Image analysis for gene expression based phenotype characterization in yeast cells

Tleis, M.

3

Hough-based Contour Extraction and Optimization

“In this chapter, we present our novel algorithm for the segmentation of ovoid objects. Our method implements a variation of the Hough Transform and Minimal Path algorithms. We propose equations and parameters to control the detection of objects through Hough Transform. The contours of these objects are extracted through minimal path algorithms implemented on a polar resampled representation of the object images. In addition, we present a contour expansion algorithm using the same polar representation of object images. Such expansion is necessary under some microscope settings.”

This chapter is based on the following publications:

- Mohamed Tleis and Fons J. Verbeek. "Extracting contours of oval-shaped objects by Hough transform and minimal path algorithms." In Sixth International Conference on Digital Image Processing, pp. 915903-5. International Society for Optics and Photonics, 2014. *(Excellent Paper Award)*.
- Mohamed Tleis and Fons J. Verbeek. "Contour Expansion preceded by the Application of Hough transform and Minimal Path Algorithm". In Fifth International Conference on Image Processing Theory, Tools and Applications, pp. 149-154. IEEE, 2015.

3.1 Background

In Chapter Two, the *Hough* transform based segmentation methods and contour optimization were introduced. This chapter explains in details the underlying algorithms of such methods and their implementation. The following section introduces *Hough* transform, generation of the cube-like accumulator, and how this accumulator is threshold in order to get information about the object locations in the image. In Section 3.3 polar transformations are described and how these are applied to ovoid objects to generate a rectangular array that is used to obtain the minimal path in the object image. In Section 3.4, the implemented minimal path algorithms are explained. The polar representation of object images is also used for the purpose of contour optimization; our novel optimization method to expand object contours is explained in Section 3.5. Section 3.6 validates the introduced methods by comparing them to other state-of-the-art solutions and assess their performance under various noise levels.

3.2 Hough Transform

In the daily practice of processing microscope images, sets of pixels yielded by edge detection methods seldom characterize the edge because of noise, breaks in the edges due to non-uniform illumination and the effects that introduce spurious discontinuities in intensity values. This is often observed with the *S. cerevisiae* cells in bright-field images.

A well-known global approach to edge linking is *Hough* transform [Gon08]. *Hough* transform is applicable to any function of the form $g(v, c)$ where v is a vector of coordinates and c is a vector of coefficients. Since many objects in systems and micro-biology such as *S. cerevisiae* yeast cells are characterized as having nearly ovoid shapes [Fel10], detecting circular arcs in yeast images would help in detecting the yeast cells. A circle is represented as in Eq. 3.1, which can be rewritten in its normalized form as depicted in Eq. 3.2.

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2 \tag{3.1}$$

$$\begin{aligned} x &= r.\cos\theta \\ y &= r.\sin\theta \end{aligned} \tag{3.2}$$

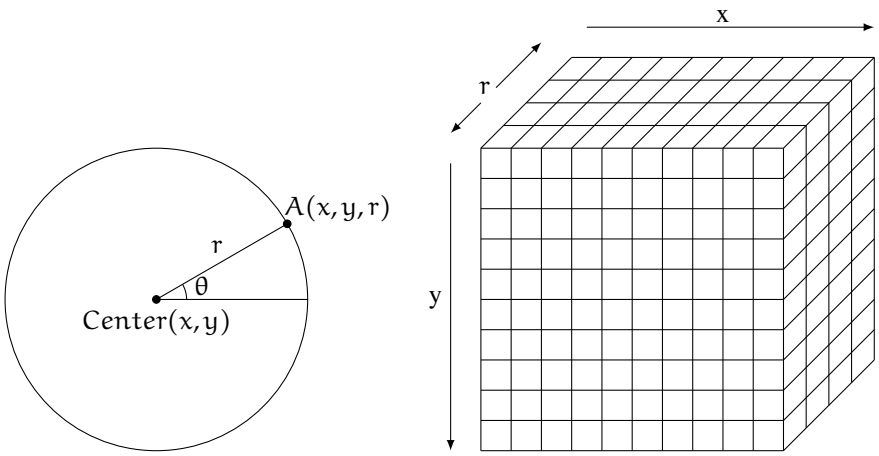
Our implementation of the *Hough* transform includes two steps. The first step is to fill the accumulator and the second is to threshold this accumulator in order to extract the circles corresponding to the estimated object locations. These steps are discussed hereafter.

