrix an Orthogonal matrix, if one further multiplies by an overall scale factor of

$\sqrt{\frac{2}{N-1}}$ , but breaks the direct correspondence with a real-even DTF.

There are other similar transformation formulas that give rise to the variants DCT-II, DCT-III, DCT-IV.... Up to DCT-VIII, and there are also the so-called Multidimensional variants

### 3.2.3 Fractal compression

As the name suggests, this is a lossy image compression method based on fractals. The method is

suitable mainly for images of nature, taking advantage of the fact that parts of an image often resemble other parts of the same image. Fractal algorithms convert these parts into mathematical data called "fractal codes" that are used to recreate the encoded image.

Mathematically, they can be described as an Iterated Function System (IFS).

**Binary images:** To begin, we take a binary image, which can be considered as a subset of  $\mathbb{R}^2$ . The IFS is then a set of contraction mappings,  $f_1, \dots, f_n$ ,  $f_i : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ .

According to this mapping functions, the IFS describes a two-dimensional set  $S$  as the fixed point of the Hutchinson Operator

$$H(A) = \bigcup_{i=1}^N f_i(A), \quad A \subset \mathbb{R}^2.$$

This is,  $H$  is an operator mapping sets to sets, and  $S$  is the unique set satisfying  $H(S) = S$ . The idea is to construct the IFS such that this set  $S$  is the input binary image. The set  $S$  can be recovered from the IFS by fixed point iteration: for any nonempty compact initial set  $A_0$ , the iteration  $A_{k+1} = H(A_k)$  converges to  $S$ .

The set  $S$  is self-similar because  $H(S) = S$  implies that  $S$  is union of mapped copies of itself:

$$S = f_1(S) \cup f_2(S) \cup \dots \cup f_N(S)$$

So, we see the IFS is a fractal representation of  $S$ . [13]

### Encoding

A current problem of fractal image representation is how to choose the  $f_1, \dots, f_n$  such that its fixed point approximates the input image, and how to do it efficiently.

An approach for doing it is the following partitioned iterated function system (IPFS):

1. Partition the image domain into range blocks  $R_j$  of size  $s \times s$

2. For each  $R_j$ , search the image to find a block  $D_j$  of size  $2s \times 2s$  that is very similar to  $R_j$
3. Select the mapping functions such that  $H(D_j) = R_j$  for each  $I$

In the second step, find a similar block so that the IFS accurately represents the input image, so it is necessary to consider enough candidates blocks for  $D_j$ . But a large search considering many blocks is computationally costly, and therefore PIFS fractal encoding is much slower than DCT and wavelet-based image representation. [14]

### 3.2.4 Image scaling

The basic concept of image scaling is to resample a two-dimensional function on a new sampling grid. Many algorithms have already been proposed for image scaling. The most widely used method is the bilinear method, which is considered a first-order sampling and resampling method. In bilinear, the output pixel value changes linearly according to the sampling position.

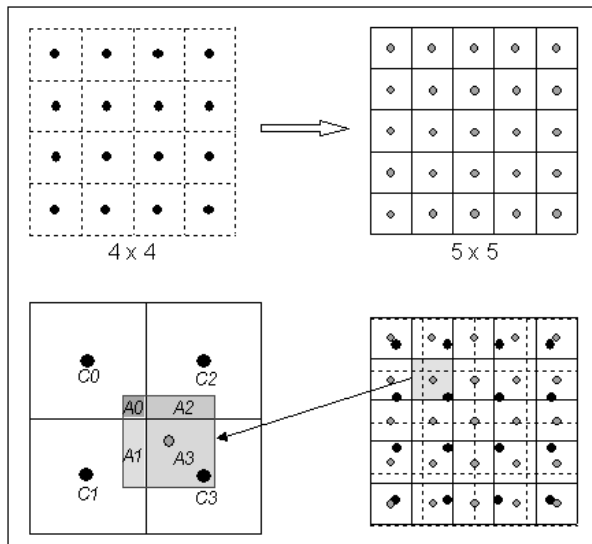


Figure 4. General concept of image scaling

There is a more complex method called bicubic. The weakness of the bilinear and bicubic methods is the blurring effect that causes a poor response at high frequencies [8].

