Automated Counting of Wheat Spikelets and Grains Project Proposal

Osagioduwa Edo-Osagie

1 Introduction

This project aims to research and create a system to automatically count the number of grains in periodically captured images of wheat plants. Agriculture is arguably one of the most important disciplines in the world. It is important that food is grown on a large scale. Large scale farmers often have multiple sites with multiple groups of plants growing. This makes it difficult for them to simultaneously monitor and coordinate all of these plants. Farmers need good tools to improve their agricultural processes, in turn improving yield. Computer science and technology has been used to grow many other disciplines but only applied to agriculture to a smaller extent. This project aims to apply computer vision techniques to automatically monitor growth rates of wheat plants. Farmers could potentially be warned when irregularities occur. In addition to this, the results of the project could also be applied in a research setting to help biological researchers better monitor their experiments.

The project involves the design and implementation of a system to automate the monitoring of wheat growth with periodically captured images. The system will function as follows: Images are periodically captured automatically through an appropriate Raspberry Pi-powered setup [1](or an alternative setup). The images are then uploaded and stored in a database. The system can then analyze the images in the database, estimating the number of spikelets in each image and draw conclusions from its analyses.

1.1 Motivation

The problem is that it is not possible to automatically count the number of grains in an image of a (wheat) plant. At the moment, it is only either a fully or partially supervised process, requiring human intervention to a significant degree. There is no way to simply create a picture of the plants' progress without any human intervention.

While it may not seem like a big deal, the problem could yield great reward if solved. Also, not only does the field of agriculture have something to gain from the success of the project but also Biology, Botany, Food Sciences and basically anything to do with plant research. The resulting system *could* be applied and extended to automatically monitor the growth/yield of (wheat) plants and raise alerts if anomalies occur. The system would function with minimal supervision. If solved, the resulting system could save farms and research institutions a lot of time and resources which can then be channeled to other pressing areas.

[1]http://www.instructables.com/id/Easy-Raspberry-Pi-Security-Cam-With-Automatic-Web-/

1.2 Aims and Objectives

The aim of this project is to develop/improve machine learning based algorithms to analyze wheat images and give a count the number of grains in the image. Its objectives are as follows:

- Apply image processing techniques to highlight and segment spikelet regions.
- Count the number of grains in each of the segmented regions.
- Make a record of the wheat plant's progress so far.

2 Literature Review & Related Work

2.1 The Counting Problem

A prevalent theme in the field of Computer Science is that of performing tasks traditional to humans, through the use of a computer. Counting is a trivial cognitive task for humans. In fact, it comes so naturally that studies have shown that children below the age of 3 demonstrate an understanding of numerosity without any knowledge of number or counting systems (Sella et al. (2015)). For a computer, the task of estimating the number of objects (of some kind) in an image is slightly more complicated and comes up in many real world applications such as counting cells in microscopic images and monitoring crowds in surveillance systems.

The most established ways of counting objects in images involve the use of image processing or computer vision techniques in conjunction with machine learning methods. While there are are many different approaches which employ different techniques to achieve these goals, they all share (to some extent) the same general outline. The image is first transformed to greyscale. The greyscale image is thresholded to yield a binary image. The binary image is segmented and the features to be used for counting are extracted and passed into the model. Figure 1 shows a generalized flowchart for the activities involved in counting objects in images with the stages described as follows:

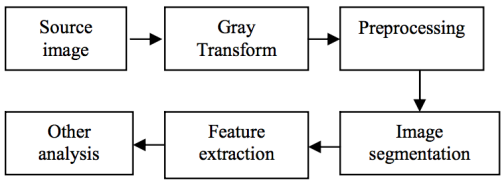

Figure 1: Generalized pipeline for approaching the counting problem

(i) Gray Transform: The image is first transformed to greyscale. This is because counting usually depends on morphological (ie. shape), spatial and textural features that have no use for colour. The exclusion of colour simplifies subsequent calculations and makes the whole process more computationally efficient. Also, a lot of the techniques used in processes further down the pipeline are designed to work on simple

two-dimensional matrices and would be very difficult, if not impossible, to extend to colour images. There are various ways to achieve this conversion. One way is to simply average the RGB pixel values to give the greyscale pixel value (Cook (2009)). A more accurate method of conversion is the use of a luminance-preserving mapping. This method gives the greyscal pixel value, given as the *luminosity* of the pixel, to be calculated as a weighted sum of the RGB pixel values (Anderson et al. (1996)).

$$Y = 0.2126R + 0.7152G + 0.0722B \tag{1}$$

Applying the formulae described above to each pixel in a colour image will result in a grayscale vesion of the image.

