



Universiteit
Leiden
The Netherlands

Pattern Recognition in High-Throughput Zebrafish Imaging

Nezhinsky, A.E.

Citation

Nezhinsky, A. E. (2013, November 21). *Pattern Recognition in High-Throughput Zebrafish Imaging*. Retrieved from <https://hdl.handle.net/1887/22286>

Version: Corrected Publisher's Version

License: [Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden](#)

Downloaded from: <https://hdl.handle.net/1887/22286>

Note: To cite this publication please use the final published version (if applicable).

Cover Page



Universiteit Leiden



The handle <http://hdl.handle.net/1887/22286> holds various files of this Leiden University dissertation

Author: Nezhinsky, A.E.

Title: Pattern recognition in high-throughput zebrafish imaging

Issue Date: 2013-11-21

3 Anchor Region Based Pattern Recognition in Image Space

Under preparation for publication and based on:

A. Nezhinsky & F.J. Verbeek: Anchor Region Based Pattern Recognition in Image Space

3.1 Introduction

We have designed and developed a generic algorithm for use in High Throughput applications in the life sciences. We want to automatically analyze images; in particular we aim at the detection and annotation of shapes of biological interest.

Shapes often seem similar to the human eye; this is a perceptual generalization as each individual (instance) is different. Size, position and a lot of other aspects differ. Global proportions and image magnification are the same. An image can contain multiple shapes and some of them are clustered together which complicates automatic analysis. Without proper localization and annotation of the regions in the shapes, the measurement of features within each instance is hampered. Additionally, each shape should be annotated in such a way that it can be compared to other retrieved shapes. For our analysis we want to establish an uniform way to overcome these problems.

The analysis is part of a High Throughput setup and therefore the starting point for our approach is, that in at least 95% of the cases, retrieved shapes should be detected and annotated correctly.

As a case study we have considered High Throughput analysis of zebrafish as it requires automated analysis of thousands of zebrafish larvae [54]. For each zebrafish larva, a brightfield image is acquired as shown in Figure 3.1. The brightfield image is used for the localization and annotation of the zebrafish shape. Until recently, these images were manually analyzed. The analysis included localization of the zebrafish shape in the brightfield images.

This chapter is structured as follows:

- In Section 3.1.2 we describe the development of the algorithm.
- In Section 3.3 we continue with the application of the algorithm to zebrafish larvae.
- In Section 3.4 we extend the application of the algorithm to other shapes.
- In Section 3.3.4 a typical case study with statistical processing of the result is presented to show the virtue of the current application.
- We close this chapter with a discussion of all results presented (Section 3.5).

3.1.1 Related Work

In order to make our approach more generic we do not design an algorithm specific for our case study, but rather design an algorithm capable of success when applied to other problems. The general starting point therefore, is to find the objects that can be globally defined by their shape. In addition, we investigate the effect of the presence of more than one instance. As prior knowledge we adapt the fact that the global shape is known but the position for each shape is not known. We do not, however, consider active snakes or other *free form* methods as a possibility.

We assume that in an image only the shapes of interest and some accidental noise are present. Therefore, if we can separate the foreground from the background the

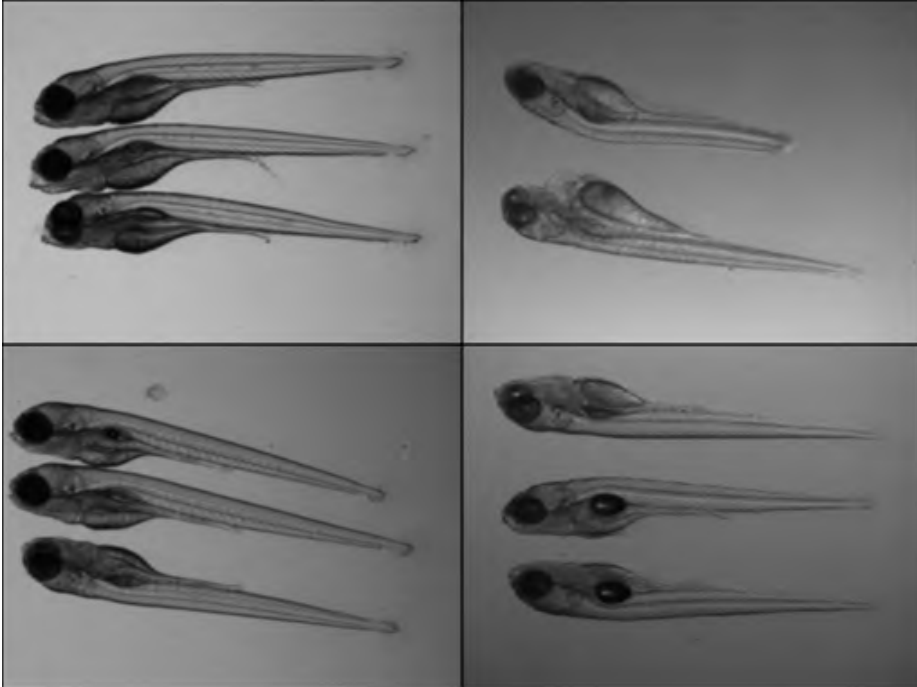


Figure 3.1: Examples of brightfield images (converted to grayscale) containing zebrafish embryos. Note the differences in illumination resulting in shades in the background and variations in intensity.

foreground object there will most likely contain the shape of interest and/or some accidental noise. An example of an approach combining both *bottom-up* and *top-down* methods is given in [22]. The authors first exploit *bottom-up* image cues to create an over-segmented representation of an image and then merge the segments by assigning labels that correspond to the object category.

By the binarization (*bottom-up*) approach we are converting the problem from the RGB domain to the binary domain, [21]. The choice of approach is problem related and is strongly dependent on the data at hand. In the cases where prior spatial information is known this characteristic can be exploited and used for setting the threshold value. We have made a small inventory of different approaches for binarization and compared the results.

Following the binarization we require a *top-down* approach for the pattern recognition step, that can simply use a sketch or a logic representation of the object we are looking for. A deformable template approach can be used for this purpose. Binary images are successfully used for template and polygon matching [17, 29].

