International Journal on Artificial Intelligence Tools
Vol. 20, No. 3 (2011) 511–530

DOI: 10.1142/S0218213011000255

World Scientific
www.worldscientific.com

ASSESSING INSECT GROWTH USING IMAGE ANALYSIS

TOM PATTEN
Department of Computer Science and Engineering, University of Nevada, Reno, USA
patten@cse.unr.edu.edu

WENJING LI
STI Medical Systems, Honolulu, Hawaii, USA
wli@sti-hawaii.com

GEORGE BEBIS
Department of Computer Science and Engineering, University of Nevada, Reno, USA
and
Computer Science Department, King Saud University, Riyadh, Saudi Arabia
bebis@cse.unr.edu

MUHAMMAD HUSSAIN
Computer Science Department, King Saud University, Riyadh, Saudi Arabia
mhussain@ccis.edu.sa

Received 12 July 2010
Accepted 17 December 2010

Image analysis represents an invaluable tool in processing and analyzing biological data in an expeditious and reliable way. This paper describes the design and implementation of an image analysis system for the automatic assessment of the growth of Heliothis-zea insects from color images. Specifically, the Heliothis zea is a corn earworm eating corn crops. Biotech researchers are interested in developing insecticidal bio-toxins with the best performance to kill or stunt the growth of the pest. Currently, assessing the effectiveness of different bio-toxin solutions is done mostly manually by biotech experts. The goal of this study is to investigate the use of image analysis for automating and improving the efficiency of this process. In this context, we have developed a prototype system for assessing insect growth from color images which contains three main stages: (1) insect segmentation from background, (2) region processing and feature extraction, and (3) categorization of insect growth. A probabilistic model based on mixtures of Gaussians has been adopted to segment the insect images. Also, back-propagation neural networks have been trained to classify the instar stage and life stage. The proposed system has been evaluated on a set of real images.

Keywords: Image segmentation; pattern classification; mixtures of Gaussians; neural networks.

1. Introduction

Processing and analyzing biological data in an expeditious and reliable way is critical in almost all practical biological applications. The relative paucity of automated techniques for reliable and rapid analysis of biological data has proved to be the rate-limiting step or bottleneck in most high-throughput experiments. Most of the experimental data in biological experiments can be acquired and represented in the form of images. Thus, automated analysis of such data entails the design and implementation of appropriate image analysis and computer vision algorithms. In this paper, we present the design and implementation of an image analysis system for the automated classification and analysis of insect images.

1.1. *Heliothis zea images*

The Heliothis zea, also called the corn earworm, is an insect which is a pest to corn crops. Figure 1 shows some pictures of Heliothis zea larvae under microscope. Biotech companies develop insecticidal proteins used to fight pests. Usually, they inject experimental proteins into plants. As the plant grows, the protein becomes part of the plant's tissue. Insects which feed off the plant get a dose of the protein. The protein causes the infected insects to die, or stunts their growth, preventing them from reproducing. The biotech insecticidal protein reduces the Heliothis zea population, which in turn reduces the damage to the corn crops.

A successful biotech insecticidal protein should meet certain criteria. It reduces damage to crops. It must be non-toxic to mammals and birds, because this same protein will be present in the food which comes to the market place. A successful biotech product has some attractive benefits. Crops which use biotech products will not need to be treated with as much chemical insecticide. Chemical insecticides

(a) (b)

Fig. 1. (color online) Heliothis zea larvae and adults (a) larvae (b) adults.

have a negative side effect called fungal toxins, so using less insecticide results in less damage from fungal toxins. Crops raised using biotech products will benefit in these ways.

1.2. *Bio-toxin analysis — The scoring of assay plate*

To investigate the performance of the insecticidal protein, the researchers in the biotech company need to test multiple levels of protein with different concentration. Usually they perform the testing in assay plates. One assay plate contains a large number of tiny wells. Each well is approximately two centimeters in diameter. Inside each well is a mixture of wheat germ mixed with concentrated biotech insecticidal protein. These two ingredients are mixed together thoroughly. A set of Heliothis zea insects is then placed into each well. The larvae are delivered on a type of silk, similar to spider silk, which is hard for the lab technician to handle, so they cannot control the quantity of insects per well, but it is usually between one and ten insects. The experiment is run by letting the insects eat the food with the protein and grow over a four-day period. At the end of the experiment, the growth rate of the insects is measured for each well. When they have quantified the growth rate of a sufficient number of sample wells, they can quantify the rating of the biotech insecticidal protein at the concentration which was tested. Figure 2 shows a 96-well assay plate after a four-day growth period.

Entomologists classify insect growth activity into stages. At infancy a Heliothis zea insect is in its first larval instar stage. About twenty-four hours later the Heliothis zea will shed its skin, and enter a second larval instar stage. At the second larval instar stage the insect body size and the size of its spots are distinctly larger than they were in the first larval instar stage. After yet another twenty-four hours

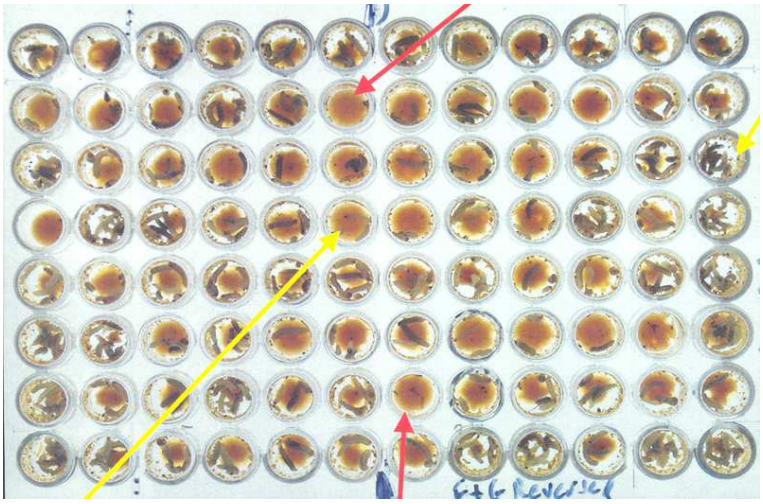


Fig. 2. (color online) An assay plate with 96 wells.

the Heliothis zea insect will again shed its skin and enter the third larval instar stage. Heliothis zea insects in the third larval instar stage are considerably large compared to the second instar stage, and as before their spot sizes are also distinctly larger. At the end of the four day experiment the Heliothis zea would normally have reached an instar stage no greater than the third instar stage. The larval instar stage classification is the most important measure which is used to quantify growth rate of insects in the experiments. After a four-day period experiment, an insect which undergoes normal growth (and is not affected by the biotech insecticidal proteins) will have grown to the third instar stage. When the well contains a high degree of biotech insecticidal protein, the insects generally die in the first instar stage. Also the insects may get stunted, or die in the second instar stage, depending on the concentration of the protein.

