Postprint of: Dembski J., Szymański J. (2020) Weighted Clustering for Bees Detection on Video Images. In: Krzhizhanovskaya V. et al. (eds) Computational Science – ICCS 2020. ICCS 2020. Lecture Notes in Computer Science, vol 12141. Springer, Cham. https://doi.org/10.1007/978-3-030-50426-7_34

Weighted Clustering for Bees Detection on Video Images

Jerzy Dembski[0000−0002−6011−1955] and
Julian Szymański[0000−0001−5029−6768]

Faculty of Electronic Telecommunications and Informatics
Gdańsk University of Technology, Gdańsk, Poland
jerzy.dembski@eti.pg.edu.pl
julian.szymanski@eti.pg.edu.pl

Abstract. This work describes a bee detection system to monitor bee colony conditions. The detection process on video images has been divided into 3 stages: determining the regions of interest (ROI) for a given frame, scanning the frame in ROI areas using the DNN-CNN classifier, in order to obtain a confidence of bee occurrence in each window in any position and any scale, and form one detection window from a cloud of windows provided by a positive classification. The process has been performed by a method of weighted cluster analysis, which is the main contribution of this work. The paper also describes a process of building the detector, during which the main challenge was the selection of clustering parameters that gives the smallest generalization error.
The results of the experiments show the advantage of the cluster analysis method over the greedy method and the advantage of the optimization of cluster analysis parameters over standard-heuristic parameter values, provided that a sufficiently long learning fragment of the movie is used to optimize the parameters.

Keywords: automatic bee's image detection, convolutional deep neural networks, weighted clustering, bee monitoring

1 Introduction

In this paper, we present the approach used for building a bee detection system that is part of a larger project on apiary monitoring with the usage of IT Technologies [1,2]. The main goal of the research presented here is to build a system that allows us for non-invasive, real-time monitoring of the bee family using video analysis. Usage of cameras and algorithms allows us to quantify the amount of the bees coming out and coming into the hive that is an important factor for a beekeeper indicating how the bee colony develops during the season.

Bees tracking is a challenging task, which is related to the specificity of this problem. There are at least four causes for this: bees move fast, they are small, on a single video frame can be a large number of items and they are very similar one each other.

The bees tracing and hive entrance monitoring can be done in different ways. For example paper [3] shows how to use for that purpose RFID Tags. In our research, we assume a usage non-invasive methods and we decided to focus on image analysis. In that domain, some works have been already done.

One of the first systems aiming at monitoring bees traffic from images has been presented in [4]. The study has been based on SVM classifier and it allows to identify the individual honeybee. The analysis of bees flight activity at the beehive entrance using tracking of flight paths has been shown in [5]. Honeybee hive health monitoring by image processing has been presented and analyzed [6] by the usage of two different approaches based on the illumination-invariant change detection algorithm and signal-to-noise ratio to estimate the number of bees at the entrance of the hives. Tracking of honeybees has been also done using an integrated Kalman Filter and Hungarian algorithm [7]. The problem of bees detection has been also tackled in 3D space. Paper [8] presents a stereo vision approach that allows detecting bees at the beehive entrance and is sufficiently reliable for tracking.

The solutions mentioned above have been constructed and most of all tested on data prepared for the particular model. The images have been taken for a fixed scene and do not show the applicability of the solutions while cameras, unlike during the training, are put in different angels to the hive as well as while light conditions are changing. Also, the images used in the above-mentioned research have been made in low resolutions that is obvious simplification due to the efficiency of processing online date in real-life applications.

Our paper is constructed as follows: In the next section we describe the data acquisition and processing used in our research. Then we describe the architecture and algorithms used in our system. In Section 4 the experiments and results are described. The paper finalizes with conclusions.

2 Data preparation

For the requirements of the project to tag the data, we have implemented own software, for fast and easy selection of windows containing bees on video images. It allows preparing examples for a window classifier training. The contours of the bees touch at least two edges of the window, which results from the fact that for simplicity the windows are square – it is easier to mark bees with square rather than rectangular windows. The selection of one individual on subsequent frames is facilitated by indicating its center and the length of the side of the square because the size of a bee changes at a lower pace than its location, so a human can mark bees with single clicks on many successive frames. Also, areas, where there are no bees, are marked on each video to generate negative examples for the initial stage of classifier learning.

