Article:

Binch, A and Fox, CW orcid.org/0000-0002-6695-8081 (2017) Controlled comparison of machine vision algorithms for Rumex and Urtica detection in grassland. Computers and Electronics in Agriculture, 140. pp. 123-138. ISSN 0168-1699

https://doi.org/10.1016/j.compag.2017.05.018

Controlled comparison of machine vision algorithms for Rumex and Urtica detection in grassland

A Binch [a,c], CW Fox [b,c]

[a] School of Computing, [b] Institute for Transport Studies,

The University of Leeds, LS2 9JT, United Kingdom.

[c] Ibex Automation Ltd, S35 7DQ, United Kingdom.

Corresponding author: C.W.Fox@leeds.ac.uk

Submitted to Computers and Electronics in Agriculture

May 16, 2017

Abstract

Automated robotic weeding of grassland will improve the productivity of dairy and sheep farms while helping to conserve their environments. Previous studies have reported results of machine vision methods to separate grass from grassland weeds but each use their own datasets and report only performance of their own algorithm, making it impossible to compare them. A definitive, large-scale independent study is presented of all major known grassland weed detection methods evaluated on a new standardised data set under a wider range of environment conditions. This allows for a fair, unbiased, independent and statistically significant comparison of these and future methods for the first time. We test features including linear binary patterns, BRISK, Fourier and Watershed; and classifiers including support vector machines, linear discriminants, nearest neighbour, and meta-classifier combinations. The most accurate method is found to use linear binary patterns together with a support vector machine.[1]

[1] This research was supported in part by the InnovateUK project IBEX2: Autonomous robot weed spraying for less favoured areas, grant number 131790.

1 Introduction

Automated robotic weeding of grassland will improve the productivity of dairy and sheep farms while helping to conserve their environments. At present grassland weeding is typically performed in two styles. Tractor-mounted bulk spraying of selective herbicides is expensive due to the volume, and cost per unit of selective chemicals. Manual backpack-mounted spraying uses lower, targeted spot spray doses of generic herbicides such as glyphosate, but requires more expensive manual labour time. Precision robots [4, 6] present an opportunity to use similarly low and targeted doses of cheap generic herbicides as in the manual case, but at much lower cost as they can drive, detect and spray automatically without the need to pay manual sprayers by the hour. Precision robots could further eliminate the need for chemical herbicide altogether by destroying detected weeds with mechanical or other non-chemical methods. Data collected about weed locations by robots can be fed into geospatial weed mapping systems to enable ecological analyses.

Autonomous weeding robots must first detect weeds at a suitable resolution and accuracy. Machine vision provides many tools and algorithms for detection, with varying performances, and can be cheap to operate using consumer-grade cameras. Several previous studies have reported results of machine vision methods to separate grass from grassland weeds, typically *Rumex obtusifolius* (dockleaf). But as is common in early stages of artificial intelligence research, they each use their own datasets and report only performance of their own algorithm rather than presenting controlled trials testing methods against one another. As proof of concept studies, many used only small data sets, did not report confidence intervals on their accuracy rates, and have not yet tested methods across lighting and weather conditions which are known to affect many vision algorithms. It is well known in artificial intelligence that unintended author bias and bottom drawer effects can creep into studies when the same author both designs and tests an algorithm, so there is a need for independent validation. We present a large-scale (tens of thousands of images), independent study of all major grassland weed detection methods evaluated on a new standardised data set, under a wider range of environment conditions. This allows for a fair, unbiased, independent and statistically significant comparison of the methods for the first time.

1.1 Previous work

Previous studies can be grouped roughly into those which classify individual windows (patches) of images (e.g. tens of pixels square) independently of one another, and those which apply morphological operations to whole images (e.g. hundreds or thousands of pixels width and height). Window-based methods compute features of the windows such as spectra or texture descriptors, while whole-image methods try to isolate shapes via segmentation algorithms. Window-based methods include: [10] used local binary pattern (LBP) texture features with a per pixel threshold Rumex/Grass classifier, under controlled artificial lighting conditions, to report between 87%-97% accuracy on a test set of 941 images of 50x40 pixels. [2] uses very large windows to obtain high accuracy, 98.5%, using texture features and support vector machine (SVM) classification. Whole-image segmentation methods include: [12] segmented images into regions of similar texture then classified the shapes of these regions, reporting 71%-95% accuracy in Rumex/(Grass and mixed herbs) classification under constant lighting conditions. [27] used thresholded and segmented Fast Fourier Transforms (FFT) to detect Rumex in grass on 161 images, reporting 94% accuracy. [33] used a similar setup to report accuracies 82%-89% for Rumex/Grass. [31] used segmentation (erosion and dilation). [17] use Gray Level Co-occurrence Matrix and Laws' filter mask texture features with linear discriminant analysis (LDA) and segmentation to report 90% Rumex/Grass. [18] use Markov Random Field based texture features and segmentation to report a 97.8% accuracy on 92 images. A summary of the key properties of each method is given in table 1 (plant types are R=Rumex, G=Grass, U=Urtica, H=mixed herbs).

study	method	classes	reported accuracy	number of test images	illumination	window size
[10]	LBP+threshold	R/G	87%-97%	941	artificial	50 × 40
[12]	Segment+shape	R/(G+H)	71%-95%	3681 windows	constant	n/a
[27]	FFT+segment	R/G	94%	161 images	constant	8
[33]	FFT+segment	R/G	82%-89%	56 images	constant	8
[31]	Segment	R/G	89%	240 images	constant	n/a
[17]	GLCM+LDA+segment	R	90%	92 images	constant	n/a
[18]	MRF segment	R/G	97.8%	92 images	constant	n/a
[2]	LBP+SVM	R	98.5%	400 images	varied	320 × 240

From the table, it is clear that making a fair comparison of these algorithms is difficult or impossible from publicly available data. Each study uses its own data sets, comprising completely different images and conditions. In many cases the separation of training and test data is not clear, with studies reporting best results having optimised parameters over the same test set used in the final result, rather than making a clean train/test separation. It is well-known [11] that optimising parameters to the test set tends to yield over-optimistic results compared to performance on new data. Some studies do not describe the variation in the lighting conditions, but are assumed to be constant conditions because they use small numbers of test images collected, presumably, on the same day. Window-based methods have used different window sizes, while whole-image segmentation methods make use of data from across the whole image to classify each local pixel, which is hard to meaningfully evaluate against windowed results. Window size is important because it represents a fundamental trade-off between detection accuracy and spatial resolution. A large window contains more information which will yield high accuracies, but at the cost of a lower spatial resolution, for example in determining what area of ground to spray with herbicide.

In our native UK grassland, Rumex is not the only common weed and almost always co-occurs with similar populations of *Urtica diotica* (stinging nettle). As such, any automated grassland weeding system needs to work with both Rumex and Urtica together. If Rumex only was precision sprayed, then a selective bulk spray for Urtica would still be needed which negates the utility of the Rumex precision system, as combined Urtica and Rumex selective chemicals are available. Previous work on automatic detection of Urtica in grassland has relied on non-visual spectral methods including near infra-red and full hyper-spectra [23, 24, 38], but these sensors are more costly than simple visual cameras. Urtica has smaller leaves than Rumex which makes it harder to detect with machine vision alone, in particular some Rumex detection methods rely entirely on obvious features of the large, smooth Rumex leaves, which may not carry over to the Urtica case. However Urtica has distinctive jagged edges on its leaves which suggest that methods based on such local shapes (rather than texture) features may be useful for detection.

With the exception of [2], all the systems in table 1 rely on vertical camera angles, i.e. cameras mounted to look directly downwards at the ground. This simplifies recognition as there is no perspective, and all parts of the ground look the same. However this imposes physical limitations on precision robots, which must either mount a camera physically outside the robot's base footprint, or inside the body of the robot looking directly under its base. Much UK grassland is found in less-favoured areas, including hilly and rocky terrain such as sheep farms. These terrains often include obstacles which robots must navigate around, and such navigation is complicated by physical extrusions beyond robot platform bases such as cameras on arms or beams. Similarly, robots designed for these terrains may need heavy, protective bases which prevent cameras or sprayers from being mounted directly downwards from them. To generalise operation beyond flat grassland to cases such as these, it is more convenient and lower-cost to use more standard camera mountings on top of the robot body, with cameras facing forwards and tilted down, as in [2]. While this is a more robust physical solution, it makes the machine vision problem harder as it must now deal with perspective. Further, the previous studies all use clean images taken by stationary cameras to ease recognition. In practice, precision robots operating in generalised terrains will be moving at speed, capturing images during motion. It is not practical to stop every time an image must be taken. While camera stabilisation systems are available at a cost, grassland and especially hill farmers typically require lower budget solutions than arable farmers, so it is of interest to test algorithms on data collected from similar moving robot platforms as would be used in practice, which can include motion blur.

Recent work has begun to explore the use of 3D lidar based sensing and detection of Rumex in grassland [3, 29, 30]. While a similar independent evaluation of such data would also be of interest, it is beyond the scope of the present machine vision study. Also beyond our scope are non-visual approaches to weed detection including hyper-spectral [23, 24, 38], and chemical sensing methods [26]. Our scope of detection of weeds in grassland is a particular sub-field of automated weed detection in general, which has developed a wider range of methods applicable to simpler cases of crops and weeds growing in flat, row-crop settings, which can typically simplify the task by initial segmentation into green and brown discrete plant and soil regions, unlike the grassland case where everything is green [7, 14, 21, 32, 36, 36].

1.2 Data and algorithm requirements

To make a fair and useful comparison between the different algorithm types proposed for UK, less-favoured area grassland weed recognition, and to extend the robustness of previous studies, the following requirements were taken into account. 1. Data should be clearly split into training and test sets. 2. Only a single run should be allowed of each algorithm on the test set. 3. The algorithms should be implemented and tested independently of their original proposers. 4. testing should be on Urtica as well as Rumex. 5. Data should be collected from a moving, robust platform, with cameras mounted on top of its body and pitched downwards from the plane. 6. Data size should be in the order order of thousands of image windows. 7. Classification should be performed on standard sized windows, including for morphological methods which should be restricted to run on the same windows as feature-based methods. 8. Windows should be of a suitable spatial size and resolution to enable precision spraying. 9. Data should be collected over a representative variety of different days, illumination, and weather conditions.

Taken together, these requirements are more challenging than settings used in the previous studies. An independent evaluation should not seek to 'sell' any one algorithm with high rates, and should not shy away from reporting low accuracies when they occur. This helps to avoid any publication bias [28] which may have acted as a filter on previous tests. Due to interactions between the requirements and differences in data types, we do not re-implement algorithms directly but instead use similar or closely related methods. This is required in particular for the morphological approaches which do not transfer directly to window classification, such as the use of watershed segmentation to represent region based methods previously run on whole images. We test classifiers that are based on and inspired by the collection of previous studies as a whole rather than directly re-implementing and competing between them.

2 Methods

The objective of the experiments is to report, to a statistically significant level, the classification performance of various classification methods for grass vs weed detection, i.e two-way classifications representing spray/no-spray decisions for a general herbicide. In general this is distinct from the problem of recognising individual weed species. A 'classification method' or 'method' means a combination of one feature type with one classifier type. Care must be taken to avoid contamination of classifier training with any information from test data.