By monitoring the growth activities of the insects in the assay plate, including number of insects in each well, the instar stages of each insect, the life information (whether the insect is dead or alive), we can score each well in the assay plate according to the concentration levels of the protein in that well. Based on these scores, the standard industry measurement, called EC50, can be computed. EC50, or Effective Concentration 50%, is computed for each biotech insecticidal protein. It is a standard form of measurement of biotech insecticidal protein potency.

To compute EC50, we need to compute the scores of the assay plate first. Now a camera mounted in a robot arm is used to take digital photos of each well. Then the scoring of the assay plate is based on interpreting these digital images.

1.3. *Why using image analysis*

Currently, the scoring of the Heliothis zea images is done by the experienced biotech experts manually. They watch each image of the well and use their vision to perform analysis. There are usually 5000 images to be analyzed weekly. Therefore, it is imperative to automate this scoring process of the insect images to reduce the workload of the biotech researchers. The objective of our work is to design and implement an auto scoring system for the Heliothis zea images. That is, given the insect images taken for each well, our system can automatically classify the insects' life and instar stages, and further score each insect image.

The proposed system consists of three stages: (i) insect segmentation from background, (ii) counting the number of insect regions, and (iii) instar and life classification of the insects. A probabilistic model based on mixtures of Gaussians has been adopted to segment the insect images. Also, back-propagation neural networks have been trained to classify the instar stage and life stage. The proposed system has been evaluated on a set of real images.

The paper is organized as follows. In Section 2, a literature review of related work is presented. Section 3 is devoted to the problem of insect image segmentation. Insect counting and instar life classification are presented in Sections 4 and 5 respectively. Experimental results are shown in Section 6. Finally, our discussions and conclusions are given in Section 7.

2. Review of Previous Work

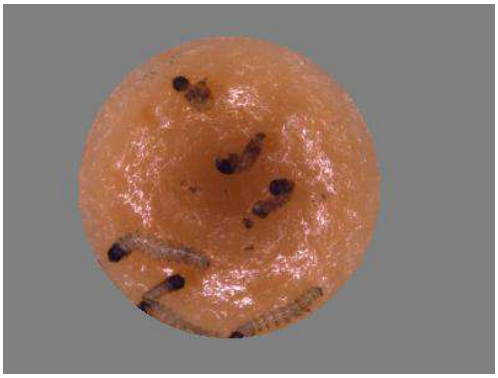
There have been many efforts to automate the analysis of image data arising from biological or medical experiments.[1–6] Research work includes using neural network[3,5,7–10] or probabilistic models.[6,11–13]

Ghosh and Chinnaiyan[12] have proposed a statistical model based on a mixture of Gaussian distributions in the context of hierarchical clustering of gene expression data from DNA microarray experiments. Jain *et al.*[13] describe a program called UCSF Spot for fully automatic quantification of DNA microarrays. The system automatically locates both, subarray grids and individual spots while requiring no user identification of any image coordinates. Manduchi *et al.*[11] describe a protocol by which discrete values are used to provide an easily interpretable description of differential expression. Novel statistical methods are proposed to attach confidence levels to the hypothesis that changes in expression levels represents true changes. In Ref. 14, linear discriminant analysis was used to classify honey samples.

In Ref. 6, the design and implementation of a computer vision system called DNAScan has been presented for the automated analysis of DNA hybridization images. A recursive segmentation procedure has been designed and implemented to extract spot-like features in hybridization images in the presence of a highly inhomogeneous background. Positive hybridization signals were extracted from the spot-like features using grouping and decomposition algorithms based on the computational geometry. A mathematical model for the positive hybridization patterns and a Bayesian pattern classifier based on the shape-based moments were proposed and implemented to distinguish between the clone-probe hybridization signals. In Ref. 15, a Bayesian network was used to discover the causal interactions among the genes and proteins in noisy expression data.

Lerner[10] has applied neural networks to automatic analysis of chromosome images, in all aspects of the analysis including segmentation, feature description, selection and extrication, and classification. He used a new approach called a classification-driven partially occluded object segmentation (CPOOS) to separate clusters of touching chromosomes. A multi-layer perceptron (MLP) was used to score and verify hypotheses. Such NN based system has classified 5500 chromosomes with a success rate of 83.6%. In Ref. 3, Cheng *et al.* used a competitive Hopfield neural network for segmenting grey scale medical images. It is a kind of Hopfield network incorporates the winner-takes-all (WTA) learning mechanism. The image segmentation is conceptually formulated as a problem of pixel clustering based upon the global information of the gray level distribution. Patel *et al.*[5] used a neural network for automating the process of grading eggs. They reported an accuracy of 92.8% for blood spots, 85% for dirt and stain detection, and 87.8% for crack detection. The features used by Patel et al. were formed by computing a histogram of the RGB color space of each image and using the counts from each histogram bin as a feature. They used 768 ($256 * 3$) histogram bins corresponding to 768 nodes in the input layer of the neural network.

(a)



(b)

Fig. 3. (color online) (a) A typical Heliothis zea image and (b) Image after Hough transform.

3. Insect Image Segmentation

Figure 3(a) shows a typical Heliothis zea insect image. It is $600 * 800$ color image captured by a digital camera mounted on a robot arm. All the images have one thing in common which is the circular shaped well in which the experiment is taking place. The insects are located inside the well with the colored background which is the biotech insecticidal protein. Our first goal is to segment all the insects from the background such that we can count the number of insects in each well. Here we propose to use the segmentation framework with two phases: Training and segmentation. The flow chart can be seen in Fig. 4.