Additionally, special areas of exclusion were marked. These are, for example, around the outlet of the hive or places where dead individuals are found. It was done to avoid excessive effort and controversial marking decisions. Exclusion

areas are ignored when negative examples are generated for training the classifier (point 3.2) and when detection is evaluated (see points 3.3 and 4).

According to the idea of adapting the system to the environment described in section 1, the system was built and tested for three different camera positions and settings. For each of these, a Full-HD (1920 x 1080) pixel movie was recorded at a frequency of 50 frames/sec. Each of the movies was manually marked with windows containing bee images, including all bee images except for exclusion areas. Large windows containing no bees were also marked, from which negative examples were generated for training purposes and selected small windows containing objects resembling bees (e.g. bee shadows, knots, inflorescences of some plants, e.g. of the genus Burnet (Sanguisorba L.), potentially be false positive examples.

Each of the original movies was divided into two almost equal parts. The first part was used for training the system, the other was intended only for the final tests of the tuned system and was not used at any stage of learning.

Statistics of the data for particular movies used for training and for test are summarized in Tab. 1.

The data and software used in this research we have made publicly available and they are accessible in the web url: https://goo.gl/KNV7sd.

Table 1. Statistics of sample movies

movie	DSC_ 0559_A	DSC_ 0559_B	DSC_ 0562_A	DSC_ 0562_B	01091_A	01091_B
num.frames	1149	630	2097	2631	1270	1634
num.pos.windows	2726	2728	6755	6750	6004	6009
num.individuals	94	92	121	224	81	79

3 System architecture

The components of our system, that aims at precisely extract bees from the background on particular movie frames has been shown in Fig. 1. It consists of three parts: procedure for determining ROI where a bee can potentially be found, the procedure for window classification by scanning particular frames within the ROI, and procedure for extracting windows that match bee images.

Our preliminary research on bee detection algorithms have been presented in [2] where the study of different color models for image representation have been given. We extend the system introducing three significant modifications that contribute to this paper:

– new models of artificial neural networks were trained separately for three selected environments (camera settings for different hives) by two-stage method of false positive error reduction described in Sect. 3.2,

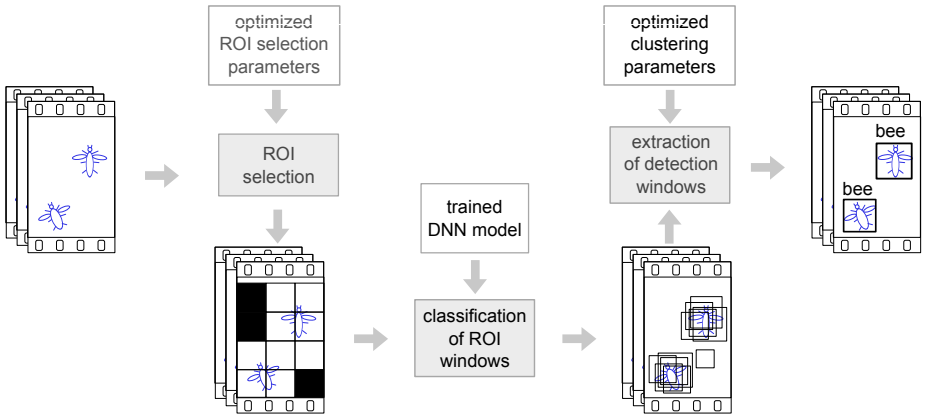

Fig. 1. Diagram of the bee detection system

- changed the greedy method of window selection to a method based on a weighted clustering algorithm with parameter tuning using the simulated annealing method described in details in Sect. 3.3,
- the function of evaluating detection results has been developed (described in Sect. 3.3).

3.1 Procedure for determining regions of interest

Due to two assumptions of the system: the camera is stationary and only bees in motion are considered, it is possible to extract regions of interest by relatively low computational cost by accepting only those windows in which motion was detected for further processing. The easiest way to detect motion is to compare the contents of the window in two adjacent movie frames. If the difference exceeds a certain threshold θ, the window is qualified as potentially containing a bee. Then it is only necessary to determine how to calculate the difference based on image features. In the simplest case, the set of image features may consist of the intensities of particular channels in the RGB model or the intensities of pixels in the B&W model. They can also consist of histograms of intensity levels or histograms of gradient directions [9]. After determining the type of feature vector, the next dilemma may be due to the function of the distance between vectors. In an extension of the simple Euclidean distance, the cosine distance can be used as less sensitive to general changes in brightness, e.g. due to obscuring the sun by clouds or Pearson distance. Additionally, the window can be divided into smaller blocks for thresholding, which reduces the system's sensitivity to local noise and lighting changes, and increases the sensitivity to displacement of objects which only occupy a part of the window. For example, a bee with open wings covers about 30% of the window surface.