(iii) Preprocessing: The presence of noise in the image will affect the counting accuracy. Because of this, certain transformations must be carried out on the image before going further. Also, depending on the objects to be counted in an image, some processes can be performed on the image in order to highlight or de-emphasize appropriate elements of the image to make detection and counting easier and more reliable. Image smoothing and sharpening are usually used to achieve this. Image smoothing can be carried out by applying a *Gaussian filter* (Kopparapu and Satish (2014)) or a *median filter* (Church et al. (2008)). However, in the process of removing noise from the image, image smoothing might also blur some of the edges in the image. Blurred edges are not conducive to segmentation, analysis and other potential follow-up processes. Hence, image sharpening is performed on the image in order to emphasize the edges in the image. Image sharpening can be carried out by applying a *high-pass filter* (Makandar and Halalli (2015)).

(iii) Binary Image Segmentation: Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics (Wikipedia (2016)). With binary image segmentation, which is what is usually used in object detection and counting, there is only one label - "edge". Pixels that form part of an edge are assigned a value of 1 and any other pixel is assigned a value of 0. Given this, image segmentation in this sense can be performed by applying an edge detection algorithm to the image. A popular edge detection algorithm used for image segmentation is the *Canny edge detection* algorithm (Canny (1986)). In simple terms, the Canny edge detection algorithm finds edges where the grayscale intensity of the image changes. These areas and changes can be approximated by calculating the gradient G in the image in the x and y directions.

$$\begin{aligned} |G| &= \sqrt{G_x^2 + G_y^2} \\ |G| &= |G_x| + |G_y| \\ \theta &= arctan(\frac{|G_x|}{|G_y|}) \end{aligned} \tag{2}$$

(iv) Feature Extraction: Feature extraction plays a very important role in the area of image processing with regards to machine learning. Feature extraction is a type of

dimensionality reduction that efficiently represents interesting parts of an image as a compact feature vector (Mathworks (2014)). These extracted vectors can then be used as inputs and elements of machine learning algorithms and techniques. The approach of using the extracted feature vectors instead of the entire image proves advantageous when image sizes are large. Using a reduced representation of the image makes it possible to perform many common and important tasks quickly. Common feature extraction techniques include Histogram of Oriented Gradients (HOG) (Dalal and Triggs (2005)), Speeded Up Robust Features (SURF) (Bay et al. (2006)), color histograms (Swain and Ballard (1992)) and Scale-InvariantFeature Transform (SIFT) (Lowe (1999)).

(v) Analysis: In this stage, the extracted feature vector representation of the image has statistical models or machine learning algorithms applied to it in order to recognize, detect or count objects within the given image. The objective of the analysis stage is to learn some useful information from the image, and most times, all of the preceding stages are carried out solely in preparation for this stage.

2.2 Existing Approaches to Counting

A lot of research has been carried out on counting objects in images. However, the objects being considered for counting differ from project to project. Because of this, there are a lot of different ideas and approaches to solving this problem. While they all seem to employ machine learning, some of the ideas take an unsupervised learning approach while others take a supervised learning approach. Supervised learning involves training a model with labeled instances of the subjects being investigated. Examples of these methods are machine learning classifiers. In unsupervised learning, the system is not introduced to any labeled training instances but learns on the job. An example of this is *clustering*.

A few approaches to solving the counting problem employ unsupervised learning by clustering or grouping similar aspects of the image. (Ahuja and Todorovic (2007)) propose a method for grouping based on self-similarities and (Rabaud and Belongie (2006)) propose a method for grouping based on motion similarities. However, the counting accuracy of such fully unsupervised methods is limited, and therefore others considered approaches based on supervised learning. These approaches fall into the following categories:

2.2.1 Counting by Detection

These approaches solve the problem by first detecting the objects in question and then counting them. Object detection is carried out by the use of a visual object detector that localizes individual instances of the object in the image. Given the localizations of the objects' instances, counting becomes trivial. However, visual object detection is far from being solved and is even more tricky when overlapping instances are involved. Several methods assume that objects are uniform and disconnected from each other by the distinct background color. Following this assumption, it is possible to localize individual instances via the Monte-Carlo process (Descombes et al. (2009)) or morphological analysis (Anoraganingrum (1999)). These methods yield reliable results when the assumption is met but are not so applicable in challenging real life situations.