When binary objects are unintentionally interconnected it is good practice to split up the binary image into a collection of binary regions based on some geometric assumptions about the data. In the literature the rules to split up large objects are the *minima rule* and the *short cut rule* [33] suggesting splitting an object at its concavities. Finding concavities is preceded by finding the convex hull of an object [6]; this method can be very effective. In such problems the choice of the algorithm that is used to split the binary objects based on their concavities is very problem dependent. The recognition step still must be performed on the resulting binary elements. We want to investigate if a more globally operating method can be found.

A brute force template matching algorithm that uses straightforward matching of a template element to every possible deformation, transition and rotation would be too computationally intensive [28]. This problem can be overcome by reducing the search space through application of predefined rules or a method should be used which is not dependent on pixel by pixel comparison. Predefined rules can be retrieved from *a priori* knowledge about the image. We therefore want to investigate a new direction in template matching approaches as addition to existing techniques. To that end we developed an algorithm capable of retrieving deformed shapes based on a prototype model.

3.1.2 Algorithm Overview

We propose a complete framework for analysis and recognition of biological shapes. It consists of multiple steps and several scenarios. The analysis pipeline is shown in Figure 3.2. As input a color or grayscale image (*Input image*) is used and in addition a *Prototype Template* of the shape that is looked for.

First, a *Preprocessing* step is applied in order to separate the object(s) from the background and by this the Input Image is converted to a *Binary image*. Segmentation alone, however, does not give satisfactory results, as we are not only interested in separating background from foreground, but we also want to recognize the particular shapes and their position and best possible representation in the image.

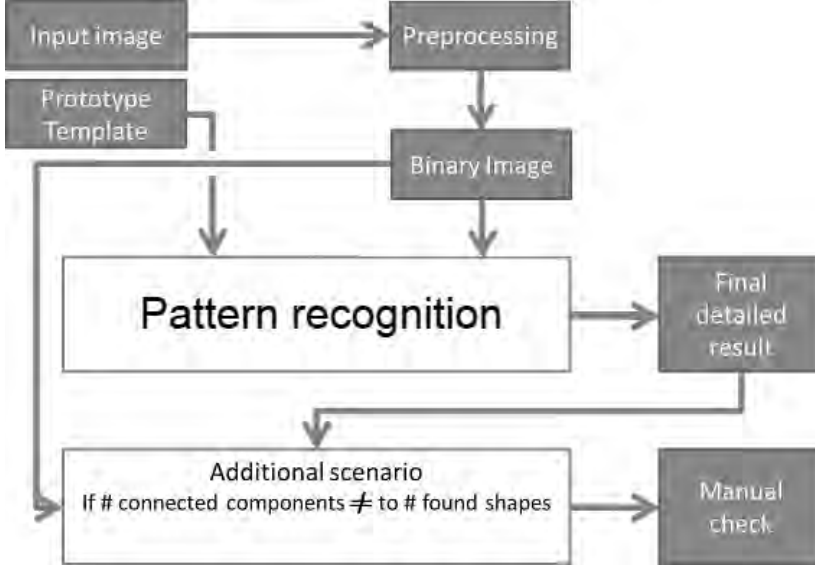


Figure 3.2: Proposed framework for the main scenario of shape localization and annotation.

The second step, a *Pattern Recognition* step, is applied in order to recognize the shapes from the binary image and annotate them. This is done by retrieving the best representation of the prototype template in the binary image. The resulting shape (*Final detailed result*) can be used for further measurements.

In order to make this model fit for High Throughput (HT) applications we propose a scenario so that the results produced by the main scenario need to be checked by a human for consistency. This scenario is an addition to the fully automated scenario in the following way. From the *Preprocessing* step the number of *Connected components* that are extracted as foreground is retrieved. In cases in which the shapes are not overlapping each other and no other objects are present in the image, the final result should present the number of objects that equals the number of connected components. Therefore we can assume that in most cases the number of connected components from the preprocessing step equals the number of objects present in the image. If the number of connected components does not equal the number of retrieved shapes from the final result, we mark this result as something that needs to be evaluated by a human observer for correctness. The complete process is shown in Figure 3.2.

3.1.3 Pre-processing

For the proposed approach we need a bottom up technique that can easily separate foreground and background. The production of a binary image I_b from a grayscale or a RGB color image differs per problem at hand and will therefore not be the focus here.

However, we will in short describe the binarization algorithm for a specific shape, since we will use it as an example throughout this chapter.

Thresholding of the typical images containing biological objects is not a trivial process. This is due to the fact that images often suffer from uneven illumination as result of microscopy, contain noise and have different color values representing the similar structures. As an example consider the images and their differences in Figure 3.1.

We have evaluated the performance of different thresholding methods for the images as used in this case study [43]. For the segmentation we did not use a texture based descriptor, since it was reported [69] that this method produced false positives when a background region contained texture similar to the object of interest. A detailed comparison of binarization methods for the images containing zebrafish is given in Chapter 2.

For the binarization we have probed a state of the art method, mean shift segmentation [59]. This approach distributes the image into connected regions (or clusters) based on color similarity. Since the mean-shift can run with different parameter settings these should be derived from the data. To accomplish this we run the algorithm with different settings.

To determine the correct settings for the mean-shift segmentation algorithm for the case study we use the assumption that the shape is never located at the edge of the image (1) and that the objects of interest are smooth shapes (2):

1. We assume the objects of interest are never located at the edge of the image. Therefore we can assume that the image border consists for the most part or solely of the background: all pixels that are part of the same cluster as the border pixels are also background. As a result of this assumption the background can be easily subtracted from the image, and in this way only the objects of interest and noise remain.
2. A smooth object will tend to have a lower compactness (Eq. 3.1.1) then edgy ones, so we can use this measurement to determine if an object is present in the image. For each collection of clusters resulting from mean-shift segmentation we start with checking the compactness of the objects (or clusters). We subtract the background clusters from the collection and of the remaining ones we calculate the average compactness. The settings that provide the highest compactness are chosen as best settings. In order to calculate compactness area and perimeter of each binary object o_i are considered:

$$compactness = perimeter(o_i)^2 / area \quad (3.1.1)$$

As an example throughout this section we use the binary image I_b as shown in Figure 3.3. This image was derived from the grayscale image through mean-shift segmentation. The result of the preprocessing is a binary image without annotation of the shapes. Moreover non specific shapes, noise and debris are still present in the image and need not be taken into account for our measurements. Therefore, additional segmentation is needed.



Figure 3.3: An example of grayscale and binary image containing zebrafish embryos and some artificial debris.