2.1 Image acquisition and pre-processing

Test plots of weeds in grass were constructed on a dairy grassland farm in South Yorkshire, UK. Slabs of Urtica approximately 0.2m squared were extracted from working fields and transplanted into a 3m squared trench (Figure 2a). This process was repeated for Rumex. Transplanting real slabs from areas of the working farm ensures maximum realism and avoids problems of growing the weeds artificially, which could lead to unrealistic soil backgrounds in images. In particular, the transplanted slabs also contain grass, soil, rocks, and other surface features of the real working grassland farm, though only in 0.2m squared slabs which human transplanters considered to be fully 'sprayable'. To make this decision, the human transplanters were instructed to collect only slabs which they would be happy to completely spray with a manual backpack sprayer if they were being employed to manually destroy weeds. The weedy turfs were watered daily for two weeks to allow the plants to stabilise before data collection. Plots were located in a region of the farm that is in direct sunlight (not in shade) throughout the day.

Stereo pair images[2] were acquired in 1080HD from each plot using auto-focusing cameras[3]

[2] Only the mono, left camera images are used in this study. Stereo images were captured for use in future comparison studies and are also made available as part of the dataset.

[3] After evaluating several industrial and consumer cameras, a pilot experiment determined that the C920 is sufficient and lowest cost for our purpose, having sufficient depth of field to cover the region of interest, and default shutter speed sufficient to give sharp images when driving the robot at around 4km/h. This camera's auto-focus is also useful as we operate on somewhat uneven terrain, sufficient to sometimes blur images with a static focus. The auto-focus explores and adjusts different settings of the focus and chooses the one which minimises the blur (maximises entropy) near the center of the image, before taking and saving the final image. 1080HD (=2megapixels) was used as we assume that real-time processing above this resolution is difficult with currently available on-board hardware. Manual inspection of the pilot images suggested that distortion from depth-of-field

(C920, www.logitech.com) mounted on a tracked robot as in fig. 1. The cameras were fixed to the robot's left side, facing out at right angle yaw to the direction of travel, and at $\pi/8$ radians (22.5 degrees) pitch down, to give a view over a roughly 1m square area of ground. The robot drove in circles around the plots whilst capturing pictures, randomised between 0m and 1m from the edge of the plots (figure 2b). This guarantees an equal balance of lighting and shadow angles in the data, because each drive around the plot contains images of the plot from all ground angles. The size of the plots, robot, and camera positions were selected such that the plot contents fill the images. This setup removes the need for manual annotation of weed classes in images, as we are assured that every part of every image is full of a weed class. One image was taken every second.

Image acquisition was arranged into epochs, where a single epoch consisted of making repeated revolutions of each plot for a period of ten minutes (yielding twenty minutes of weed image acquisition from the two weed types together), followed by fifteen minutes of grass image acquisition. The open grassland contained a mixture of grasses including *Lolium multiflorum, Festuca pratensis, Phleum pratense*, and *Holcus lanatus* with some *Trifolium repens* (clover). Approximately half of the epochs were acquired under overcast weather conditions, whilst the other half was acquired under bright or sunny weather conditions. Data capture was staged over four days, with 10 epochs in total captured at random times of day from sunrise to sunset during May 2016. Images were inspected manually and a small portion ($< 0.1\%$) removed due to recording problems. Approximately a third of a terabyte of usable image data was thus acquired in total, to our knowledge this is the largest and most multi-conditioned data set of its kind.

Images were pre-processed in three steps: colour calibration, perspective dewarping, and windowing.

In colour calibration, images are transformed to compensate for variations due to lighting and weather conditions. Colour calibration acts to colour-standardise images (to a certain extent) to simplify classification. It was performed using a colour bar present in all images, recorded as part the robot camera frames (Figure 3). The colour bar was composed of five coloured squares (red, blue, yellow, white and green) using standard colours of paint. A single

and motion-blur effects were usually small compared to other factors such as perspective distortion. The camera also makes adjustments for lighting but we override these with our own colour normalisation as detailed below.

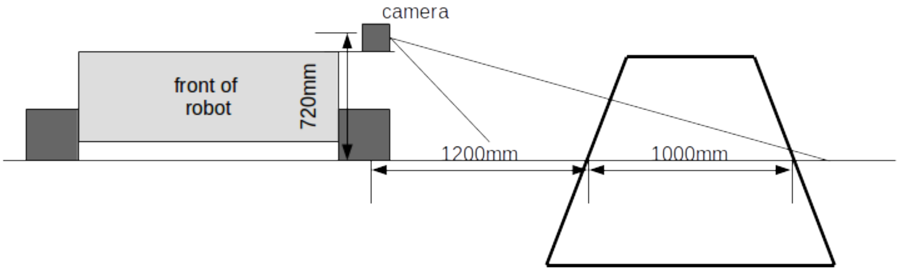
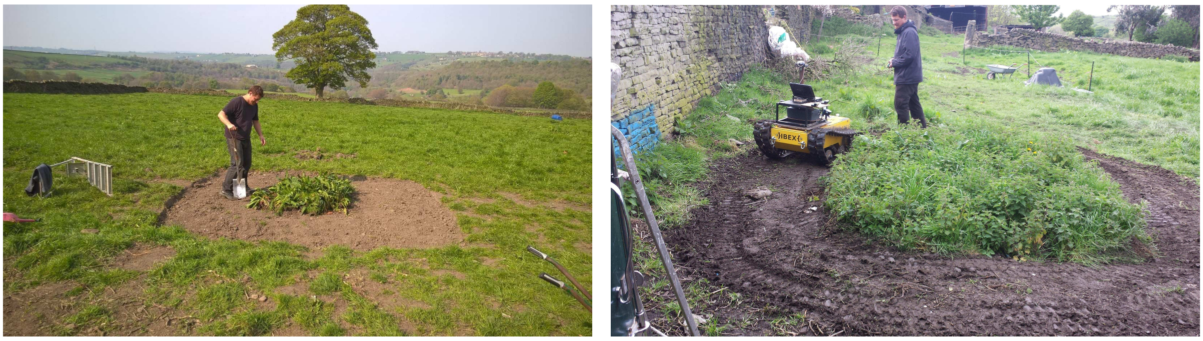


Figure 1: **Camera geometry.** Showing position of camera on tracked robot (viewed from the front), facing sideways. The thick black square shows the area on the ground, in perspective, used later in the perspective transform.



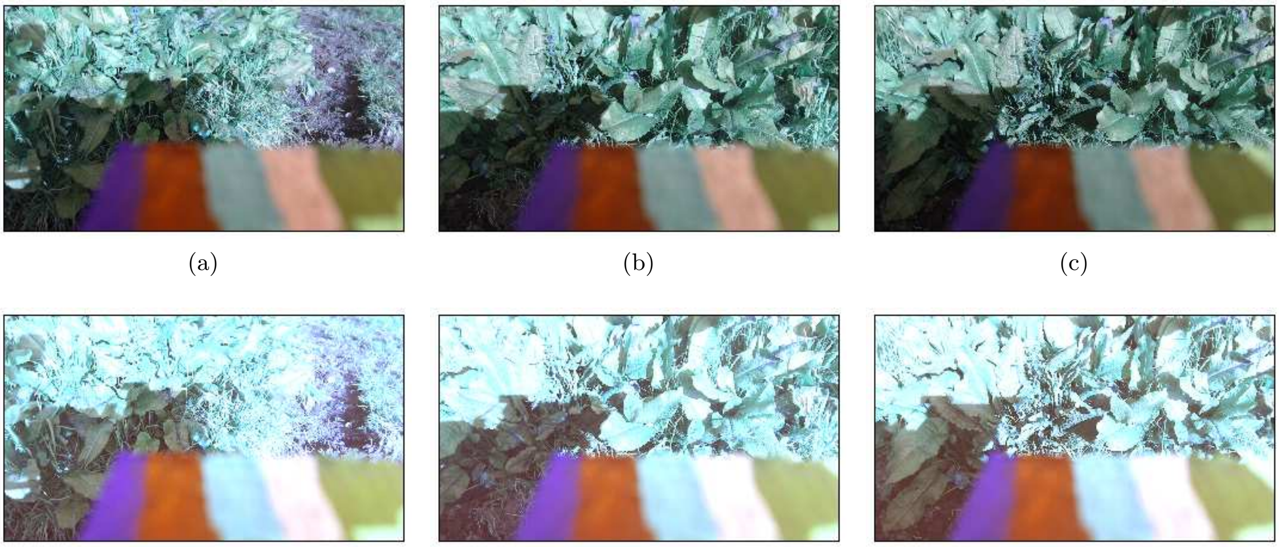
(a)

(b)

Figure 2: **Image acquisition.** a) Weed plot construction. b) Images were obtained using cameras mounted to the side of a tracked robot. The robot repeatedly made revolutions of each plot whilst taking pictures.

reference image was selected to calibrate all other images to. A measure of the blue, red and green light intensities from each coloured square on the bar was obtained as the mean value of the red, green and blue channels within the square. For each channel a vector of intensities was constructed with values for each coloured square, yielding the vectors $\boldsymbol{b}_{\mathrm{r}}$, $\boldsymbol{g}_{\mathrm{r}}$ and $\boldsymbol{r}_{\mathrm{r}}$ for the blue, red and green channels, respectively. The subscript r indicates that these values are from the reference image. Repeating this procedure for a comparison image c (i.e. an image that is to be colour calibrated) gives vectors $\boldsymbol{b}_{\mathrm{c}}$, $\boldsymbol{g}_{\mathrm{c}}$ and $\boldsymbol{r}_{\mathrm{c}}$. A linear relationship between the intensities of the blue, red and green channels is assumed, measured from the reference image and the intensities of the blue, red and green channels measured from the comparison image. Given this assumption, the parameters for each channel, for example blue, are obtained as $\hat{\boldsymbol{\beta}}(b)$,

$$\hat{\boldsymbol{\beta}}(b) = (\mathbf{X}(b)^{\mathbf{T}}\mathbf{X}(b))^{-1}\mathbf{X}(b)^{\mathbf{T}}\mathbf{b}_{\mathrm{c}}, \qquad (1)$$



(a)

(b)

(c)

(d)

(e)

(f)

Figure 3: **Colour Calibration.** Figures a, b and c are raw images of dock leaves before any pre-processing has taken place. Figures d, e and f are colour calibrated versions of figures a, b and c, respectively.

where

$$\mathbf{X}(b) = \begin{pmatrix} \mathrm{red}_{\mathrm{r}}(b) & 1 \\ \mathrm{blue}_{\mathrm{r}}(b) & 1 \\ \vdots & \vdots \\ \mathrm{green}_{\mathrm{r}}(b) & 1 \end{pmatrix}, \mathbf{b}_c = \begin{pmatrix} \mathrm{red}_{\mathrm{c}}(b) \\ \mathrm{blue}_{\mathrm{c}}(b) \\ \vdots \\ \mathrm{green}_{\mathrm{c}}(b) \end{pmatrix}, \tag{2}$$

where the values in the left hand column of the matrix $\mathbf{X}(b)$ is the vector $\boldsymbol{b}_{\mathrm{r}}$. These parameters are used to colour calibrate the blue channel of the comparison image as:

$$I'_{\mathrm{c}}(b) = \frac{I_{\mathrm{c}}(b) - \hat{\boldsymbol{\beta}}(b)[1]}{\hat{\boldsymbol{\beta}}(b)[2]}, \tag{3}$$

where the term in square brackets indexes a value in $\hat{\boldsymbol{\beta}}(b)$, I is the original intensity and I' is the adjusted intensity. Performing these operations for the green and red channels also yields a fully colour calibrated image.