3.1. *Hough transform*

Since the area of interest of the insect image is the area inside the circular well, so the first step of preprocessing to detect the rim of the well and remove the well from the image. The commonly used curve detection methods are Hough transform and its variants.[16,17] Here the Hough Transform for circle detection is used to extract the

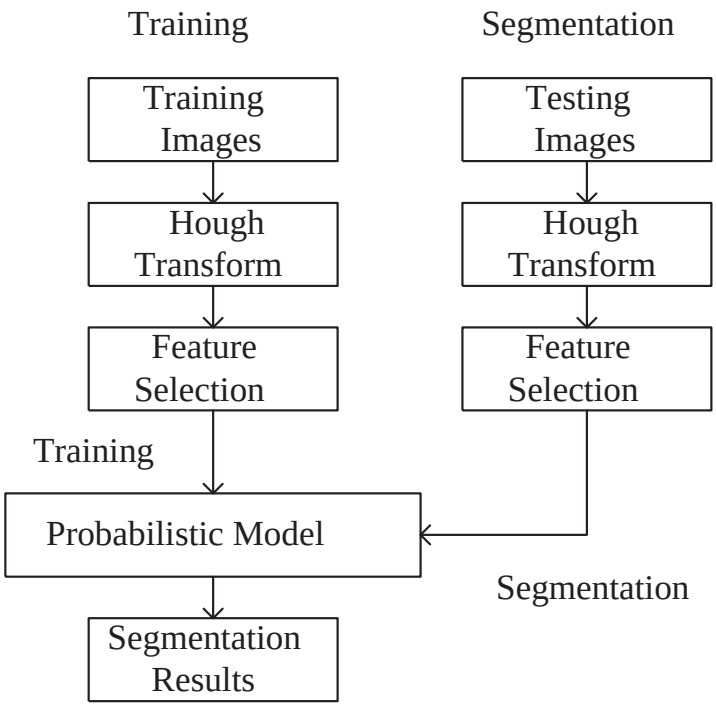


Fig. 4. Flow-chart of the segmentation stage.

circles underlying the rim of the well. A circle of the form $(x-a)^2+(y-b)^2=r^2$ is represented by the 3D Hough accumulator denoted by (a,b,r). At the end of voting procedure, the cells in the (a,b,r) accumulator with a value greater than a certain threshold represent circles in image space. Once the well is detected, we remove it from the image. Figure 3(b) shows the results of the Hough transform applied to the image in (a). Because the camera may not exactly perpendicular to the assay plate during shooting, in reality we sometimes do not get perfect results, but we always get pretty good estimates.

3.2. *Feature selection*

Feature selection is always a crucial problem in pattern classification. The basic rule in feature selection is to choose features that would result in large *between-class distance and small* within-class variance in the feature vector space. To judge if a pixel in the image belongs to the insect or not, we need to use multiple criteria for each pixel. Our feature selection method is based on a small window or sub-image centered on that pixel. In this way, we are able to compute many features based on this window. After careful investigation and test through experiments, we came up with the following five features for each image window:

- Average hue (hue is a numerical representation of a color, e.g. yellow = 1.57)
- Average saturation (grey has saturation 0.0, and vivid colors like red, yellow, blue have 1.0)
- Grey-scale percentage (measured by saturation and a threshold)
- Intensity variance
- Count of spots (if the windows contains something that looks like a spot)

Here we convert the image from RGB color space to HSV space[18,19] to compute the average hue value and the average saturation value.

3.3. *Training the probabilistic model*

Mixture models are a type of density model which comprises a number of component functions, usually Gaussian. These component functions are combined to provide a multi-modal density. In the past, they have been applied to learn the distribution of the object class in probabilistic visual learning for object representation.[20] They have also been employed to model the color distribution of objects for real-time segmentation and tracking.[18] The task can be made more robust by generating a mixture model corresponding to background colors in addition to foreground model and employing Bayes' rule to perform pixel classification.

Let the conditional density for the sample data ξ belonging to a multi-colored object O be a mixture with M component densities:

$$p(\xi|O) = \sum_{j=1}^{M} p(\xi|j)\pi(j) \tag{1}$$

where a mixing parameter $\pi(j)$ corresponds to the prior probability that data ξ was generated by component j and where $\sum_{j=1}^{M} \pi(j) = 1$. Each mixture component is a Gaussian with mean μ and covariance matrix Σ, i.e. in the case of a N dimensional space:

$$p(\xi|j) = \frac{1}{(2\pi)^{\frac{N}{2}}|\Sigma_j|^{\frac{1}{2}}} e^{-\frac{1}{2}(\xi-\mu_j)^T\Sigma_j^{-1}(\xi-\mu_j)}. \tag{2}$$

Expectation-Maximization (EM) algorithm provides an effective maximum-likelihood algorithm for fitting such a mixture model to a set of training data.[21] The EM algorithm is iterative with the mixture parameters being updated. It monotonically increases the likelihood in each iteration, converging to a local maximum. Mixture models provide greater flexibility and precision in modeling the underlying statistics of sample data. Once a model is generated, posterior probabilities can be computed according to the Bayes' rule. Given density estimates for both an object O, and the background S, the probability that a sample data, ξ belongs to the object is given by the posterior probability $P(O|xi)$:

$$P(O|\xi) = \frac{p(\xi|O)P(O)}{p(\xi|O)P(O) + p(\xi|S)P(S)}. \tag{3}$$

4. Insect Counting

Based on the segmented results of the insect images, in this stage, we attempt to count the number of insects present in each image. A couple of image processing techniques have been applied to refine the segmented regions to achieve this goal.