## 3.3 Lossless Image-compression alternatives

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

### 3.3.1 Burrows-Wheeler Transform (BWT)

Also called Block-sorting compression, is an algorithm used to prepare data to be used with a data compression technique such as bzip2. [15]

When a string of characters is transformed using this algorithm, the transformation "permutes" the order of the characters. If the original string contains several sub-strings that repeat frequently, then the transformed string will have several places where a single character repeats multiple times in a row. For example:

Input

```
SIX.MIXED.PIXIES.SIFT.SIXTY.PIXIE.DUST.
BOXES
```

Output

```
TEXYDST.E.IXIXIXSSMPPS.B..E.S.EUSFXDII
OIIIT
```

The Output is easier to compress as it has many repeated characters. In this example, the transformed string contains 6 sets of identical characters: XX, SS, PP, ..., II and III which together add up to 13 of the 44 characters in the string.

#### Example:

The transform is done by sorting all the circular shifts of a text in lexicographic order and by extracting the last column and the index of the original string in the set of sorted permutations  $S$ .

Given an input string  $S = \text{^BANANA|}$ , rotate it  $N$  times, where  $N = 8$  is the length of the  $S$  string including the symbol  $\text{^}$ , representing the start of the string and the red  $|$  character representing the EOF pointer; these rotations, or circular shifts, are then sorted lexicographically. The output of the encoding phase is the last column,  $L = \text{BNN^AA|A}$  and the index  $I$  of the row containing the original string  $S$ , in this case  $I = 6$ .

		Transformation		
1.Input	2.All Rotations	3.Sort into lexical order	4.Take the last column	5.Output
^BANANA	^BANANA     ^BANANA A   ^BANAN NA   ^BANA ANA   ^BAN NANA   ^BA ANANA   ^B BANANA   ^	ANANA   ^B ANA   ^BAN A   ^BANAN BANANA   ^ NANA   ^BA NA   ^BANA ^BANANA     ^BANANA	ANANA   ^B ANA   ^BAN A   ^BANAN BANANA   ^ NANA   ^BA NA   ^BANA ^BANANA     ^BANANA	BNN^AA   A

## Dynamic Burrows-Wheeler transform:

When a text is edited, its Burrows-Wheeler transform will change. Salson et al [16] proposes an algorithm that deduces the Burrows-Wheeler transform of an edited text from that of the original text, doing a limited number of local reordering in the original Burrows-Wheeler transform, which can be faster than constructing the Burrows-Wheeler transform of the edited text directly.

The Time Complexity of the code is  $O(n^2 \log n)$  because the method builds a suffix array which with  $O(n)$  time for strings comparisons, and  $O(n \log n)$  for the sorting algorithm.

The following pseudocode gives a simple way to

### 3.3.2 LZ77 and LZ78

<b>function</b> BWT (string s)  create a table, where the rows are all possible rotations of s sort rows alphabetically  <b>return</b> (last column of the table)
<b>function</b> inverse BWT  create empty table  <b>repeat</b> length(s) <b>times</b> // first insert creates first column  insert s as a column of table before first column of the table  sort rows of the table alphabetically  <b>return</b> (row that ends with the EOF character)

calculate the BWT and its inverse. It assumes that the input string  $s$  contains a special character EOF which is the last character and occurs nowhere in the text: data."

Also known as LZ1 and LZ2, are two lossless data compression algorithms, published by Abraham Lempel in 1977 [17] and Jacob Ziv in 1978 [18]. They form the basis of many variants, including LZW, LZSS, LZMA and others. They are the basis of such general systems as GIF and DEFLATE, the latter used in PNG and ZIP.

LZ77 performs compression by replacing repeated occurrences of data with a reference to a single copy of that repeated data existing in the original uncompressed data. A match is encoded with a pair of numbers called a "length-distance pair" equivalent to the following phrase: "each of the following length characters is equal to exactly the previous length-distance characters in the original uncompressed

To detect matches, the program must keep a record of a certain amount of the most recent data, for example, the last 2KB, 4KB or 32KB. The structure in which this data is stored is called a sliding window, which is why LZ77 is also known as sliding window compression.