Perspective normalisation was performed via the projective transform shown in fig. 4a to those in fig. 4b. This maps a 1.16m width by 1.0m depth ground area into a 700 pixel width by 600 pixel height image, whose geometry is identical to that of a vertical, overhead camera as used in previous studies. However the image is not exactly the same as that of an overhead

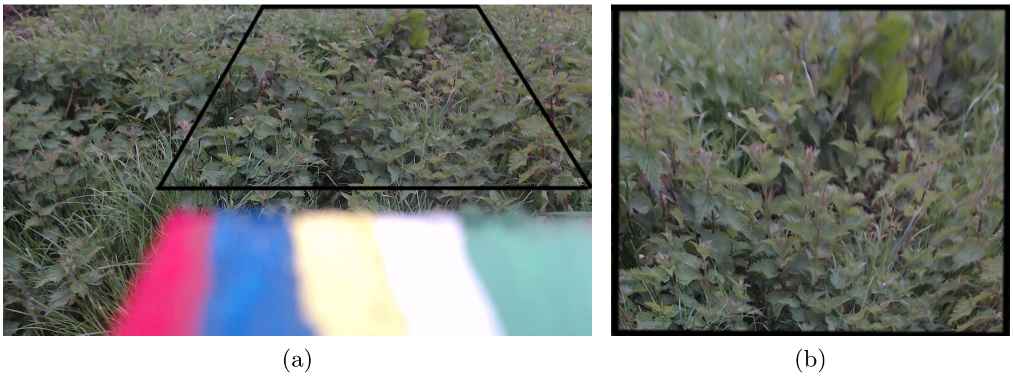


Figure 4: **Perspective normalisation.** a) Raw image from left camera, with grid annotation showing 1.16m width x 1m depth on the ground. b) Affine transformation to remove perspective, grid annotation showing locations of the same rectangle after perspective transformation into a 600x700 pixel image.

camera due to the three dimensional structure of the plants. In particular, tall plants at the front of the image are inflated in size because they are warped as if they were further back. A key research question asks whether this will have a detrimental effect over pure overhead imaging.

Finally the dewarped image was split into regular 28×28 square or 64×64 square pixel windows for classification, corresponding to 46mm or 106mm squares of ground space respectively. Windows were contiguous and non-overlapping, and were stored for analysis. (These sizes were chosen to be around the scale of a single spray target radius, or ground size of a single weed or clump of weeds. 64 is a power of two which speeds up FFT based methods; 28 rather than 32 is chosen for the smaller window size to enable future comparisons with neural network methods, where 28^2 pixel windows are a common standard for historical and technical reasons [16].)

2.2 Dataset definitions

To avoid test data contaminating training processes, the set of epochs was first partitioned into training epochs and test epochs. This prevents, for example, classifiers from learning to recognise the lighting conditions rather than the weed types. Partitioning was performed manually to ensure a balance of weather conditions and weed types in each partition.

After partitioning the epochs, we defined training and hyper-training datasets by sampling random windows from the training epochs, and test and hyper-test data sets by random sampling windows from the test epochs. (Separate hyper-training and hyper-test sets are used during hyper-parameter optimisation before training proper, to avoid contamination by test set data, as described in section 2.5.) A data set consisted of a set of images, where half of those were images of grass and the other half were images of weeds. A data set could contain an individual weed type (Rumex or Urtica) or a mixture of weed types (Rumex and Urtica). Similarly, a data set could contain images obtained under individual weather conditions (overcast or sunny) or a mixture of weather conditions (overcast and sunny). The data set sizes are shown in the table below,

Table 1: Dataset sizes.

Window size	Dataset type	Epoch set sampled	Number of windows
28^2	Hyper-training	Training	10,000
28^2	Hyper-test	Test	1,000
28^2	Training	Training	200,000
28^2	Test	Test	20,000
64^2	Hyper-training	Training	2,000
64^2	Hyper-test	Test	200
64^2	Training	Training	40,000
64^2	Test	Test	4,000

These sizes are still small compared with the total amount of raw data collected, but are orders of magnitude larger than data used in previous studies.

[4] The test set size was chosen to yield significant confidence in the results, while the training set size was selected to enable all the software implementations to train within one day of processing on a single 3GHz Intel core. Test and hyper-test datasets are 10% size of their corresponding training and hyper-training sets. The 28^2 sets are set to contain 5 times as many images as the 64^2 sets so that they contain the same amount of total pixel data ($28^2 \approx 64^2/5$),

[4] While the number of images is large, the number of actual plants imaged is smaller, because the many images are obtained by driving many times around the same plots of plants. However it is unlikely that images of the same plant are ever identical due to several factors. First, images were taken over several days, with collection days spaced several days apart. This allows the plants time to grow and move around between epochs. Second, within epochs, the plants are outdoor and exposed to wind, which moves their leaves constantly. Third, epochs are taken at different times of day, with different weather and light conditions. Fourth, the radius of the robot around the plots was randomised to +/-0.5m from the center line on each drive, ensuring different views. Fifth, the angles of the cameras shifted slightly between epochs during robot maintenance; this was deliberately not reduced, to introduce addition variation. Sixth, images were taken once per second, and it is unlikely that the robot would be at exactly the same angle around the plot at any two imaging moments. We assume that the additional variability from using more individual plants would be small compared to the variability introduced by these factors.

to allow results to be meaningfully compared across window sizes.

2.3 Feature extraction

The windowed images contain 64^2 or 28^2 pixels of RGB data, which are too large to use directly as input vectors to most classifiers. Therefore, features were first computed from the data, as in the previous studies. The selection of features used is detailed below and was selected to represent most previously proposed feature choices. All features ran on greyscale versions of the windows (obtained as the mean of the RGB channels), which is justified as all images are primarily all the same shade of green (unlike the case in detection studies of arable crop weeds in brown soil).

2.3.1 Fourier Transform

The Fourier Transform [39] represents an image in the basis of its orthogonal harmonic frequency components. For digital images the discrete Fourier transform (DFT) is used, whose basis is a set of two dimensional harmonics large enough to fully describe the spatial domain image. The number of frequencies corresponds to the number of pixels in the spatial domain image. For a square image of size $N \times N$, the two-dimensional DFT is given by:

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) e^{-\iota 2\pi (\frac{ki}{N} \frac{lj}{N})}, \tag{4}$$

where $f(i,j)$ is the image in the spatial domain and the exponential term is the basis function corresponding to each point $F(k,l)$ in the Fourier space. The basis functions are two dimensional sinusoidal waves of increasing spatial frequencies, i.e. $F(0,0)$ represents the DC-component of the image which corresponds to the average brightness and $F(N-1, N-1)$ represents the highest frequency. The absolute values of the DFT yield the image's magnitude of frequency spectrum which is used as the feature vector for classification. We denote this method of feature extraction as FT. The Fourier transform is illustrated in figure 5, which shows that broad leaves tend to contain stronger low spatial frequencies while grass' thin blades give rise to higher spatial frequencies.

2.3.2 Local binary patterns

Local binary patterns (LBP) are a texture description feature [15]. Local means they are computed on local sub-windows of an image only, as a function of a center pixel and either its immediate or r pixel radius neighbours; binary means that the feature vector is binary, with each feature classed as either present or absent. Windows are converted to greyscale, then for each pixel $x_{i,j}$ in the window, LBP computes an 8-element binary vector,

$[(x_{i,j} > x_{i,j+r}), (x_{i,j} > x_{i-r,j+r}), (x_{i,j} > x_{i-r,j}), (x_{i,j} > x_{i-r,j-r}), (x_{i,j} > x_{i,j-r}), (x_{i,j} > x_{i+r,j-r}), (x_{i+r,j} > x_{i,j}), (x_{i,j} > x_{i+r,j+r})]$.

There are $2^8 = 256$ possible values of this feature vector, with oriented edge and corner detection present as special cases. LBP computes feature vector for each pixel in the window, then computes a 256-point histogram of the obtained values over the window. The shapes of these histograms are considered to be characteristic of the texture classes and are given as input to classifiers. As well as using the eight near neighbours as above, a hyper-parameter n_{points} can also be used to generalise to other numbers of quantised comparison points equally spaced on a circle around the center. (Many other variations on the LBP concept have also been proposed [13] but are beyond the scope of the present comparison study.)

2.3.3 Interest points and k-means

The above features treat every pixel in the window as equally important, and are considered to represent texture-like properties. An alternative approach is to locate only 'interesting' points within the window and base classification on these. In particular, weeds such as Urtica have many distinctive jagged corners which might form useful points on which to base classification. Interest point method contain two feature extraction steps prior to classification. First, interest points are located; second, the local region at each of these points is used to extract a feature descriptor. Points are considered interesting if they contain a mixture of colours and can be uniquely located, i.e. a corner is interesting because there is only one location where it exists, while an edge is less interesting because it exists along a line of locations. Feature classes used to describe these points are usually wavelet-like, combining color, frequency and size information.

Precise definition of useful interest point detectors and descriptors normalising these properties has been and remains an active area of machine vision research [25], but the present study arbitrarily selects the state-of-the-art Scale-Invariant Center-Surround Detectors (CenSurE) [1] interest point detector and the Binary Robust Invariant Scalable Keypoints (BRISK) [20] descriptor to represent the general class of methods. CenSurE finds an approximation to the set of corner-like points defined by,

$$\{\lambda_d(H(x)_{i,j}) > t, d \in \{1,2\}\}, \tag{5}$$

where λ_d are the two eigenvalues of the Hessian matrix H of the image x at each pixel i, j, and t is a threshold. Both eigenvalues are maximal at corners of any rotation. Rather than compute this computationally intensive (due to eigenvalue finding) test for every pixel, CenSurE first performs a faster pre-screening step, using a set of scaled filters to approximate the Laplacian (total curvature) at each pixel, then only computing the Hessian test at local maxima of this curvature [1].

BRISK descriptors are similar to the LBP vectors above, but using pixel intensity comparisons,

$$f_n = (x_{i,j} > x_{i'_n,j'_n}), \tag{6}$$

at a larger set of 256 offsets $\{a_n, b_n\}$ giving $(i'_n = i + a_n, j'_n = j + b_n)$. Unlike LBP, these offsets are not equally spaced around a circle, but may form any arbitrary pattern. Together with the larger number of points, this may capture potentially higher-order information than in LBP. Standard values of the offset patterns are used as provided in [20].