4.1. *Morphological operations*

Goutsias *et al.*[22] proposed morphological operations as a method of merging pixels into sets of pixels which share the same properties based on their densities and proximities. Upon completion of segmentation procedure, we have the set of original images, plus we also have a set of images which were output by the classifier to be used as maps. Figure 5(a) is an example of a segmented image map, which is considered as one of inputs to this stage. We use two morphological operations,

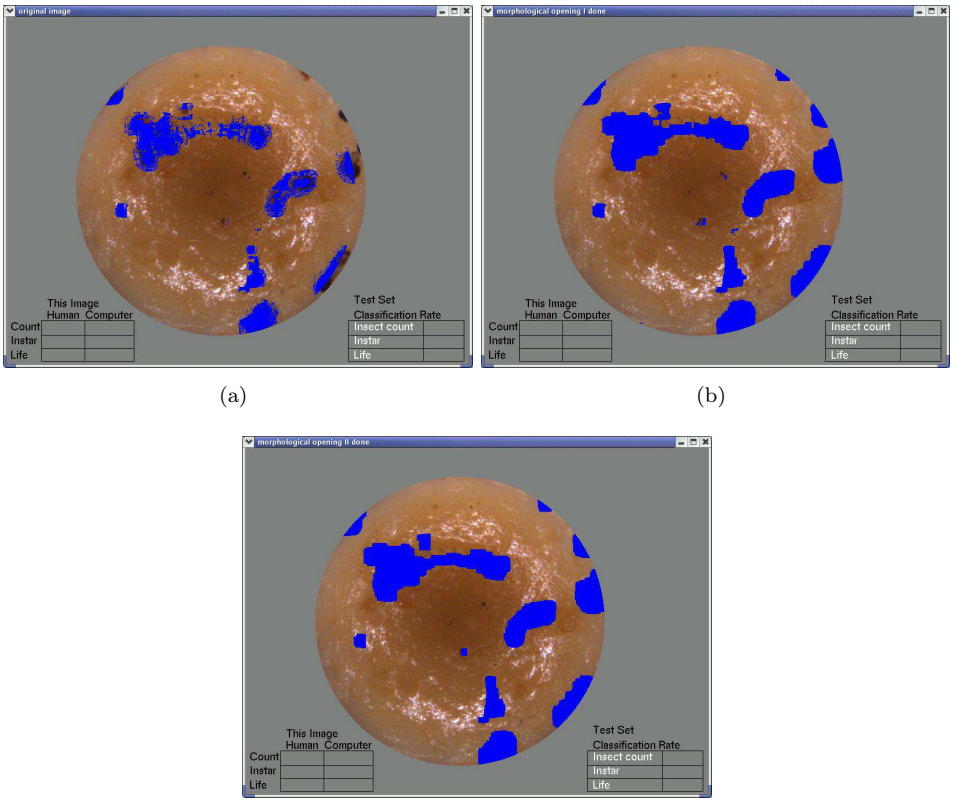


Fig. 5. (color online) An example of morphological operations. (a) Segmented image map, (b) Morphological opening and (c) Morphological closing.

"morphological opening" and "morphological closing". Each of these operations consists of processes: dilation and erosion.

The first process is morphological opening. The goal of morphological operations is to convert areas of the image which consists of dense points of foreground into solid regions each of which represent the insects.

The process of dilation traverse the mapped image pixel-wise. Each pixel not already marked blue is checked to see if there is any other pixel which is marked blue within a distance of nine pixels. If there is, then we change the pixel to blue. We process the entire image of approximately 50,000 pixels in this way. When we have completed dilation on the entire image, we have the benefit that the dense regions have become solid, but it comes at a cost: the solid regions of blue pixels which represent mappings of insect areas are over-sized, and need to be trimmed. The simple solution is erosion. Erosion visits each pixel marked blue, and checks to see if there is any pixel within a distance of nine pixels not marked blue. If so, then mark this pixel as background, instead of blue. This is done by consulting the original image to return the pixel to its original color. These two steps, dilation followed by erosion is called morphological opening. It has the useful result of converting dense sections of blue pixels into solid sections of pure blue. Figure 5(b) shows the resulting output.

After morphological opening is complete, the next step is the reverse of morphological opening. It is morphological closing. It is the same as morphological opening, except the erosion and dilation steps are reversed. The purpose of morphological closing is that the erosion step will eliminate tiny false positive regions which are too small to be considered insect. Figure 5(c) shows the output of this step. Notice that a few tiny sections were removed.

After the above morphological operations, we can build our lists of regions in each image. Smaller regions which include less than 1,000 pixels are filtered out. These regions are considered as the noise or peeled skin of the insects.

4.2. *Snake algorithm*

The snake algorithm[23] is also called the deformable contour algorithm. The idea of the snake algorithm is to start with a chain of boundary pixels of a region, and repeatedly adjust the positions of the points along the boundary for the purpose of finding the best possible boundary of the region. The output of the snake algorithm is an improved boundary of the insect region.

The snake algorithm processes the boundary by applying an energy function as it adjusts the boundary points. The energy function is computed on each boundary point, and it continues to process all points over and over again until the boundary stops moving. The energy function tries to simultaneously enforce three components: (1) continuity energy, (2) smoothness energy, and (3) image energy. The three components of the energy function are shown in Equations (4)–(6).

$$E_{\text{continuity}} = (\bar{d} - \|p_i - p_{i-1}\|)^2 , \tag{4}$$

$$E_{\text{image}} = -\|\nabla I\,, \tag{5}$$

$$E_{\text{smoothness}} = \|p_{i-1} - 2p_i + p_{i+1}\|^2\,, \tag{6}$$

where $\bar{d}$ is the average distance between any two consecutive points. p_i represents the point whose energy is being computed. p_{i-1} and p_{i+1} represent the immediate neighbors on the boundary, and ∇I represents the gradient of the brightness between the boundary point and its neighboring pixels (not other boundary points but the immediate pixels next to it inside the region). The continuity energy component has the purpose of making the boundary continuous, and its effect is to prevent sets of points from getting bunched together. The effect of the smoothness energy component forces the contour to be smooth, and avoid a lot of jagged edges. The image energy component is the one component which pushes the boundary towards the contour of the object. The gradient or derivative of the brightness is how it finds the image boundary. Its effect is to be attracted to areas of large change in image brightness. The resulting boundary after applying this energy function is a more precise boundary of the insect regions.