The following is the pseudocode of the algorithm:

```
while input is not empty do
    prefix := longest prefix of input
    that begins in window

    if prefix exists then
        d := distance to start of prefix
        l := length of prefix
        c := char following prefix in
input
    else
        d := 0
        l := 0
        c := first char of input
    end if

    output (d, l, c)

    discard l + 1 chars from front of
window
    s := pop l + 1 chars from front of
input
    append s to back of window
repeat
```

The LZ78 performs compression by replacing repeated occurrences of data with references to a dictionary that is constructed based on the input data. Each dictionary entry is of the form

Dictionary[...] = {index, character}

Where index is the index to a previous dictionary entry and character is appended to the end of the string

represented by dictionary[index]. For example, "abc" will be stored (in reverse order), as follows:

Dictionary[k] = {j, 'c'}, dictionary[j] = {i, 'b'}, dictionary[i] = {0, 'a'}

Where index 0 represents the first character of a string. The algorithm initializes last matching index = 0 and next available index = 1. For each character in the input string, the dictionary is searched for a match: {last matching index, character}. If a match is found, then last matching index is matched to the index of the matching input, and there is no output. If no match is found, a new dictionary entry is created: dictionary{next available index} = {last matching index, character}, and the algorithm writes last matching index followed by character, makes last matching index = 0 and increments next available index. When the dictionary is full no new entries are added. When the end of the input data is reached, the algorithm writes last matching index. The strings are stored in reverse order, which the decoder must consider.

The complexity of the algorithm was calculated as:  $O(n(1 + \log s / \log \log n))$ , where  $s = O(1)$ , by Paolo Ferragina, Igor Nitto and Rossano Venturini, "On the bit-complexity of Lempel-Ziv compression"

### 3.3.3 Huffman coding

Huffman coding is a lossless data compression algorithm. The idea is to assign variable length codes to the input characters; the lengths of the assigned codes are based on the frequencies of the corresponding characters. The most frequent character receives the smallest code and the least frequent the largest.



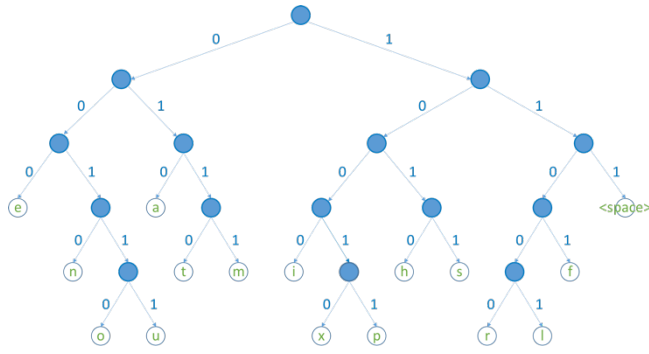


Figure 5. Huffman tree generated from the exact frequencies of the text "this is an example of a Huffman tree". Encoding the sentence with this code requires 135 bits, as opposed to 288 (or 180) bits if 36 characters of 8 (or 5) bits were used. [10]

Variable-length codes assigned to input characters are prefixed codes, meaning that the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of the code assigned to any other character. This is how Huffman coding ensures that there is no ambiguity when decoding the generated bit stream. The complexity is  $O(n \log n)$  where  $n$  is the number of unique characters. [9]

### 3.3.4 Delta encoding

This algorithm represents compressed pixel streams as the difference between the current and previous pixel.

The first pixel in the delta-encoded file is the same as the first pixel in the original image. All subsequent pixels in the encoded file are equal to the difference (delta) between the corresponding value of the input image and the previous value of the input image. In other words, delta coding increases the probability that the value of each pixel is close to zero and decreases the probability that it is far from zero [11].

## 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<sup>1</sup>.

### 4.1 Seam carving Algorithms

In this work, we propose the Seam Carving compression algorithm which is based in the removal of non relevant parts of the image. We give some examples and explain how the algorithm works.

