Rapid and accurate developmental stage recognition of *C. elegans* from high-throughput image data

Amelia G. White[1, 2], Patricia G. Cipriani[1], Huey-Ling Kao[1], Brandon Lees[1], Davi Geiger[3], Eduardo Sontag[2, 4], Kristin C. Gunsalus[1], Fabio Piano[1].

[1] Center for Genomics and Systems Biology and Department of Biology, New York University, New York, NY, USA. [2]BioMaPS Institute, Rutgers University, Piscataway, NJ, USA.[3] Department of Computer Science, New York University, New York, NY, USA.[4] Department of Mathematics, Rutgers University, Piscataway, NJ, USA.

Abstract

We present a hierarchical principle for object recognition and its application to automatically classify developmental stages of C. elegans animals from a population of mixed stages. The object recognition machine consists of four hierarchical layers, each composed of units upon which evaluation functions output a label score, followed by a grouping mechanism that resolves ambiguities in the score by imposing local consistency constraints. Each layer then outputs groups of units, from which the units of the next layer are derived. Using this hierarchical principle, the machine builds up successively more sophisticated representations of the objects to be classified. The algorithm segments large and small objects, decomposes objects into parts, extracts features from these parts, and classifies them by SVM. We are using this system to analyze phenotypic data from C. elegans high-throughput genetic screens, and our system overcomes a previous bottleneck in image analysis by achieving near real-time scoring of image data. The system is in current use in a functioning C. elegans laboratory and has processed over two hundred thousand images for lab users.

1. Introduction

C. elegans is a one of the major animal model organisms used to study fundamental questions in development and behavior [1]. Indeed, *C. elegans* was the first animal whose genome was completely sequenced [2], leading to new tools for genome-wide functional analysis. With the discovery of RNAi in *C. elegans*[3, 4], and the development of new resources [5, 6], it is possible to interfere with the function of most of the ~20,000 genes *in vivo*. These advancements have led to a new wave of genome-wide screening, leading to the identification of entire sets of genes required for a number of key developmental and physiological processes.

A feature of *C. elegans* that enables such high-throughput functional genomic analyses is the ability to cultivate it in liquid media. Given their small size (1mm as a mature adult), *C. elegans* can be cultured in a well of a 96-well plate. Using automated liquid handling robots, it is possible to conduct thousands of experiments per week, assaying various perturbation conditions and mutations, and to analyze the resulting effect on – or phenotype of – the organism.

Phenotypic analysis is the most complex aspect of a large-scale screen and is currently carried out mostly by expert manual annotation. This tends to be a pain-staking process that is slow, qualitative and, for large-scale projects, potentially error-prone. The lack of automated quantitative analysis of complex phenotypes from image data is thus a major obstacle to high-throughput screening. From the computer vision perspective, the challenge remains largely unmet because the images are not easily segmented and are complex in terms of object recognition. This type of challenge is common in biological image analysis, and addressing this major hurdle would remove one of the primary bottlenecks in this type of work.

Some progress in analyzing images from screens has been made. Quantitative phenotypes based on the measurement of numerous morphological characters using fluorescent markers have been analyzed in the single celled yeast *S. cerevisiae* [7]and in cell lines from *Drosophila* or human [8]. In *C. elegans*, quantitative phenotypic analysis has been applied to locomotion [9, 10], subcellular phenotypes in the developing embryo [11], and behavior [12, 13]. One of the most fundamental phenotypes to measure is survival or, conversely, lethality. This basic phenotype can be used to test different genetic perturbations or environmental conditions, including temperature or the effects of small chemical molecules and drugs.

Here we present a hierarchical approach to measure the number of embryos, larvae and adults in an image within seconds, and therefore perform *C. elegans* developmental stage recognition (DevStaR). The input images are from

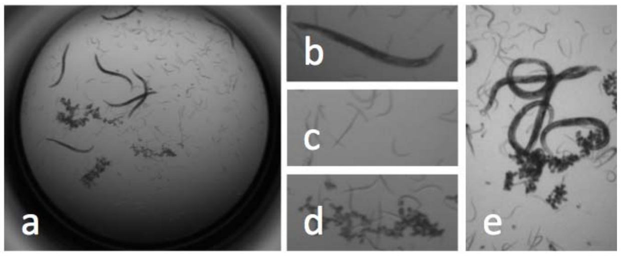


Figure 1: *a) Original image. b-d) Magnified views of: b) an adult worm, c) larvae, d) a clump of eggs. e)Overlap and occlusion of worms and eggs.*

96-well plates and the output is a quantitative measurement of the classified objects within each well. These data enable the calculation of the lethality or survival phenotype from images of mixed *C. elegans* populations. Importantly, this algorithm is implemented in a prototype system that is currently in use in a functioning *C. elegans* laboratory.

The goal of DevStaR is to label each pixel in the image as one of four classes: adult worm, larva, egg/embryo (for this application, we consider these as interchangeable), or background (Figure 1). The successful segmentation and labeling of these images is a significant challenge in computer vision. There is a large variation in contrast and illumination both across individual images, as well as between images. The computer vision algorithm needs to be effective on all images. We must be able to detect and categorize all animals (objects) in the images. Occlusions, overlaps, and deformations of the objects occur because the animals are alive and moving (Figure 1e shows a complex example). The objects may be in different focal planes since the animals are swimming in liquid, and the focal length may differ between images due to manual refocusing by the biologist.