2.2.2 Counting by Regression

These approaches sidestep the hard detection problem altogether and instead, attempt to form a direct mapping from some features or groups of features of the image to the number of objects contained within it. This is a standard regression problem that can be addressed by many different machine learning tools such as neural networks. This approach however has to discard any available information about the location of the objects, using only its 1-dimensional statistics (total number) for learning. As a result, a large number of training images with the supplied counts needs to be provided during training (Lempitsky and Zisserman (2010)).

2.3 Learning and Modeling

In order to solve the counting problem, some way of actually determining the number of objects present in a preprocessed representation of an image has to be established. This is usually done by building a model to be used to perform this determination using machine learning techniques. As explained above, supervised learning models yield better results and perform better in practice. A supervised learning model is built from previously encountered and labeled instances and is then applied to subsequently encountered instances.

One possible tool for building such models is the *support vector machine* (SVM). The Support vector machine is a machine learning tool that is usually used for classification tasks but can be extended to regression tasks as well. An SVM model is a representation of the training examples as points in space, mapped so that the examples of the separate classes are separated by a hyperplane (or set of hyperplanes in high dimension problems) while maximizing the distance from the hyperplanes to the nearest training point. Given a set D of n training points,

$$D = \{(x_1, x_2) \dots (x_n, y_n)\} \tag{2}$$

with a hyperplane described by the equation

$$\langle w, x \rangle + b = 0 \tag{3}$$

To minimize the margin of the hyperplane, the idea is to solve the optimization problem

$$\text{minimize } ||w|| \text{ such that } y_i(\langle w, x \rangle - b) \geq 1, i = 1 \dots n \tag{4}$$

For non-linearly separable scenarios, a loss function may be applied to the SVM. Usually the hinge-loss function is used for this. SVMs can be extended to regression problems by the introduction of an alternative loss function (Smola et al. (1996)). The alternative loss function must be modified to include a distance measure. Some possible loss functions are the quadratic function, laplacian function and Huber function.

Another machine learning tool that can be used for modeling is the *artificial neural network*. Artificial neural networks are a family of machine learning models inspired by the way that biological nervous systems, such as the brain, process information. A neuron is a cell that has several inputs that can be activated by some outside process. Depending on the amount of activation, the neuron produces its own activity and sends this along its

outputs (Stergiou and Siganos (1996)). In addition, specific input or output paths may be "strengthened" or weighted higher than other paths. The artificial neural network equivalent of a neuron is a ***node***. A node receives a set of (weighted) inputs, applies its activation function ϕ to their sum, and passes the result of the activation function to nodes further down the network. The activation function can thus be represented as such

$$\phi = \sum_{i=1}^{n} w_i.a_i \tag{5}$$

Many such nodes are chained together to form a network, passing the outputs of their activation functions along. The network is trained with labeled data by feeding inputs into the network and fine-tuning the weights until the network always yields the expected class label as its output. This is not very different from regression in that parameters are being tuned to create a function that yields certain values. Hence, artificial neural networks can easily be adapted to solve regression problems.

3 Proposed Approach

The proposed approach to solving this problem is to use a "counting-by-regression" approach to build a model which can estimate the number of grains in a given image. This approach was chosen over one using "counting-by-detection" because of the nature of the contents of the image. Counting-by-detection methods usually assume that objects are uniform and disconnected from each other by a distinct background. The images in our problem contain bushes of wheat plants with the wheat plants overlapping and occluding each other. This makes the problem very difficult for counting-by-detection methods.

This approach evades the complications brought on by trying to detect individual spikelets and grains by instead, using a regression approach. The high-level idea of the approach is extremely simple: given N training images with their grain counts provided, the goal is to recover a grain density function F as a real function of pixels in the images. The grain density function learns the relationship between the density of spikelets in areas of the image and provides a mapping between said density and the number of grains in the image. New images can then be provided as input to F and an estimate of the number of grains in the provided image returned as output.

The grain density function (which serves as the predictive model) is built by constructing feature vectors from the textural properties of the images. The vectors are used to build a regression model employing ensemble learning. The framework is described in more detail in the following sections.