Seam carving (or Liquid rescaling) is an algorithm for content-aware image rescaling, developed by Shai Avidan of Mitsubishi Electric Research Laboratories (MERL), and Ariel Shamir of the Interdisciplinary Center and MERL. It can be used for both reduction and expansion. A *seam* is an optimal 8-connected path of pixels on a single image, from top to bottom or from left to right, where optimality is defined by an image energy function. [21]





Seam Carving supports several types of energy functions, such as gradient magnitude, entropy, visual saliency, eye-gaze movement and more.

It functions by establishing a number of seams (or paths of least importance), in an image, and automatically removes seams to reduce image size, or inserts seams to extend it. It allows for manually defining areas in which pixels may not be modified and features the ability to remove whole objects from photographs.


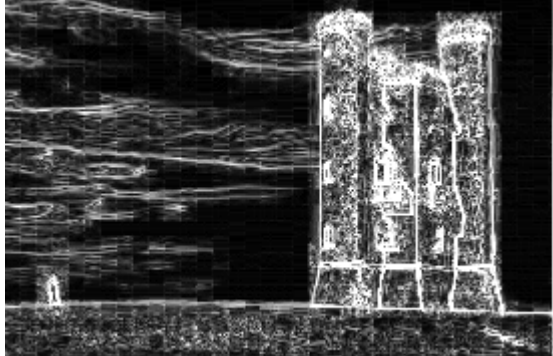
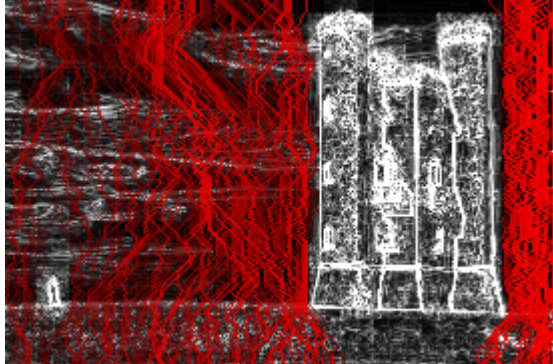
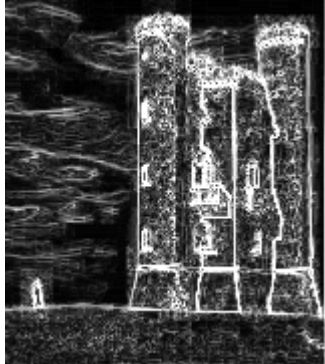

The purpose of the method is "Image retargeting" which is the problem of displaying images without distortion on media of various sizes, as cell phones or projection screens.

An example of the method is explained in the following images:

<sup>1</sup><https://github.com/homorenom/ST0245-003/tree/master/proyecto>

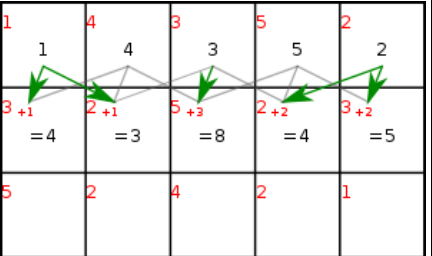
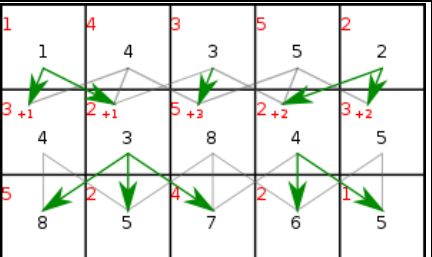
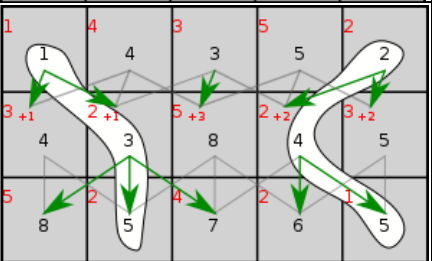
	<p>Original image</p>
	<p>Image scaling is not appropriate, as the castle is distorted from its original dimensions.</p>
	<p>Image cropping, does not work in this case, as the image of the castle is cut in half.</p>
	<p>Seam carving, removes non relevant information between the castle and the person, conserving all the relevant features of the original image.</p>