Figure 6 shows the improved boundaries of the insect regions. The green lines are the region boundaries before running the snake algorithm. The blue lines are the boundaries output by the snake algorithm. The snake algorithm seems to perform well on insect regions which have high contrast with the background, but in case of the large insect the snake does not perform very well in the center, because the insect brightness at that point has a low contrast with its background.

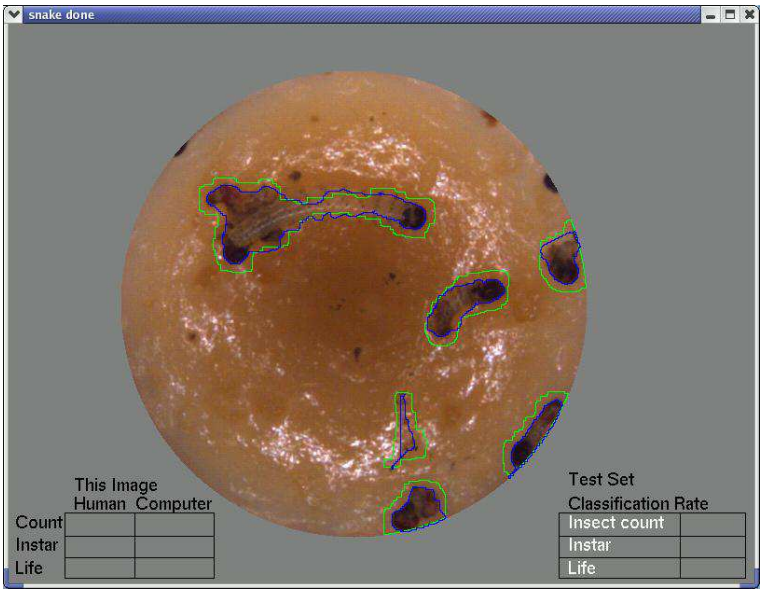


Fig. 6. (color online) An improve boundary by the snake algorithm.

5. Instar and Life Classification

5.1. *Insect spot detection*

For instar classification, the size of spots is considered to be the key features. So we need to detect the spots of the insects first. For spot detection, we start by visiting each pixel in the region. We use the BFS (Breadth-First Search) graph searching algorithm to grow sub-regions, such that each sub-region contains sets of pixels with similar intensity, or brightness. We continue in this way until each region pixel is assigned to one sub-region. This will partition the set of region pixels into pair-wise disjoint subsets, each subset being a sub-region of an image region. Figure 7 shows the sub-regions created for an insect region. The roles that the colors play is to allow the viewer visually distinguish the sub-regions from each other.

From the sub-regions, we next try to find candidate sub-regions which might contain insect spots. We do this by visiting each sub-region, then visiting its surrounding field of pixels, and evaluate the degree of contrast between the region and its field. When we find sub-regions which are candidate spots, we append all of the sub-region pixels onto a large list of pixels. We use a queue data structure for this list. A tricky problem we will encounter occurs when two spots get too close to each other. We need an algorithm which will identify each spot separately. That is the reason for the queue. From here forward we no longer use the sub-regions for anything. Instead we will define spot regions out of the candidate pixels in the queue.

Think of the spots in the image as a contour map like in Fig. 8. We can find the spots by their darkness, or intensity values. Each pixel has an elevation cor-

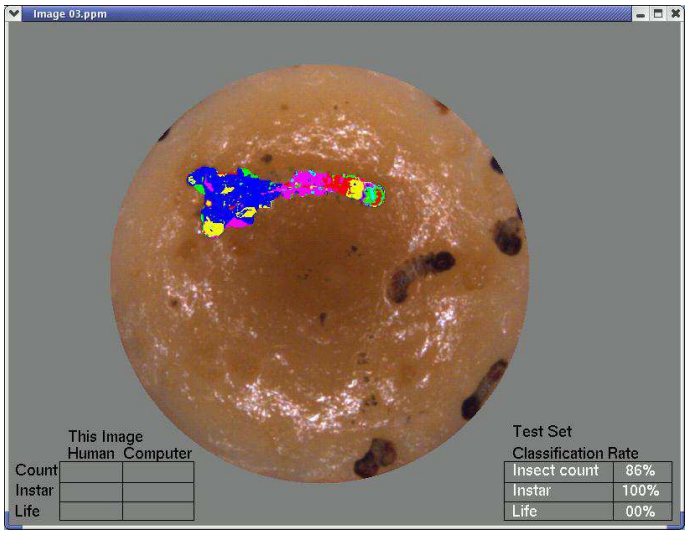


Fig. 7. (color online) Sub-regions extracted for spot detection.

