Abstract.

On-trap moth automated identification suffers the problems of varieties of moth pose, life stage and sex, which finally lead to incomplete feature extraction and mis-identification. Due to the intra-species variance, a pose estimation-dependent automated identification method using deep learning architecture for on-trap field moth sample is proposed in this paper. To solve the segmentation task with cluttered background and uneven illumination, a two-level moth automated segmentation method for field moth trap was developed to obtain separate moth sample image from each trap image. Then based on the segmentation results, moth pose estimation was conducted to assign each moth to one of the two types of moth pose, which are top view and side view. Regarding to the refined moth pose types, suitable combination of texture, color, shape and local features were extracted for further moth description. Finally, an improved pyramidal stacked de-noising auto-encoder (IpSDAE) architecture was proposed to build a deep neural network for moth identification. The identification results on 428 field-based testing images of nine species with 480 lab-based training images achieved the identification precision value of 98.13%, which demonstrates the effectiveness of the proposed identification method and its potential application in integrated pest management. The results also indicated that the pose estimation process is effective for improving the moth identification results.

Keywords.

Field moth; automated identification; segmentation; feature extraction; deep learning

1 Introduction

Integrated pest management (IPM), an effective and accurate way for pest control, applies control methods based on the presence of pest in the field rather than regular spray. For example, moth pest control and monitoring work in field are implemented by calculating the number of pests on traps, which are placed around the orchard by growers or IPM consultants. According to the pest number on traps, population and distribution of moth pests in orchard field can be estimated, and pest control strategies are developed based on these data. Regular checking trap is very time-consuming when there are a great number of traps spread out around the orchard, and identifying the species of moth requires biological knowledge. Moreover, it’s urgent to build spraying strategy as soon as possible because some pests are emerging in a very short term, but the best time to spray may be missed because a lot of time is wasted in checking traps and identifying the insects.

The paper proposes a pose estimation-dependent method to identify the field moth species and ultimately aims to solve the problem of field moth pest automated identification regarding to intra-species variance. The results of the study will lay the foundation to achieve real-time pest insect monitoring in orchard. We firstly introduce a moth automated segmentation method for field moth traps to obtain separate insect sample in each image, which the method can solve the segmentation task under cluttered background with grid lines, moth scales and broken parts of moth. Then moth pose estimation based on segmentation result is conducted to estimate the possible pose of moth. Regarding to the moth pose type, suitable features are extracted to represent each moth sample. An improved pyramidal stacked de-noising auto-encoder (IpSDAE) model, a deep learning architecture, was proposed for moth identification. Finally, the field traps images were tested based on the above process for automated moth identification using lab based training images.

1.1 Automated insect identification and classification

Automated insect identification and classification can improve the identification efficiency and accuracy, and many relevant studies have been published (Arbuckle et al., 2001; Russell et al., 2005; Weeks et al., 1999; Wen et
Recently, more results related to automated insect identification and classification have been released using different feature combination and identification models. Wang et al. proposed a content-based image retrieval (CBIR) method to identify butterfly species at family level based on shape, color and texture features (Wang et al., 2012). Artificial neural networks and support vector machine were compared for identifying nine common orders and sub-orders insect (Wang et al., 2012). Wen and Guyer presented an image-based method for field moth identification based on the combination of global and local features (Wen and Guyer, 2012). Yaakob and Jain developed a quality threshold ARTMAP neural network model for classifying insect, and six different types of moment invariant shape features were compared and analyzed (Yaakob et al., 2012). Larios et al. developed an automated approach for stonefly larvae species identification with combination of three region detectors (Larious et al., 2010). Kang et al. identified five butterfly species using shape feature, which is characterized by partitioning and weighting the entropy profile (Kang et al., 2012). Guarnieri et al. developed an automatic electronic trap, which is used to monitor the flight of codling moth males and identify the pest by the information of trapped males remotely (Guarnieri et al., 2011).