The process is as follows:

<p>1. Start with an original image</p>	 The original image shows a large, multi-towered stone castle with crenellated roofs, situated on a green grassy hill under a clear blue sky. A small figure of a person is visible on the left side of the hill for scale.
<p>2. 3 parameters are calculated for each pixel: weight, density and energy, using different algorithms, as gradient magnitude, entropy, visual saliency or eye-gaze movement. In this example gradient magnitude was used.</p>	 A grayscale edge detection image of the castle. The edges of the castle's towers and the horizon are highlighted in white against a black background.
<p>3. From the energy calculated for each pixel, make a list of seams, ranked by energy: those with low energy are the least important to the content of the image. Seams are calculated via dynamic programming explained below.</p>	 The edge detection image from the previous step, but with numerous red lines overlaid. These lines represent the detected seams, which are paths of low-energy pixels through the image.
<p>4. Low energy seams are then removed.</p>	 The edge detection image, but with the red seams removed. The image is mostly black, with only the most prominent edges of the castle remaining in white.
<p>5. Final image: low energy areas, to the right of the castle, and between the person and the castle, are removed from the image. The relevant image (the castle) is not touched.</p> <p>The seams to remove, depends only on the dimension (height or width) one wants to shrink. It is also possible to invert sept 4 so the algorithm enlarges in one dimension by copying a low energy seam and averaging its pixels with its neighbors.</p>	 The final image after seam removal. The castle remains in its original position and size. The area to the right of the castle and the area between the person and the castle have been removed, resulting in a narrower image where the castle is closer to the person.

Below, we explain the seam carving process:

Computing seams: It consists of finding a path of minimum energy from one end of the image to another. There are different algorithms to do it, like Dijkstra's Algorithm,

<p>The top row has nothing above it, so the energies are the same as the source image</p>	<p>Algorithm Direction</p> 
<p>For each pixel in the rest of the rows, the energy is its own energy plus the minimal of the three energies above. Repeat until the bottom is reached.</p>	<p>Algorithm Direction</p> 
<p>For the lowest energies we have at the end, work back up the minimals to recover the seam with minimal energy.</p>	<p>Algorithm Direction</p> 

The approach to content aware resizing is to remove pixels in a judicious manner, but, how to choose the pixels to be removed? The goal is to remove unnoticeable pixels that blend with the surroundings. This leads to the following energy function explained by the authors in their paper:

$$e_1(\mathbf{I}) = \left| \frac{\partial}{\partial x} \mathbf{I} \right| + \left| \frac{\partial}{\partial y} \mathbf{I} \right|$$

If we need to reduce the image width without distortion, the authors propose the following function, where  $\mathbf{I}$  is an  $n \times m$  image and a *vertical seam* is defined as:

$$\mathbf{s}^x = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, \text{ s.t. } \forall i, |x(i) - x(i-1)| \leq 1,$$

Where  $x$  is a mapping  $x: [1, \dots, n] \rightarrow [1, \dots, m]$ . That is, a vertical seam is an 8-connected path of pixels in the image, from top to bottom, containing one, and only one, pixel in each row of the image. Similarly, if  $y$  is a mapping  $y: [1, \dots, m] \rightarrow [1, \dots, n]$ , then a horizontal seam is

$$\mathbf{s}^y = \{s_j^y\}_{j=1}^m = \{(j, y(j))\}_{j=1}^m, \text{ s.t. } \forall j, |y(j) - y(j-1)| \leq 1.$$

greedy algorithm, graph cuts, or dynamic programming, which we explain in the next section.

Dynamic Programming is a programming method that stores the results of sub-calculations in order to simplify a more complex result and can be used to compute seams. If attempting to compute a vertical seam (or path) of lowest energy, the energy for each pixel in a row is computed, plus the energy of one of the three possible pixels above it.

We present the adaptation of a python program to implement seam carving. The original program was taken from [geekforgeeks.org](http://geekforgeeks.org).