3.2 Pattern Recognition: The Anchor Region Based Method

Now that we have removed the background with the pre-processing step we need to recognize and annotate the shapes that are segmented from the binary image. As *Pattern Recognition* step we have developed a Deformable Template Matching method: the anchor region based algorithm. In Figure 3.4 an overview is shown.

We investigate the feasibility of a new Deformable Template Matching approach compared to solutions presented in Chapter 2.

The template matching takes a binary image I_b as obtained from the preprocessing step. Binary image I_b consists of two types of regions: foreground regions R_0, \dots, R_i and the background.

Foreground regions R are defined as a collection of interconnected pixels with value 1. Background regions B are defined as a collection of interconnected pixels with value 0.

3.2.1 Prototype Template

The prototype template consists of template elements (t_0, t_1, \dots, t_n) . The sequence of elements in the tuple defines the order of these objects. Each element t is defined by its size r and a length limitation l :

- r is the radius of a circle that inscribes the object to be determined
- l is the distance between the inscribed circle centers of t

Graphically this is depicted in Figure 3.5.

Since we want to focus on biological shapes the size and length may vary substantially. Therefore we represent r and l as variables instead of constants. We assume, however, that the ratio of different template elements as relative to each other is the same, while the object itself might be scaled. In this way we can define a collection of shapes with variable width and length that all match the shape of interest.

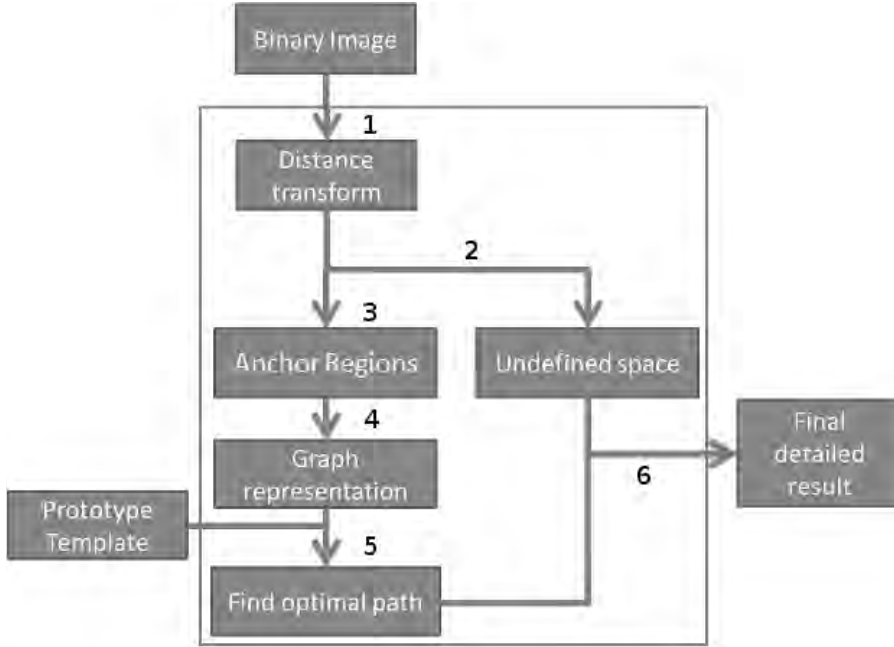


Figure 3.4: Overview of the proposed framework for Object Recognition. The first input is a binary image that consists of fore- and background regions that are obtained from the pre-processing. Second input, a prototype template, represents the ideal shape of the object that we would like to retrieve. As output detailed results are provided in the form of object instance locations. The numbers $(1, \dots, 6)$ refer to the explanatory text.

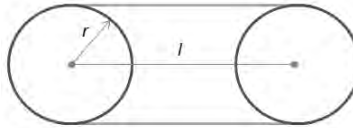


Figure 3.5: A template element $t(r, l)$, defined by the inscribed circle radius r and inner length l (extension within which this circle can shift).

Example Template

For example, if we want to localize all elements that look like a zebrafish larvae as depicted in Figure 3.3 we can define the prototype template as a head object t_0 followed by a body object t_1 followed by a tail object t_2 . The prototype template shape sequence is then $[t_0 t_1 t_2]$. Now we construct the template as follows.

For the head region instead of a constant value for r we define $r_0 \geq 0, r_0 < r_1$. This means that the head element must have a radius of inscribed circle smaller than for the body region.

The radius of the tail region at its largest should be smaller than the body region, with a minimum of 70%, thus, $0 < r_2 < 0.7r_1$.

We assume that the length of the head and body region together should be less than half of the entire shape, thus, $l_0 + l_1 < 0.5 \cdot (l_0 + l_1 + l_2)$.

The zebrafish larvae shape can be represented as:

$$T_0 = \begin{cases} [t_0 t_1 t_2] & \text{sequence} \\ r_0 \geq 0, r_0 < r_1, 0 < r_2 < 0.7r_1 & \text{radius relationship} \\ r_{min} < r_i < r_{max} & \text{radius limits} \\ (l_0 + l_1) < 0.5 \cdot (l_0 + l_1 + l_2) & \text{length relationship} \\ l_{min} < (l_0 + l_1 + l_2) < l_{max} & \text{length limits} \end{cases} \quad (3.2.1)$$

The values of $r_{min}, r_{max}, l_{min}, l_{max}$ are here to restrict the allowed object size or length for parts of the zebrafish object. If we set $r_{min} = 0, r_{max} = \infty, l_{min} = 0, l_{max} = \infty$, the representation becomes fully scale independent.

3.2.2 Conversion from Image to Graph

In this section we elaborate on the steps 1, 2, 3 and 4 from Figure 3.4. The template matching step consists of searching for characteristic regions in I_b as they are defined in the prototype template T_0 . In order to do so we need to convert the search space into a representation that can be used with the template T . To that end we need an approach that converts the representation of the objects in the image from binary shapes to a graph that can be traversed and evaluated. The representation must be reversible, that is, upon localization of the template elements in the graph we must be able to convert graph nodes to the original shape.

The concept of anchor points [57] is very helpful to obtain the template elements. Anchor points are defined as a collection of non-removable pixels during the thinning process. In [57] a set of anchor points (RCDM) based on a reduced set of *center of maximal disks* (CMDs) is retrieved for reversible skeletonization.