After the preliminary experiments, the simplest feature vector variant was adopted in the form of a pixel intensities vector divided into R, G and B channels,

Euclidean distance between vectors and a division into blocks of $16x16$ pixels appeared to be sufficient. For each of these blocks, the distance is zeroed when the actual distance value does not exceed the threshold of η.

For each movie, optimization of the parameters θ and η was performed using a training fragment of the movie e.g. with A ending. We used the "brute-force" searching approach consisting of calculation the criterion for each point of the grid of 100×100 pairs of parameter values, which gives about 10000 cases. Both parameters were in the $(0, 1)$ range of values. As a criterion for optimization we used the weighted error sum $E_w = w_{fn} * E_{fn} + w_{fp}E_{fp}$, where w_{fn}, w_{fp} - partial error weights, E_{fn} - false negative error (the ratio of the sum of manually marked windows representing bees recognized as the background to the sum of all manually marked windows representing bees), E_{fp} - false positive error (the ratio of the sum of windows containing the background considered ROI to the sum of all windows containing the background). These windows are generated from larger areas that certainly do not contain bees. The weight values were set at $w_{fn} = 0.95, w_{fp} = 0.05$, which means that it is more important to leave bees in the ROI than skipping the background windows, which only results in increased calculations. Additionally, the maximum error of false negative $E_{fn}^{max} = 0.005$ was assumed. It is not zero due to the occurrence of isolated cases when the bee is almost motionless in two successive frames.

3.2 Windows classification procedure by sliding window method

After determining the ROIs, these regions are scanned with the usage of the square sliding window at different scales, shifted in horizontal and vertical directions. Each window was classified whether it contains a bee or not. By default, scanning begins with windows that are 64 pixels wide and tall, and then the window width and height is multiplied by 1.2 factor up to 440 pixels. The window shift step is 20% horizontal and vertical of frame width. In total, for a Full-HD format frame, this gives 51644 of windows for evaluation. Each window is transformed to the standard resolution of 48×48 pixels and is fed as an input image to the deep artificial neural network with convolutional layers (DNN-CNN). The last softmax layer of DNN-CNN returns the probability that there is a bee in the window.

The two-step DNN-CNN model training was proposed to match the model to the specific camera and environment settings. In the first stage, positive examples are used in the form of bee images extracted from manually selected windows in the training movie. Negative examples are extracted randomly from manually selected areas in various places in a training movie where there were definitely no bees. After training, the first version of the model it was used as a window classifier on all frames of the training movie with non-ROI skipped windows and outside the exclusion areas. As a result of this scanning process, some background windows are classified as containing bee e.g. with false positive decision. The images from these windows are used as negative examples in the second step of DNN-CNN model training process. The final result of the two-step learning process was the DNN model with a significantly reduced number of false positive

classifications for specific camera settings and the environment. The scheme of the learning system is shown in the Fig. 2. The 6-layer artificial neural network

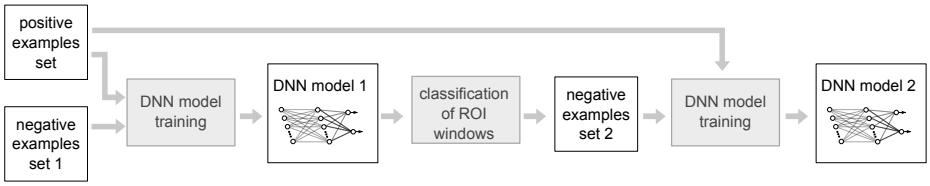

Fig. 2. Diagram of the two-step learning process of the DNN-CNN model as a window classifier

with three convolutional layers and three fully connected layers was used for building window classifier. As an input representation, the RGB three-channel model was chosen from 4 different color models, which was evaluated in the experiments described in [2]. The network was trained using a dropout technique with a keep probability 0.5 in all fully connected layers apart from the last one. We used cross entropy as a loss function and ADAM optimization. All learning parameters were selected experimentally as part of the work [2]. The output layer with softmax activation function returns the probabilities whether an input contains bee and background. To generate negative examples, as well as to calculate the classification error, it was assumed that the classification of the window in terms of bee content occurs when the probability of a bee occurrence is greater than or equal to 0.5. The network diagram is shown in Fig. 3. The