3.1 Preprocessing

The first step in the proposed approach is preprocessing which involves applying image processing techniques to the wheat images in order to highlight the areas of interest (grains and spikelets) and de-emphasize other areas before applying the framework to the images. This is intended to make the system more reliable. First, the image is smoothed by applying a Gaussian filter to it (Kopparapu and Satish (2014)). This will reduce noise in the image. This is particularly important because arbitrary specks in the image could

be mistaken for grains and it would be best to minimize such occurrences. However, smoothing usually blurs the edges of objects within the image. To make up for this, image sharpening is subsequently performed on the wheat images by applying a high-pass filter to them (Makandar and Halalli (2015)). This will re-emphasize the edges in the images while maintaining it in a state of reduced noise. Next, binary image segmentation is performed on the images by means of the *Canny edge detection* algorithm. This process isolates the boundaries of the wheat plants, yielding an outline of the stalks, spikelets and grains. The segmented binary image is then superposed with the smoothed and sharpened full image, resulting in a new image with the important areas visibly highlighted. Figures 2, 3 and 4 show an illustration of the process.

Figure 2: Wheat image

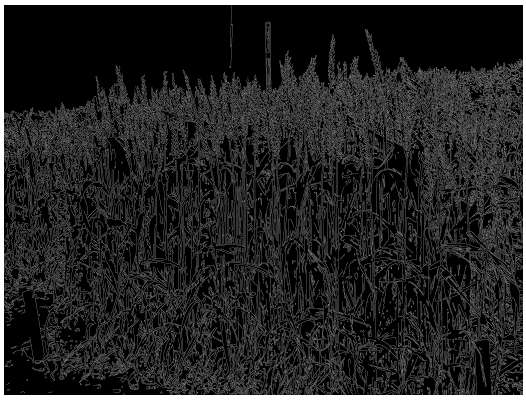

Figure 3: Segmented wheat image

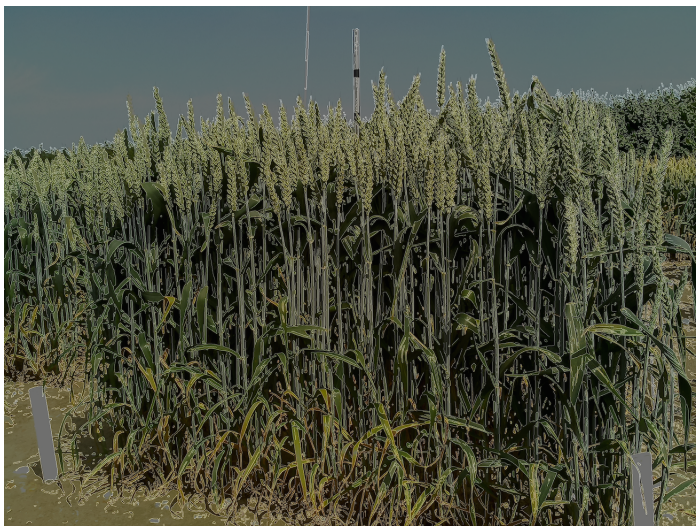

Figure 4: Superposed image

3.2 Texture Analysis for Feature Extraction

Image texture analysis is a process which applies certain techniques to extract texture parameters from an image in order to quantify or qualify its perceived texture. Image texture gives us information about the spatial arrangement of color or intensities in an image or selected region of an image (Singh et al. (2005)). Generally speaking, textures are complex visual patterns that have characteristics such as brightness, colour and contrast. Because of this, texture is easily perceived by humans and is believed to be a rich source of

visual information. Thus, texture is can be used as a metric for similarity between images.

The proposed approach makes use of ***Gray Level Co-occurrence Matrix (GLCM)*** texture analysis. A co-occurrence matrix can be defined as the distribution of co-occurring values at a given offset. It reflects the luminance distribution (as well as position distribution) between the same pixels or pixels with similar luminance values. The Gray Level Co-occurrence Matrix (GLCM) is formed from a gray-scale image and calculates how often a pixel with gray-scale intensity i occurs (horizontally, vertically or diagonally) adjacent to pixels with gray-scale intensity j. Simply put, a GLCM characterizes an image by calculating how often pairs of pixels with specific values (and in a specified spatial relationship) occur in an image, where the spatial relationship could be horizontally adjacent, vertically adjacent or diagonally adjacent. The spatial relationship between pixels can be denoted using direction θ and distance d. For example, using this notation, the spatial relationship between two pixels i and j that are 3 pixels apart diagonally could be denoted as $(d, \theta) = (3, 45°)$.