When BRISK feature vectors have been computed for all images in the training set they are passed to the k-means algorithm which clusters them into K regions in feature space, where feature space is a finite p-dimensional vector space with each dimension representing a BRISK feature of an image. Initiating K random clusters we firstly calculate the Euclidean distance between the *ith* BRISK vector $\mathbf{x}_i$ and the *kth* cluster $\mathbf{C}_k$

$$d_{i,k} = \left[\sum_{j=1}^{p}(\mathbf{x}_{ij} - \mathbf{C}_{kj})^2\right]^{1/2}. \tag{7}$$

After performing this operation for all clusters we can then assign the BRISK vector to the cluster that minimises d (Equation 7). This operation is then performed for each BRISK vector. Knowing the members of each group we can now compute the new centroid of each group based on these new memberships. New centroids are the average coordinates among new members,

$$\mathbf{C}_k = \frac{\sum_{n=1}^{N_k} \mathbf{X}_{n1}}{N_k}, ..., \frac{\sum_{n=1}^{N_k} \mathbf{X}_{np}}{N_k}, \tag{8}$$

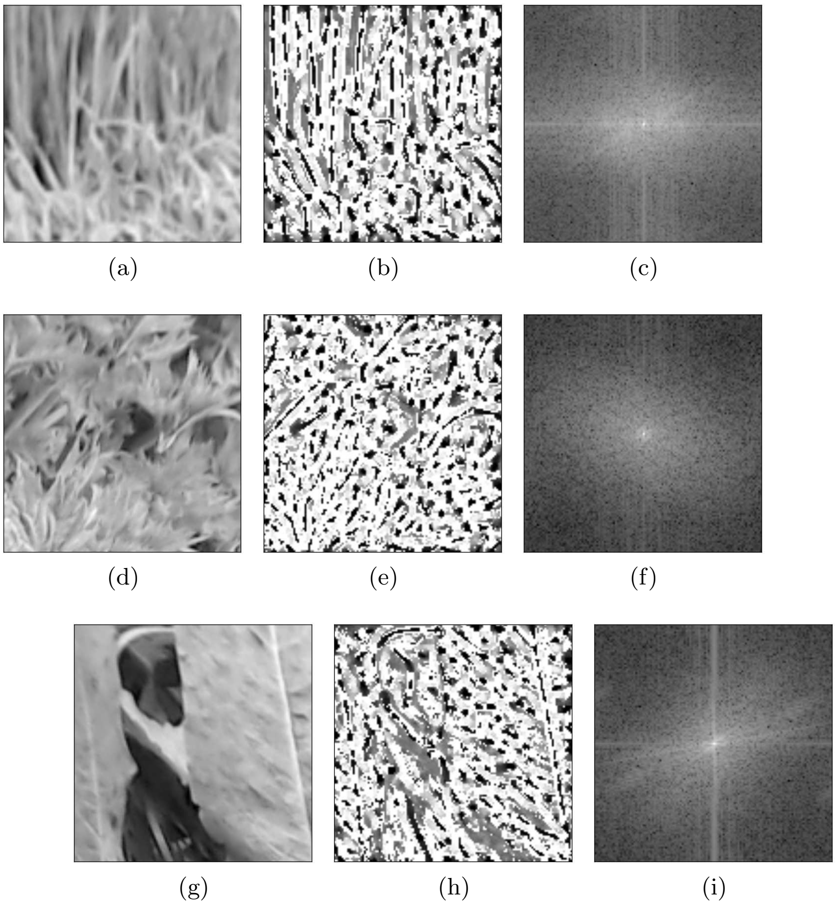
where N_k is the number of BRISK vectors in cluster k. This whole process is then repeated until the BRISK vectors cease to move groups (i.e. until the computation of the k-means clustering has reached stability).

Clustering observed BRISK vectors in this way defines K discrete types of BRISK feature. For any given image, we may now extract each of its interest points, compute a BRISK descriptor at these points, then replace each of their BRISK descriptors with one of these quantised types. This allows us to then count how many c_k of each discrete BRISK types $k = 1 : K$ appear in the image. The vector of these counts, $\{c_k\}_{k=1:K}$ is then used as a feature descriptor of the whole image.

We refer to this feature as $B > K$, (for 'BRISK followed by k-means').

2.3.4 Watershed segmentation

We wish to test window-based features against region-growing type methods as proposed in previous studies. To make a fair comparison it is necessary to substitute pure region growing with a similar but window-based method. Otherwise the region growing methods could be accused of accessing more data to make classifications of each region, from the whole image, rather than just from its local window. For this purpose, we use a watershed method as a close substitute. Watershed segmentation [37] was originally developed for the purpose of separating touching objects in an image rather than for classification, but may also be used as a region-growing type classifier. The watershed transform finds 'catchment basins' and 'watershed ridge lines' in an image by treating it as a surface where light pixels are high and dark pixels are low. Segmentation using the watershed transform works better if a human operator can first identify, or 'mark', pixels from foreground objects and background locations. The marker-controlled watershed segmentation used in the present study follows a multi-step procedure.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 5: **Feature extraction.** Figures a, d and g are fully pre-processed images containing grass, Urtica and Rumex leaves, respectively. Figures b, e and h are the local binary patterns (LBPs) of images a, d and g, respectively (with $n = 24$, $r = 4$). Figures c, f and i are the magnitude of frequency spectra of images a, d and g, respectively.

Given a grey-scale image as input, we apply Otsu's thresholding [37] to segment the background from the foreground. Then we compute the Euclidean distance transform which computes the Euclidean distance to the closest zero (i.e. background pixel) for each of the foreground pixels. Doing this yields the distance map d. Next we apply a function $f(d, d_{\text{min}})$ that finds the peaks (local maxima) in the distance map, and ensures that we have at least a d_{min} pixel distance between each peak. Then we apply a connected component analysis using 8-connectivity to the output of f, the output of which gives us our markers, which we then feed in to the watershed function. The watershed function returns a matrix of labels, an array with the same width and height as the input image. Each pixel value has a unique label value. Pixels that have the same label belong to the same object. For a given image, we then count the number of *unique* labels or *segments*. Performing these operations for multiple values of d_{min} yields multiple counts of segments for a given image, and thus a feature vector for classification. We denote this method of feature extraction as W.

2.4 Classifiers

The following classifiers were used to represent previously proposed architectures. Each classifier (SVM, LDA, NN) can take as its input any feature type obtained from windows (LBP, FT, B⟩K, W). This yields 12 distinct classification methods, LBP⟩SVM, FT⟩SVM, B⟩K⟩SVM, W⟩SVM, LBP⟩LDA, FT⟩LDA, B⟩K⟩LDA, W⟩LDA, LBP⟩NN, FT⟩NN, B⟩K⟩NN and W⟩NN. For example, LBP⟩SVM means that we pass the local binary pattern feature vector as input to a support vector machine, whilst FT⟩LDA means that we pass the image's frequency spectrum magnitude feature vector as input to linear discriminant analysis.

2.4.1 Support Vector Machines

A Support Vector Machine [35] models the a classification problem as finding a non-linear partition of the feature vector space into classes (e.g. grass or weeds), formed as a linear partition of a higher-dimension space formed by non-linear high-dimensional projection of the feature vectors. To understand how SVMs work it is useful to first briefly describe the support vector classifier (SVC). The SVC separates images into their classes by finding the *linear affine* hyper-plane that maximises the distance (known as the margin M) between the two image classes in feature space. Observations that fall on the boundaries of the margin are the support vectors.

The linear affine hyper-plane is defined by the following inner product,

$$\mathbf{b} \cdot \mathbf{x} + b_0 = 0, b_0 \neq 0, \tag{9}$$

where $\mathbf{x}$ is a $p-$dimensional training image feature vector with associated weights $\mathbf{b}$. Now consider a set of n $p-$dimensional training image feature vectors, $\mathbf{x}_i$, each with an associated class label $y_i \in \{-1, 1\}$. Introducing new hyper-parameters; n ϵ_i values (known as slack values) and a hyper-parameter C (known as the budget), then we wish to maximise M across $b_1, ..., b_p$,

$\epsilon_1, ..., \epsilon_n$ such that

$$\sum_{j=1}^{p} b_j^2 = 1, \tag{10}$$

$$y_i(\mathbf{b} \cdot \mathbf{x} + b_0) \geq M(1 - \epsilon_i), \forall_i = 1, ..., n, \tag{11}$$

$$\epsilon \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C, \tag{12}$$

where C, the budget, is a non-negative 'tuning' hyper-parameter and hyper-parameters ϵ_i allow the individual observations (i.e. training images) to be on the wrong side of the margin or hyper-plane. C collectively controls how much the individual ϵ_i can be modified to violate the margin.

SVMs are an extension of SVCs that results from a non-linear enlargement of the feature space through the use of functions known as kernels. This enlargement of the feature space means that observations from different classes can be separated in many more ways than they could be otherwise. To obtain the SVM, firstly we note that it is possible to show that a linear support vector classifier for a particular observation can be represented as a linear combination of inner products for the subset ℓ of training observations that represent the support vectors,

$$f(\mathbf{x}) = b_0 + \sum_{i \in \ell} \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle, \tag{13}$$

where α_i are the coefficients. Replacing the inner product $\langle \mathbf{x}_i, \mathbf{x}_k \rangle$ with a more general inner product 'kernel' function $K = K(\mathbf{x}_i, \mathbf{x}_k)$, we can modify the SVC representation to use non-linear kernel functions. One example is the radial (RBF) kernel,

$$K(\mathbf{x}_i, \mathbf{x}_k) = \exp\left(-\gamma \sum_{j}^{p} (\mathbf{x}_{ij} - \mathbf{x}_{kj})^2\right), \gamma > 0. \tag{14}$$

Intuitively, the γ parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The γ parameter can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.

The algorithmic solution for the SVM is one that finds optimal values for the coefficients α and the slack variables ϵ_i. Typically gradient decent algorithms are used. The hyper-parameters C and γ are set/optimised by the user.

Finally, a test image is classified according to whether its feature vector $\mathbf{x}^*$ results in a positive or negative sign when passed into the function $f(\mathbf{x}^*)$. Note that feature vectors were normalised before being passed into the support vector machine. We denote this classifier as SVM.

2.4.2 Linear Discriminant Analysis

As with SVCs/SVMs, Linear Discriminant Analysis [22] models the classification problem by creating a feature space with a dimension for each feature. However, in LDA, observations from 2 separate classes are assumed to be sampled from 2 separate multivariate Gaussian distributions in feature space with different means but the same covariance matrix. Given those assumptions we have a linear hyperplane perfectly separating the means of the 2 distributions. This means that any observation that is situated above the hyperplane has a higher probability of being a sample from the Gaussian whose mean is situated above the hyperplane than being a sample from the Gaussian whose mean is located below the hyperplane.

To explain LDA in some more detail, firstly we write Bayes' rule for the classification problem

$$P(i|\mathbf{x}) = \frac{P(\mathbf{x}|i)P(i)}{\sum_j P(\mathbf{x}|i)P(i)}, \tag{15}$$

where the likelihood function $P(\mathbf{x}|i)$ gives the probability that the observation $\mathbf{x}$ is a sample from the Gaussian representing the class i and $P(i)$ is the prior probability of the class i. From Bayes' rule and the assumptions outlined we can derive the linear discriminant analysis formula

$$f_i = \mathbf{m}_i \mathbf{C}^{-1} \mathbf{x}_k^T - \frac{1}{2} \mathbf{m}_i \mathbf{C}^{-1} \mathbf{m}_i^T + \log(p_i), \tag{16}$$

where $\mathbf{m}_i$ is a vector containing the mean of each feature for the class i and $\mathbf{C}$ is the pooled within group covariance matrix which is a weighted mean of the covariance matrix $\mathbf{C}_i$ for each class. For a total of n observations, N classes and n_i observations in each class $\mathbf{C}$ is

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{N} n_i \mathbf{C}_i. \tag{17}$$

Then we simply assign a test image k to group i that has maximum f_i. We denote this classifier as LDA.