Most of these methods work well when the insects are with good pose and under the uniform lighting situation. Comparing to the sample images in above methods, the images of field insects have higher complexity, such as varied insect pose and cluttered background.

### 1.2 Pose variance problem

From our previous study (Wen and Guyer, 2012), it has been found that the insect pose changes before the insect completely lays down on the trap. Through the wind tunnel experiments, we also studied the time-dependent pose change of insect on trap. Moth automated identification on trap suffers the problem of variety of moth pose, which finally leads to incomplete feature extraction and mis-identification. When dealing with the insect classification and identification task, the insect image with bad pose are usually discarded, for example the side view pose insect image. Insufficient features will be obtained under this situation, and finally leads to incorrect decision. Three types of typical moth poses are commonly observed on moth trap, which are top view with wing opened, top view with wing closed and side view (as shown in Figure 1).

![Figure 1. Common poses of moth. (a) Top view with wing opened. (b) Top view with wing closed. (c) Side view](image)

As we can see from Figure 1(a) and (b), relative complete and clear texture and color feature of moth are obtained for both top view poses insect, but the shape and size are different. Poor texture and color feature are obtained for side view pose insect, and the shape and size features couldn’t reveal the species of the moth. Moreover, intra-species variances also brings difficulties for moth identification, for example the same species at different life stage and the same species but different sex. With the inter-species similarity and intra-species variances, it’s obviously that no single feature is able to identify the moth successfully. To solve these problems, stable and robust feature extraction and description according to the moth pose, and better learning model for identification are needed to deal with the intra-species variances brought by moth pose, life stage and sex.

### 1.3 Deep learning structure

Theoretical results suggested that deep architectures can be an effective way to represent high-level abstractions via learning some kind of complicated functions. Deep architectures are composed of multiple levels of non-linear operations. Automatically learning features at multiple levels of abstraction allows a system to learn complex
functions mapping the input to the output directly from data, without depending completely on feature extracted. (Bengio, 2009)

Over the last few years, huge amount of research on visual recognition has focused on learning low-level and mid-level features using unsupervised learning, supervised learning, or a combination of above two. The ability to learn multiple levels of good feature representations in a hierarchical structure helps to construct sophisticated recognition systems (Kavukcuoglu et al., 2010). Lee et al. presented the convolutional deep belief network, a hierarchical generative model which scales to realistic image sizes (Lee et al., 2009). Hinton et al. presented a deep belief network (DBN), which is a multilayer generative model where each layer encodes statistical dependencies among the units in the layer. DBNs have been successfully applied to learn high-level structure in a wide variety of domains (Hinton et al., 2006). Krizhevsky et al. presented that a large deep convolutional neural network is capable of working on a highly challenging dataset using purely supervised learning (Krizhevsky et al., 2012). Cire et al. presented a deep hierarchical architecture which achieve the best published results on benchmarks (NORB, CIFAR10) for object classification and handwritten digit recognition (Ciresan et al., 2011). Turaga et al. presented a machine learning approach to computing an affinity graph using a convolutional network (CN) trained using the ground truth provided by human experts (Turaga et al., 2010).

2 Materials and Image Acquisition

Cydia pomonella (Codling moth), Choristoneura rosacea (Obliquebanded leafroller), Argyrotaenia velutinana (Redbanded leafroller), Grapholita molesta (Oriental fruit moth), Platyctena idaeusalis (Tufted apple budmoth), Spilonota ocellana (Eyespotted budmoth), Rhagoletis pomonella (Apple maggot), Rhagoletis cingulata (Cherry fruit fly), and Grapholita prunivora (Lesser appleworm) are top common moth pests in the orchard, and require to be intensively monitored and controlled. It has been widely reported on research of moth insect automated identification and classification based on samples in lab. In this paper, we are focusing on the moth samples obtained from field, which image background and moth poses are more complicated. Field moth insect samples were obtained by sticky pest traps, which were distributed in the field and with pheromones on the trap to attract male moth. For each moth specie, several sticky traps for nine species were collected.