3.2.1 Filling the Accumulator

Figure 3.1(a) illustrates the geometrical interpretation of the parameters x, y, θ and r. The x, y, r parameter space is sub-divided into accumulator cells forming a 3-D cube-like cells and accumulator of the form $A[x, y, r]$. The x and y dimensions of this accumulator are related with the width and height of image I and the r dimension is defined by the constraint of the possible object radius values. This accumulator is illustrated in Fig. 3.1(b). The procedure is to increment x and y and solve the equations in Eq. 3.2 for θ at a pre-specified value of the radius r and update the accumulator cell associated with the triplet (x, y, r), i.e. increment accumulator cell by 1 if a foreground edge pixel is found at angle θ for the circle of center (x, y) and radius r. The flowchart in Fig. 3.2 illustrates the filling process in more detail. At radius r specified from a range of radii, all the pixels in the image are checked and if a certain pixel is an edge pixel, i.e. foreground in the binary image, the accumulator cell is incremented by the value of one. After filling the *Hough* accumulator array, we need to decide on which pixels in the binary image are to be considered as center of circles. This required specifying a rule to find a threshold for the accumulator. The following sub-section discusses our approach to find this accumulator threshold.

3.2.2 Accumulator Threshold

Hough values in the accumulator array are related to the number of pixels located in the circumference of a certain circle with a specified center c and radius r where a maximum of $2\pi r$ pixels can belong to that circumference. A threshold value is set to allow detection of structures that lie a pre-specified percentage of the circumference. This threshold starts at $2\pi r$ and decrements for smaller values of the radius. The threshold value is calculated according to



(a) Geometrical interpretation of parameters x, y ,θ and r.

(b) Cube Accumulator $A[x, y, r]$.

Figure 3.1: *The Hough Accumulator.*

the equations depicted in Eq. 3.3 and 3.4, where r is the radius of the circle, r_{max} and r_{min} are the maximum and minimum specified radius values, r_{index} is an index to track the current radius value. α and β are control values. Parameter p in Eq. 3.4 ensures that the threshold value is relatively high for smaller circles. This allows detecting some deformed shapes at a smaller radius $(r-1)$ if not detected at radius r.

$$T = 2\pi r - \{2\pi r \times \alpha + p\} \tag{3.3}$$

$$p = \beta \times (r_{max} - r_{min}) - r_{index} \tag{3.4}$$

The flowchart in Fig. 3.3 walks-through the process of using the threshold value to get the geometrical arcs that form part of circles in the binary image. Starting from the largest radius and decrementing until the minimal radius, we loop through *Hough* values in every cell of the cube accumulator starting from the maximum *Hough* value until the specified threshold given in Eq. 3.3. As long as there are pixels associated with that *Hough* value, we check if the circles occupying that space overlap with the previously detected circles. If not, then the space occupied by the new circle is reserved, preventing subsequent circles from occupying this space. The parameters α and β control the threshold equation depicted in Eq. 3.3 and 3.4.

The parameter α is a weighting factor in the range $[0,1]$ and is used to determine the percentage of pixels that must exist as edge pixels in order to consider them as being part of a closed circumference of a circle; e.g. at $\alpha = 0.5$, at least 50% of the pixels located on the circumference must be edge pixels. The parameter β acts similar to α except that it increases in value at the examination of circles with lower radius values, allowing more shape deformation at lower radii.