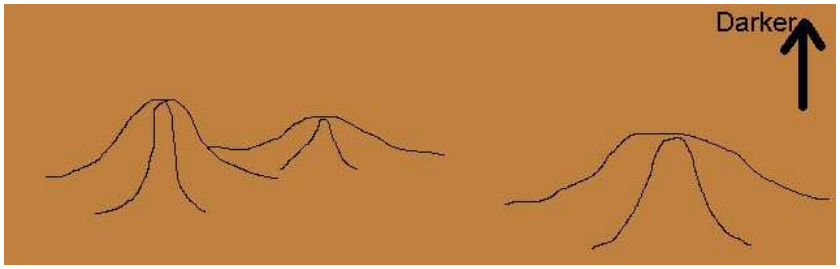
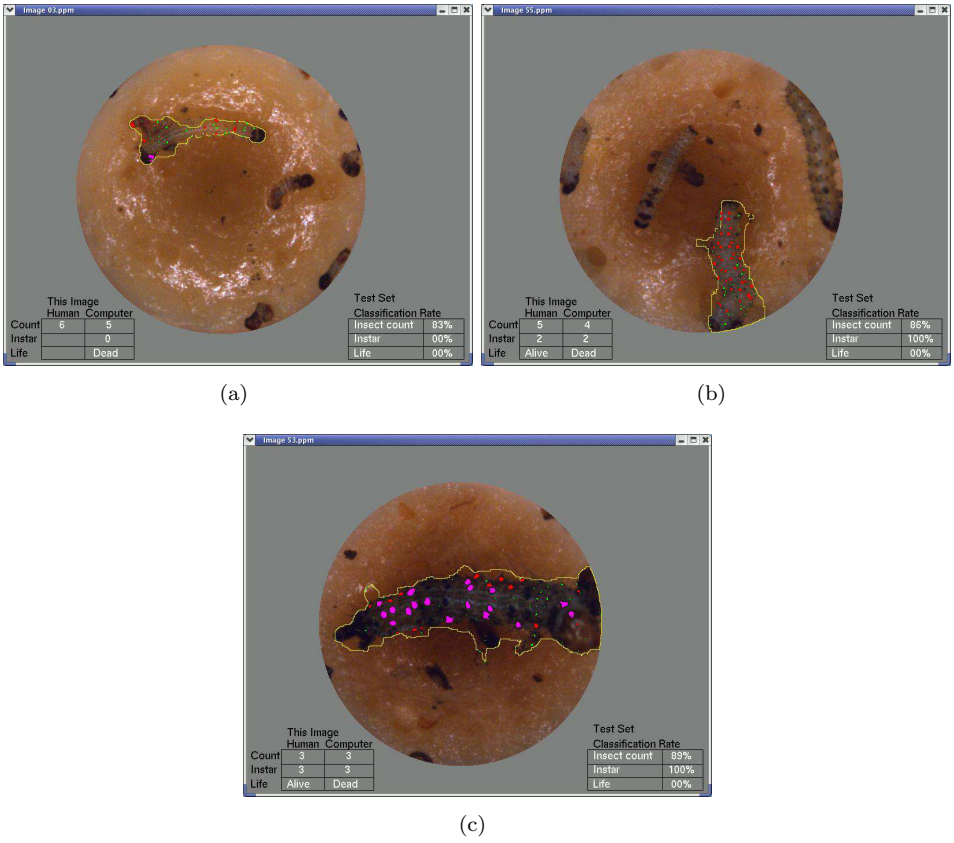


Fig. 8. (color online) Spots represented as a contour map.

responding to its darkness. The dark pixels on the spots form hills. We can find the spot regions by finding these hills. At the present point in processing, we have a list which holds most of the pixels which comprise the hills. We take a pixel off the queue and pass it as a seed to Breadth First Search, or BFS grid-searching algorithm. BFS grows an area of contiguous pixels, and returns the result as a list of pixels. We traverse the list to find its darkest pixel. This pixel is the "top of a hill". We pass this "top of the hill" pixel again as a seed to BFS, and this time the BFS procedure enforces the constraint that as grows the spot, it is not allowed to go uphill. This is exactly how we separately identify two spots close to each other as two distinct spots. The resulting list returned by BFS is now a candidate spot, and we put it on a master list of spot regions. We also mark these pixels on our mapped image as "spot", We return to our original list of spot candidate pixels, and retrieve the next pixel. Pixels on this queue already marked on our mapped image as spot are disregarded, and we move on to the next pixel on the list. When we find a pixel not yet marked as a spot pixel, we repeat the procedure described above. When we have processed the entire list of candidate spot pixels in this way, we will have a list of spots on an insect region.

We can validate spots for roundness. First, we find the size of the min-max box around the spot. Then we compute the relation between the count of spot pixels, and the count of pixels in the min-max box. A perfectly round circle will have the relation $circle/square = \pi/4$. Of course, we don't look for perfectly round spots. We use a roundness threshold, which has the effect of filtering out regions which are not round enough to be spots. This allows us to consider only round spots for classification purposes.

The most important goal of the classifier is to identify the larval instar stage of the insects. We have noticed that even though the insects grow rapidly, the size of their spots seem to remain consistent within each instar stage. One possible explanation of this may come from the fact that they shed their skin each time they change instar stages. We can use the pixel count of the spots as a measure of spot size. We use thresholds for size classifications which distinguish between the classes. We have defined three spot size classifications: (small, medium, large). This size gives us a good feature for correctly classifying the instar stage of the insects.



(a) (b)

(c)

Fig. 9. (color online) Spot detection (a) first instar, (b) second instar and (c) third instar.

Figure 9 shows some examples of the spot detection. In the figure, the green colored spots are classified as small. The red spots are classified as medium. The magenta colored spots are classified as large.

5.2. *Features used for insect classification*

With the master list of spots completed, we are ready to extract features. We define one feature vector for each region. For each region, we look for the most useful features to extract, and compute the region features. These are the features which we chose:

- count of small spots
- count of medium spots
- count of large spots
- ratio of spot pixels to region pixels
- skin brightness near the spots

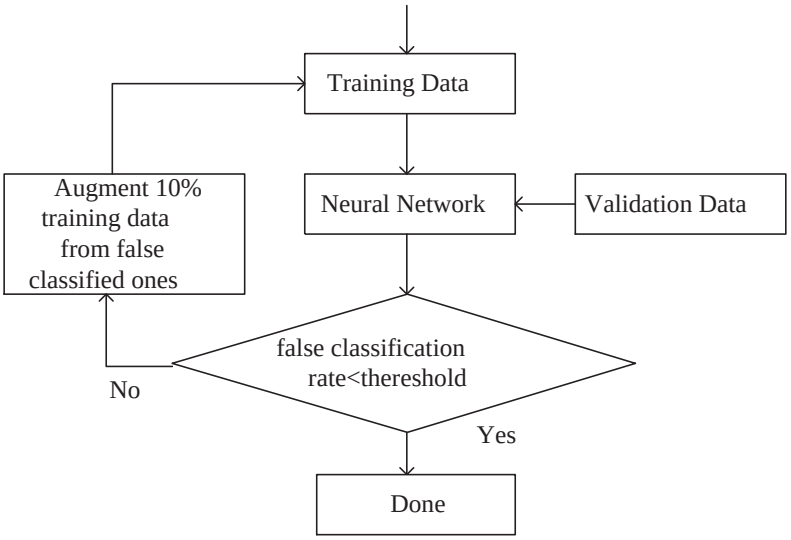


Fig. 10. Neural network training with bootstrapping.