Our approach is based on retrieval of a set of CMDs followed by the reduction of the set in such a way that the global characteristics of the shape still can be described while the search space is drastically reduced. We adapt the basic method to our needs.



Figure 3.6: *Euclidean Distance transform D* of the binary image from Figure 3.3. An inverted lookup table is used, therefore the lighter pixels correspond to low DT function values (closer to the background) and dark pixels correspond to high values (further from the background).

Compute Distance Transform

The Euclidean Distance Transform is a tool common in shape matching and doing measurements related to distance and CMD retrieval [67, 5, 13] and therefore it lends itself perfectly for the problem at hand. The Euclidean Distance Transform (EDT, or DT for short here) is a generally used operator which stands for the calculation of the smallest Euclidean distance from each pixel to a region of interest (cf. [15, 5, 14]).

As input image we use the binary image mask I_b . As output we construct an image that will represent a DT map which will denote the radius of each region (step 1 in Figure 3.4). The resulting Distance Transform has the same dimensions as I_b and is denoted as D . The value of each pixel in the DT image corresponds to the distance to the background d . The result of a DT applied to the grayscale image in Figure 3.3 is shown as in Figure 3.6. In this image the locations of a larger inscribed circle for I_b have lighter color.

Convert Distance Transform to Anchor Regions

Now from the DT the Anchor points are obtained (step 3 in Figure 3.4); the DT is reduced to a set of anchor points through repetitive dilation of region in D . This object will be called D_{best} .

We set the starting $d_i = r_{min}$ (smallest radius of the object we are looking for) and increase by $d_{i+1} = d_i + \delta$ while $d_i \leq r_{max}$ with stepsize $\delta = 1$. We choose $\delta = 1$ in order to simplify the calculation, as a consequence the resulting anchor points may have a thickness of more than one pixel and might not be accurately reversible. In our case, the representation is used for recognition of the general outline of the object during the template matching step and complete reversibility is not required. Note that the anchor points in this case will be represented as clusters of non-removable pixels rather than one-pixel points. Therefore we prefer the term *anchor regions* instead.

For each iteration the following is done. First we threshold D with distance value d_i . The resulting image $D[i]$ will contain only the regions that were created with element width r_i . For each pixel at $D(x, y)$ the following is done:

$$D[i](x, y) = \begin{cases} 1 & D(x, y) = d_i \\ 0 & otherwise \end{cases} \quad (3.2.2)$$

$$D_{best}(x, y) = \max(D[i](x, y), D_{best}(x, y)) \quad (3.2.3)$$

Now, D_{best} contains only the largest radius regions.

To retrieve the approximation of the shape that is covered by the retrieved radius region we perform backtracking, therefore $D[i]$ is subsequently dilated with a circular structuring element E_i with a radius d_i . The result is subtracted from D in order to prevent evaluating at the same area in the next iteration:

$$D' = D - (D[i] \oplus E(d_i)) \quad (3.2.4)$$

Each result is the input to the next iteration. At completion of iterations (stop criterion defined as $d_i \leq r_{max}$) D_{best} will contain only the global anchor regions a_0, a_1, \dots, a_n . In Figure 3.7 the resulting D_{best} is depicted.

In order to define each anchor region a from D_{best} in such a way that it can be used as a node in a graph, an anchor region a is represented in the following manner: $a(d, F, c)$, where:

- d is the radius of the circular structuring element belonging to the anchor region. For each a we store the $d(a) = d_i$ that was used for its creation.
- F is the shape representing the anchor region. It is a collection of pixels that contribute to the anchor region shape.
- $c(c_x, c_y)$ is the anchor point, which also is the center of mass of the shape.

The overview of the entire process is given in the following pseudo code fragment:

1. $d = r_{min}$
2. $i = 0$
3. **while** $d \leq r_{max}$ **do**



Figure 3.7: D_{best} : a collection of anchor regions a_0, \dots, a_n selected from the Distance Transform of example in Figure 3.3. d for each a is encoded as a grayscale value. Image saturation contrast has been improved for visual purposes.

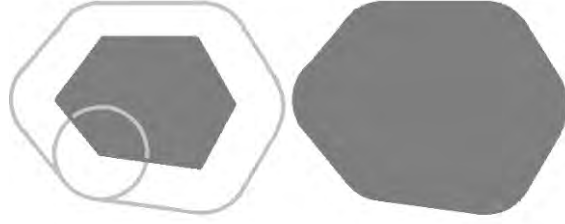


Figure 3.8: Example of a dilation process of an anchor region by a circular structuring element. Dark area to the left is the anchor region being dilated, resulting in the dark area to the right.

```

4.    $i = i + 1$ 
5.    $d = d + 1$ 
6.   for  $x = 0$  to  $ImageWidth$  do
7.     for  $y = 0$  to  $ImageHeight$  do
8.       if  $D(x, y) = d$ 
9.          $D[i](x, y) = 1$ 
10.      else
11.         $D[i](x, y) = 0$ 
12.      end for
13.    end for
14.     $D_{best}(x, y) = \max(D[i](x, y), D_{best}(x, y))$ 
15.     $D = D - (D[i] \oplus E(d))$ 
16.  end while

```

Determine Undefined Space

Now that we have determined all anchor regions in D_{best} we can consider the area it represents. A consequent dilation of every anchor region a_k will result in such area. A graphical example of a dilation process of an anchor region is shown in Figure 3.8.

Taking Figure 3.7 as an example, if we perform a dilation of each a_k with a circular structuring element with radius $d(a_k)$ this will result in a configuration as depicted in Figure 3.9:

$$F(a_k) \oplus E(d(a_k)) \quad (3.2.5)$$



Figure 3.9: Dilation of every anchor region shape in D_{best} from Figure 3.7.

The result obtained by application of Eq. 3.2.5 captures the original shape in a global manner but is missing out on the small details; more specifically, objects that have $d < r_{min}$ are discarded.

This error is introduced through the usage of non linear filters. For retrieving the global shape of the objects this does not give any problem.