Image acquisition system established aims to afford by growers, which consists of a computer, a commercial webcam camera (Creative Inc., USA) and light components. The insect images captured by this imaging system yielded the problems of uneven lighting and background complexity, such as scales distributed around the insect when insect struggled to fly away the sticky trap. Examples of lab images of lab moths and webcam images of field moths were given (as shown in Figure 2). Lab microscope images of lab moth (Figure2(a) and (b)) were captured using microscope and digital camera to obtain images of insects colonies, and webcam images of field moths (Figure 2(c) and (d)) were captured by webcam camera with moth collected from orchard field (only shown a part of sticky trap image). Here, various poses of field moth, clutter background, for example grid lines, moth scales and broken body parts can be observed on sticky traps.

Figure 2. Examples of lab images of lab moths and webcam images of field moths

3 Automated Segmentation and Moth Pose Estimation

For each field moth image, the segmentation and classification process includes image segmentation, feature extraction and identification stages (as shown in Figure 3). Firstly, an automated segmentation method, which solves the segmentation task under the cluttered background, such as grid lines, moth scales and broken body parts of moth, is introduced to obtain separate insect sample within each image from field moth trap image. Then,
moth pose estimation based on segmentation result is conducted to estimate the possible pose of moth. Regarding to the moth pose type, suitable features are extracted as inputs of deep learning model for moth identification. Finally, the field traps images are tested for automated moth identification using lab based training images.

### 3.1 Automated segmentation of moth

An image block with individual insect is obtained by following aspects: firstly capture the image of sticky trap, following a morphological opening operation to eliminate the background, and use adaptive thresholding segmentation method to locate the approximate center of each individual insect.

Considering the complex background and uneven lighting of field sample image, we developed a morphological-based method to segment insect from each image block. Firstly, an adaptive thresholding segmentation method was applied to segment the sub-image and following an erosion operation in horizontal and vertical direction to take off the grid lines on the traps. Then, a morphological opening operation was performed to eliminate the small size objects, such as insect scales, impurities and broken legs of moth following a dilation operation to compensate the effect from erosion operation. Then, fill the hole and complete the segmentation. Finally, cut out sub-image (referring to image block above) with the pixel size of 160*120 around each moth center above.

Figure 4 shows the examples of fine segmentation results on images of Argyrotaenia velutinana samples collected from field, in which the white area (pixel 1) is the insect object and the black area (pixel 0) is the background.

![Image](image.png)

**Figure 4** Segmented images of field moths using morphological method

From Fig. 4, we found that the grid lines, scales of moth and broken body parts as well as moth itself can be observed in original field moth images. The placing relationships between the moth and the grid line were also complicated (on or off the line). The morphological-based segmentation method proposed can split insect from the background well, and segmentation results shown that the method can handle the problems of background cluttering, grid lines elimination, and relatively complete object segmented from the background. However, this method also removes the insect’s antennae and legs since erosion operation is used to eliminate small size objects, and some of morphological feature of moth will disappear after segmentation. Considering only shape and size related information but not antennae and legs are used as identification features, above problems will not affect the result of feature extraction and identification.

### 3.2 Moth Pose Estimation
3.2.1 Central axis detection

The central axis detection works as follows: (a) Segment the original image to be binary image. (b) Extract the edge of the insect. (c) Search inscribed triangle with maximum area. (d) Divide the insect into two parts by central axis of the inscribed triangle with maximum area. More details show in Figure 5.

![Figure 5. Central axis detection for moth](image)

3.2.2 Side view pose estimation by structural similarity calculation

As shown in Figure 6, we divide the side view moth image and top view moth image into the left and the right side. For top view pose images, there are relatively highly symmetric relationships between the left and right side in shape and texture. For side view pose images, there are weakly symmetric relationships between the left and right side in color, shape and texture. So, we can separate the moth image with side view pose from the moth image with top view pose using these symmetric relationships.