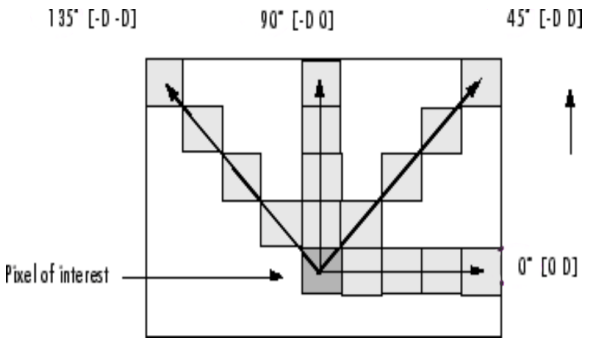

Figure 5: Illustration of GLCM spatial analysis (Hayes (2007))

Below are the textural descriptors that can be derived from a GLCM p with size M by N:

- *Energy*

$$ASM = \sum_{i=1}^{M} \sum_{j=1}^{N} p(i, j|d, \theta)^2$$

  is computed as the sum of squared elements in the GLCM. It is also known as *uniformity*, *uniformity of energy*, and *angular second moment* and it represents texture coarseness.

- *Contrast*

$$CON = \sum_{i=1}^{M} \sum_{j=1}^{N} |i - j|^2 p(i, j|d, \theta)^2$$

  returns a measure of the intensity contrast between a pixel and its neighbor over the whole image. It is also known as *variance* and *inertia*.

- *Homogeneity*

$$HOM = \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{p(i,j|d,\theta)}{1+|i-j|}$$

represents the uniformity of the image and measures the change of local image texture.

- *Correlation*

$$CORR = \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{(i-\mu_i)(j-\mu_j)p(i,j|d,\theta)}{\sigma_i \sigma_j}$$

Returns a measure of how correlated a pixel is to its neighbour over the whole image.

In the proposed system, each of these descriptors is computed for when $\theta = 0°, \theta = 45°, \theta = 90°$ and $\theta = 135°$ and the distance, $d = 0$ to build a 16-dimension feature vector describing the image.

3.3 Linear Regression Analysis for Count Estimation

The prediction model is developed by performing linear regression analysis on the set of extracted texture-based features. The model learns the relationship between the textural features of the image and the number of grains (and potentially spikelets) in the image. Regression analysis is a classic statistical method, and its basic idea is that given some value y and another value x, which is a property of y, a function $y = f(x)$ is derived. This function f can then be used to determine the value of $y\prime$ given $x\prime$, where $x\prime$ is a newly encountered value of x and $y\prime$ is its corresponding value of y.

The proposed framework makes use of a linear regression model with the count as the dependent variable and for each of the 4 texture descriptors in the 16-dimension feature vector, selects the most important values as independent variables. Recall that for each descriptor (*energy*, *contrast*, *homogeneity* and *correlation*), the value of the descriptor is computed in 4 different spatial arrangements (d, θ). This gives the values ASM_{00}, ASM_{01}, ASM_{11}, ASM_{10}, CON_{00}, CON_{01}, CON_{11}, CON_{10}, HOM_{00}, HOM_{01}, HOM_{11}, HOM_{10}, $CORR_{00}$, $CORR_{01}$, $CORR_{11}$ and $CORR_{10}$. Before building the regression model, Principal Component Analysis (PCA) is applied to the four values for the arrangement of each descriptor to yield one representative value for each texture descriptor giving ASM', CON', HOM' and $CORR$'. An artificial neural network is applied as described in the previous section to build the regression model. The linear regression model is of the form:

$$y = \alpha ASM' + \beta CON' + \gamma HOM' + \delta CORR' \tag{6}$$

where α, β, γ and δ are constants calculated by applying the artificial neural network to the set of training feature vectors and y is the count of grains (or spikelets) in the image.

3.4 Further Considerations

The reliability of the proposed framework could be improved by employing ensemble learning to build the model (grain density function). Ensemble learning is a machine learning technique that uses a set of models, each of which is obtained by applying a learning process to a given problem. This set of models - known as an ***ensemble*** - is integrated in some way to obtain the final prediction (Mendes-Moreira et al. (2012)). For this application, a *Support Vector Machine* (SVM) and an artificial neural network could be used to build the ensemble. These techniques are chosen because they are well suited to mathematical modeling. For instance, artificial neural networks automatically handle nonlinearities, which one would need to explicitly model using transformations (splines etc.) in linear regression. SVMs can also model nonlinearities without too much effort using the kernel method. The SVM and ANN could both be trained with all of the training images or each of them could be trained with a disjoint subset of the training images. The result of the ensemble would then be calculated as the weighted average of the outputs of its constituent models.