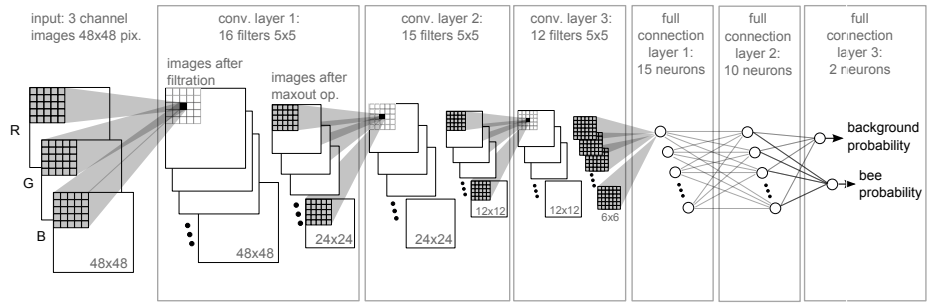

Fig. 3. Diagram of the DNN-CNN model - artificial neural network with convolutional layers determining the probability value that a bee is in the window

training process has been done using the TensorFlow library. The classification error of the final model version for particular test movies is given in Tab. 2.

Table 2. DNN-CNN generalization error calculated for test movies

DSC_0559_B			DSC_0562_B			01091_B		
error	fp	fn	error	fp	fn	error	fp	fn
0.0044	0.0070	0.0018	0.0115	0.0018	0.0212	0.0022	0.0000	0.0043

3.3 Window extraction form classification windows using weighted clustering algorithm

In many detection systems, both classic [10] and compact [11,12,13] the non-maximum suppression (NMS) method is used for window extraction. NMS is based on greedy removal of windows with a lower probability of object occurrence and more similar to other windows in the sense of covering. [14] describes the improved version of Soft-NMS, but still concerns a simple elimination of windows. There are more advanced methods for window extraction using different grouping methods – clustering. In our work we propose an original grouping method with the possibility of adapting its parameters to specific environmental conditions and camera settings.

We consider the window classification as positive if the probability returned by DNN-CNN network that it contains a bee image is greater or equal 0.5. However, during scanning the frame by sliding window the classifier positively classifies not only the window perfectly coinciding with the bee image, but also the windows slightly shifted and scaled relative to the true bee image. The reason for this is not the classifier inaccuracy, but rather the inaccuracy associated with manual selection of windows containing bees. For this reason, after the scanning process, we are obtaining a cloud of positive detection windows, instead of one window, perfectly matching the image of a bee. It became necessary to use an algorithm for determining windows that coincide with bee images.

The idea of the algorithm is to bring each dense window cluster to a single window with average parameters that should match the image of a bee. The algorithm should break the clusters into two windows, thus allowing detection of the case while two bees are very close together. It should also ignore clusters with a small number of windows that may arise due to false positive classification. Our algorithm was designed as an extension of the K-means clustering, adopting the following assumptions:

- Each square window in the movie frame is a point in the 3-dimensional space $p \in \mathcal{P}$ with the parameters x, y, d, where x, y - the coordinates of the center of the window, d - the length of its side,
- each cluster center $c \in \mathcal{C}$ is also a point in 3-dimensional space,
- the p_i window is assigned to the cluster with center c_j, when $n(p_i) = j$,
- the IOU function decides about the window assignment to the cluster, given by the formula $\mathrm{IOU}(p, q) = \mathrm{area}(p \cap q)/\mathrm{area}(p \cup q)$ for two windows p and q, item the probability that i-th window contains a bee - $Pr_{bee}(p_i)$ is used to calculate the window weight from the formula $w_i = Pr_{bee}(p_i)^{\beta}$, where β

- one of the clustering parameters, item the window weight w_i is used for calculation of the cluster center parameters, as well as decides if the cluster is sufficiently represented by the weighted sum of the windows that belong to it.