2.4.3 'Nearest Neighbour' Classifier

Nearest neighbour is a very simple classifier, used here to provide a baseline to compare with the two more sophisticated classifiers (SVM and LDA) above. In its training phase, the nearest neighbour classifier computes the median feature vector for each class (grass or weeds). A test image is then classed as grass if its feature vector minimises the sum of absolute errors between it and the median feature vector computed from grass images. Likewise a test image is classed as containing weeds if its feature vector minimises the sum of absolute errors between it and the median feature vector computed from images containing weeds. We denote this classifier as NN.

2.5 Hyper-parameter optimisation

Some of the classifiers and features have hyper-parameters which define how training is computed. Previous studies have mostly reported the best obtained results of methods on test sets, and stated the hyper-parameter values used to give them. However it is unclear whether the hyper-parameters in these cases have been set in advance of the evaluation on the test sets, or if they have been fit to the test data by running method multiple times on the test data and reporting only the best result.

To avoid this potential bias, careful use was made of hyper-training and hyper-test datasets, independent of both training and test datasets, to select hyper-parameters in advance of the main training and test phases. Hyper-parameters were optimised on these sets, so that each system only saw the final test set only once for its reportable evaluation score.

In theory, hyper-training could be performed by training many versions of a classifier on the full training dataset, then scoring them against a hyper-test set, and selecting the best performer. (The hyper-test sets are sometimes known as 'validation sets'). However this requires running time-consuming training many times. So given the large ratio of data to compute resources available for this study, a smaller hyper-training dataset, of 5% size of the full training dataset, was used in place of the training dataset. This greatly reduces the required computation time but was found to still give a reasonable indication of good hyper parameters to use within available compute resources.

SVMs have two hyper-parameters (C and γ) that should be optimised. We set C and γ in exponentially growing sequences, $C = 2^{-5}, 2^{-3}, ..., 2^{15}$, $\gamma = 2^{-15}, 2^{-13}, ..., 2^{3}$, which has been shown to be a practical method for identifying good parameters [19]. In addition the parameters (number of points n_{points} and radius r) of the LBP should also be optimised. For all experiments we set $n_{\text{points}} = 2, 4, ..., 30$. For experiments with 28^2 pixel windows we set $r = 1, 2, ..., 8$ while for experiments with 64^2 pixel windows we set $r = 2, 4, ..., 16$ (r has maximum value equal to a quarter of the image width). For all experiments involving the B⟩K method of feature extraction we optimised K for values $K = 1, ..., 28$. For all experiments involving the W method of feature extraction we optimised the minimum distance term d_{min} in growing sequences of integers $[1], [1, 2], ..., [1, ..., 15]$. For the sequence $[1, 2]$, for example, we would generate a feature vector by first setting $d_{\text{min}} = 1$, passing an image into the W method and retrieving the number of segments. Then the process would be repeated for $d_{\text{min}} = 2$ and both segment values would be appended, yielding a feature vector of length 2. Thus the length of a particular sequence is the length of the feature vector generated by the W method of feature extraction.

In addition to numerical hyper-parameters, SVMs can further use various kernels and LDA can further use various types of 'solver' and both SVMs and LDA have an option to apply a 'shrinking'/'shrinkage' heuristic. For the LBP⟩SVM method we reduced the number of parameter permutations to consider by assuming some independence between parameters. Thus we first optimised C and γ, (considering all permutations of C, γ) for a fixed $n_{\text{points}} = 15$, $r = 4$ for 28^2 pixel windows and a fixed $n_{\text{points}} = 15$, $r = 8$ for 64^2 pixel windows, for each kernel with the shrinking heuristic turned on and off. Then we fixed C, γ and the kernel at their optimal values and optimised n_{points} and r (considering all permutations of n_{points}, r). For all other classification methods we considered all possible parameter permutations.

LDA's r and BRISK's CenSurE approximation variables were also treated as hyper-parameters. Details of the results on the hyper-test set from hyper-training that are used to set hyper-parameters for full training are shown in the Appendix.

2.6 Experiments

After hyper-parameter optimisation, we evaluated the performance of each feature-classifier combination by training from scratch on the full training dataset and testing for the first and only time on the test dataset.

To enable a fair comparison of systems running on the two different window sizes, more windows were present in the 28^2 pixel training and test sets than in the 64^2 pixel sets. This is because each 64 pixel image contains roughly five times as much visual information as each 28 pixel image ($28^2/64^2 = 0.19$); each 64^2 pixel image is effectively five 28^2 pixel windows joined together. Therefore, in the 28^2 pixel case, we used a training set of 200,000 windows and a test set of 20,000 windows, each comprised of half grass and half weed windows; while in the 64^2 pixel case we used 1/5 as many windows: 40,000 training and 4,000 test, also comprised of half grass and half weeds.

The most important practical question for weeding robots is the performance in mixed weeds (Rumex+Urtica) and in mixed weather (sunny+overcast), for the two window sizes. Window size is important because it controls the spatial resolution at which the robot could spray the weeds – in square windows of 106mm or 56mm. As the key research question, performance was evaluated for every one of the twelve feature-classifier combinations on both window sizes.

It is sometimes the case in machine learning that improved accuracies can be obtained by fusion results from multiple methods into *meta-classifiers* (also known as ensemble learning). Many combination algorithms are available with different and subtle assumptions which are still sometimes debated [8, 9].[5] To give a simple illustrative, though non-optimal, idea of what performance improvements could be available, three simple, standard fusion methods were tested. First, a simple voting scheme, META-VOTE, assigns an equal weight to each classifier's output, and yields the classification with the most votes. (In the case of a tie, the best classifier's output is given the deciding vote.) Second, META-ACC weights the votes of each classifier by its accuracy. Finally, META-LDA considers the output of each classifier as an element of a Boolean

[5] Until recently it was often assumed that optimal combination could be achieved via Bayesian Model Averaging (BMA), which makes class predictions c of input x from models M_i and training data D as $P(c, x) = \sum_i P(c|M_i, x)P(M|D)$. However it is now known that for large data sets, BMA simply converges to the outputs of the single best classifier in the ensemble, ignoring this others [9], hence it is not used here. This problem with BMA is caused by its underlying assumption that the ensemble contains the perfect, ground-truth model rather than just a set of approximations.

feature, and trains a new LDA classifier to predict ground truth class from these vectors.[6]

Secondary questions of interest include the effects of perspective, weather type, weed type, and windowing. As a full training process can take several days, these questions were examined using only the best feature+classifier system and assumed to be independent of one another.

To examine the effect of perspective unwarping, the test set (containing grass under mixed weather conditions, and mixed weeds under mixed weather conditions), was split into new test sets according to their windows' vertical locations (row numbers) in the camera images for individual scoring. Low row numbers indicate windows from the base of the image, corresponding to space close to the robot cameras, while high row numbers indicate windows at the top of the image, from space furthest from the robot cameras.

To examine the effect of weather, each epoch was classified as sunny or overcast, and the original test set was split into two test sets comprised of windows of these weather types for individual evaluation.

To examine the effect of weed type on classifier performance, two set sets were created which contained only grass-and-Urtica and grass-and-Rumex respectively, for individual evaluation.

To give an idea of performance in the limiting case of large windows full of weeds or grass, we assessed the performance of the B⟩K⟩SVM classification method on full sized (600 × 700) pixel windows from data sets containing grass vs mixed and individual weed types under mixed weather conditions. We conducted this experiment because BRISK features are more usually extracted from full-view images than from the standardised windows used in the rest of this study. Thus we wished to asses the performance of this particular classification method under its own ideal conditions. As with other experiments, hyper-parameters were optimised on hyper-training and hyper-test datasets (though of new 600 × 700 windows and set sizes hyper-training=1000, hyper-test=200, training=10,000,test=2000 , before training and testing on the training and test datasets.

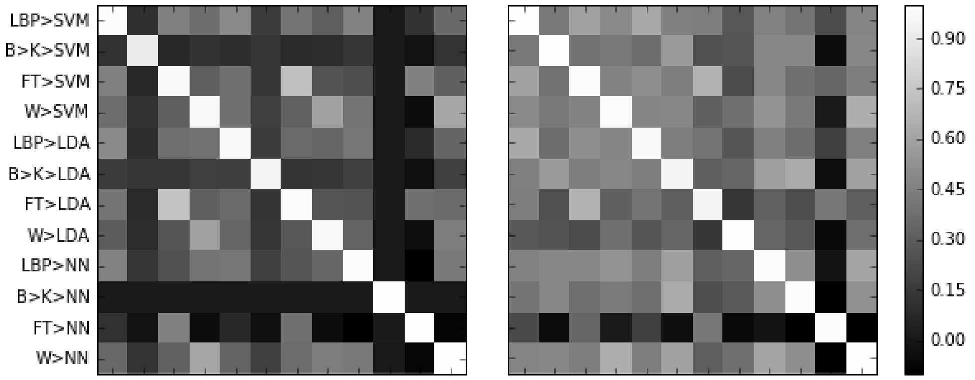
[6] To estimate META-LDA performance on new data without corruption by training the meta-classifier on test data, the test set was split into two random partitions with one used to train the new LDA and the other to test it. This means the result quoted from only a subset of the original test set. However the original and partitioned test sets are sufficiently large to maintain tight Bayesian confidence internals in the accuracy posteriors to be comparable with the other results.

Table 2: Results of applying all 12 classification methods to 28^2 pixel window test dataset containing grass vs mixed weeds, mixed weather, after training on 28^2 pixel window training dataset. ACC is over all accuracy, i.e. the probability that a random image is correctly classified. CI is the confidence interval in the estimate of ACC. $GRASS$ and $WEED$ are probabilities that images of grass, or weed, respectively, are correctly classified. 'NA' stands for 'Not Applicable', 'shk' for 'shrinking'.