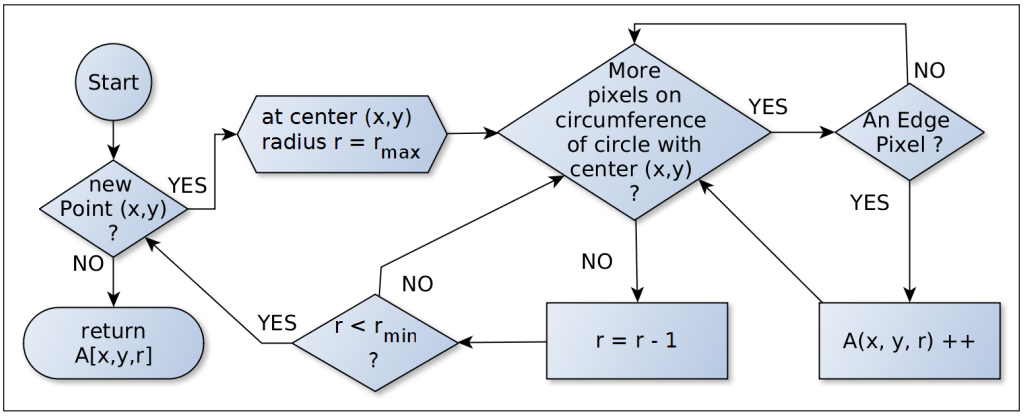


Figure 3.2: *Filling the Hough Accumulator.*

3.3 Polar Transformation

After estimating the initial position of the ovoid objects such as *S. cerevisiae* yeast cells in bright-field microscope images through *Hough* transform, our objective is to extract the exact contour of that object. For this, we resample each object into a polar representation for each object where a polar transformation is applied on the image part as a *RoI* for that object. On this polar image, we extract the minimal path corresponding to the actual contour of the object. In this section, we explain the Polar Coordinate system? How the transformation is achieved from cartesian to polar system? And how is that applied on actual objects such that *S. cerevisiae* cells? The minimal path algorithm is discussed later in Section 3.4.

3.3.1 Polar Coordinate system

In 2D space, the polar coordinate system is a two-dimensional coordinate system in which each point on a plane is determined by a distance from a reference point and an angle from a reference direction. The reference point is analogous to the origin of a cartesian system. It is known as the "pole", and the ray from the pole in the reference direction is known as the "polar axis". The distance from the pole is called the radial coordinate or radius and we will refer to it as r, and the angle is the angular coordinate, polar angle, or azimuth and we will refer to it as θ [Bro97].

3.3.2 Cartesian to Polar System

The cartesian coordinates x and y can be converted to polar coordinates r and θ with $r \geq 0$ and θ in the interval $(-\pi, \pi]$ using Eq. 3.5 and 3.6. The $\mathtt{atan2}$ notation in Eq. 3.6 is a common variation of the $\mathtt{arctangent}$ function as defined in Eq. 3.7. The geometrical interpretation of the relationship between the polar and cartesian coordinates is further illustrated in Fig. 3.4.

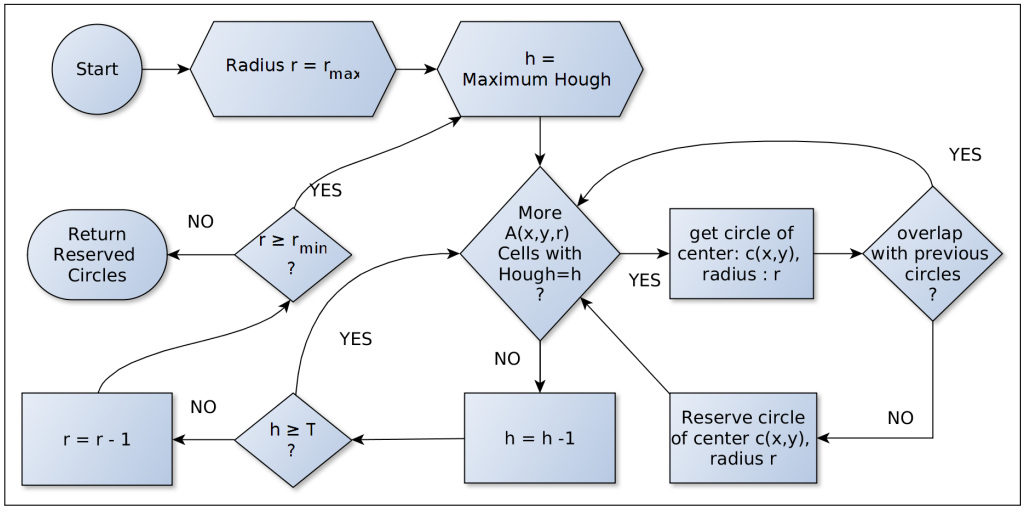


Figure 3.3: *Extracting the Circles from Image.*

$$r = \sqrt{(x^2 + y^2)} \tag{3.5}$$

$$\theta = \operatorname{atan2}(y, x). \tag{3.6}$$

$$\operatorname{atan2}(y,x) = \begin{cases} \arctan(\frac{y}{x}), & \text{if } x > 0 \\ \arctan(\frac{y}{x}) + \pi, & \text{if } x < 0 \text{ and } y \geq 0 \\ \arctan(\frac{y}{x}) - \pi, & \text{if } x < 0 \text{ and } y < 0 \\ \frac{\pi}{2}, & \text{if } x = 0 \text{ and } y > 0 \\ -\frac{\pi}{2}, & \text{if } x = 0 \text{ and } y < 0 \\ \text{undefined}, & \text{if } x = 0 \text{ and } y = 0 \end{cases} \tag{3.7}$$