It also has been discovered that one half side of the moth with side view pose has fewer pattern since the scale on the wing is hidden because of this specific pose. We assumed that the texture difference between the two parts of moth body with side view pose is huge. Fundamentally, the luminance, contrast, and the structural between the two images are different, a structural similarity index similarity (SSIM) is used to compare the local patterns of pixel intensities which are normalized for luminance and contrast. The basic idea of SSIM is based on the fact that the structures of the objects in the scene are independent of illumination (L), contrast (c), and structure (s) (Silvestre-Blanes, 2011). Suppose L and R are the left and right part image of an insect. The SSIM index is defined as:

\[
SSIM(L, R) = [(L(R)c(L, R)S(L, R)].
\]

The luminance component is:

\[
l(L, R) = \frac{2\mu_L\mu_R + \sigma^2_L}{\mu^2_L + \mu^2_R + \sigma^2_L},
\]

where the mean intensity of the left and right part image is \(\mu_L = \frac{1}{N}\sum_{i=1}^{N} L_i\), and \(\mu_R = \frac{1}{N}\sum_{i=1}^{N} R_i\), and the constant \(C_1 = (k_1 I)^2\).

The contrast component is:

\[
c(L, R) = \frac{2\sigma_L\sigma_R + \sigma^2}{\sigma^2_L + \sigma^2_R},
\]

where \(\sigma_L = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(L_i - \mu_L)^2}\), and \(\sigma_R = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(R_i - \mu_R)^2}\), and the constant \(C_2 = (k_2 I)^2\).

The structure component is:

\[
s(L, R) = \frac{\sigma_{LR}\sigma^2}{\sigma^2_L + \sigma^2_R},
\]

where \(\sigma_{LR} = \frac{1}{N}\sum_{i=1}^{N}(L_i - \mu_L)(R_i - \mu_R)\), and constant \(C_2 = (k_2 I)^2\).

The SSIM index is:

\[
SSIM(L, R) = \frac{(2\mu_L\mu_R + C_1)(2\sigma_{LR} + C_2)}{(\mu^2_L + \mu^2_R + C_1)(\sigma^2_L + \sigma^2_R + C_2)}
\]
With (5), the SSIM index calculation based on two part images from each moth image was conducted, and a setting threshold value by experiences is used to determine the pose of the moth.

4 Feature Selection and Deep Learning Architecture

4.1 Moth pose-dependent feature Selection

With moth pose estimation, each moth image is identified to side view pose or top view pose. Then a feature set shown in Table 1 is extracted for each image (Wen and Guyer, 2012).

<table>
<thead>
<tr>
<th>Feature type</th>
<th>Descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometric features</td>
<td>area, perimeter, eccentricity, circularity ratio (compactness measure), major axis length, minor axis length, convex area, solidity and equivalence diameter</td>
</tr>
<tr>
<td>Shape features</td>
<td>Power spectrums at the first two spatial frequencies of the Fourier descriptor on moth contour.</td>
</tr>
<tr>
<td>Invariant moments</td>
<td>Seven moments derived from the second and third moments</td>
</tr>
<tr>
<td>Texture features</td>
<td>Twenty three GLCM texture features; Three statistical moments of the intensity histogram</td>
</tr>
<tr>
<td>Color features</td>
<td>Mean hue in HSV color space, standard deviation of hue in HSV space, mean saturation in HSV color space, standard deviation of saturation in HSV color space, mean hue in LCH color space, standard deviation of hue in LCH color space, mean saturation in LCH color space, standard deviation of saturation in LCH color space, mean lamination in LAB color space, standard deviation of saturation in LAB color space; the three largest elements of the 4-bin red-blue chromaticity histogram in RGB space</td>
</tr>
<tr>
<td>Local features</td>
<td>100 dimension normalized histogram bins from scale invariant feature transformation</td>
</tr>
</tbody>
</table>

4.2 Deep Learning Structure

Compared with the auto-encoder (AE), denosing auto-encoder (DAE) adds the noise to the train data, which means that the DAE learns to detach noise to get real data. It makes the encoder learning to be more robust to the expression of the input signal. As a result, the DAE gains better generalization ability than general AE. The stacked denoising auto-encoders (SDAE) is a deep network which is constructed by several stacked DAE.