METHOD	ACC	CI	GRASS	WEED	PARAMETERS
LBP$\rangle$SVM	68.75	3.27×10^{-3}	74.1	63.6	$n_{\text{points}} = 20$, $r = 5$, kernel=RBF(shk), $C = 2^{13}$, $\gamma = 2^{-1}$
B$\rangle$K$\rangle$SVM	52.70	3.53×10^{-3}	38.8	67.1	$K = 4$, kernel=linear(shk), $C = 2^{-5}$
FT$\rangle$SVM	**71.80**	3.18×10^{-3}	79.0	64.7	kernel=RBF(shk), $C = 2^{1}$, $\gamma = 2^{3}$
W$\rangle$SVM	64.67	3.38×10^{-3}	72.1	57.1	$d_{\text{min}} = [1, ..., 11]$, kernel=RBF(shk), $C = 2^{15}$, $\gamma = 2^{-3}$
LBP$\rangle$LDA	65.25	3.37×10^{-3}	71.1	59.1	$n_{points} = 28$, $r = 3$, solver=svd
B$\rangle$K$\rangle$LDA	53.97	3.52×10^{-3}	63.6	44.1	$K = 20$, solver=lsqr(shh)
FT$\rangle$LDA	68.54	3.28×10^{-3}	78.9	63.2	solver=lsqr(shk)
W$\rangle$LDA	61.63	3.44×10^{-3}	69.4	53.8	$d_{\text{min}} = [1, ..., 5]$, solver=lsqr(shk)
LBP$\rangle$NN	62.93	3.41×10^{-3}	69.6	66.3	$n_{\text{points}} = 2$, $r = 4$
B$\rangle$K$\rangle$NN	50.00	3.54×10^{-3}	0.0	100.0	$K = 17$
FT$\rangle$NN	61.79	3.44×10^{-3}	62.3	60.8	NA
W$\rangle$NN	65.77	3.35×10^{-3}	65.6	66.1	$d_{\text{min}} = [1, ..., 10]$
META-VOTE	71.38	3.20×10^{-3}	76.5	66.3	NA
META-ACC	71.90	3.18×10^{-3}	76.5	67.3	NA
META-LDA	**73.47**	4.41×10^{-3}	75.8	71.1	NA

Table 3: Results of applying all 12 classification methods to 64^2 pixel window test dataset containing grass vs mixed weeds, mixed weather, after training on 64^2 pixel window training dataset. ACC is over all accuracy, ie. the probability that a random image is correctly classified. CI is the confidence interval in the estimate of ACC. $GRASS$ and $WEED$ are probabilities that images of grass, or weed, respectively, are correctly classified. 'NA' stands for 'Not Applicable', 'shk' for 'shrinking'.

METHOD	ACC	CI	GRASS	WEED	PARAMETERS
LBP$\rangle$SVM	**82.88**	5.96×10^{-3}	87.1	78.7	$n_{\text{points}} = 24$, $r = 4$, kernel=RBF(shk), $C = 2^9$, $\gamma = 2^3$
B$\rangle$K$\rangle$SVM	69.15	7.27×10^{-3}	70.4	69.0	$K = 17$, kernel=RBF, $C = 2^1$, $\gamma = 2^8$
FT$\rangle$SVM	79.40	6.39×10^{-3}	84.6	74.2	kernel=RBF(shk), $C = 2^3$, $\gamma = 2^1$
W$\rangle$SVM	73.23	7.00×10^{-3}	79.0	67.5	$d_{\text{min}} = [1, ..., 10]$, kernel=RBF(shk), $C = 2^{11}$, $\gamma = 2^3$
LBP$\rangle$LDA	75.50	6.80×10^{-3}	82.3	68.7	$n_{points} = 16$, $r = 4$, solver=lsqr(shk)
B$\rangle$K$\rangle$LDA	70.65	7.16×10^{-3}	81.6	60.9	$K = 17$, solver=lsqr(shk)
FT$\rangle$LDA	73.15	7.01×10^{-3}	82.6	63.7	solver=eigen(shk)
W$\rangle$LDA	63.13	7.63×10^{-3}	88.1	38.2	$d_{\text{min}} = [1, ..., 15]$, solver=svd
LBP$\rangle$NN	72.43	7.06×10^{-3}	77.0	68.0	$n_{\text{points}} = 10$, $r = 12$
B$\rangle$K$\rangle$NN	70.55	7.40×10^{-3}	71.5	63.7	$K = 18$
FT$\rangle$NN	63.08	7.63×10^{-3}	58.7	67.5	NA
W$\rangle$NN	74.48	6.89×10^{-3}	76.0	73.1	$d_{\text{min}} = [1, ..., 5]$
META-VOTE	78.63	6.48×10^{-3}	89.4	67.9	NA
META-ACC	80.90	6.21×10^{-3}	88.4	73.8	NA
META-LDA	**83.40**	8.42×10^{-3}	i84.9	81.9	NA



(a) (b)

Figure 6: **Covariance Matrices.** a) Covariance between the 12 classification methods when applied to 28 pixel squared windows (grass vs mixed weeds, mixed weather). b) Covariance between the 12 classification methods when applied to 64 pixel squared windows (grass vs mixed weeds, mixed weather).

3 Results

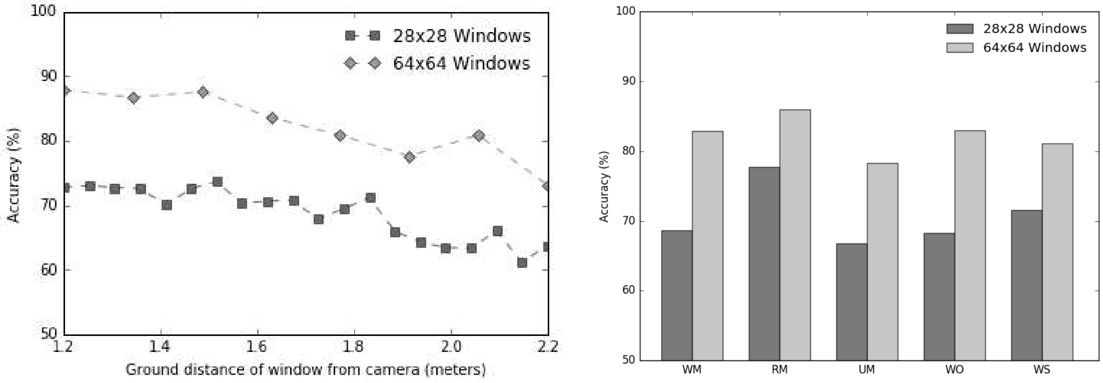
Tables 2 and 3 give the results for training and testing the 12 classification methods on 28^2 and 64^2 pixel windows, respectively, both with mixed weather and mixed weed types. ACC shows the overall accuracy of each method, as the proportion of images correctly classified. Bayesian confidence intervals (CI) computed as standard deviations of the Beta distribution posteriors over belief in the accuracy [5], assuming flat priors, are also given for each classification method, which justify the significance of the accuracy percentages to two decimal places. (We also list breakdown accuracies for grass and weed image presentations, which indicate rates of false positive and false negatives.) These experiments were conducted to determine which classifier was the most accurate in predicting test images from data sets containing mixed weeds and mixed weather conditions.

The LBP⟩SVM method performed better than the other classification methods for experiments using 64^2 pixel windows, with an accuracy of 82.88%. This was achieved with the SVM kernel set to RBF with the shrinking heuristic turned on, the hyper-parameters of the LBP set to $n_{\text{points}} = 24$, $r = 4$ and the hyper-parameters of the SVM set to $C = 2^9$, $\gamma = 2^3$. The FT⟩SVM method performed better than the other classification methods for experiments using 28^2 pixel windows, with an accuracy of 71.80%. This was achieved with the SVM kernel set to RBF with the shrinking heuristic turned on, and the SVM hyper-parameters set to $C = 2^1$, $\gamma = 2^3$.

The META-VOTE meta-classifier yielded worse results than the single best method in both

28^2 and 64^2 pixel windowed cases. This may be due to averaging of the best method with the less good methods dragging down the overall result. This typically occurs when all or most classifiers are acting on the same inherent information in the data but with different accuracies, rather than acting on different types of information per method. Similarly, META-ACC gives on a tiny improvement over the best method for 28^2 windows (71.90 vs 71.80), and is worse than the best 64^2 pixel method (80.90 vs 82.88). META-LDA is the best of the meta-classifiers, and is the only one to give significant improvements in both the 28^2 pixel (73.74 vs 71.80) and 64^2 pixel (83.40 vs 82.88). (Significance can be seen by comparing the small CIs with the larger accuracy differences). META-LDA's weights are more principled, and optimal under its assumptions, than the heuristic META-VOTE and META-ACC, so its better performance is expected. However the gain from using META-LDA over using just the single best method, in each window case, is small. Again, this suggested that all the methods are operating on similar information within the images rather than with different information. Further insight into this possibility is gained by examining the correlation matrix of the 12 methods' predictions in fig. 6. Here, each grass/weed classification in the test set is considered to have a value of 0 or 1 for grass/weed, and correlations over the test data are presented. It can be seen that the methods are less correlated with one another in the 28^2 pixel case than in the 64^2 pixel case, which explains why meta-classification works better for 28^2 than 64^2 windows. There are stronger correlations between methods sharing the same feature type than methods sharing the same classifier type, as can be seen by the secondary diagonal patterns. The FT>NN method has a low correlation with the others because it is a very poor accuracy method.

Results for the distance experiment are shown in Figure 7a. The best performing feature-classifier combination - the LBP$\rangle$SVM method - is here run again on mixed weed and weather test sets, separated as a function of the distance of the window from the robot camera's ground location (for both 28^2 and 64^2 pixel-squared windows). Classification performance decreased smoothly as the distance increased, for both 28^2 and 64^2 pixel windows, by a considerable amount (by around 15% absolute for 64^2 windows, and 10% absolute for 28^2 windows.) Weeds closest to the cameras were predicted with a 87.85% accuracy (for 64^2 pixel windows), which is more in line with the high accuracies reported by the previous studies than with accuracies at far distances. It should also be noted that this result was obtained for a mixture of Rumex and Urtica under a mixture of weather conditions, unlike those studies.



(a)

(b)

Figure 7: **Effects of distance, weed type and weather.** a) Classification performance of LBP⟩SVM as a function of the distance from the robot cameras (experiments on grass vs mixed weeds, mixed weather). b) Classification performance of LBP⟩SVM as a function of weed type (W: mixed weeds, R: Rumex, U: Urtica) and weather type (M: Mixed weather, O: Overcast, S: Sunny).

Results of the weed type and weather experiments are shown in 7b, again using the LBP⟩SVM method. Here W stands for mixed weeds, R stands for Rumex and U stands for Urtica; M stands for mixed weather, O stands for overcast weather and S stands for sunny weather. Rumex classification was more accurate than Urtica or mixed weed classification, and mixed weed classification was better than Urtica classification. For 28^2 pixel windows classification under sunny weather conditions was better than classification under overcast weather conditions. For 64^2 pixel windows the opposite weather pattern was found.

Finally, table 4 gives the results of applying the B⟩K⟩SVM classification method on full sized (600×700) windows for data sets containing grass vs mixed and individual weed types under mixed weather conditions. Again Rumex classification was the most accurate (97.9%), while Urtica classification was the least accurate (94.65%).

Table 4: Results of applying the B⟩K⟩SVM method to full sized (600×700) windows (mixed and individual weed types, mixed weather).

Experiment	Accuracy	Optimum K	Optimum C	Optimum γ
WM	95.1	16	2^{-3}	2^3
RM	97.9	19	2^{11}	2^1
UM	94.65	28	2^1	2^3

4 Conclusion

For our data set and the requirements upon which it is based, the best performing method for the overall spray/no-spray decision is Linear Binary Patterns with Support Vector Machine classification on 64^2 pixel windows.

LBPs are texture-based, rather than shape-based, features, and SVM is a highly nonlinear model. This suggests that when a mixture of Rumex and Urtica is present and spray/no-spray decisions are required, texture is more informative than shape, and that the discriminating distribution of texture features has some nonlinear component. In particular, linear classification of the same features with LDA performs less well.