The algorithm is described by a pseudo-code:

Require:
$\mathcal{P}$ – window set with positive classification
$\mathcal{C}$ – initial set of cluster centers
$\alpha, \beta, \gamma, \delta$ – clustering parameters
while the specified number of cycles has not been reached **do**
  for all $p_i \in \mathcal{P}$ **do** ▷ 1. Assignment of windows to clusters
    if $\max_{c_j \in \mathcal{C}} \mathrm{IOU}(p_i, c_j) > \alpha$ **then**
      $n(p_i) \leftarrow \operatorname{argmax}_{j, c_j \in \mathcal{C}} \mathrm{IOU}(p_i, c_j)$
    else
      $n(p_i) \leftarrow$ null
    end if ▷ $n(p_i)$ – number of cluster which contain window p_i
  end for
  for all $c_j \in \mathcal{C}$ **do** ▷ 2. Modification of location of cluster centers
    $x(c_j) \leftarrow \sum_{i, n(p_i)=j} x(p_i) w_i / \sum_{i, n(p_i)=j} w_i$
    $y(c_j) \leftarrow \sum_{i, n(p_i)=j} y(p_i) w_i / \sum_{i, n(p_i)=j} w_i$
    $d(c_j) \leftarrow \sum_{i, n(p_i)=j} d(p_i) w_i / \sum_{i, n(p_i)=j} w_i$
    $\sigma_x(c_j) \leftarrow \sum_{i, n(p_i)=j} x(p_i)^2 w_i / \sum_{i, n(p_i)=j} w_i - x(c_j)^2$
    $\sigma_y(c_j) \leftarrow \sum_{i, n(p_i)=j} y(p_i)^2 w_i / \sum_{i, n(p_i)=j} w_i - y(c_j)^2$
  end for
  for all $c_j \in \mathcal{C}$ **do** ▷ 3. Removing clusters from a set $\mathcal{C}$
    if $\sum_{i, n(p_i)=j} w_i < \gamma$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \setminus \{c_j\}$
    end if
  end for
  for all $c_j \in \mathcal{C}$ **do** ▷ 4. Adding new cluster with high std.dev.
    $\sigma_{max} \leftarrow \max(\sigma_x(c_j), \sigma_y(c_j))$
    if $\sigma_{max} / d(c_j)^2 > \delta$ **then**
      $coord \leftarrow \operatorname{argmax}_{x,y}(\sigma_x(c_j), \sigma_y(c_j))$ ▷ selection of coordinate
      $c_{new} \leftarrow c_j$
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_{new}\}$ ▷ adding a new cluster
      $coord(c_j) \leftarrow coord(c_j) - \sqrt{\sigma_{max}}/3$ ▷ shifting of new clusters centers
      $coord(c_{new}) \leftarrow coord(c_j) + \sqrt{\sigma_{max}}/3$
    end if

end for
end while
return $\mathcal{C}$

To the input of the weighted clustering algorithm is given a set of positively classified $\mathcal{P}$ windows by DNN-CNN network, initial cluster centers, and clustering parameters. The initial locations of the cluster centers are determined using the greedy algorithm described in [2] employing the gradual averaging of parameter pairs of overlapping windows. The main algorithm loop consists of four nested "for" loops. The first and second of them are analogous to the *K-means*and allow the assignment of windows to clusters alternately with correction of a location of cluster centers based on their windows. However, due to the specifics of the application, there are two main differences. The first is the additional condition of window assignment to the cluster, which is based on a test if the maximum match in the sense of IOU is less than the α threshold. If so, then the window is significantly outside the bee images represented by the cluster centers and should therefore not have any assignment as probably false positive classification. The second difference is the usage of weight for each i-th window. The next two nested "for" loops allow us to remove and break the clusters. The first option allows for removing false positive detections, which are usually represented by a small number of windows. Breaking the cluster into two allows extracting images of two bees that are very close together or even partially overlapping. In the algorithm, this is solved by adding a copy of the primary cluster and then moving these two clusters towards the coordinate with the maximum standard deviation. This procedure was carried out when the maximum coordinate standard deviation of σ_{max} normalized with the window area exceeds the given threshold value δ.