Our DevStaR algorithm runs on average in 4 seconds per image (1200x1600 pixels) on a single 3GHz Intel Xeon processor (we currently use a quad core machine with 16GB RAM). We have demonstrated that the algorithm can clearly distinguish between groups of images with differing levels of lethality, and are now applying it to assess data from high-throughput RNA interference (RNAi) screens.

1.1. Previous Work in Object Recognition

A common approach in object recognition suggests modeling an object and then applying a form of matching or alignment to the image. Usually a search for the model in the image is required, e.g., [14-16]. Such an approach is 'top down', in the sense that a model is being searched in the image. The search procedure has to consider a large space of model variability and thus can be computationally expensive [17-19]. Some approaches go further in depth and organize objects as a combination of parts, such as [20]. We think the main limitation of these approaches is the lack of intermediate levels[21] and the inflexibility to adapt to data exemplars.

Another common approach in object recognition suggests learning to recognize the objects from examples [22-24]. This is a 'bottom up' approach, after the learning phase, as there is no explicit model of an object. Note that the absence of a model suggests recognition to be a discriminative process. Often the discrimination is based on a distance function from a feature list extracted from the image to the set of examples (in which each example is also represented by a feature list). This approach includes the "bag of words" [25]. For most problems, learning machines without grouping mechanisms will be too complex. We note that in some successful learning approaches grouping mechanisms are applied – but not necessarily acknowledged – many times in the architecture of the neighbor structure of the learning (hidden) layers. If one unit in a hidden layer can a priori only "talk" to a selected neighbor set of lower layer units, this choice typically induces grouping mechanisms. However, learning does not usually extract or represent the full global shape information of objects.

Hierarchies of layers of computation have been proposed in the literature [22, 23, 26-29], but they do not specify the need for a grouping mechanism at each layer. There is a compositional approach[30] that considers grouping mechanisms with some form of Bayesian Network, but does not offer the flexibility to use at different layers different methods (either discriminative or model-based) depending on the specific nature of the sub-problems of the layers in the hierarchy. A question we ask is: what is the interplay between learning (i.e. discriminative) methods, model-based methods, and grouping methods? What is the role of grouping with respect to discriminative and model-based approaches? Are there any principles than can help guide us to construct such hierarchies? We have not found in the literature such a clear principled description of a hierarchical approach.

1.2. Our Approach: Hierarchical Principle

We argue that an object recognition machine is composed of layers. Each layer consists of units upon which: (i) Evaluation functions output a label score, (ii) Grouping mechanism of the units resolve ambiguities in the score.

The output of each layer is groups of units and from these groups we get the units of the next layer. We will refer to this routine as a Hierarchical Principle (HP).

Evaluation functions can use either model-based or learning (discriminative) methods to output the label score. Grouping mechanisms can apply local geometrical and topological constraints that neither a pure learning nor a model-based method can do. Grouping does not produce new scores, but rather organizes the scores (for example, neighbor units can be constrained to take similar labels). These constraints eliminate ambiguities in the label scores assigned to the units, producing coherent structures.

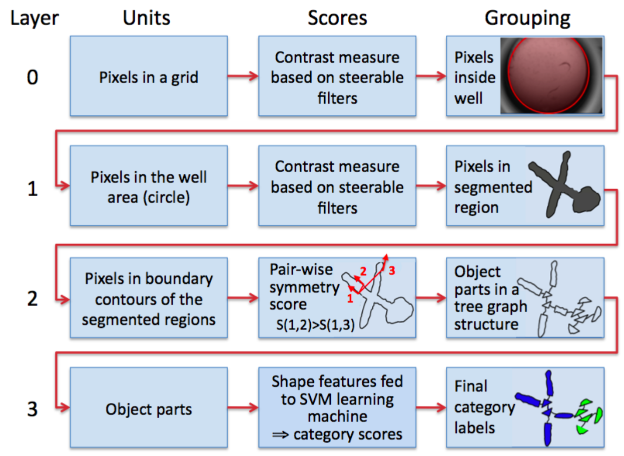
1.3. Specific Contributions

We think our primary contribution to be the development of our DevStaR system, which can replace the need for a human to spend many hours manually analyzing images. In fact thousands of images are now being produced per day, and thus it is no longer possible for humans to visually process them all.

Many previously developed vision algorithms can be described as following our HP including algorithms used here, such as the symmetry axis. In this paper we introduce a novel method for labeling objects by simultaneously grouping and labeling object parts. This method also follows our HP. We use the min cut algorithm as a grouping mechanism to resolve object label ambiguities left by scores of the learning machine. Our approach of using learning methods to output scores followed by a graph algorithm to group units and resolve score ambiguities can be applied to many other computer vision problems.