The first three features are useful for instar classification, and the last two features are intended to be useful for life classification, but we use all five features for performing each classification task. As in the segmentation classification task, our feature vectors forms a five dimensional feature space, and data points in this feature space will form clusters unique to each class of data.

5.3. *Training neural network model*

Two back-propagation neural networks are trained to perform the insect classification and life classification separately. Both of them are 3-layer networks with 5 nodes in the input layer and 10 nodes in the hidden layer. Bootstrapping strategy is also applied to improve the training performance.

First, we divide the image set into 3 sub-sets, training set, validation set and test set. We use the data in the training set to train the neural network. Once the training is done, the false classification rate of the data in the validation set is computed. If the rate is larger than some predefined threshold, we augment the training set with the false classified data from the validation set and repeat the training, until the false classification rate is below the threshold. This technique is useful for enhancing the training process to learn from its mistakes without using the test data set.

6. Experimental Results

A total of 88 images were used in our experiments. We use five-fold cross validation method to measure the accuracy statistically. In detail, we randomly divide the

whole set of images into three subsets. Subset 1 is called the training data set consisting of 50% of the images, which are used to train both the neural network model and the probabilistic model. Subset 2 is the validation set consisting of 25% of the images, which are used in the bootstrapping to augment the training set. Subset 3 is the test set consisting of the last 25% of the images, which are used to test the performance of the classifiers. We do the partition of the three sub-sets five times, and then to compute the average performance of the classifiers to make the results statistically correct.

6.1. *Results for image segmentation*

For insect image segmentation, first we get the ground truth images for each insect image by hand using a GUI. The set of the ground truth images are used to measure the accuracy of segmentation methods. We measure the accuracy of the segmentation based of the following:

$$\text{accuracy} = \frac{2 * Seg_i}{(Seg_n + Seg_h)} \tag{7}$$

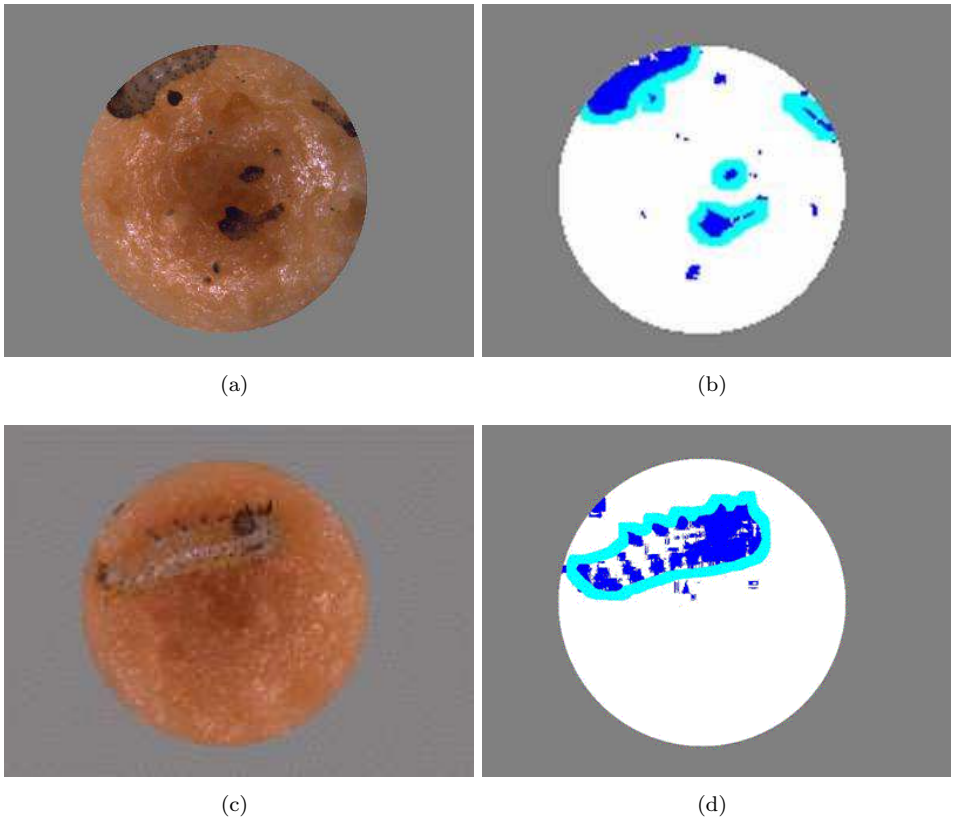
where Seg_h is the number of insect pixels per ground truth image. And Seg_n is the number of insect pixels segmented by the probabilistic classifier. Seg_i is the intersection of Seg_h and Seg_n. In our experiments, we have modelled background with one Gaussian and foreground (the insects) with three Gaussians for each partition. Table 1 summarizes our segmentation results. Columns "S1" to "S5" indicate the five-time partition of the images. The accuracy of the classifiers on the test set of each partition is shown in each column. The last column shows the average accuracy for the different classifiers. As it can be observed, the probabilistic model has the better performance than the neural network for the insect image segmentation. Figure 11(b)and (d) shows the segmented results of the images in Fig. 11(a) and (c).

Table 1. Segmentation results.

Methods	S1	S2	S3	S4	S5	Average
NN	90%	58%	90%	91%	87%	83%
PM	89%	90%	89%	90%	86%	89%

6.2. *Results for instar and life classification*

Table 2 summarizes the results of the neural network classifiers in the case of instar and life classification. The second column indicates the accuracy for the insect counting. We got 84% of accuracy on average. There are a few reasons which make the accuracy of insect counting not very high. First, the insect images have so much noise included, such as the skin, and the head cap peeled off from the



(a) (b)

(c) (d)

Fig. 11. (color online) Segmentation results, (a), (c) Original Image after Hough transform (b), (d) Segmented results.

Table 2. Classification results.

Partition	Insect Count	Instar	Life
S1	81%	97%	67%
S2	89%	95%	69%
S3	83%	93%	56%
S4	85%	96%	67%
S5	83%	95%	71%
Average	84%	95.2%	66%

insects. Second, insects may overlap with each other. This is the most important factor affecting the accuracy of the insect counting. Currently, the system does not consider the decomposition of the overlapped insect regions. This problem will be addressed in the future work.