3.4 Clustering parameter optimization

The optimization of $\alpha, \beta, \gamma, \delta$ parameters, similarly as in the case of optimization of ROI determination parameters and DNN-CNN model learning, takes place each time for a given camera setting on the training movie. We used simulated annealing as optimization algorithm due to its simplicity, although of course other optimization methods such as genetic algorithms or PSO could be used. The applied algorithm of simulated annealing, after adopting the initial parameter values, works periodically, randomly shifting the parameter vector in each cycle, and then after calculating the evaluation, can reject or accept new solution with probability $Pr_{accept} = (1+\exp(\Delta E/bT))^{-1}$. This probability value depends on the change in the evaluation value of the solution $\Delta E = E(t)-E(t-1)$ in step t and the randomness factor T – the temperature that initially allows greater exploration of solutions (acceptance of worse solutions). The b constant differentiate the effect of temperature on the acceptance probability and the magnitude of the parameter vector shift. Determining the assessment of the solution in a given step of the annealing algorithm t: $E(t)$ is based on running a weighted clustering algorithm with new parameters for a given number of cycles, and

then calculating the coverage error of the obtained detection windows in relation to the windows determined manually in the training movie. Unfortunately, the detection error measures known from the literature such as Mean Average Precision (mAP) do not change for small window shifts, which is necessary to direct the search in optimization. The second reason why we do not use mAP is the difficulty in determining the probability that the window contains a bee because these windows are not determined directly by the DNN network, but are obtained as a result of extraction - averaging of many windows. Due to the requirements of the learning system, we used its own measure of coverage error represented by the Eq. 1 together with the algorithm to calculate it.

$$E = \frac{\sum_{k=1}^{K} |\mathcal{O}_k| + |\mathcal{C}_k| - m_{\mathcal{O}_k} - m_{\mathcal{C}_k}}{\sum_{k=1}^{K} |\mathcal{O}_k| + |\mathcal{C}_k|} \tag{1}$$

where K – the number of movie frames, $|\mathcal{O}_k|, |\mathcal{C}_k|$ – the number of window sets determined manually and by means of the weighted clustering algorithm on the frame k, $m_{\mathcal{O}_k}$ - the sum of the coverage degrees of windows determined manually with windows determined algorithmically, $m_{\mathcal{C}_k}$ - the sum of the coverage degrees of the windows determined algorithmically with windows determined manually on the frame k. If the manually selected windows perfectly coincide with the windows determined by the clustering algorithm than $E = 0$. The error will be non-zero when the bees are not detected and in false positive cases when the system classifies the background image as containing the bee.

The sums of coverage degrees on the frame are calculated by the algorithm:

Require:
$\mathcal{O}$ – a set of windows containing manually selected bee images
$\mathcal{C}$ – set of windows obtained by the weighted clustering algorithm
$\mathcal{R} : \mathcal{O} \times \mathcal{C} \equiv \{(o, c) | o \in \mathcal{O}, c \in \mathcal{C}\}$ – set of window pairs
$m_{\mathcal{O}} \leftarrow 0$
$m_{\mathcal{C}} \leftarrow 0$
while $|\mathcal{R}| > 0$ **do**
  $(a, b) \leftarrow \underset{i,j}{\operatorname{argmax}} \operatorname{IOU}(o_i, c_j)$ ▷ pair with the highest matching degree
  $m_{\mathcal{O}} \leftarrow m_{\mathcal{O}} + \operatorname{IOW}(o_a, c_b)$
  $m_{\mathcal{C}} \leftarrow m_{\mathcal{C}} + \operatorname{IOW}(c_b, o_a)$
  for all $(o_i \in \mathcal{O}, c_j \in \mathcal{C})$ **do** ▷ for all window pairs (o_i, c_j)
    if $i = a \vee j = b$ **then**
      $\mathcal{R} \leftarrow \mathcal{R} \setminus \{(o_i, c_j)\}$ ▷ removing a pair $\{(o_i, c_j)\}$
    end if
  end for
end while
return $m_{\mathcal{O}}, m_{\mathcal{C}}$

The IOW function determines the coverage degree of the first window by the second and can be represented by the formula: $\operatorname{IOW}(p, q) = \operatorname{area}(p \cap q)/\operatorname{area}(p)$.

Standard parameter values and values optimized for three training movies are shown in the Tab. 3.

Table 3. Cluster analysis parameter values obtained as a result of optimization for particular training movies

symbol	interpretation	standard par.	DSC_ 0559_A	DSC_ 0562_A	01091_A 01091_A
α	IOU threshold	0	0	0.044748	0.02259
β	power at probability that bee	2	-1.1917	2.1359	-2.6423
γ	sum of window weights threshold	5	4.8284	4.7435	4.7063
δ	normalized std. dev. threshold	0.5	2.1140	39.1152	18.5411