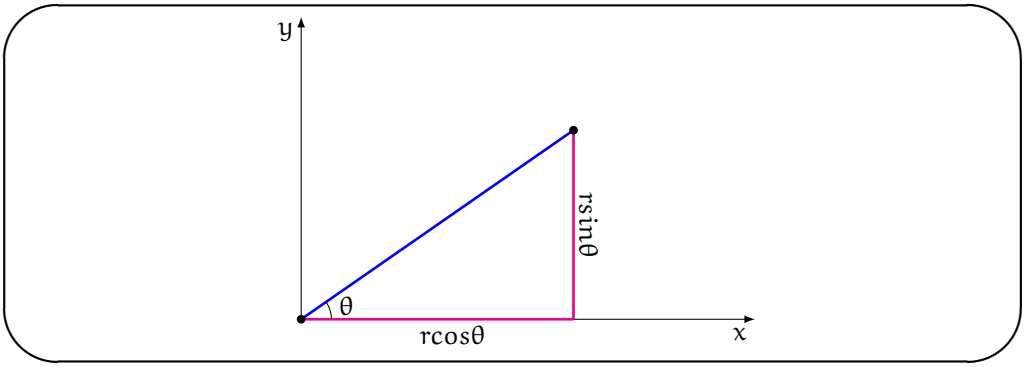


Figure 3.4: *Geometrical interpretation of relationship between polar and cartesian coordinates.*

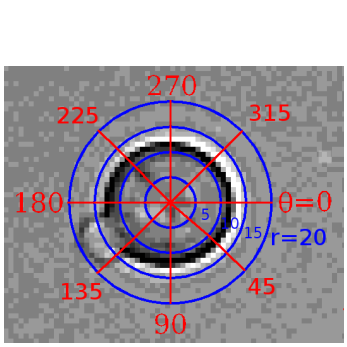
3.3.3 Polar Image of Yeast Cells

Figure 3.5(a) shows an example of a yeast cell after locating its center point by *Hough* transform. The *red* circles and *blue* lines are labels to illustrate the resampling of the polar image of the pixels surrounding the center point of the cell as shown in Fig. 3.5(b). The radius at an angle θ in the original image plane is transformed into a column in the polar image. The circles surrounding the center point, i.e. *red* circles of radius r in the original image plane in Fig. 3.5(a), are transformed into rows of the polar image, i.e. red horizontal lines in Fig. 3.5(b).

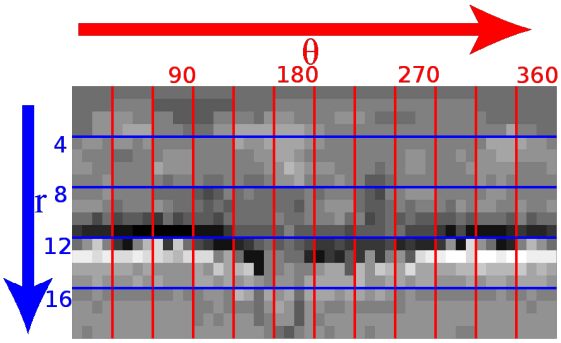
As a rule, the height of the polar image I is determined based on the scale information of the images, and a priori generic knowledge on the objects, i.e. scale × maximum radius. The width of I is dependent of the length of the contour. In order to have a sufficient region to evaluate, we use a resampling length equal to twice the height of I. The minimal path algorithm then finds the circular shortest path from the first to the last angular coordinate in image I.

3.4 Minimal Path Algorithms

Once all the center of objects, i.e. cells, are estimated as circle centers using the *Hough* transform, a polar image I relative to each circle center is resampled from the original intensity image [Kva08]. In our work, we adopted dynamic programming to extract the object contours by finding the minmal path in the polar representation of these objects. In this section, we discuss our implementation of a minimal path algorithm that uses grey weighted distance transform followed by our novel approach to extract a circular shortest path for the polar image.



(a) An example of a cell after detection of the center point by Hough transform.



(b) A polar image is generated for the yeast cell in Fig. 3.5(a). The columns correspond to the pixels along the radius at an angle θ of the largest possible circle (red lines). The rows correspond to the circles surrounding the center point (blue lines, blue circles in Fig. 3.5(a)).