We developed IpSDAE, an improved version of the pSDAE, based on our previous work (Xie et al., 2014). Compared to original pSDAE, the different fraction of corruptions, hidden layers, learning rates and the nodes of each layers are taken into consideration to optimize the architecture. All these parameters will finally influence the performance of the deep learning architecture, more detailed analysis of the model parameters are given in Section 5.

A deep learning architecture generally contains two stages: unsupervised pre-training and supervised fine-tuning. In our architecture, the pre-training stage is implemented by the IpSDAE, which each layer is basically trained as a DAE. The output of the lower layer is used as the input of the next higher layer. Through this process, the relationship among different layers is built up from the bottom to the top and the weights and biases of each layer are obtained. In short, the IpSDAE achieves the reconstruction of the original data. The latter is implemented by a feed-forward neural network, we use it for moth identification. The trained weights and biases are assigned to a neural network as the initial value, covering the random value, the output of the top layer correspond to nine pests. The algorithm of moth identification is shown in Table 2.
Table 2 Detailed algorithm of identification process

**Algorithm: Pattern recognition**

<table>
<thead>
<tr>
<th>Input:</th>
<th>Train_x, Train_y, Test_x, Test_y, n_layer, num_iter, W[n_layer], B[n_layer] n_action, n_hidden[n_layer], corruption[n_layer], LRate[n_layer]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>Error, Labels</td>
</tr>
</tbody>
</table>
1.      | X = Train_X; // construct a SDAE, achieve the trained weights and biases                                                                 |
2.      | For i=1 to n_layer                                                                                                                                 |
3.      | X′ = SDAE [i].Corrupt(X, corruption[i]);                                                                                                    |
4.      | SDAE[i].Initparams(X, n_hidden[i], LRate[i]);                                                                                                 |
5.      | For k=1 to num_iter                                                                                                                        |
6.      | Y = SDAE.Representation(X′, W[i], B[i])                                                                                                     |
7.      | Z = SDAE.Reconstrut(Y, W'[i], B'[i])                                                                                                         |
8.      | Cost = SDAE.Compare(X, Z)                                                                                                                   |
9.      | LRate = SDAE.Update(LRate, Cost)                                                                                                             |
10.     | End                                                                                                                                              |
11.     | SDAE[i].w = W[i]. SDAE[i].b = B[i].                                                                                                          |
12.     | X = Y;                                                                                                                                          |
13.     | End                                                                                                                                              |
14.     | NN = nnsetup(Train_x, n_hidden[layer], n_action); // use a feed-forward neural network for pattern recognition                                  |
15.     | For i=1 to n_layer                                                                                                                          |
16.     | NN[i].W = SDAE[i].w, NN[i].B = SDAE[i].b                                                                                                |
17.     | End                                                                                                                                              |
18.     | For k=1 to num_iter                                                                                                                          |
19.     | NN = nntrain(Train_x, Train_y);                                                                                                             |
20.     | NN.LRate = Update(NN.LRate)                                                                                                                  |
21.     | End                                                                                                                                              |
22.     | [Error, Labels] = NN.Predict(Test_x, Test_y);                                                                                                 |

**5 Experimentation and Results**

**5.1 Training and testing dataset**

The paper aims to identify the insect samples collected from the orchard field at species level, however there is a fact that to obtain large number of insect sample for certain species is difficult. Considering the highly similarity between the field samples and lab colonies, the training dataset is composed of lab colonies. The lab insects were frozen and then randomly placed on a white balance panel under the reflectance light base of a Nikon stereoscopic zoom microscope SMZ1000 (Nikon, Tokyo) with Plan Apochromat 0.5x objective. A DS-Fi1 color digital camera (Nikon, Tokyo) was mounted on the microscope. Illumination was provided by a gooseneck light guide powered by a Schott-Fostec Eke Pheostat 150W light source (Schott North America Inc., NY). The example images of the nine species sample used for training is shown in Figure 6. The sample numbers for training and testing for each species are shown in Table 3.