However, since we need to reason about the retrieved regions, we save all these regions as undefined regions U (step 2 in Figure 3.4). The region U can be determined by a subtraction of the dilated D_{best} from the original binary image:

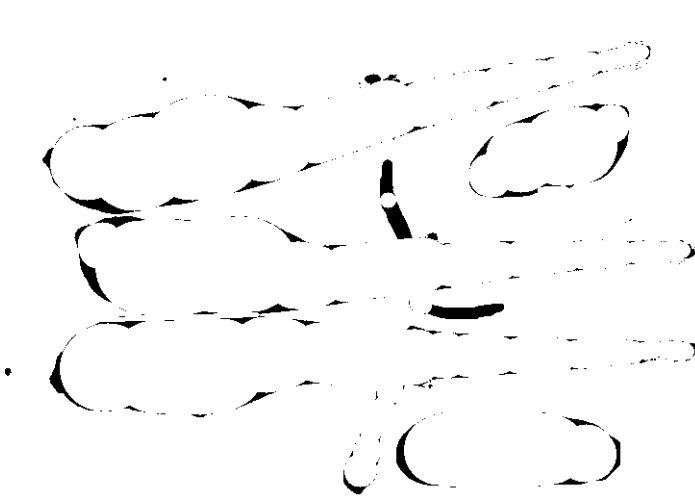
$$U = I_b - (D[i] \oplus E(d_i)) \quad (3.2.6)$$

for all i . For the example in Figure 3.10 the resulting undefined region image is depicted.

Connecting Nodes

The next step consists of creation of an undirected graph G connecting the anchor regions (step 4 in Figure 3.4). This is accomplished by connecting all neighboring anchor regions through their centers of mass.

We define a_m to be a neighbor of a_n if the elements of the line from the center of mass of a_m to the center of mass of a_n ($c(a_m)$ to $c(a_n)$) are entirely within regions belonging to the dilated sequences of these anchor regions (or nodes). An example of this neighboring definition is shown in Figure 3.11. This definition is comparable to the definition of Voronoi (or Delaunay) neighbors: if two Voronoi regions share a boundary,


 Figure 3.10: Undefined space U .

the nodes of these regions are connected with an edge. In our case the regions themselves can not be seen as Voronoi regions, however the neighboring property is the same.

In order to handle the undefined space the previous rule is extended as follows. We define a_m to be a neighbor of a_n if the elements of the line from $c(a_m)$ to $c(a_n)$ are entirely within regions belonging to the dilated sequences of these anchor regions (or nodes) or to undefined regions, cf. Eq. 3.2.7. An example is shown in Figure 3.12.

$$\overline{c(a_m)c(a_n)} \in F(t_m) \oplus E(d(a_n)) \cap F(a_n) \oplus E(d(a_n)) \cap U \quad (3.2.7)$$

In Figure 3.13 we show G as it is created for the example of the zebrafish larvae.

The rule as described in Eq. 3.2.7 is always applicable for neighbor definition except in pathological cases, for example due to poor image quality.

Convert Anchor Regions to Template Elements

Since a path in G is a sequence of anchor regions these need to be converted to template elements. One template shape element t_i can consist of multiple anchor regions. To that end a collection of anchor regions $[a_0..a_n] \in t_i$ if:

- All anchor regions are of the same radius as the template element:
 - for all i with $0 \leq i \leq n$, $d(a_i) = r_i$
- The length between the start and the end of the anchor region chain fits the template element length definition:

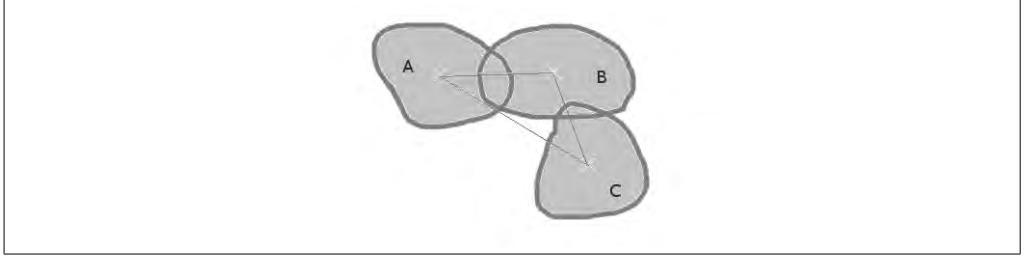


Figure 3.11: Consider the areas A , B and C . Elements of the line from $c(A)$ to $c(B)$ are entirely within regions A and B . This makes A and B neighbors. Elements of the line from $c(B)$ to $c(C)$ are entirely within regions B and C . This makes B and C neighbors. Some elements of the line from $c(A)$ to $c(C)$ are not within regions A and C . A and C are therefore not neighbors.

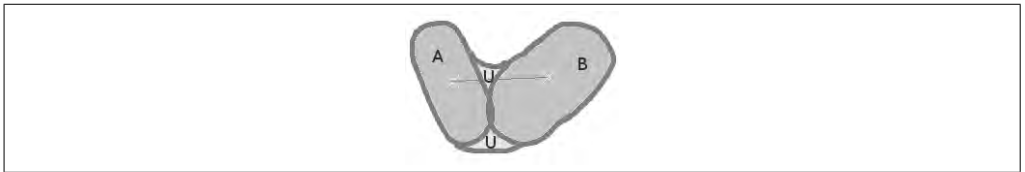


Figure 3.12: Consider the areas A , B and U . U represents the undefined regions. Elements of the line from $c(A)$ to $c(B)$ are not entirely within regions A and B . However, elements of the line from $c(A)$ to $c(B)$ are all within regions A , B and U . A and B are therefore treated as neighbors.