Another contribution of our work is the expansion of the symmetry axis algorithm [33]. We expand the algorithm by enabling its application to shapes with more than one boundary contour. In particular a shape with holes is described by the external shape contour and the internal hole contours. Our insight was to introduce a cut connecting the external and internal contours as shown in Figure 3f and thus reduce to a one contour representation. This enables the use of a dynamic programming solution.

2. Hierarchical Approach



Layer
Units
Scores
Grouping
0
Pixels in a grid
Contrast measure based on steerable filters
Pixels inside well
1
Pixels in the well area (circle)
Contrast measure based on steerable filters
Pixels in segmented region
2
Pixels in boundary contours of the segmented regions
Pair-wise symmetry score S(1,2)>S(1,3)
1
2
3
Object parts in a tree graph structure
3
Object parts
Shape features fed to SVM learning machine ⇒ category scores
Final category labels

Figure 2: *Architecture of the four-layer machine: each layer is composed of a set of input Units, a computer vision component that outputs Scores, and a Grouping activity to group the units and resolve score ambiguities.*

Let us now discuss our hierarchical approach in a more precise way as applied to DevStaR. Our starting point is a graph representation of the image. A graph is represented by G(v,e) where v is a vertex in the graph and e is an edge in the graph. The image is defined in a graph $G_p(v,e)$, where v represents a pixel and stores the grey level, and e represents an edge between eight neighboring pixels. The image grid is defined by the set (v,e).Below we describe our approach (summarized in Figure 2) in terms of Units, Scores, and Grouping of each layer.

2.1. Layer 0: Attention (Area of Interest)

We must first choose within each image the area of interest in which we will perform the search for objects. The objects (Figure 1) are all inside the well, which has a more or less circular shape in the image (Figure 3a).

Units: The nodes of the pixel Graph $G_p(v,e)$.

Scores: Features such as the contrast surrounding the pixel can be extracted from the image and stored in the nodes v in $G_p(v,e)$. Steerable filters (SF) [31] measure the image contrast in different scales and directions and are the score assigned to each node v. We chose to model these filters, though we could instead have derived them from learning [29].

Grouping: We want to group the pixels inside the well (pw). We use a circle shape as the geometric constraint to group the SF responses. The best circle (group) has the largest sum of SF responses sampled uniformly along 32 vertices on the circumference, and in the perpendicular direction of the circumference (Figure 3a). We output the graph $G_{pw}(v,e)$.

2.2. Layer 1: Filtering and Segmentation

The Units of this layer are the nodes of $G_{pw}(v,e)$. The main problem here for segmentation is illumination and contrast variation, both within and across images. We apply this layer separately to segment the small and large objects. To remove the small objects we apply a Gaussian blur with sigma = 6. The blurring visually removes small objects by mixing background and object grey level pixels. After the large objects are segmented we remove them from $G_{pw}(v,e)$ and use this graph as the units for the small objects.

Units: The nodes of the graph $G_{pw}(v,e)$.

Scores: The score combines the SF responses to the image at each vertex v into one quantity, the energy E(v).

Grouping: We select pixels based on a threshold value for E(v). The well area is sub-divided into sectors, and a

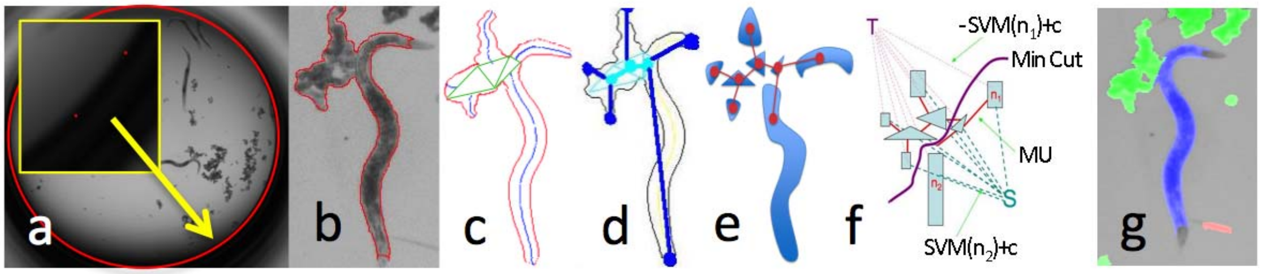


Figure 3: a) Identification of the location of the well area as the optimal circle. The total score is the sum over 32 uniformly sampled points on the circumference (red dots, magnified in yellow box). B) Boundary of segmented region (red). c) Extraction of the symmetry axis (blue line); bifurcations are shown with green triangles. d) Tree structures where edges store object part information. e) Output: the dual tree structure of d, represented as $GT_{op}(v, e)$.f) A tree graph of object parts with the addition of sink (T) and source (S) nodes. g) Final results of the category labels after application of the min-cut algorithm. The egg clump (green) is clearly separated from the adult worm (blue).