Figure 3.5: *Polar transform of a yeast cell image.*

3.4.1 Grey-weighted Distance Transform

Grey-weighted distance transform algorithm was developed originally [Vin88] to find the path in grey-scale images. Let I be a polar image, which is formally a matrix of weights $C_{(r,\theta)}$; then let A denotes the first column of pixels in image I and B denotes the last column of pixels. The objective is to find the minimal path from A to B. Considering I as a neighbouring graph, let E be the edge between two neighbouring pixels p and q. E is assigned a value $V_I(E) = V_I(p,q) = I(p) + I(q)$. Let P be a path in this graph, and let $C_I(P)$ be a cost associated with path P. This cost equals to the sum of all the edge values. This provides a new metric in image I for which the distance d_I between two pixels p and q is given by: $d_I(p,q) = \min C_I(P)$, P: path between p and q. A pixel p belongs to the minimal path if and only if $d_I(p,A) + d_I(p,B) = d_I(A,B)$, therefore, the grey-weighted distance transform to A for each pixel is created by computing $d_I(p,A)$ for every pixel p, and similarly creating the grey-weighted distance transform to B. The two distance functions are added. Subsequently, a column-wise threshold is applied to the resulting image taking the minimum pixel value of the column as a threshold in order to keep the pixels p belonging to the minimal cost between A and B. This result represents the actual contour of the object. We ensure a closed contour by assigning the coordinates of the extracted pixels $p \in I$ as vertices of a polygon region of Interest (RoI), which is filled to create a binary mask representing a closed object.

Figure 3.6(a) shows a sample polar image, where the first and last columns are labelled as A and B as shown in Fig. 3.6(b). Figure 3.6(c) shows the grey-weighted distance transform to A for each pixel created by computing $d_I(p,A)$. Similarly, Fig. 3.6(d) shows the grey-weighted distance transform to B. The addition of the two distance images is shown as Fig. 3.6(e). The result of the application of column-wise threshold to obtain the minimal cost between A and B is shown in Fig. 3.6(f). This minimal path is the acutal contour of the cell. Figure 3.7 shows another example for a cell whose contour is extracted from within a group of clumped cells.

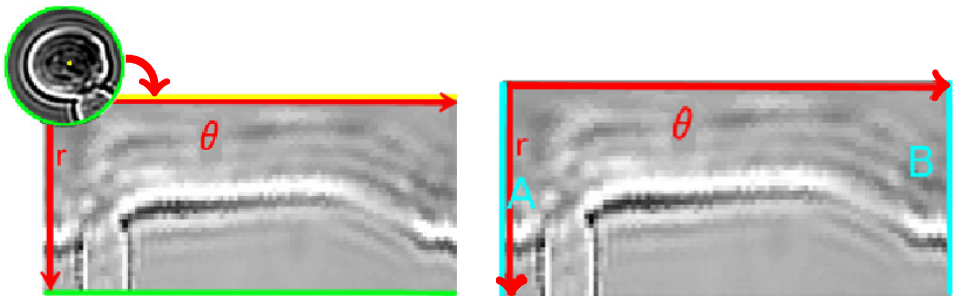
3.4.2 Circular Shortest Path

Object contours are extracted from the computation of the minimal path through distance transform as discussed in the previous subsection. However, this extracted contour is not circular as the pixels p_A and p_B of column A and B in the polar image I are actually the same pixels (at $\theta = 0$ and $\theta = 360°$). Hence, we developed an algorithm to extract the circular shortest path from column A to column B in the polar image with the constraint that a path that starts at row r_i must also end at r_i.

While the standard shortest path problem aims at finding a shortest path between two known nodes in a graph or an image grid, the shortest circular path problem is to find a closed path with minimum cost. This problem is more difficult than the standard shortest path problem since no explicit start or end node is known. Our circular shortest path algorithm uses the concept of ordinary shortest path obtained with dynamic programming [Buc97] but with the constraint that the first and last pixels p_A and p_B of the detected path P are at the same radial coordinate in order to ensure a circular path (cf. Fig. 3.8).