4 The experiments and results

The experiments were carried out for the same ROI regions and DNN-CNN models trained separately for each movie at earlier stages. The results of the experiments presented in the Tab. 4 allow us to compare movies registered in different conditions and three window selection methods: cluster analysis with optimized parameter values, cluster analysis with heuristically accepted – standard parameter values and the greedy method. Each experiment consisted of usage of a fragment of the movie to learn, for example DSC_0559_A to determine ROI areas, DNN classifier training and in the case of parameter optimization – searching for optimal clustering parameters with the criterion given by the Equation 1. Then for the test movie, for example DSC_0559_B, the error was calculated according to the Eq. 2:

$$E_{0.5} = \frac{\sum_{k=1}^{K} |\mathcal{O}_k| + |\mathcal{C}_k| - 2N_k}{\sum_{k=1}^{K} |\mathcal{O}_k| + |\mathcal{C}_k|}, \tag{2}$$

where N_k - the sum of pairs of windows determined manually and algorithmically (o_i, c_j) in the frame k, such that $\mathrm{IOU}(o_i, c_j) \geq 0.5$, assuming that each window determined manually could only coincide with one window determined algorithmically and vice versa. The 0.5 threshold value is the value most commonly used in other works related to image detection. For the method with the optimization of cluster analysis parameters, the table also provides the detection error of false positives $E_{0.5}^{fp} = \sum_{k=1}^{K}(|\mathcal{C}_k| - N_k)/\sum_{k=1}^{K}|\mathcal{C}_k|$, which can be interpreted as the proportion of algorithmically determined windows that do not cover any manually designated windows. False negative error $E_{0.5}^{fn} = \sum_{k=1}^{K}(|\mathcal{O}_k| - N_k)/\sum_{k=1}^{K}|\mathcal{O}_k|$ is the proportion of manually marked windows that were not covered after window extraction process. As it can be seen, the

Table 4. Detection error of windows containing bees for test movies

movie name	methods of extraction windows with bee images				
	parameters optimized on training movie			standard parameters	greedy method from [2]
	$E_{0.5}$	$E_{0.5}^{fp}$	$E_{0.5}^{fn}$	$E_{0.5}$	$E_{0.5}$
DSC_0559_B	0.0688	0.0504	0.0865	0.0619	0.0926
DSC_0562_B	0.0626	0.0546	0.0705	0.0646	0.0686
01091_B	0.427	0.494	0.340	0.417	0.524

error when using cluster analysis is always smaller than while using the greedy method. The error in two out of three setting cases was smaller for standard cluster analysis parameters, which is because tests were done on a different fragment of the movie from the fragment for parameters optimization. Such weak generalization may be related to short length of the movies, which may be confirmed by a fact that in both cases with weak generalization: DSC_0559_A and 01091_A the movies were much shorter than in the case of DSC_562_A where results after optimization is slightly better than with standard parameters. This may lead to the conclusion that the movies used for parameter optimization should be longer. The significantly greater error for the movie 01091 is probably due to greater consideration of the outlet area combined with a large number of mutually obscuring bees.

Fig. 4 shows the dependence of the test error on the assumed IOU threshold at which positive detection was considered. As can be seen in the case of DSC_0559 and DSC_562 for $x > 0.5$, the error increases very quickly, but in some applications, such as bee counting at low density, high detection precision is not required and even a lower threshold at the level of $0.1 \div 0.2$ can be used. In the case of further image analysis, e.g. to determine if a bee carries pollen, precision should be increased. Sample images after subsequent stages of the detection process are shown in Fig. 5.

5 Conclusions

The results of the experiments indicates that it is possible to build an effective bee detection system that can adapt to the specific camera settings and environmental conditions. This system can be implemented using the classical method in the form of three subsystems: the ROI area subsystem, the window classification subsystem along with the procedure of scanning individual frames of the video stream, and the subsystem for determining positive detection windows by the weighted clustering method proposed in this paper. The disadvantage of the current system is a longtime scanning and window classification does not yet allow the system to be operated on-line on high resolution images. Two-step training process of the window classifier allows the elimination of false positive