local threshold value is defined by the median value E(v) in each image sector. The threshold thus varies both within and between images to account for variation in illumination. Pixels that pass the local threshold value are grouped into connected components. The convex hull graphs of each connected component are found. More precisely, $\bigcup_i G_{CH}^{i}(v,e)\,, i = 1, \ldots, L_{CH}$, where G_{CH}^{i} is the convex hull graph of one connected component index by i, and assuming there are L_{CH} large objects. To obtain an accurate segmentation we apply the min cut algorithm. We add an edge connecting the vertices of $G_{CHi}(v,e)$ to the source and to the sink, giving a weight value (E(v) – Th) to the edge, where Th is the median value of all energies E(v) within $G_{CHi}(v,e)$. In this way local illumination will affect the Th value of each $G_{CHi}(v,e)$. An empirically optimized edge weight μ is also added to edges between neighboring pixels, which encourages them to adopt the same label (μ is the same for all images). We then apply a min-cut algorithm [32] to obtain the final segmentation. Due to overlaps of objects, a region may contain more than one object. The output of this layer is the set of boundaries of these segmented regions. Each boundary in a region is a graph: an ordered list of pixel coordinates, where the coordinates are stored in the pixel vertices and the graph edges connect one pixel vertex to the next, yielding the order. Thus, the output of this layer for the large objects is $\bigcup_i \Gamma_{CH}^{i}(v,e)\,, i = 1, \ldots, L_{CH}$. The output for the small objects is $\bigcup_j \Gamma_{CH}^{j}(v,e)\,, j = 1, \ldots, S_{CH}$, where S_{CH} is the number of small objects found in the image. Therefore, total number of objects found in the image is$N_{CH} = L_{CH} + S_{CH}$. Results of the segmentation are shown in Figures 3 and 6.

2.3. Layer 2: Object parts and deconstruction I

In the second layer we apply a model-based component, a shape mechanism able to break regions into object parts (Figure 4). The idea is that even if multiple objects are overlapping and creating one region, the object parts can be identified. We use the symmetry axis method to break regions into object parts.

Units: The set of region boundaries described by the graph $\bigcup_i \Gamma_{CH}^{i}(v,e)\,, i = 1, \ldots, N_{CH}$.For each region boundary $\Gamma_{CH}^{i}(v,e)$ we have an ordered list of pixels. This list can be parameterized in two “opposite” ways: counter-clockwise with parameter *s* or clockwise with parameter *t* (Figure 4c), yielding $\Gamma_1(s)$ and $\Gamma_2(t)$.

Scores: The symmetry scoring for a pair of nodes in the graph $\Gamma_{CH}^{i}(v,e)$ is a co-circularity measure (see Figure 4b):

$$\boldsymbol{S}(s,t) = \big(x(s) - \tilde{x}(t) \odot (\tau(s) + \tilde{\tau}(t)\big) + \big(x(s) - \tilde{x}(t) \odot (\tau(s) + \tilde{\tau}(t)\big)^{\perp}, \quad (1)$$

where $x(s)$ and$\tilde{x}(t)$ are the node coordinates for the respective parameterizations, τ and $\tilde{\tau}$ are the corresponding unit tangent vectors and $\odot$ is the dot product of two vectors. Perfect symmetric elements have score $\boldsymbol{S}(s,t)=0$ and the more symmetric a pair is, the lower is the score.

Grouping: A tree graph structure is imposed, and we apply the dynamic programming algorithm from [33] to find the best pairwise matching of elements to obtain the optimal symmetry pairing, taking into account the score cost of equation (1) and a penalty for the number of object parts created (Figure 4d,e). Other algorithms for this task can be considered (e.g.[34]).

The first output of this grouping mechanism is a tree graph, in which graph edges represent object parts. From this tree, a dual tree is built so that object parts become nodes on this dual tree and the bifurcations become edges linking object parts (Figure 3d, e). We will refer to this tree graph as $GT_{op}(v, e)$, where v = part of an object and e = edge connecting nodes that represent adjacent object parts. The dual tree representation, $GT_{op}(v, e)$, is a natural representation for the object categorization process.

There is one parameter which controls how easy it is to create a branch in the tree, otherwise any little bump in the shape will lead to a small branch of the tree. The algorithm is extremely robust, as we set the parameter once for all images. Figures3 and 5 show typical results. A problem with this method occurs when shapes have more than one boundary contour (see Figure5). This occurs quite often, due to overlaps of objects and even simply due to deformation. We thus had to create a simple novel solution. Our insight was to select a point in each boundary contour, and then to apply a "topological surgery" to link the two boundary shapes, creating a final long unique boundary shape (Figure 4f). A greedy method is used to select the closest pair of points, in Euclidean distance, where one point is on the internal contour and the other on the outline. We then applied the same dynamic programming algorithm to the new unique boundary shape. Figure 5 shows an illustrative result.