![Figure 6. Example images of (a) Cydia pomonella, (b) Choristoneura rosaeceana, (c) Argyrotaenia velutinana, (d) Grapholita molesta, (e) Platynota idaeusalis, (f) Spilonota ocellana, (g) Rhagoletis Pomonella, and (h) Rhagoletis Cingulata. (i) Grapholita Prunivora.](image-url)

We extracted 154 features, including 100 global features and 54 local features (given in Section 4.1) from each image to construct the feature set, and the species number for identification is 9. A 3 hidden layered IpSDAE model was constructed for moth identification. The model iterates 100 times and learns the pre-training weights,
and then adopts BP algorithm to obtain a trained network for species identification. Experimental results are shown in terms of the network structure parameter optimization, and adaptive learning rate optimization. Also, the performance comparison between our model and other methods are given.

### Table 3 Image samples used for training and testing

<table>
<thead>
<tr>
<th>Species</th>
<th>Class 1</th>
<th>Class 2</th>
<th>Class 3</th>
<th>Class 4</th>
<th>Class 5</th>
<th>Class 6</th>
<th>Class 7</th>
<th>Class 8</th>
<th>Class 9</th>
<th>Total number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td>40</td>
<td>45</td>
<td>76</td>
<td>26</td>
<td>40</td>
<td>42</td>
<td>41</td>
<td>30</td>
<td>140</td>
<td>480</td>
</tr>
<tr>
<td>Testing set</td>
<td>21</td>
<td>25</td>
<td>36</td>
<td>18</td>
<td>28</td>
<td>26</td>
<td>19</td>
<td>15</td>
<td>60</td>
<td>248</td>
</tr>
</tbody>
</table>

### 5.2 Evaluating performance of the deep learning model

#### 5.2.1 Network structure parameter optimization

Firstly, we optimize the relation between the training accuracy and different fraction of corruption and different layers. Figure 7(a) shows training performance as we increase the fraction from 0 to 0.8 and the layer from 1 to 3. As the fraction of 0 is the traditional stacked auto-encoder, we can clearly see that de-noising pertaining works better than the auto-encoder pertaining. Under a certain fraction, the more layers, the better performance. As the lower layers may need a bigger fraction, we should choose different fraction of each layer. Here we choose 0.30 for the 1st hidden layer, 0.12 for the 2nd hidden layer and 0.06 for the 3rd hidden layer. Figure 7(b) shows the much better evolutions clustering around quarter of input nodes. Finally, we choose 196 neurons in the 1st hidden layer, 64 neurons in the 2nd hidden layer, and 25 neurons in the last hidden layer.

![Figure 7. (a) The performance on different number of layers by increasing the fraction of corrupted input. (b) The first layer’s performance by increasing the number of nodes.](image)

#### 5.2.2 Adaptive learning rate optimization

In original pSDAE model, the learning rate is initialized by 0.2 in all layers. It is obvious to see that the learning rate is adaptively adjusted by the mean squared error on training set. In IpSDAE architecture, we use adaptive learning rate to reduce the mean squared error. Define that $h(x)$ is the mean squared error of the $i$’s iteration epoch of SDAE. $h’(i)$ is the difference between the $i$’s and $(i - 1)$’s mean squared error. Define $h''(i)$ is the difference between $h’(i)$ and $h’(i-1)$, which is the change rate of the mean squared error. Define $S_{LR}$ is the scaling factor and $D_{LR}$ is the fine-tuning factor. The adjustment procedure of learning rate is:

$$h'(i) = \begin{cases} h(i) - h(i - 1) & i > 1 \\ 0 & i = 0 \end{cases}$$

$$h''(i) = \begin{cases} h'(i) - h'(i - 1) & i > 2 \\ 0 & i = 0, 1 \end{cases}$$
\[ S_{LR}(i) = \begin{cases} S_{LR}(i-1) - h'(i), & h'(i) > 0 \\ S_{LR}(i-1) + h'(i), & h'(i) \leq 0 \end{cases} \]  