#### 4.2.2 Lossless image-compression algorithm LZ77 Compressor: Lempel-Ziv 77

In a few words, what it does is look through the file you need to compress looking for sequences of characters that recur over and over again and will attempt to reuse them as much as possible. For example, if a text has the word “computerphile” somewhere, and the word “computer” somewhere else, the compressor will remember, using a pointer, where the word first occurred, and how long the word was.

This pointer has the form  $\langle j, l \rangle$ , where  $j$  is the number of characters to “jump-back” to find this word in the text, and how long ( $l$ ) will it be in that position. But these brackets are not in an LZ77 file. It is just our form to explain it. In

reality, it could be something like a 2 bytes (16 bits) word, with the number of characters to “jump back” in the first byte and the length of the word in the second, for example, (30,8) would mean go back 30 characters from here and read 8 characters for the word that goes here. But, if we use 2 bytes words, it means that we could go back a maximum of 255 characters (8 bits) and replace a word 255 characters long. As normally we do not have words this long, is better to split the 2 bytes in a first word of 12 bits for the number of characters to go back, and a second word of 4 bits for number of characters to look for. This gives 4096 characters to go back and 16 characters to look for.

In practice, this is done with a dictionary that relates where to find the substring that goes in some place in the text. Using this technique, it is possible to replace a full word any number of characters long for a two-byte word which points to where to find it in the text.

LZ77 is a lossless compression algorithm, which means that when you compress something and then decompress it, you get back exactly what you had.

### 4.3 Complexity analysis of the algorithms

To achieve a good balance between time and memory consumption, we first apply the seam carving algorithm and then the LZ77 to work with smaller images, thus the image will be a bit lossy. In the following we explain the complexity of each algorithm and the total

Algorithm LZ77	Time Complexity
Compression	$O(N^2 * M^2)$
Decompression	$O(N^2 * M^2)$

**Table 1:** Time complexity of image compression and decompression algorithms using LZ77. Where N is the number of rows and M the number of columns of the matrix containing the image pixels.

Algorithm LZ77	Memory Complexity
Compression	$O(N * M)$
Decompression	$O(N * M)$

**Table 2:** Memory complexity of image compression and decompression algorithms using LZ77. Where N is the number of lines and M the number of columns of the matrix containing the image pixels.

Seam Carving	Time Complexity
Compression	$O(M * N)$
Decompression	$O(M * N)$

**Table 3:** Time complexity of image compression and decompression algorithms using seam carving. Where N is the number of rows and M the number of columns of the matrix containing the image pixels.

Seam carving	Memory Complexity
Compression	$O(M * N)$
Decompression	$O(M * N)$

**Table 4:** Memory complexity of image compression and decompression algorithms using seam carving. Where N is the number of lines and M the number of columns of the matrix containing the image pixels.

### 4.4 Design criteria of the algorithm

For the design of the algorithm, we considered both its time and memory complexity, as well as the purposes for which the algorithm would be used, in this case for an artificial intelligence model which is trained to determine whether the livestock is sick or healthy according to the images.

In the case of the lossy image compression algorithm, seam carving was chosen because, firstly, considering the use of the image, rescaling wasn't ideal, as it removes objects from the image. Fractal compression also wasn't an option because it can cost more in terms of memory. Bilinear interpolation can have a similar memory consumption, but images are blurrier. Therefore, we chose a content aware algorithm, in this case seam carving, which leaves the most important information in the image. We thought this would be ideal for the AI models purpose and its training [9][7].

On the other hand, we chose the LZ77 as the lossless algorithm. Studying other algorithms, such as Huffman's algorithm, we found out it has a time complexity of  $O(N * M * \log(N * M))$  for both compression and decompression, so we could notice that although the compression is faster compared to LZ77, the decompression is slower and, it uses much more memory, although their compression rates are usually very similar. In the case of LZ78, we found this one was better in terms of memory consumption but took more time, and considering that the



priority was time, we didn't choose this algorithm. Finally, the Burrows-Wheeler algorithm has a time complexity of  $O(N*M*P)$  where  $N$  are the rows,  $M$  the columns of the matrix and  $P$  the longest repetition in the whole input string, so it can sometimes have a higher complexity than the LZ77 although its decompression in memory complexity  $O(N*M)$  is equal to the LZ77[13][15].