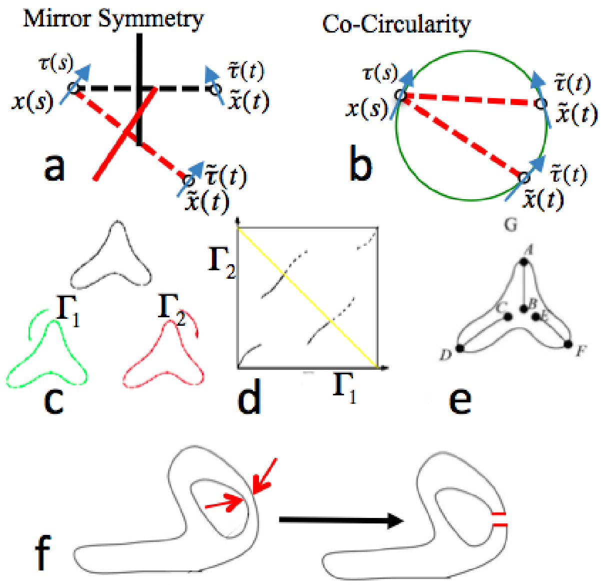


Figure 4:*Scoring and grouping of layer 2. The measure of symmetry, of two elements on the shape boundary, is given by a co-circularity score (a) or the equivalent concept of a mirror symmetry score (b). c) A shape boundary with two parameterizations: $\Gamma_1$counter-clockwise (green), and Γ_2 clockwise (red). d) The matching space is created where the x-axis is the shape boundary parameterized by $\Gamma_1$and the y-axis is the same shape parameterized by Γ_2. The matching scores are given by the measure in Equation 1. The grouping mechanism is a dynamic programming algorithm that takes into account the scores and a penalty for bifurcation, and extracts the optimal matching path as shown in the diagram. e) A tree structure is created in which bifurcations on the matching space correspond to nodes in the tree graph (breaking the object in parts). f) Our novel contribution is a Topological surgery to yield one long and unique contour from two contours.*

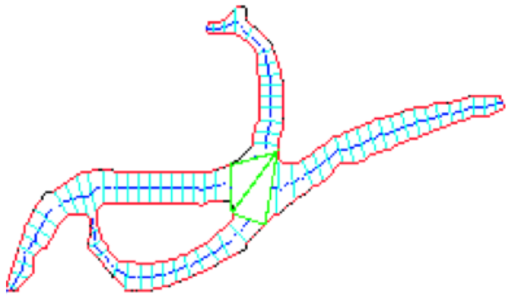


Figure 5: *In addition to the outline of regions, internal region boundaries also appear frequently in images due to overlaps of animals and deformations (worms can loop around themselves). This figure illustrates examples of our algorithm using the novel idea of introducing a "topological surgery" between multiple contours. Every 5th pair-wise match is shown by lines connecting matching pixels on each boundary.*

2.4. Layer 3: Object part labeling and deconstruction II

We will now address the problem of object categorization. Our object label categories are "worm" or "egg clump" or "egg" or "larva". For overlapping objects of different categories, it would make no sense to output one object label or category for an entire region. Therefore, we must focus on the problem of first labeling the object parts and then grouping object parts to produce the final object labels.

Units: nodes in the graph $GT_{op}(v, e)$.

Scores: We desire to produce scores for the four categories. We know that large objects are only worms or egg clumps, and that small objects are only larvae or individual eggs. We therefore focus on the more difficult problem of large object categorization. From examining many examples of worms and egg clumps, we concluded that the characteristics that distinguish the two shapes are their thickness, elongation, size, and the smoothness of the boundary contour. Instead of determining empirically how to optimally combine this information, we apply a learning procedure. The input features of the object parts are the following list: area, length of the symmetry axis, length of the boundary contour, and change in width(calculated as $\sum(\Delta_w)$, where Δ_w is the difference between the widths of consecutive matches along the object part).This last feature was calculated at three different scales, taking the difference in width for adjacent matches, every three matches, and every five matches. We also consider the sum of the number of times that Δ_w changes sign, i.e. how often Δ_w switches direction: a very "bumpy" boundary contour will give a high value, whereas that of a smooth circle will give a very low value.

In total we extracted 13 features for each object part. An SVM learning method [35] was trained using ~2000 examples. The SVM assigns a score to each category – a negative score for one label and a positive score for the other. The magnitude of this score tells us how well decided the algorithm is about the label category.

Grouping: We observed that some object part label scores favor the "wrong" category, so that thresholding the score would assign some incorrect categories. We have more information than just object part scores; we also have the tree graph, from which the proximity of object parts is easily accessed (Figure 3e). Our insight was to use this proximity in the tree graph to better resolve the scores. Typically the nodes with scores that would give the wrong label had a neighbor node with stronger scores that carried more discriminative power, and so could help correct the score of their neighbor. To produce final scores that exploit this neighbor structure, we applied the min-cut algorithm (Figure 3f) using edge weights between the source and sink determined from the SVM output, and edge weights μ between the nodes based on their proximity in the tree graph. This grouping process gave a clear improvement in our results (Figure 3g). Only one parameter, μ, was used for grouping. We used the same value for μ for all images tested.

We are thus proposing a novel method to label objects in the presence of overlapping, occlusions and clutter. Assuming a segmentation of the overlapping objects is done, we use the symmetry axis to break the object in parts, followed by labeling the object parts with a learning method. The final label of the objects is then accomplished via the grouping of the parts and their scores into final separate objects and final labels.