We apply the shortest path algorithm just once starting at the first angular coordinate $\theta = 0$ of each row r_i in image I and only evaluating the pixels p_h that would have a high probability of belonging to the path. The global minimum $\min\{C_I(P)\}$ of these shortest paths $C_I(P)$ is guaranteed to be the global circular shortest path *CSP* [App03]. Our algorithm does not require evaluating all the pixels while checking for shortest paths. It actually visits the pixels $(\frac{w^2 l}{4})$ times instead of $w^2 l$ times stated in the standard circular shortest path algorithm [Sun03], where w and l are the width and height of image I respectively.

In **Box 3.1** the pseudo-code for the circular shortest path algorithm is depicted. For each radial coordinate r in the polar image I (cf. line 2), we create a matrix $C_{(r,\theta)}$ having the same dimensions as I, i.e. $w \times l$ (cf. line 3). Ultimately all contour points need to get the status *final*.



(a) Polar representation of *S. cerevisiae* yeast cell.

(b) Labeling First and Last Column as A and B.

(c) I_A = Distance Transform to Column A.

(d) I_B = Distance Transform to Column B.

(e) Add grey-weighted distance transform to A (I_A) and that to B (I_B), i.e. $(I_A + I_B)$.

(f) Threshold $(I_A + I_B)$ in Fig. 3.6(e).

Figure 3.6: *Hough Transform and Minimal Path Algorithm to Extract Cell Contours.*

Input : A polar image I of size $w \times l$.
Integer variables i, j, i_2, k and $Cost_{min}$.
Output : Integer array CSP of length w.

Initialization: $Cost_{min} \leftarrow$ Maximum Value; $i_2 \leftarrow w-1$;
for *each radial coordinate* r **do**
  Create cost matrix $C_{(r,\theta)}$ of size $w \times l$;
  /* Initialize first column of $C_{(r,\theta)}$ to the value 1 */
  $C_{(r,\theta)}[0...l-1,0] \leftarrow 1$;
  /* For each column in the left half of the image */
  for i *from* 1 *to* $(\frac{w}{2}-1)$ **do**
    /* And for each row within the possible range R */
    for j *from* $(r - i\%\frac{w}{2})$ *to* $(r + i\%\frac{w}{2})$ **do**
      $C_{(r,\theta)}[i,j] \leftarrow I[i,j] + \min(C_{(r,\theta)}[i-1,j+k]) \quad \forall k \in [-1,1]$;
      $K[i,j] \leftarrow \underset{k}{\operatorname{argmin}}(C_{(r,\theta)}[i-1,j+k]) \quad \forall k \in [-1,1]$;
    endfor
  endfor
  /* For each column in the right half of the image */
  for i *from* $(\frac{w}{2})$ *to* $(w-1)$ **do**
    /* And for each row within the possible range R */
    for j *from* $(r - i_2\%\frac{w}{2})$ *to* $(r + i_2\%\frac{w}{2})$ **do**
      $C_{(r,\theta)}[i,j] \leftarrow I[i,j] + \min(C_{(r,\theta)}[i-1,j+k]) \quad \forall k \in [-1,1]$;
      $K[i,j] \leftarrow \underset{k}{\operatorname{argmin}}(C_{(r,\theta)}[i-1,j+k]) \quad \forall k \in [-1,1]$;
      $i_2 \leftarrow i_2 - 1$;
    endfor
  endfor
  /* Keep track of the global minimal Cost */
  if $C_{(r,\theta)}[w-1,r] < Cost_{min}$ **then**
    $Index_{min} \leftarrow r$;
    $Cost_{min} \leftarrow C_{(r,\theta)}[w-1,r]$;
  end
endfor
/* BackTrace path at row r */
$CSP[w-1] \leftarrow Index_{min}$;
for j *from* $(w-2)$ *to* 0 **do**
  $CSP[j] = CSP[j+1] + k[CSP[j+1], j+1]$;
endfor
return CSP.