Finally, we found LZ77 to be a good algorithm and its results were quite optimal. And finally, we applied a combination of seam carving and LZ77 since, considering that the dataset had very large images, this would cause the compression to take much longer, so we decided to first reduce the size of the image using seam carving and then applying the LZ77. This ideally reduces the LZ77's running time. Although the images will have a little loss due to the seam carving, we consider that it will not affect the accuracy of the model much, especially considering that seam carving is a content aware algorithm.

It's important to add that we could have only used the LZ77 algorithm because the images aren't that big, however using seam carving first can optimize the process.

## 5. RESULTS

### 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.

	<i>Average execution time (s)</i>	<i>Average file size (KB)</i>
<i>Compression</i>	500s	121KB
<i>Decompression</i>	0.58 s	121KB

**Table 6:** Execution time of the joined seam carving and LZ77 algorithm. We took the average out of processing 10 color images. We took their execution time and then calculated the average

### 5.3 Memory consumption

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

	<i>Average memory consumption (MB)</i>	<i>Average file size (MB)</i>
<i>Compression</i>	634 MB	121KB

<i>Decompression</i>	9 MB	878.12 MB
----------------------	------	-----------

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

### 5.3 Compression ratio

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

	<i>Healthy Cattle</i>	<i>Sick Cattle</i>
Average compression ratio	3:2	3:2

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

	<i>Healthy Cattle</i>	<i>Sick Cattle</i>
Average compression ratio	3:1	3:1

**Table 9:** Rounded Average Compression Ratio of all the images of Healthy Cattle and Sick Cattle with seam carving.

## 6. DISCUSSION OF THE RESULTS

We consider that the memory and time consumption is appropriate considering the image sizes. The most significant result is that the LZ77 works faster after applying the seam carving algorithm. However, we think it would be appropriate studying seam carving furthermore to obtain better results. We could compress a 121 KB image to 300 bytes, however this still took longer than what would be ideal for multi image processing in an AI model.

### 6.1 Future work

We would like to improve the accuracy of the compression considering that we don't want to interfere with the execution of the AI model. We would also like to work on the model as well, meaning train an AI with data to determine whether the cattle are sick or not. A model that is conscious about time and space complexity would be ideal. We would also like the possibility of working not only with images, but also video compression of cattle.

## ACKNOWLEDGEMENTS

We would like to express our gratitude to Mauricio Toro who helped and supported us by explaining all the important concepts relevant to this project. Also, we want to express our thanks to Simon Marin and Kevin Lopez who supported us, teaching us new techniques and how to optimize each code.