3. Experiments

The biological images to which we applied our algorithm are inherently challenging for automated image analysis: they show high variation in illumination and contrast, and images often contain complicated occlusions and deformations of the objects to be classified. To perform DevStaR we need to segment and label each *C. elegans* developmental stage from an image with good performance even in these difficult conditions (Figure 6). The pixel area of each developmental stage can be used to calculate a quantitative measure of lethality in a mixed population of adults, larvae, and eggs.

In order to analyze the performance of our DevStaR system, we first evaluated performance in terms of pixel labels. We compared pixel values labeled by the algorithm with manual coloring of objects by a human expert for the same set of images. The latter is extremely time consuming and was performed specifically for this evaluation; when a human scores images for lethality, s/he

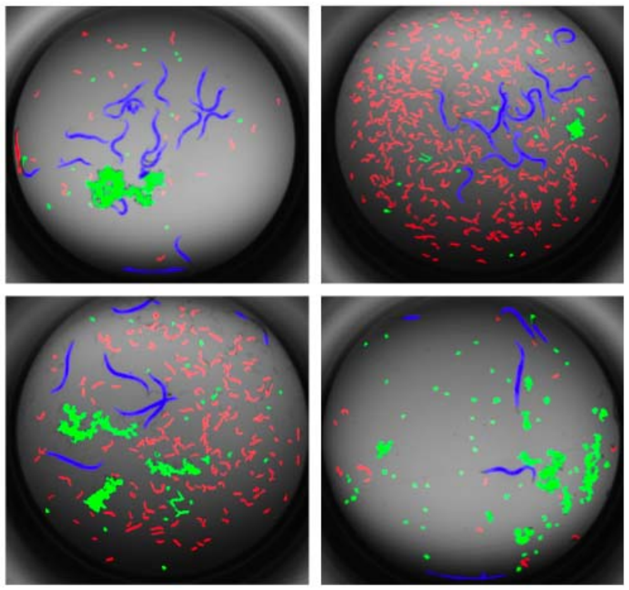


Figure 6:*Typical results of segmentation and labeling from DevStaR showing adults (blue), larvae (red), and eggs (green).*

usually estimates the number of animals at each developmental stage, reserving quantification for exceptional cases. Thus while this comparison gives us an idea of the object recognition error of DevStaR, it does not represent a direct comparison with typical results of human scoring. We assume that the biologist's labels are correct and calculate the number of false positive (FP), false negative (FN), true positive (TP), and true negative (TN) pixels segmented and labeled by DevStaR.

We evaluated precision and recall for the separation of objects from background and for the segmentation and labeling of each developmental stage (Figure 7). For the comparisons of objects/background and adult labeling, our algorithm has both high recall and precision – i.e. DevStaR accurately matches a human estimation of both total objects and the location of adult worms in an image. For eggs, there is only one clear low outlier in both measures: an image containing a large clump of eggs that is mislabeled. Since most of the eggs in an image often clump together to form one object (see Figure 6), if it is mislabeled then a large proportion of the eggs will not be accurately detected. This problem is hard to avoid, as it arises from a single object labeling error. We can approach this problem by using the scores from layer 3 as weights to each category label and computing pixel area of each category as a weighed sum of the pixels. Precision and recall are lowest for larvae, mainly due to differences between DevStaR and the human in labeling boundary pixels on these relatively small objects. Larvae are very small and only contain roughly 250 pixels; thus an error of one pixel around the object boundary can translate into a large error in total larval area. The set of images colored by the biologist also contained relatively few larvae, so

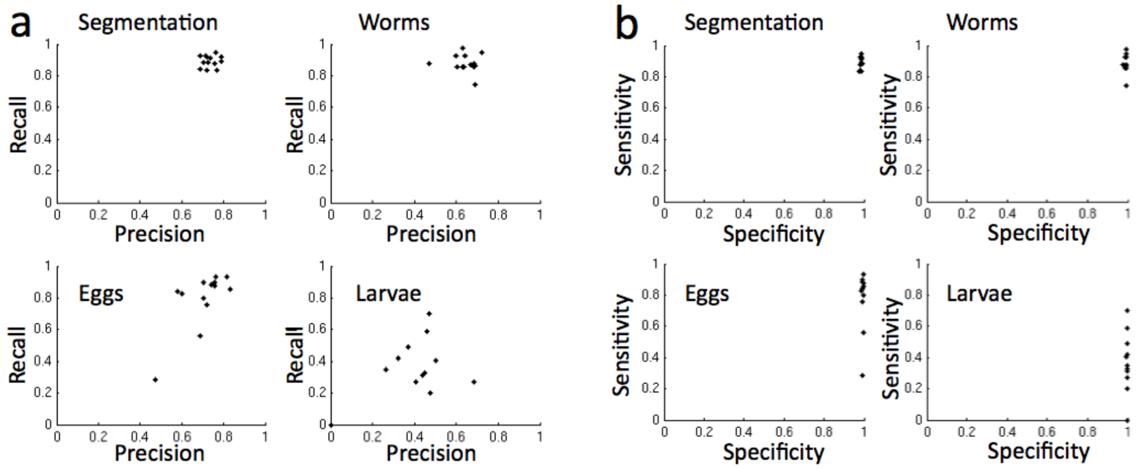


***Figure** 7:a) Precision vs. recall plots for foreground-background segmentation (top left) and for segmentation and labeling of adults (top right), eggs (lower left), and larvae (lower right). Precision = TP/(TP + FP). Recall = TP/(TP + FN). Each point corresponds to an image .b) Sensitivity vs. specificity plots for the foreground-background segmentation and for segmentation and labeling of adults, eggs, and larvae. Sensitivity = TP/(TP + FP). Specificity = TN/(TN + FP). Each point corresponds to an image.*

any discrepancy would represent a large proportion of the total pixels segmented and labeled as larvae. This error can largely be overcome later by counting isolated labeled objects of appropriate sizes as individual animals.