All the accuracies in our independent re-implementations are lower than those reported in the papers which originally proposed them. This may be due to several factors. First, our data is more difficult to classify, even by human eye, than data used in the original studies. Apart from [2], previous work has used vertical, downward-pointing cameras giving clear and equal views of each point on the ground by removing the need for perspective correction. Our data is more challenging, requiring additional invariance to perspective distance due to the requirement to operate with cameras mounted on top of robot bodies rather than protruding from them. Second, we required our data to come from a moving vehicle without expensive image stabilisation, so intentionally included some blurred images which confuse edge-based detection methods in particular, as these edges become blurred and no longer trigger these detectors. Third, our data is required to come from a wide mixture of lighting and weather conditions as would be encountered in real-world applications. Fourth, we did not allow fitting of any parameters to the test set, and allowed each algorithm to see the test data only once, and report only these results. Fifth, we have removed all possible experimenter, data set selection, and publication bias by operating as an independent controlled study rather than setting out to show the benefits of any one method.

For some applications, such as treatment by individual species-selective herbicides, finer classification of weed type into Rumex and Urtica may be required. Correct classification of Urtica is harder to achieve than of Rumex, using the overall best LBP-SVM method. This is likely because Rumex has larger, flatter leaves which present more obvious differences to most features than Urtica's smaller and more contoured leaves. The BRISK-KMEANS-SVM method shows less difference in performance between Urtica and Rumex when run on very large windows, as expected this may be due to its ability to pick up the jagged edges of Urtica leaves. However it does not work well for the regular 64^2 and 28^2 pixel windows, because it depends on the ability to select good interest points from a large image. With 64^2 pixels the choice of interest points is very limited and with 28^2 is almost non-existent, resulting in few or no interest points being found to classify. LBP and BRISK are closely related, with LBP viewable as a special case of BRISK that treats *every* pixel as an interest point and forces it to be included in the classification, which explains why LBP outperforms BRISK for the smaller windows. The fact that Rumex is in general easier to classify than Urtica may explain the existence of the many more published method-proposing studies of Rumex vision than Urtica vision.

Evidence for the contribution of perspective effects to reducing accuracy is given by the distance experiment result, which shows a considerable drop in accuracy as a function of distance from the camera. If all plants were part of a perfectly flat ground surface then the affine transformation would yield identical images to those taken by a vertical overhead camera as used in previous studies. However real plants and ground are not flat and in particular the vertical structure of plants near the camera results in them being enlarged out of proportion by the perspective transform. The distortion is tolerable for short distances but makes the system less useful beyond distances of around 1.5m. This suggests that for robots that are not able to mount vertical overhead cameras, for example rough terrain specialist robots for which it is undesirable to have overhanging parts that could be damaged by collisions, it may be preferable to concentrate visual processing power only on nearby regions of ground space. Computation is a limited resource for most mobile robots, which must trade off frame rate for size of spatial area to process and battery power consumption. Designers of these robots should consider increasing frame rates to obtain multiple views of the same nearby terrain up to around 1.5m away, at the expense of ignoring further away terrain. It is possible that some improvements to distant recognition will be possible using higher resolution cameras, to produce less pixel distortion during dewarping; by using cameras with smaller apertures to gain deeper depth of field; and/or by mounting cameras at higher positions such as one poles above the robot.

The effect of weather conditions on classification appears somewhat ambiguous from the tests conducted here. The classifiers were trained on mixed sunny and overcast data, then tested on mixed, sunny-only and overcast-only data. Overcast weather yields mostly diffuse lighting from the whole sky, while sunny weather comprises mostly directional light from the sun's position in the sky, which gives rise to distinct shadows. In some cases the shapes of shadows may assist classification (eg. the shapes of Urtica leaf shadows include the same distinctive jagged edges as the leaves themselves), while in other cases shadows may act as noise over the features of the real leaves. There is no clear contributing weather factor to these results, unlike the weed type breakdown which gave clear evidence that Urtica are more responsible than Rumex for lowering performance. Future work could try training classifiers on sunny-only and on overcast-only data, or on more nuanced partitions of weather type, and test them on matched conditions. However unlike the present experiment, this would require online robots to first classify the overall weather condition in order to choose which classifier to use, which introduces further complexity.

More fine grained weather and time-of-day classifications could be made. These can affect both the spectrum of light illuminating the plants, and how the light interacts with the plants, for example casting shadows, or cases of viewing low sunlight through leaves. The current dataset contains time-of-day information which could be used to break down performance in this way; epochs might also be further sub-classified into more detailed weather conditions, or more epochs obtained from new weather and time-of-year conditions. The data sets are all collected from the same weeds plots, and while we have argued that these do produce substantial variation in the images, it would be useful to validate the methods on completely separate plots in the future.

Window size is obviously an important factor on accuracy, because larger windows contain more information than smaller ones. However there is a trade off because they represent larger spatial regions which reduce the available accuracy of precision spraying. Robot designers can choose between 83% accuracy at 106mm resolution, or 72% accuracy at 56mm resolution, from the present studies. Very large windows can give near-perfect results as in the large-window BRISK experiments yielding 95% accuracy for mixed weeds and 97.9% accuracy for Rumex. In practice of course, classifications of single windows are unlikely to take place completely independently of one another. Rather, in the field, a live robot would perform both spatial averaging of neighbouring window classes, as well as temporal averaging as multiple images of the same regions are taken over time from moving robot locations, for example via Markov Random Fields as in [18], which would improve accuracy. The windows used here are non-overlapping

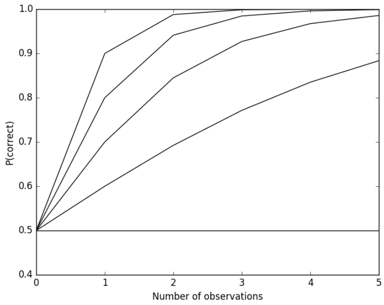


Figure 8: **Multi-observation fusion.** Showing the effect of Bayes-fusing multiple observations of a window. A live moving robot is likely to make several observations and classifications of each window from different poses, which when fused together will increase accuracy. Each line represents fusion of n observations of equal accuracy, whose per-observation accuracy is seen in the 'number of observations = 1' case ranging from 0.5 to 0.9 in steps of 0.1.

and live robots could further make use of overlapping windows to provide additional information about the spatial frequencies across window boundaries that is not used in the present study. An indication of the strength of accuracy amplification by multi observation fusion (which could be temporal or spatial or both) is shown in fig. 8. This shows that the present LBP-SVM accuracy is easily amplified into the mid or high 90s percentages by 2 to 5 observations, as could be obtained by a moving robot. This is computed by Bayes-fusing accuracies with themselves repeatedly, where the Bayesian fusion of two evidence probabilities p and q is given by,

$$\frac{pq}{pq + (1-p)(1-q)} \tag{18}$$

It can be seen from fig. 8 that fused data from two or three windows, observed over time and/or space, is sufficient to bring mot methods to 95%+ accuracy which is generally sufficient (e.g. [34]) for spraying use in the field.

Running multiple classifiers and combining their results with an LDA meta-classifier yields a slightly higher accuracy than pure LBP-SVM. The effect is only slight because all the feature-classifier methods mostly work with similar image information to each other, with varying levels of success, rather than working with different types of information. This information is presumably Fourier or wavelet-like, and mostly linear, though with some nonlinearities which enable the nonlinear SVM to outperform the other methods. All of the classification methods tested here run comfortably in real time for live use, however running them all simultaneously for meta-classification on a mobile robot platform would likely require parallel processors which consume additional and valuable battery power.

As part of this publication we are making our training and test sets available for non-commercial research by others under Creative Commons licence CC BY-NC 3.0 US. We hope that the standardised setup presented here will enable performance to be improved upon through fair evaluation of new methods and implementations, and though other researchers adding data from new conditions to the set.

References

1. Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In European Conference on Computer Vision, pages 102–115. Springer, 2008.

2. Faisal Ahmed, Md Hasanul Kabir, Shayla Bhuyan, Hossain Bari, and Emam Hossain. Automated weed classification with local pattern-based texture descriptors. Int. Arab J. Inf. Technol., 11(1):87–94, 2014.

3. T Anken, M Holpp, W Venn, HJ Kutterer, et al. Automatic detection of broad-leaved dock in grassland. In International Conference on Agricultural Engineering-AgEng 2010: towards environmental technologies, Clermont-Ferrand, France, 6-8 September 2010. Cemagref, 2010.

4. Avital Bechar and Clément Vigneault. Agricultural robots for field operations: Concepts and components. Biosystems Engineering, 149:94–111, 2016.

5. José M Bernardo and Adrian FM Smith. Bayesian theory, 2001.

6. Simon Blackmore, Bill Stout, Maohua Wang, and Boris Runov. Robotic agriculture–the future of agricultural mechanisation. In Proceedings of the 5th European Conference on Precision Agriculture, pages 621–628, 2005.

7. James S Cope, David Corney, Jonathan Y Clark, Paolo Remagnino, and Paul Wilkin. Plant species identification using digital morphometrics: A review. Expert Systems with Applications, 39(8):7562–7573, 2012.

8. Thomas G Dietterich. Ensemble methods in machine learning. In International workshop on multiple classifier systems, pages 1–15. Springer, 2000.

9. Pedro Domingos. Bayesian averaging of classifiers and the overfitting problem. In ICML, volume 2000, pages 223–230, 2000.

10. L Dürr, T Anken, H Bollhalder, J Sauter, KG Burri, and D Kuhn. Machine vision detection and microwave based elimination of rumex obtusifolius l. on grassland. In 5th European Conference on Precision Agriculture, Uppsala, Sweden, page 5, 2005.

11. Jerome Friedman, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning, volume 1. Springer series in statistics Springer, Berlin, 2001.

12. Steffen Gebhardt, Jürgen Schellberg, Reiner Lock, and Walter Kühbauch. Identification of broad-leaved dock (rumex obtusifolius l.) on grassland by means of digital image processing. Precision Agriculture, 7(3):165–178, 2006.

13. Lei Guo and Qingshan Li. Lbp and its variations for image classification. In Proceedings of the 2012 International Conference on Electronics, Communications and Control, pages 712–715. IEEE Computer Society, 2012.

14. Esmael Hamuda, Martin Glavin, and Edward Jones. A survey of image processing techniques for plant extraction and segmentation in the field. Computers and Electronics in Agriculture, 125:184–199, 2016.

15. Dong-Chen He and Li Wang. Texture unit, texture spectrum, and texture analysis. IEEE transactions on Geoscience and Remote Sensing, 28(4):509–512, 1990.

16. Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. Neural computation, 18(7):1527–1554, 2006.

17. S Hiremath, G Van der Heijden, FK Van Evert, and A Stein. The role of textures to improve the detection accuracy of rumex obtusifolius in robotic systems. Weed research, 52(5):430–440, 2012.

18. Santosh Hiremath, Valentyn A Tolpekin, Gerie van der Heijden, and Alfred Stein. Segmentation of rumex obtusifolius using gaussian markov random fields. Machine vision and applications, 24(4):845–854, 2013.

19. Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. Tech. report, https://www.csie.ntu.edu.tw/ cjlin/, 2003.

20. Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In 2011 International conference on computer vision, pages 2548–2555. IEEE, 2011.

21. Cheryl McCarthy, Steven Rees, and Craig Baillie. Machine vision-based weed spot spraying: a review and where next for sugarcane? In Proceedings of the 32nd Annual Conference of the Australian Society of Sugar Cane Technologists (ASSCT 2010), volume 32, pages 424–432. Australian Society of Sugar Cane Technologists, 2010.