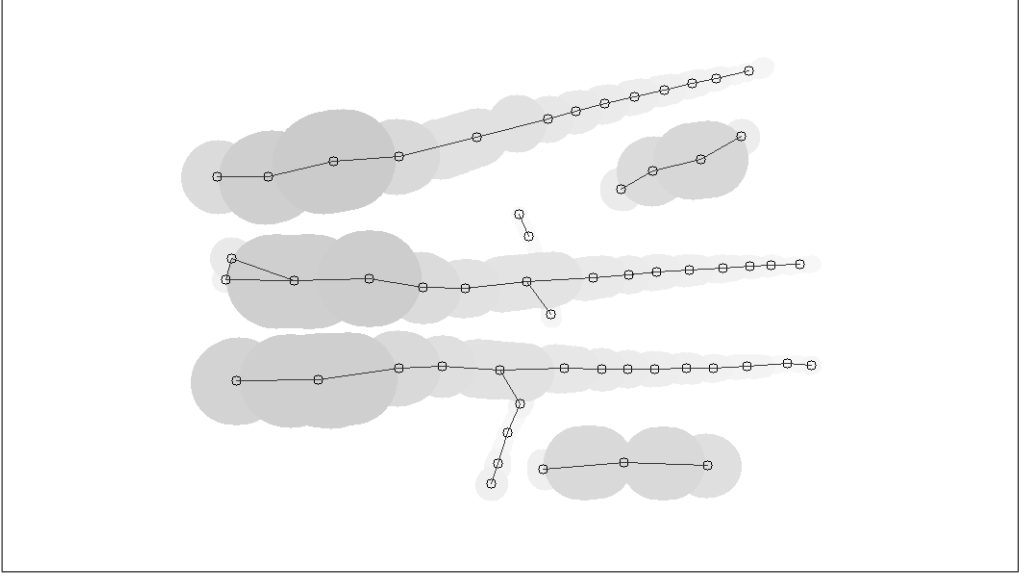


Figure 3.13: Graph G depicted on image containing the dilation of D_{best} .

$$-l_{min}(i) \leq \overline{c(a_0)c(a_1)} + \overline{c(a_1)c(a_2)} + \dots + \overline{c(a_{n-1})c(a_n)} \leq l_{max}(i)$$

Now that we have a conversion approach for anchor regions to template elements we can assume that graph G consists of template elements t .

3.2.3 Bayesian Formulation

In the graph G , representing the input image, we will search for the presence of the prototype template. Therefore G is traversed. An optimal solution determines the location of the best matching shape.

An optimal solution is defined as one with the highest probability being a deformed instance of the prototype template. The probability of a shape T_i being present in an image is expressed as: $P(T_i|G)$:

$$P(T_i|G) = \frac{P(G|T_i)P(T_i)}{P(G)} \quad (3.2.8)$$

Let δ be any deformation of the prototype template T_0 . Then, according to [60] $P(G|T_i)$ can be expressed as marginal probability sum of all probabilities for a deformation to happen joint with the probability for its validity $P(G, \delta_i|T_i)$. This sum is approximated by $\int P(G, \delta_i|T_i)d\delta$ and can be rewritten in the following way:

$$P(G|T_i) = \int P(G, \delta_i|T_i)d\delta = \int P(G|\delta_i, T_i)P(\delta_i|T_i)d\delta \quad (3.2.9)$$

Then, in order to maximize the probability that given graph G we can identify our template T in it can be related to the maximum of the following expression:

$$P(G|\delta_i, T_i)P(\delta_i|T_i) \quad (3.2.10)$$

$P(G|\delta_i, T_i)$ represents the *likelihood*. $P(\delta_i|T_i)d\delta$ is the *prior probability* of a deformation. We can reduce the problem to minimizing external and internal energy [60], related respectively to likelihood and prior probability:

$$E_{ext} = -\log P(G|\delta_i, T_i) \quad (3.2.11)$$

$$E_{int} = -\log P(\delta_i|T_i) \quad (3.2.12)$$

This yields a total energy function that needs to be minimized being $E = E_{ext} + E_{int}$.

Likelihood

The likelihood is a measurement of the similarity between an underlying object in G and the deformed template T_i in the image [24].

A deformed object of interest is a path in graph G that contains all the template shape elements that are present in T_0 (and these elements are encountered in the same sequence).

A solution s is an ordered list of template elements $[t_0, \dots, t_k]$. Cases where $s = [t_0, \dots, t_k] \notin T_0$ are not taken into consideration as a valid solution, therefore

$$P(G|\delta_i, T_i) = \begin{cases} 0, & \text{if } s \notin T_0 \\ 1, & \text{otherwise} \end{cases} \quad (3.2.13)$$

For the zebrafish test case we assume that its basic structure should always be the same. Therefore we do not introduce any fuzzy classifier for the external energy and only consider the cases where $P(G|\delta_i, T_i) = 1$. Then energy function that needs to be minimized becomes:

$$E = E_{int} \text{ if } s \notin T_0 \quad (3.2.14)$$

Prior Probability

Prior probability [24] is model driven and is the probability of a deformed T_i matching the original T_0 . We provide a prior that is applicable to the zebrafish larva, being the main test case of this chapter, based on the following assumptions:

- The template, as described in Eq. 3.2.1 assumes a regular (healthy) zebrafish is straight. Bended instances are allowed, but the straight ones should be preferred.

- The template, as described in Eq. 3.2.1, is represented in such a way, that shorter instances could be (mis)interpreted as correct matches, while being a subset of the correct match. For example a zebrafish match with a short tail can be considered a solution, while the tail actually is longer. In order to prevent this we need to prioritize the solutions that represent longer shapes.

As a result of these assumptions we construct the following priors:

- We favor the less deformed shapes over the heavily deformed ones. For two sequential nodes t_a and t_{a+1} we check the angle $\theta_{a-1,a+1}$ between nodes as compared to their predecessor node t_{a-1} :

$$\theta_{a-1,a+1} = \arctan(c(t_{a-1})_x - c(t_a)_x, c(t_{a-1})_y - c(t_a)_y)(180/\pi) - \arctan(c(t_a)_x - c(t_{a+1})_x, c(t_a)_y - c(t_{a+1})_y)(180/\pi) \quad (3.2.15)$$

In order to consider the less deformed solutions we look at:

$$\theta_{largest} = \max(\theta_0 \dots \theta_{k-1}) \quad (3.2.16)$$

We prioritize the candidates with the lowest $\theta_{largest}$. This means the prior probability increases as $\theta_{largest}$ decreases.

- Additionally solutions that represent longer shapes need to be prioritized. Length of a solution l is represented as the sum of distances from anchor regions t_a to t_b mass centers:

$$l = \sum \overline{c(t_a)c(t_b)} \quad (3.2.17)$$

We prioritize the candidates with the highest l . Therefore prior probability increases when the solution path length increases.

To that end we set the internal energy to:

$$E_{int} = e^{-ml} \cdot e^{-k(1-\theta_{largest})} \quad (3.2.18)$$

with m, k normalizing constants. In Eq. 3.2.14 we state to consider only cases where likelihood equals 1 therefore $E = E_{int}$.