We also evaluated sensitivity vs. specificity for the object/background separation and for the segmentation and labeling of adults, larvae, and eggs (Figure 7; sensitivity is the same as precision above). The high specificity for the object/background separation and all developmental stages tells us that DevStaR accurately labels background (which constitutes many more pixels than each class of object).

In order to evaluate whether DevStaR can accurately measure the lethality or survival phenotype and discriminate between images with different levels of lethality, we used data from mutant *C. elegans* strains containing temperature sensitive (*ts*) alleles. A *ts* allele is a mutation in a gene that causes loss of or severely reduced function at a restrictive (high) temperature, but retains normal or near-normal function at a permissive (normal) temperature. Using a *C. elegans* strain carrying a *ts* allele for an embryonic lethal gene, we can control lethality by adjusting the temperature in which the animal is raised. The expected level of lethality at different temperatures for different mutant strains is known.

We use the total labeled pixel area for each developmental stage as a proxy for the number of *C. elegans* progeny that are alive (larvae) or dead (eggs), and calculate ratios of these areas as a measure of lethality and survival. These relative areas give us sufficient discriminative power that we do not need to calculate survival in terms of the percentage of individuals (for which we would need to divide the total area by the pixel area per animal). In testing on more than 1700 images, DevStaR can clearly distinguish different levels of lethality and survival. Figure 8 shows the distribution of lethality and survival rates for a strain carrying a *ts* mutation in a gene that is essential for embryonic survival, measured at the restrictive (22.5ºC) and permissive (15ºC) temperatures. The spread of the distribution at each temperature arises from both biological variation and measurement error made by DevStaR. Even with both sources of error, the distributions at the two temperatures are clearly distinct.

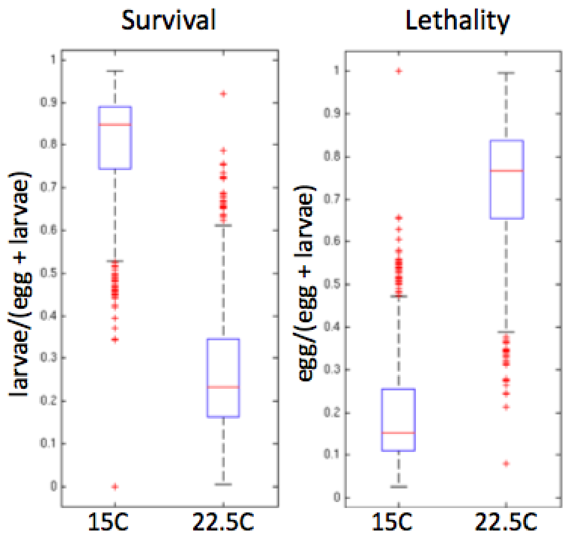


***Figure 8:** Box plots of pixel area distributions for survival (left) and lethality (right) for a strain with an embryonic lethal mutation in the spn-4 gene, at permissive (15ºC, n=1056 images) and restrictive (22.5ºC, n=672 images) temperatures. The distributions are significantly different (p = 0, Student's t test).*

4. Conclusion

We have presented an object recognition machine that can accurately measure the amounts of adult worms, larvae and embryos from high throughput image data and therefore produce a quantitative measure of the lethality phenotype. The techniques applied in this application can be generalized to other object recognition problems, especially image analysis of high throughput biological image data.

5. Acknowledgement

Data acquisition was supported in part by a grant from NICHD to FP and KCG (R01HD046236). We thank R. Cole for useful discussions on the symmetry axis algorithm. We thank the reviewers for useful comments.