22. Geoffrey McLachlan. Discriminant analysis and statistical pattern recognition, volume 544. John Wiley & Sons, 2004.

23. D Moshou, D Kateris, XE Pantazi, and I Gravalos. Crop and weed species recognition based on hyperspectral sensing and active learning. In Precision agriculture13, pages 555–561. Springer, 2013.

24. Xanthoula-Eirini Pantazi, Dimitrios Moshou, and Cedric Bravo. Active learning system for weed species recognition based on hyperspectral sensing. Biosystems Engineering, 2016.

25. Akash Patel, DR Kasat, Sanjeev Jain, and VM Thakare. Performance analysis of various feature detector and descriptor for real-time video based face tracking. International Journal of Computer Applications, 93(1), 2014.

26. Gerassimos G Peteinatos, Martin Weis, Dionisio Andújar, Victor Rueda Ayala, and Roland Gerhards. Potential use of ground-based sensor technologies for weed detection. Pest management science, 70(2):190–199, 2014.

27. Gerrit Polder, Frits K van Evert, Arjan Lamaker, Arjan De Jong, GWAM Van der Heijden, LAP Lotz, T Van der Zalm, and C Kampenaar. Weed detection using textural image analysis. Plant Research International, PO Box, 16:6700, 2007.

28. Robert Rosenthal. The file drawer problem and tolerance for null results. Psychological bulletin, 86(3):638, 1979.

29. Dejan ŠEATOVIĆ. 3d-object recognition, localization and treatment of Rumex obtusifolius in its natural environment. In 1 st International Conference on Machine Control & Guidance, page 169, 2008.

30. Dejan Šeatović, Hansjörg Kutterer, and Thomas Anken. Automatic weed detection and treatment in grasslands. In ELMAR, 2010 PROCEEDINGS, pages 65–68. IEEE, 2010.

31. Muhammad Hameed Siddiqi, Irshad Ahmad, and Suziah Bt Sulaiman. Weed recognition based on erosion and dilation segmentation algorithm. In 2009 International Conference on Education Technology and Computer, pages 224–228. IEEE, 2009.

32. ND Tillett, John A Marchant, and A Hague. Autonomous plant scale crop protection. AgEng96, European Society of Agricultural Engineers A, 96:23–26, 1996.

33. FK Van Evert, G Polder, GWAM Van Der Heijden, C Kempenaar, and LAP Lotz. Real-time vision-based detection of rumex obtusifolius in grassland. Weed Research, 49(2):164–174, 2009.

34. Frits K van Evert, Joost Samsom, Gerrit Polder, Marcel Vijn, Hendrik-Jan van Dooren, Arjan Lamaker, Gerie WAM van der Heijden, Corné Kempenaar, Ton van der Zalm, and Lambertus AP Lotz. A robot to detect and control broad-leaved dock (rumex obtusifolius l.) in grassland. Journal of Field Robotics, 28(2):264–277, 2011.

35. Vladimir Vapnik. The nature of statistical learning theory. Springer Science & Business Media, 2013.

36. Anup Vibhute and SK Bodhe. Applications of image processing in agriculture: a survey. International Journal of Computer Applications, 52(2), 2012.

37. Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE transactions on pattern analysis and machine intelligence, 13(6):583–598, 1991.

38. Els Vrindts and Josse De Baerdemaeker. Optical weed detection and evaluation using reflection measurements. In Photonics East (ISAM, VVDC, IEMB), pages 279–289. International Society for Optics and Photonics, 1999.

39. Eric W Weisstein. Fast fourier transform. Wolfram World of Mathematics (online resource), 2015.

Appendix

Here we give the intermediate results used for the parameter optimisation stage of classification for all 12 methods when trained on hyper-training and tested on hyper-test datasets containing grass vs mixed weeds under mixed weather conditions. This gives an indication of the effect of hyper-parameter optimisation on performance.

Tables 5 and 12 give the results for all classification methods except those involving the NN classifier (results for the NN classifier are given in the text) used on 28^2 pixel windows. 'nan' indicates an experiment abandoned due to unreasonable computation time, while NA stands for 'not applicable'. Tables 13 and 20 give the equivalent results for 64^2 pixel windows. For LBP$\rangle$SVM, tables 5 and 13 give the results for the first stage of hyper-parameter optimisation, in which the hyper-parameters of the SVM C and γ were optimised. The second stage of optimisation for this method yielded optimised values for the hyper-parameters of the LBP (n_{points}=20,r=5,accuracy=72.50% for 28^2 pixel windows, $n_{\text{points}} = 24$, r=4,accuracy=88.00% for 64^2 pixel windows). For LBP$\rangle$NN, optimised parameters for the LBP were $n_{\text{points}} = 2$, $r = 4$ for 28^2 pixel windows and $n_{\text{points}} = 10$, $r = 12$ for 64^2 pixel windows. For B$\rangle$K$\rangle$NN, the optimal value of K for the B$\rangle$K method of feature extraction was 17 for 28^2 pixel windows and 18 for 64^2 pixel windows. For FT$\rangle$NN there were no parameters to optimise. For W$\rangle$NN, the optimal value of d_{min} for the W method of feature extraction was $[1, ..., 10]$ for 28^2 pixel windows and $[1, ..., 5]$ for 64^2 pixel windows.

Table 5: Results for the classification method LBP$\rangle$SVM with hyper-training and hyper-test datasets, on 28^2 pixel windows.

Kernel	Shrinking	Accuracy(%)	Optimum C	Optimum γ
linear	off	65.6	2^{11}	NA
linear	on	65.6	2^{11}	NA
rbf	off	68.3	2^{13}	2^{-1}
rbf	on	68.4	2^{13}	2^{-1}

Table 6: Results for the classification method B$\rangle$K$\rangle$SVM with hyper-training and hyper-test datasets, on 28^2 pixel windows.

Kernel	Shrinking	Accuracy	Optimum K	Optimum C	Optimum γ
linear	off	53.1	4	2^{-5}	NA
linear	on	53.1	4	2^{-5}	NA
rbf	off	53.1	4	2^{-5}	2^{-15}
rbf	on	52.5	4	2^{-5}	2^{-15}

Table 7: Results for the classification method FT⟩SVM with hyper-training and hyper-test datasets, on 28^2 pixel windows.

Kernel	Shrinking	Accuracy(%)	Optimum C	Optimum γ
linear	off	67.1	2^3	NA
linear	on	67.1	2^3	NA
rbf	off	69.3	2^1	2^3
rbf	on	69.4	2^1	2^3

Table 8: Results for the classification method W⟩SVM with hyper-training and hyper-test datasets, on 28^2 pixel windows.

Kernel	Shrinking	Accuracy (%)	$d_{\min}$	Optimum C	Optimum γ
linear	off	65.7	$[1, ..., 9]$	2^3	NA
linear	on	65.7	$[1, ..., 9]$	2^3	NA
rbf	off	66.6	$[1, ..., 11]$	2^{15}	2^{-3}
rbf	on	66.7	$[1, ..., 11]$	2^{15}	2^{-3}

Table 9: Results for the classification method LBP⟩LDA with hyper-training and hyper-test datasets, on 28^2 pixel windows.

Solver	Shrinkage	Accuracy(%)	Optimum n	Optimum r
svd	off	67.6	28	3
lsqr	off	67.6	28	3
lsqr	on	67.6	28	4
eigen	on	67.6	28	3

Table 10: Results for the classification method B⟩K⟩LDA with hyper-training and hyper-test datasets, on 28^2 pixel windows.

Solver	Shrinkage	Accuracy	Optimum K
svd	off	54.3	4
lsqr	off	54.1	23
lsqr	on	54.9	20
eigen	on	54.9	23

Table 11: Results for the classification method FT⟩LDA with hyper-training and hyper-test datasets, on 28^2 pixel windows.

Solver	Shrinkage	Accuracy(%)
svd	off	65.5
lsqr	off	65.5
lsqr	on	65.6
eigen	on	65.4

Table 12: Results for the classification method W⟩LDA with hyper-training and hyper-test datasets, on 28^2 pixel windows.

Solver	Shrinkage	Accuracy	$d_{\min}$
svd	off	63.1	$[1, ..., 5]$
lsqr	off	56.5	$[1]$
lsqr	on	63.3	$[1, ..., 5]$
eigen	on	nan	nan

Table 13: Results for the classification method LBP⟩SVM with hyper-training and hyper-test datasets, on 64^2 pixel windows.

Kernel	Shrinking	Accuracy(%)	Optimum C	Optimum γ
linear	off	80	2^5	NA
linear	on	80	2^5	NA
rbf	off	82	2^9	2^3
rbf	on	82	2^9	2^3

Table 14: Results for the classification method B⟩K⟩SVM with hyper-training and hyper-test datasets, on 64^2 pixel windows.

Kernel	Shrinking	Accuracy	Optimum K	Optimum C	Optimum γ
linear	off	59	17	2^3	NA
linear	on	61.25	17	2^5	NA
rbf	off	68.5	17	2^1	2^8
rbf	on	63	17	2^3	2^3

Table 15: Results for the classification method FT⟩SVM with hyper-training and hyper-test datasets, on 64^2 pixel windows.

Kernel	Shrinking	Accuracy(%)	Optimum C	Optimum γ
linear	off	76.5	2^5	NA
linear	on	76.5	2^5	NA
rbf	off	79	2^3	2^1
rbf	on	79	2^3	2^1

Table 16: Results for the classification method W⟩SVM with hyper-training and hyper-test datasets, on 64^2 pixel windows.

Kernel	Shrinking	Accuracy (%)	d_{min}	Optimum C	Optimum γ
linear	off	72.5	$[1, ..., 10]$	2^{13}	NA
linear	on	72.5	$[1, ..., 10]$	2^{13}	NA
rbf	off	76.5	$[1, ..., 10]$	2^{11}	2^3
rbf	on	76.5	$[1, ..., 10]$	2^{11}	2^3

Table 17: Results for the classification method LBP⟩LDA with hyper-training and hyper-test datasets, on 64^2 pixel windows.

Solver	Shrinkage	Accuracy(%)	Optimum n	Optimum r
svd	off	81.5	28	4
lsqr	off	81.5	28	4
lsqr	on	82	16	4
eigen	on	81.5	16	4

Table 18: Results for the classification method B⟩K⟩LDA with hyper-training and hyper-test datasets, on 64^2 pixel windows.

Solver	Shrinkage	Accuracy	Optimum K
svd	off	76.5	20
lsqr	off	77.5	17
lsqr	on	78	17
eigen	on	77.5	22

Table 19: Results for the classification method FT⟩LDA with hyper-training and hyper-test datasets, on 64^2 pixel windows.

Solver	Shrinkage	Accuracy(%)
svd	off	55.5
lsqr	off	53
lsqr	on	61
eigen	on	62

Table 20: Results for the classification method W⟩LDA with hyper-training and hyper-test datasets, on 64^2 pixel windows.

Solver	Shrinkage	Accuracy	$d_{\min}$
svd	off	67	$[1, ..., 15]$
lsqr	off	57	$[1, ..., 13]$
lsqr	on	66.5	$[1, ..., 13]$
eigen	on	nan	nan