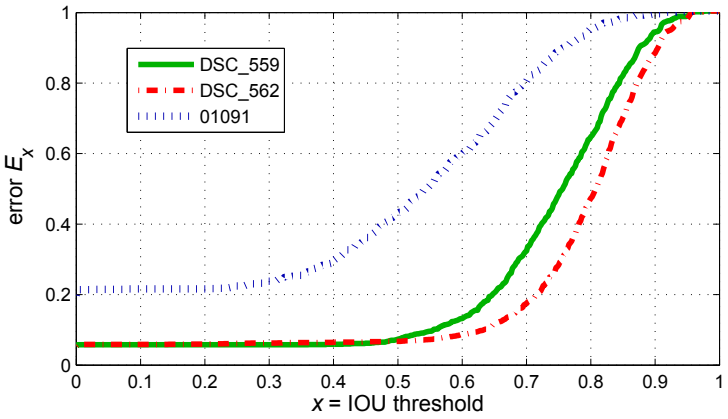

Fig. 4. Test error E_x depending on x - IOU threshold, above which detection is considered as positive

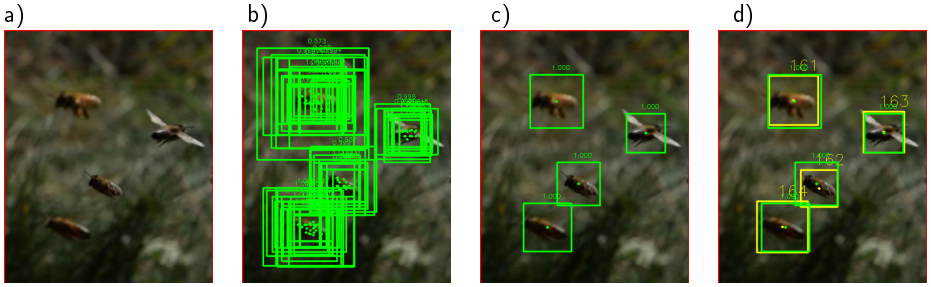

Fig. 5. Sample images from frame 1203 from the movie DSC_0562_B: a) original image, b) window clouds after window classification stage, c) windows after extraction by weighted clustering algorithm, d) comparison of positive detections (green) with windows marked manually (yellow)

windows in specific camera setting and in a specific environment, so that after the determination of positive detections there is almost no false positive detection despite the simple, only 6-layer architecture of DNN-CNN network. Weighted clustering is always better than the greedy method of windows selection, and the additional optimization of its parameters allows to achieve better results in the case of sufficiently long training movies.

Acknowledgements

The work has been supported by founds of Faculty of Electronics,Telecommunications and Informatics, Gdańsk University of Technology.

References

1. Cejrowski, T., Szymański, J., Mora, H., Gil, D.: Detection of the bee queen presence using sound analysis. In: Asian Conference on Intelligent Information and Database Systems, Springer (2018) 297–306
2. Dembski, J., Szymański, J.: Bees detection on images: Study of different color models for neural networks. In: International Conference on Distributed Computing and Internet Technology, Springer (2019) 295–308
3. de Souza, P., Marendy, P., Barbosa, K., Budi, S., Hirsch, P., Nikolic, N., Gunthorpe, T., Pessin, G., Davie, A.: Low-cost electronic tagging system for bee monitoring. Sensors **18** (2018) 2124
4. Chen, C., Yang, E.C., Jiang, J.A., Lin, T.T.: An imaging system for monitoring the in-and-out activity of honey bees. Computers and electronics in agriculture **89** (2012) 100–109
5. Magnier, B., Gabbay, E., Bougamale, F., Moradi, B., Pfister, F., Slangen, P.: Multiple honey bees tracking and trajectory modeling. In: Multimodal Sensing: Technologies and Applications. Volume 11059., International Society for Optics and Photonics (2019) 110590Z
6. Tashakkori, R., Ghadiri, A.: Image processing for honey bee hive health monitoring. In: SoutheastCon 2015, IEEE (2015) 1–7
7. Ngo, T.N., Wu, K.C., Yang, E.C., Lin, T.T.: A real-time imaging system for multiple honey bee tracking and activity monitoring. Computers and Electronics in Agriculture (2019) 104841
8. Chiron, G., Gomez-Krämer, P., Ménard, M.: Detecting and tracking honeybees in 3d at the beehive entrance using stereo vision. EURASIP Journal on Image and Video Processing **2013** (2013) 59
9. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). Volume 1. (2005) 886–893 vol. 1
10. Viola, P., Jones, M.J.: Robust real-time face detection. International journal of computer vision **57** (2004) 137–154
11. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition (2014)
12. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
13. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. Lecture Notes in Computer Science (2016) 21–37
14. Bodla, N., Singh, B., Chellappa, R., Davis, L.S.: Soft-nms — improving object detection with one line of code. 2017 IEEE International Conference on Computer Vision (ICCV) (2017)