The third column indicates the accuracy for the instar classification. We got 95.2% accuracy on average, which is quite acceptable. The last column is the accuracy for the life classification. We did not get good accuracy for the life classification based on the following reasons. First, life classification is difficult. The insects assay plate was frozen before taking pictures (i.e., otherwise the insects may climb out of the well during the shooting). The frozen process may change the skin color of the insects. It is worth mentioning that it is hard even for humans to make a correct decision when the insects are dying. Second, the features we used to train the classifier may not be optimal. However, compared to instar classification, life classification plays a less important role in the final scoring system to get the EC50.

7. Conclusions

We have presented a system for the analysis of the Heliothis zea insect images. The analysis has been shown mainly consists of three stages: insect segmentation from background, counting the number of insect regions, and the instar and life classification of the insects. Two different models were used in the segmentation and classification. One is the back-propagation neural network model, the other is the probabilistic model based on mixture of Gaussians. We used the probabilistic model for segmentation and neural network model for classification.

The effectiveness of our system was demonstrated using five-fold cross validation. Our system was designed and implemented for the Heliothis zea insect images. However, it could be extended to handle other types of insect images easily, such as Spodoptera exigua (Beet armyworm) and Trichoplusia ni (Cabbage looper) images. Future work includes the decomposition of the overlapped insect regions (e.g., using tensor voting[24]), the optimization of feature selection, more powerful density estimation models,[25] and classifiers such as Support Vector Machines.[26]

Acknowledgment

The authors would like to thank Verdia Corporation for providing the insect image data set and the valuable discussions.

References

1. F. Meyer, Automatic screening of cytological specimens, *Computer Vision, Graphics and Image Processing* **35** (1986) 356–369.
2. A. Carothers and J. Piper, Computer-aided classification of human chromosomes: A review, *Stat. Comput.* **4** (1994) 161–171.
3. K.-S. Cheng, J.-S. Lin, and C.-W. Mao, The application of competitive hopfield neural network to medical image segmentation, *IEEE Transactions on Medical Imaging* **15**(4) (1996) 560–566.
4. G. Agam and I. Dinstein, Geometric separation of partially overlapping nonrigid objects applied to automatic chromosome classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19** (1997) 1212–1222.

5. V. Patel, R. McClendon, and J. Goodrum, Color computer vision and artificial neural networks for the detection of defects in poultry eggs, *Artificial Intelligence Review* **12** (1998) 163–176.
6. S. M. Bhandarkar, T. Jiang, N. Li, and K. Verma, Automated analysis of DNA hybridization images for high-throughput genomics, *Machine Vision and Applications* **15**(3) (2004) 121–138.
7. J. Graham, P. Errington, and A. Jennings, A neural network chromosome classifier, *Journal of Radiat. Res.* **33** (1992) 250–257.
8. W. S. Sweeney, Jr., M. Musavi, and J. Guidi, Classification of chromosomes using a probabilistic neural network, *Cytometry* **16** (1994) 17–24.
9. B. Lerner, H. Guterman, I. Dinstein, and Y. Romen, Human chromosome classification using multilayer perceptron neural network, *International Journal of Neural Systems* **6** (1995) 359–370.
10. B. Lerner, Toward a completely automatic neural-network-based human chromosome analysis, IEEE Transactions on Systems, *Man and Cybernetics* **28**(4) (1998) 544–552.
11. E. Manduchi, G. Grant, S. McKenzie, G. Overton, S. Surrey, and C. Stoeckert, Generations of pattern from gene expression data by assigning confidence to differentially expressed genes, *Bioinformatics* **16**(8) (2000) 685–698.
12. D. Ghosh and A. Chinnaiyan, Mixture modeling of gene expression data from microarray experiments, *Bioinformatics* **18** (2002) 275–286.
13. A. Jain, T. Tokuyasu, A. Snijders, R. Segraves, D. Albertson, and D. Pinkel, Fully automatic quantification of microarray image data, *Genome Research* **12** (2002) 325–332.
14. L. A. Dias *et al.*, An electronic tongue for honey classification, *Microchimica Acta* **163**(1-2) (2008) 97–102.
15. G. Yap, A. Tan, and H. Pang, Learning causal models for noisy biological data mining: An application to ovarian cancer detection, *22nd National Conference on Artificial Intelligence* **1** (2007) 354–359.
16. R. Duda and P. Hart, Use of hough transformation to detect lines and curves in pictures, *Commun. ACM* **15**(1) (1972) 11–15.
17. J. Illlingworth and J. Kittler, A survey of the hough transform, *Computer Vision, Graphics and Image Processing* **43** (1988) 221–238.
18. S. J. McKenna, Y. Raja, and S. Gong, Tracking color objects using adaptive mixture models, *Image and Vision Computing* **17** (1999) 225–231.
19. S. Sura, G. Qian, and S. Pramanik, Segmentation and histogram generation using the hsv color space for image retrieval, *Proceedings of IEEE International Conference on Image Processing* (2002) 589–592.
20. B. Moghaddam and A. Pentland, Probabilistic visual learning for object representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(7) (1997) 696–708.
21. T. K. Moon, The expectation-maximization algorithm, *IEEE Signal Processing Magazine* **13**(6) (1996) 47–60.
22. J. Goutsias and K. Sivakumar, Morphological transformations of image sequences, Master's thesis, Department of Electrical and Computer Engineering, John Hopkins University (2004).
23. M. Kass, A. Witkin, and D. Tezopoulos, Snakes: Active contour models, *International Journal of Computer Vision* **1** (1988) 321–331.

24. L. Loss, G. Bebis, and B. Parvin, Tunable tensor voting improves grouping of membrane-bound macromolecules, *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*.
25. A. Tavakkoli, M. Nicolescu, G. Bebis, and M. Nicolescu, Non-parametric statistical background modeling for efficient foreground region detection, *Machine Vision and Applications* **20**(6) (2009) 395–409.
26. S. Theodoridis and K. Koutroumbas, *Pattern Recogntition* (Academic Press, 2009).