When \( h(i) < 0 \)

\[ D_{LR} = \begin{cases} \max(\text{ceiling}, -h''(i)) & h'(i) < 0 \\ \text{floor} & \text{otherwise} \end{cases} \]  

When \( h'(i) \geq 0 \)

\[ D_{LR} = \begin{cases} \max(-\text{ceiling}, h(i-1)) & h'(i) > -h'(i-1) \\ \min(-\text{floor}, -h''(i)) & \text{otherwise} \end{cases} \]  

Here, the current learning rate \( LR(i) = LR(i-1) \times (S_{LR}(i) + D_{LR}) \). To ensure the learning rate under a fit ratio, we select ceiling = 0.02 and floor = 0.01. Figure 8(a) shows the comparison among different fixed learning rates (0.1, 0.3, 0.5, 0.7 and 0.9) and adaptive learning rate method. The result shows that the adaptive learning rate performs better than the fixed learning rate by achieving the lower mean squared error. Figure 8(b) shows the corresponding change of learning rate.

![Mean squared error comparison among different learning rates.](image)

**Figure 8(a)** Mean squared error comparison among different learning rates. (b) The corresponding change of learning rate.

### 5.3 Performance comparison

#### 5.3.1 Results comparing to other identification methods

The performance comparison between our model and other popular identification methods are given in this part. We compare IpSDAE architecture with the classical supervised learning methods, including support vector machine (SVM), linear regression and Bayes networks (BayesNet), RBF network, as well as an ensemble learning method, random forest (REFTrees). In IpSDAE model, a SDAE structure includes a depth of three layer architecture and 50 times iterations for rebuilding the feature information, and a BP neural network for insect identification.

Table 3 shows the evaluation criteria, including precision, recall and F-measure, for identification performance of six methods. With a pose estimation step followed by segmentation step, we found that the IpSDAE model has a precision value of 98.13\% on the testing dataset, and it outperformed other five methods. Following methods are SVM with a precision value of 93.42\% and REFTrees with a precision value of 93.18\%. And the BayesNet presented the lowest precision value for the same testing data.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision (without)</th>
<th>Precision</th>
<th>Recall (without)</th>
<th>Recall</th>
<th>F measure (without)</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>93.42%</td>
<td></td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REFTrees</td>
<td>93.18%</td>
<td></td>
<td>99.8%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear Regression</td>
<td>97.5%</td>
<td>98.0%</td>
<td>96.0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BayesNet</td>
<td>90.0%</td>
<td></td>
<td>92.0%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3 Comparison results given by different identification methods (\%)
The corresponding confusion matrix of the identification result given by IpSDAE with pose estimation is shown in Figure 9. As shown in the figure, six out of nine species, which are *Cydia pomonella*, *Choristoneura rosaceana*, *Spilonota ocellana*, *Rhagoletis Pomonella*, *Rhagoletis Cingulata*, and *Grapholita Prunivora* achieved 100% correct identification rate. For these six species, there are no misidentification happened even for species from the same genus, such as *Rhagoletis Pomonella* and *Rhagoletis Cingulata*. 6% of *Argyrotaenia velutinana* samples are mis-classified to be *Platynota idaeusalis* samples because of the highly similarity of the shape and size between two species. 11% of *Grapholita molesta* samples are mis-classified to be *Cydia pomonella* samples because of the highly similarity of the texture feature between the two species. The same situation, 3% *Platynota idaeusalis* a samples are mis-classified to be *Argyrotaenia velutinana* samples because of the highly similarity of the shape and size between the two species. Also 3% *Platynota idaeusalis* a samples are mis-classified to be *Spilonota ocellana* samples because of the similarity of the texture between the two species.