3.2.4 Finding an Optimal Solution

The Bayesian approach leads to a list L_{all} that contains all solutions in the dataset. The following rules are considered while looking for optimal solutions:

- Create a list L_{best} that will contain the optimal solutions
- Iteratively select the solution s with lowest E from L_{all} . To prevent cycles, solutions with multiple instances of the same node are not considered.
 - Save s to L_{best}
 - Remove s from L_{all}

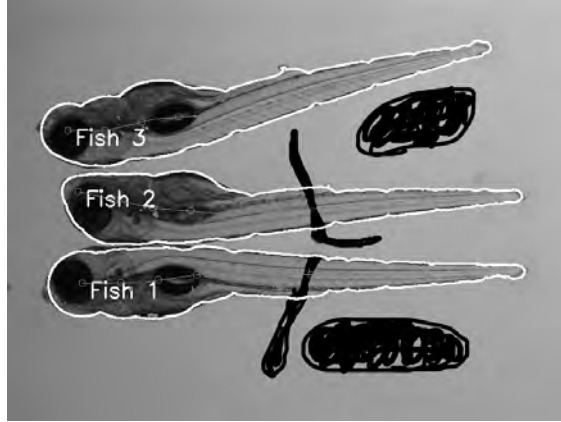


Figure 3.14: A visual representation of retrieved zebrafish larvae. Detected shapes are delineated with white.

- Remove all solutions that share at least one node with s from L_{all}
- Continue looking for solutions, while L_{all} is not empty (contains solutions)

The global shape of the objects can now be selected from the solutions in L_{best} . The shape is then retrieved by applying a dilation to the anchor regions.

The shape is not exact as it was an approximation derived from a distance transform and dilation. In order to accurately retrieve our objects we miss out on the small objects in the image at the locations where the opened regions are overlapping each other. These small objects appear because opening does preserve the main geometric features, but excludes features that are smaller than the structuring element radius. These small objects (or artifacts) do remain in the undefined space U . Therefore we add elements from U to the solution under evaluation. In order to know which elements to add we select all elements that are the neighbors (cf. Eq. 3.2.7) of the retrieved regions (step 5 in Figure 3.4).

In our example the optimal result with three best solutions is shown in Figure 3.14. Note that the algorithm correctly annotated the three zebrafish shapes, while the artifacts were not included.

3.3 Case Study 1: Zebrafish infection analysis

We have applied our approach for shape retrieval and annotation in a case study regarding the spread of induced bacterial infection within the zebrafish larvae. In Chapter 5 a more detailed description is given.

To better understand the infection progression large quantities of zebrafish larvae images needed are infected and after a few days of infection images are taken. All these images need to be analyzed. Until recently, these images were analyzed manually. Of

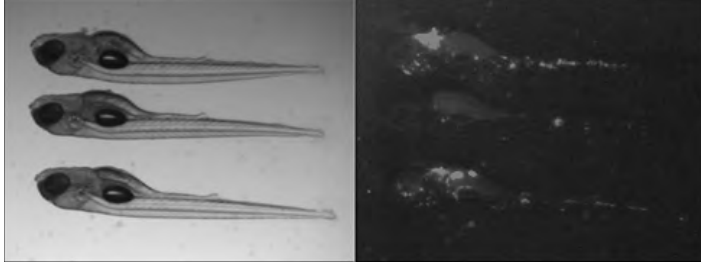


Figure 3.15: A brightfield image (in grayscale representation) and a matching fluorescent image (in grayscale representation). Images were taken with Leica DC500 microscope.

each location containing the zebrafish larvae two images were taken: one brightfield image in order to establish the location of the larvae, and one fluorescent image containing only location of the infection. In Figure 3.15 example of such an image pair is shown.

The analysis includes localization of the zebrafish shape in the brightfield image and quantification of the granuloma cluster size and spread from the fluorescent images. This enables the necessary HT automated approach.

For this case study we use a comparison of infection pattern of the wild-type of the *Mycobacterium marinum* (MM) that made the zebrafish sick with a mutant strain (714M) that induced less infection [42].

A brightfield image is used for finding and annotating the zebrafish shape. Additionally, locations of subareas can be estimated. We annotate the zebrafish larvae shape based on the prototype template representation as shown in Eq. 3.2.1. By using this template as prototype the algorithm estimates the location of the head, body and tail region of the zebrafish larvae.

3.3.1 Zebrafish Orientation

The template we have used for the zebrafish does not discriminate between the top and the bottom of the zebrafish. Therefore an additional step is needed to retrieve the orientation of the zebrafish larvae shape; this is necessary for a good assessment of the infection.

In order to be able to determine that the following approach is suggested. We localize all the convex deficiency locations. The bottom of a zebrafish shape will usually have larger convexity defects than the smooth top. If the largest convexity defect is then located below the shape path we assume the fish is on its belly, otherwise its on its back.

The annotation allows us to retrieve some other characteristic locations within a shape. For example the heart of the larva and the point at which usually injections are performed can be not automatically retrieved. This could be done by defining the locations of these points of interest in the prototype template. For example, the heart is located between the head and the body nodes at the bottom of a larva. The injection point is located

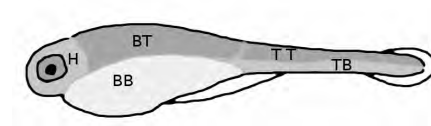


Figure 3.16: Graphical representation of the separation of a zebrafish into regions.

approximately halfway the tail area. An automatic annotation of these locations can then be performed as we know the locations of the head, body and tail regions and we have obtained the shape orientation from our analysis.

3.3.2 Zebrafish Region Annotation

Now for the assessment of the infection, head (H), body bottom (BB), body top (BT), tail bottom (TB) and tail top (TT) are defined. From the position assessment the orientation of the zebrafish could be easily retrieved. This made it possible to divide areas into the *top* and *bottom* part. Separation of these regions was based on [63].

The estimated location of these areas is shown in Figure 3.16. After annotation the found regions were masked with the fluorescent image of the same fish (cf. Chapter 4) in order to define infection location and amount.

3.3.3 Data

For this case study 189 zebrafish larval shapes were retrieved and analyzed. The images are the input for the analysis framework which consisted of two steps. First the zebrafish are localized by our algorithm and then used as a mask for the fluorescent images. Infection spread and size is saved to a comma separated file. In Figure 3.17 and Figure 3.18 an example of the result is shown. Based on this annotation the infection patterns were established and described in more detail in Chapter 4

3.3.4 Analysis and Results

In our case study we set out to analyze the relationship between mutant and wild-type (MM) in the amount of clusters, spatial distribution and the cluster size. The results presented here followed from an in depth statistical analysis that is described in Chapter 5.