## REFERENCES

- [1] Berckmans D. (2014). Precision livestock farming technologies for welfare management in intensive livestock systems. *Revue scientifique et technique (International Office of Epizootics)*, 33(1), 189–196. <https://doi.org/10.20506/rst.33.1.2273>
- [2] PIERS Online, Vol 5, No. 1, 2009: Kae Hsiang Kwong, Tsung Ta Wu, Hock Guan Goh, Bruce Stephen, Michael Gilroy, Craig Michie and Ivan Andonovic. Tomado de: Vol5No1Page31to35.pdf(piers.org)
- [3] Precision Livestock Farming: An international Review of scientific and commercial aspects (T M Banhazi, H Lehr, J L Black, H Crabtree, P Schofield, M Tschärke, D Berckmans. Tomado de <http://ijabe.org/index.php/ijabe/article/view/599/498>
- [4] Viazzi, Stefano & Bahr, C. & Schlageter Tello, Andres & Van Hertem, Tom & Romanini, C.E.B. & Pluk, Arno & Halachmi, Ilan & Lokhorst, C & Berckmans, Daniel. Analysis of individual classification of lameness using automatic back posture measurement in dairy cattle. *Journal of dairy science*, 2012, 96. 10.3168/jds.2012-5806.
- [5] Chao Zuo, Liang Han, Pin Tao, and Xiang lei Meng. 2020. Livestock Detection Based on Convolutional Neural Network. In 2020 the 3rd International Conference on Control and Computer Vision (ICCCV'20). Association for Computing Machinery, New York, NY, USA, 1–6.
- [6] DOI:<https://doi.org/10.1145/3425577.3425578> Wikipedia contributors. (2020, 7 November). Seam carving. Wikipedia. <https://en.wikipedia.org/wiki/Seamcarving>
- [7] Domingues, D., Alahi, A., & Vanderghenst, P. (2010). *Stream carving: An adaptive seam carving algorithm*. 2010 IEEE International Conference on Image Processing. doi:10.1109/icip.2010.5653984
- [8] Chun-Ho Kim, Si-Mun Seong, Jin-Aeon Lee and Lee-Sup Kim, "Winscale: an image-scaling algorithm using an area pixel model," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 6, pp. 549-553, June 2003, doi: 10.1109/TCSVT.2003.813431.
- [9] GeeksforGeeks. "Huffman Coding | Greedy Algo-3." *GeeksforGeeks*, 23 Mar. 2021, [www.geeksforgeeks.org/huffman-coding-greedy-algo-3](http://www.geeksforgeeks.org/huffman-coding-greedy-algo-3).
- [10] Ghent University. "Huffman Coding." *Dodona*, 2021, [dodona.ugent.be/en/activities/601074743](http://dodona.ugent.be/en/activities/601074743).
- [11] Wahba, Walaa Z., & Maghari, Ashraf Y. A. (2016). Lossless Image Compression Techniques Comparative Study. *International Research Journal of Engineering and Technology (IRJET)*, e-ISSN 2395-0056, Volume: 3, Number: 2, <http://hdl.handle.net/20.500.12358/25131>
- [12] Ahmed, Nasir; Natarajan, T. ; Rao, K.R. (January 1974), Discrete Cosine Transform, *IEEE Transactions on Computers*, C-23 (1): 90-93. doi: 10.1109/T-C. 1974.223784
- [13] May, Mike (1966). "Fractal Image Compression". *American Scientist*. **84** (5): 442-451. Bibcode: 1966AmSci..84..442
- [14] Saupe, Dietmar; Hamzaoui, Raouf (November 1994). "A review of the fractal image compression literature" *ACM SIGGRAPH Computer Graphics*. **28** (4): 268-276. doi:10.1145/193234.193246
- [15] Burrows, Michael; Wheeler, David J. (1994), *A block sorting lossless data compression algorithm*, Technical Report 124, Digital Equipment Corporation
- [16] Salson M , Lecroq T, Leonard M, Mouchard L (2009). "A Four-Stage algorithm for updating a Burrows-Wheeler transform". *Theoretical Computer Science*. **410** (43): 4350-4359. doi:10.1016/j.tcs.2009.07.016
- [17] Ziv, Jacob; Lempel, Abraham (May 1977). "A universal algorithm for Sequential Data Compression" *IEEE Transactions on Information Theory*. **23** (3): 337-343. CiteSeerX 10.1.1.118.8921. doi:10.1109/TIT.1977.1055714
- [18] Ziv, Jacob; Lempel, Abraham (September 1978). "Compression of individual sequences via Variable-Rate coding". *IEEE Transactions on Information Theory*. **24** (5): 530-536. CiteSeer X 10.1.1.14.2892
- [19] Andonovic, Ivan; Michie, Craig; Gilroy, Michael; Guan Goh, Hock; Hsiang Kwong, Kae; Sasloglow, Konstantinos; Wu, Tsungta. "Wireless sensor networks for cattle Health Monitoring". Centre for Dynamic Communication, Department of Electronic and Electrical Engineering, University of Strathclyde, 204 George Street, Glasgow, United Kingdom. doi: 10.1007/978-3-642-10781-8\_3
- [20] Smith, Kevin; Martinez, Angel; Craddolph, Roland; Erickson, Howard; Andresen, Daniel; Warren, Steve. "An Integrated Cattle Health Monitoring System", IEEE, (2006), doi 10.1109/IEMBS.2006.259693
- [21] Avidan, Shai & Shamir, Ariel. (2007). Seam Carving for Content-Aware Image Resizing. *SIGGRAPH*. 26. 10.1145/1276377.1276390.