References

[1] Brenner, S., *The genetics of Caenorhabditis elegans.* Genetics, 1974. **77**(1): p. 71-94.
[2] *Genome sequence of the nematode C.elegans:a platform for investigating biology.*Science,1998. **282**(5396): p.2012-8.
[3] Fire, A., et al., *Potent and specific genetic interference by double-stranded RNA in Caenorhabditis elegans.* Nature, 1998. **391**(6669): p. 806-11.
[4] Guo, S. and K.J. Kemphues, *par-1, a gene required for establishing polarity in C. elegans embryos, encodes a putative Ser/Thr kinase that is asymmetrically distributed.* Cell, 1995. **81**(4): p. 611-20.
[5] Kamath, R.S., et al., *Systematic functional analysis of the Caenorhabditis elegans genome using RNAi.* Nature, 2003. **421**(6920): p. 231-7.
[6] Rual, J.F., et al., *Toward improving Caenorhabditis elegans phenome mapping with an ORFeome-based RNAi library.* Genome Res, 2004. **14**(10B): p. 2162-8.
[7] Ohya, Y., et al., *High-dimensional and large-scale phenotyping of yeast mutants.* Proc Natl Acad Sci U S A, 2005. **102**(52): p. 19015-20.
[8] Jones, T.R., et al., *Scoring diverse cellular morphologies in image-based screens with iterative feedback and machine learning.* Proc Natl Acad Sci USA, 2009.**106**(6): p.1826-31.
[9] Geng, W., et al., *Automatic tracking, feature extraction and classification of C elegans phenotypes.* IEEE Trans Biomed Eng, 2004. **51**(10): p. 1811-20.
[10] Tsibidis, G.D. and N. Tavernarakis, *Nemo: a computational tool for analyzing nematode locomotion.* BMC Neurosci, 2007. **8**: p. 86.
[11] Ning, F., et al., *Toward automatic phenotyping of developing embryos from videos.* IEEE Trans Image Process, 2005. **14**(9): p. 1360-71.
[12] Cronin, C.J., Z. Feng, and W.R. Schafer, *Automated imaging of C. elegans behavior.* Methods Mol Biol, 2006. **351**: p. 241-51.
[13] Huang, K.M., P. Cosman, and W.R. Schafer, *Automated detection and analysis of foraging behavior in C. elegans.* J Neurosci Methods, 2008. **171**(1): p. 153-64.
[14] Binford, B.a., *ACRONYM.* 1981.
[15] Forsyth, D.A.a.F., M. M., *Automatic Detection of Human Nudes.* IJCV, 1999. **32**(1): p. 14.
[16] Grimson, W.E.L., *Object Recognition by Computer-The Role of Geometric Constaints*. The MIT Press Classics Series. 1990, Cambridge, MA: MIT Press.
[17] Jean Ponce, M.C., Sung-il Pae and Steve Sullivan., *Shape Models and Object Recognition.* Shape, Contour and Grouping in Computer Vision, 1999.
[18] Forsyth, S.I.a.D.A., *Probabilistic methods for finding people.* IJCV, 2001. **43**(1): p. 23.
[19] K. Mikolajczyk, B.L., and B. Schiele, *Multiple Object Class Detection with a Generative Model.* CVPR, 2006.
[20] Fei-Fei, L.a.F., R. and Perona, P, *Learning Generative Visual Models for 101 Object Categories.* Computer Vision and Image Understanding, 2006.
[21] V. Ferrari, L.F., F. Jurie, and C. Schmid, *Groups of Adjacent Contour Segments for Object Detection.* INRIA Technical Report, Grenoble, 2006.
[22] LeCun, H., and Bottou, *Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting.* CVPR, 2004.
[23] T. Serre, L.W., S. Bileschi, M. Riesenhuber and T. Poggio, *Object recognition with cortex-like mechanisms.* IEEE PAMI, 2007.
[24] Ullman, S., Sali, E. & Vidal-Naquet, M, *A fragment-based approach to object representation and classification.* . Arcelli, L.P. Cordella & G. Sanniti di Baja (eds.), Int. Workshop on Visual Form, Berlin, 2001: p. 15.
[25] S. Lazebnik, C.S., and J. Ponce, *A Discriminative Framework for Texture and Object Recognition Using Local Image Features.*
[26] Geman, Y.J.a.S., *Context and hierarchy in a probabilistic image model.* CVPR, 2006: p. 8.
[27] Jedynak, D.G.a.B., *An active testing model for tracking roads from satellite images.* IEEE Trans. Pattern Anal. Mach. Intell, 1996. **18**: p. 14.
[28] Jones, P.V.a.M., *Rapid object detection using a boosted cascade of simple features.* Conference on Computer Vision and Pattern Recognition, 2001: p. 8.
[29] Kevin Jarrett, K.K., Marc'Aurelio Ranzato and Yann LeCun, *What is the Best Multi-Stage Architecture for Object Recognition?* Proc. International Conference on Computer Vision (ICCV'09), 2009.
[30] Geman, Y.A.a.D., *A computational model for visual selection.* Neural Computation, 1999. **11**: p. 14.
[31] W. T. Freeman, E.H.A., *The Design and use of Steerable Filters.* IEEE Transactions on PAMI, 1991. **13**: p. 15.
[32] Boykov, Y., *An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Computer Vision.* IEEE Transactions on PAMI, 2004. **13**: p. 9.
[33] D.Geiger,T. Liu, R.V. K., *Representation and Self-Similarity of Shapes.*IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003. **25**(1): p. 86.
[34] Kimia, K.S.a.B.B., *Parts of Visual Form: Computational Aspects. IEEE Transactions o*n Pattern Analysis a*nd Machine Intelligence, 1995. **17**(3): p. 12.*
[35] Burges, C., *A Tutorial on Support Vector Machines for Pattern Recognition.* Data Mining Knowledge Discovery, 1998. **2**: p. 46