In all regions of the zebrafish the wild-type gives a higher area of infection compared to the 714M; overall the infection area is 5 times larger. In the head region the same percentage of the granulomas in both MM and 714M are located. In the body bottom region a higher percentage of the granulomas of the MM then 714M are located. In the tail bottom region a higher percentage of the granulomas of the 714M then MM are located. A graphical representation of these results is shown in Figure 3.19. In all regions of the zebrafish the MM gives a higher amount of clusters then the 714M; overall the amount of clusters is 3 times higher. The spread of the clusters is not significantly



Figure 3.17: Output for a single image as created by the our algorithm. The found shapes are denoted with red line. The purple area indicates the presence of granuloma formation at the current location. Optimal graph G is printed within each shape. Blue lines represent node connections that were classified as tail area, green lines as body area and red as head. This image is created in an automated fashion.

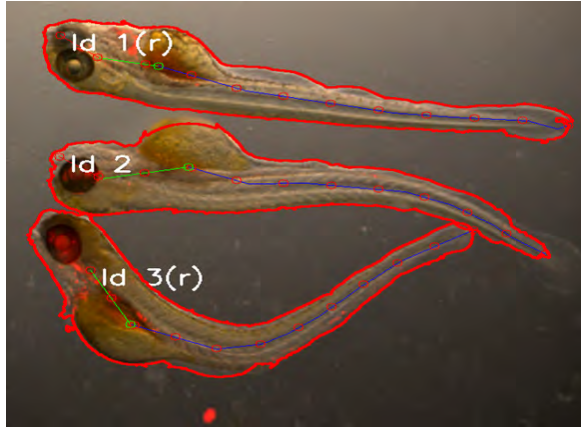


Figure 3.18: Output for a single image as created by our algorithm. The found shapes are denoted with red line. Note that the shapes in this example are heavily deformed, yet the shapes were found and annotated correctly. This image is created in an automated fashion.

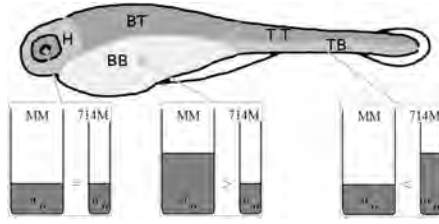


Figure 3.19: Graphical representation of the amount of infection (see Chapter 5 for more information on these infographics). While there is always more MM infection than 714M, the percentages of their presence in certain regions are different.

different for MM and 714M. In this manner we can derive statements on the behavior of mutant strains.

3.4 Case Study 2: Other Model Organisms

For an additional test of the algorithm we test the possibility to separate different shapes/organisms in the same image. An experiment was set up with two different shapes. We have collected images containing both *Xenopus laevis* and *Danio rerio*. In early stages (20-30) the *Xenopus* has approximately a length in the same order of magnitude as a zebrafish embryo in larval stage. However, the shape of the organisms is different: the tail of the *Xenopus* has the same width as its body, which is different from the zebrafish. In Figure 3.20 we present an image containing the zebrafish in larva stage and the *Xenopus* in stage 26.

We have used Figure 3.20 as input for our algorithm with a prototype template $T_{\text{zebrafish}}$ as presented earlier in this paper. As result the zebrafish larva was found while the *Xenopus* shape was rejected as a solution. The result is shown in Figure 3.21. As can be seen the non zebrafish shapes were not selected, while zebrafish shapes were annotated.

In order to be able to find the *Xenopus* we have adapted the prototype template to the following:

That is, the r of the tail of the *Xenopus* should in general be close or equal to its body thickness (biggest part of the tail should be or at least 70% of it):

$$T_{\text{xenopus}} = \begin{cases} [t_0 t_1 t_2] & \text{sequence} \\ r_0 \geq 0, r_0 < r_1, r_2 > 0, r_2[\text{biggest}] > 0.7r_1 & \text{radius relationship} \\ r_{\min} < r_i < r_{\max} & \text{radius limits} \\ (l_0 + l_1) < 0.5 * (l_0 + l_1 + l_2) & \text{length relationship} \\ l_{\min} < (l_0 + l_1 + l_2) < l_{\max} & \text{length limits} \end{cases} \quad (3.4.1)$$



Figure 3.20: Brightfield image of a zebrafish in larval stage (on top of the image) as well as the *Xenopus* in stage 26. Image was provided by N.Bardine.



Figure 3.21: Results of applying or algorithm to an image of a zebrafish larva and the *Xenopus* larva with a prototype template $T_{0_{zebrafish}}$.



Figure 3.22: Results of applying or algorithm to an image of a zebrafish larva and the Xenopus larva with a prototype template $T_{0_{xenopus}}$.

Results of the segmentation are shown in Figure 3.22. As can be seen with this template the Xenopus shape is annotated, while the zebrafish shape was ignored.

To show that our algorithm is independent in scale and size and only shape based we also run the algorithm for an image that is containing the zebrafish larva and the Xenopus in a later stage of development (see Figure 3.23). In stage 45 Xenopus shape is very similar to the zebrafish and its shape has approximately the same characteristics. The difference, however is the size, as the Xenopus in this stage is several magnitudes bigger then the zebrafish.

Results of the segmentation are shown in Figure 3.24. As can be seen both the Xenopus as the zebrafish are annotated. This is due to the fact that their shapes are similar and thus both fit the prototype template of the zebrafish larva regardless of their size.

3.5 Conclusions

The algorithm is robust; it can deal with variation in position and orientation which makes it a perfect candidate to use for the complex shapes and scenes as apparent in life sciences: if no special instrumentation is used the location and orientation will not be known.

We have successfully used the algorithm in a case study for HT of automatic recognition of zebrafish larva, specifically for in depth analysis of granuloma cluster spread. From statistical analysis of the output data we gained insight on the distribution pattern



Figure 3.23: Brightfield image of a zebrafish in larval stage (on top of the image) as well as the *Xenopus* in stage 45. Image was provided by N.Bardine.



Figure 3.24: Results of applying or algorithm to an image of a zebrafish larva and the *Xenopus* larva with a prototype template $T_{0_{zebrafish}}$.

3 Anchor Region Based Pattern Recognition in Image Space

of the wild-type strain and how it was different from mutant 714 strain.