4 Project Plan

4.1 Timeline

The project will be carried out over the space of 31 weeks. Figure 6 shows a Gantt Chart showing the timeline of activites that will be carried out in the project.

4.2 Issues, Challenges and Risks

The wheat images are very dense in the sense that they contain many stalks cramped together, with some stalks partially or wholly occluding others. This will make it very difficult to get an exact count of the spikelets. Because of this, it is more likely that an estimate of the spikelet count will be obtained rather than an exact count. The proposed framework is intended to make the estimate as accurate and precise as possible.

Also, because the framework uses a supervised learning approach to identify and count grains and spikelets, a lot of images will need to be collected to be used to train the system. In addition to the images that will need to be collected for training, more images would need to be collected for testing and evaluating the implemented framework as well. This makes a certain amount of field work necessary to properly implement the framework.

5 Conclusion

In this paper, a framework for counting the number of grains and spikelets in images of wheat plants is proposed. The proposed framework calculates estimates of grain and spikelet counts from the textural properties of the image. The textural properties of wheat images labeled with their grain and spikelet counts are used to build and train a regression model using an artificial neural network. The number of grains and spikelets in subsequent images can then be estimated by applying the built model to their textural

Figure 6: Project Gantt chart

features. In future work, investigations will be made into building the regression model using ensemble learning in order to improve the reliability of the results.

References

Ahuja, N. and Todorovic, S. (2007). Extracting texels in 2.1 d natural textures. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

Anderson, M., Srinivasan, C., Motta, R., and Stokes, M. (1996). A standard default color space for the internet": srgb. Technical report, Technical report, International Color Consortium.

Anoraganingrum, D. (1999). Cell segmentation with median filter and mathematical morphology operation. In *Image Analysis and Processing, 1999. Proceedings. International Conference on*, pages 1043–1046. IEEE.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer.

Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698.

Church, J. C., Chen, Y., and Rice, S. V. (2008). A spatial median filter for noise removal in digital images. In *Southeastcon, 2008. IEEE*, pages 618–623. IEEE.

Cook, J. D. (2009). Three algorithms for converting color to grayscale. *http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.

Descombes, X., Minlos, R., and Zhizhina, E. (2009). Object extraction using a stochastic birth-and-death dynamics in continuum. *Journal of Mathematical Imaging and Vision*, 33(3):347–359.

Hayes, J. (2007). Image classification - gray level co-occurrence matrix (glcm). *http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Hayes_GreyScaleCoOccurrenceMatrix.pdf*.

Kopparapu, S. and Satish, M. (2014). Optimal gaussian filter for effective noise filtering. *arXiv preprint arXiv:1406.3172*.

Lempitsky, V. and Zisserman, A. (2010). Learning to count objects in images. In *Advances in Neural Information Processing Systems*, pages 1324–1332.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.

Makandar, A. and Halalli, B. (2015). Image enhancement techniques using highpass and lowpass filters. *International Journal of Computer Applications*, 109(14).

Mathworks (2014). Feature extraction for compact representation of image data in computer vision. *http://uk.mathworks.com/discovery/feature-extraction.html*.

Mendes-Moreira, J., Soares, C., Jorge, A. M., and Sousa, J. F. D. (2012). Ensemble approaches for regression: A survey. *ACM Computing Surveys (CSUR)*, 45(1):10.

Rabaud, V. and Belongie, S. (2006). Counting crowded moving objects. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 705–711. IEEE.

Sella, F., Berteletti, I., Lucangeli, D., and Zorzi, M. (2015). Spontaneous non-verbal counting in toddlers. *Developmental science*.

Singh, K., Ma, M., Park, D. W., and An, S. (2005). Image indexing based on mpeg-7 scalable color descriptor. In *Key Engineering Materials*, volume 277, pages 375–382. Trans Tech Publ.

Smola, A. J. et al. (1996). Regression estimation with support vector learning machines. *Master's thesis, Technische Universit at M unchen*.

Stergiou, C. and Siganos, D. (1996). Introduction to neural networks. *https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#Introduction%20to%20neural%20networks*.

Swain, M. J. and Ballard, D. H. (1992). Indexing via color histograms. In *Active Perception and Robot Vision*, pages 261–273. Springer.

Wikipedia (2016). Image segmentation. *https://en.wikipedia.org/wiki/Image_segmentation*.