![Confusion Matrix](image)

**Figure 9. The confusion matrix of IpSDAE architecture**

### 5.3.2 Pose estimation analysis

For both training and testing process, the pose estimation is applied to achieve two estimated classes for each species sample. Then, there are actually 18 class for 9 species, and at the end of the testing process, the results with different pose estimation regarding to the same species are added together. Here, it needs to point out that our ultimate objective is to identify the insect with the results of the pose estimation. The accuracy of pose estimation itself is not emphasized in this paper. Therefore, we compared the identification accuracy of different identification models with and without pose estimation process.

Identification results without pose estimation are also given in the Table 3. Here, it can be observed that in general the pose estimation process helps to improve the identification accuracy on average for all the identification methods. For example, the precision value of identification by IpSDAE is 95.24%, and it improves almost 3% after pose estimation process. However, we also found that the pose estimation step improves the result in different degree for different identification method.

<table>
<thead>
<tr>
<th>pose estimation)</th>
<th>pose estimation)</th>
<th>pose estimation)</th>
<th>measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>90.1</td>
<td>93.42</td>
<td>92.13</td>
</tr>
<tr>
<td>REFTrees</td>
<td>89.57</td>
<td>93.18</td>
<td>91.72</td>
</tr>
<tr>
<td>Liner regression</td>
<td>85.93</td>
<td>92.22</td>
<td>91.42</td>
</tr>
<tr>
<td>BayesNet</td>
<td>84.67</td>
<td>88.43</td>
<td>86.22</td>
</tr>
<tr>
<td>RBF network</td>
<td>82.28</td>
<td>87.47</td>
<td>85.87</td>
</tr>
<tr>
<td>IpSDAE</td>
<td>95.24</td>
<td>98.13</td>
<td>96.29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>measure</th>
<th>90.1</th>
<th>93.42</th>
<th>92.13</th>
<th>89.41</th>
<th>92.77</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>90.1</td>
<td>93.42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REFTrees</td>
<td>89.57</td>
<td>93.18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Liner regression</td>
<td>85.93</td>
<td>92.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BayesNet</td>
<td>84.67</td>
<td>88.43</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RBF network</td>
<td>82.28</td>
<td>87.47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IpSDAE</td>
<td>95.24</td>
<td>98.13</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6 Conclusions

In this paper, a pose estimation-dependent automated identification method based on a deep learning architecture was proposed to identify nine moth species collected from orchard field. The method consists of four stages: (1) morphological-based segmentation, (2) insect pose estimation, (3) feature extraction, and (4) identification process. The segmentation method is designed especially for field moth image can solve the problems of impurities, scales, and information confusion caused by uneven lighting and reflectance of background on field image. The SSIM-base pose estimation method is proposed to estimate the top view pose or side view pose of the insect, and helps to divide each species into two types regarding to the specific pose. An IpSDAE method based on a deep learning architecture was developed for moth identification.

Features of 154 extracted from each sub-image were transmitted to the IpSDAE for discriminate analysis. Images of 248 insect samples collected from field orchard were tested for identification preformation, and image of 480 insect samples of lab colonies were used for training. An average precision value of 98.13% was achieved by the proposed method. The identification results show our method outperformed other popular identification methods regarding to same testing data. Also, the performance comparison is also given regarding to the use of pose estimation step for different methods. We found a different degree of performance improvement for different identification methods, and indicated that the pose estimation-dependent automated moth identification works better than without pose estimation process.

Acknowledgements

Authors would like to thank the support from the Department of Entomology, Michigan State University, and the USDA-ARS Postharvest Lab, East Lansing, MI. Especially thank Dr. Daniel E. Guyer from Michigan State University. The work is partially supported by Research Fund for the Doctoral Program of Higher Education (20130121120048).

References