Box 3.1: *Find Circular Shortest Path.*

For each angular coordinate less than $\frac{w}{2}$, we check all pixels that have a radial coordinate in the possible range $R \in [(r - i\% \frac{w}{2}), (r + i\% \frac{w}{2})]$, where i is the current angular coordinate. When a pixel meets the aforementioned conditions, its value is added to the minimum cost at the previous angular coordinate and a radial coordinate in the range of $r \pm 1$ (cf. line 7), where r is the radial coordinate of the current pixel. The distance from the current radial coordinate to the minimum cost at the previous angular coordinate is stored in an index matrix K (cf. line 8). For the right part of the image I, the same procedure is performed with the constraint that the range of R is dependent on the integer variable i_2 instead of the current angular coordinate i. i_2 decrements from $w - 1$ as i increments from $\frac{w}{2}$ to $w-1$ (cf. lines 11- 17). After computing the cost matrix $C_{(r,\theta)}$, we update the minimal circular shortest path cost $\mathrm{Cost}_{\mathrm{min}}$ by first checking if the minimal cost at the last angular coordinate is less than the global variable $\mathrm{Cost}_{\mathrm{min}}$ (cf. line 18). If this is the case, the current radial coordinate is stored as an index for the global minimal path $\mathrm{Index}_{\mathrm{min}}$ (cf. line 19). The minimum cost value is also stored by updating the global variable $\mathrm{Cost}_{\mathrm{min}}$ (cf. line 20). Now that all the radial levels are examined in the polar image I, we created a rule to extract the circular minimal path by back-tracing the pixels of this path starting from the index of the global minima $\mathrm{Index}_{\mathrm{min}}$ that will be the radial index for the last contour point (cf. line 23). As an additional rule at each current angular coordinate, the radial index of the circular shortest path is stored by checking the radial index at the next angular coordinate and adds to it the distance stored in the index matrix K (cf. line 25).

A sample application of the circular shotest path algorithm is applied on the *S. cerevisiae* yeast cell shown previously in Fig. 3.5(a). The result of this application is illustrated in Fig. 3.9(a), where the final extracted contour is displayed as *cyan* points on the resampled image. The backprojection of the contour on the original image is shown in Fig. 3.9(b).

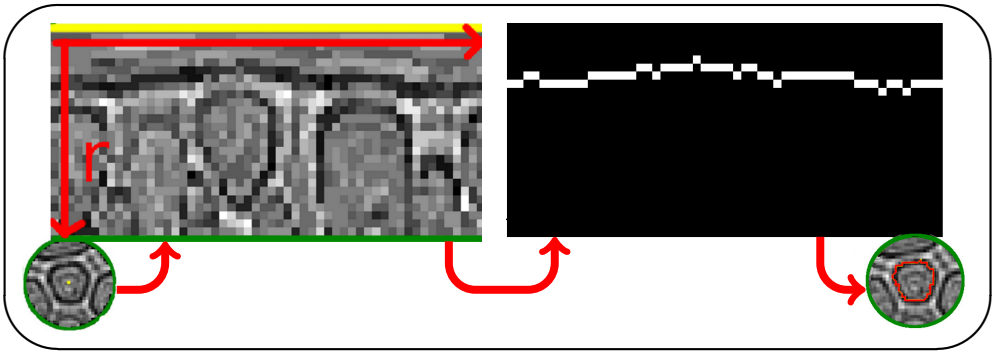
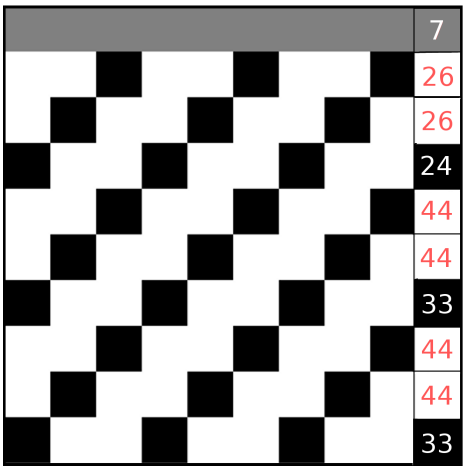


Figure 3.7: *Hough transform and grey-weighted distance transform to extract cell contours.*

3.5 Contour Optimization

Our objective is to find reliable methods for the extraction of an accurate contour delineating the object. We have shown to be successful in achieving this objective through dynamic programming (DP) [Ger86], where we used *Hough* transform and set a cost matrix where we applied a minimal path algorithm to estimate the contour location [Tle14]. We described two minimal path methods, consequently we have two segmentation algorithms. The first is the



(a) Pathological cost function from [App03]. Last column shows CSP cost for each row.

(b) Shortest path at first row with a total weight=7.

(c) Shortest path at 3rd row with a total weight=24.

(d) Shortest path at last row with a total weight=33.

Figure 3.8: *Finding the Circular Shortest Path (CSP). Black, grey and white squares represent pixel intensity values of 0, 1 and 11 respectively (similar to [App03]). Pixels marked with X are not evaluated. The path with the minimum weight is the global CSP.*

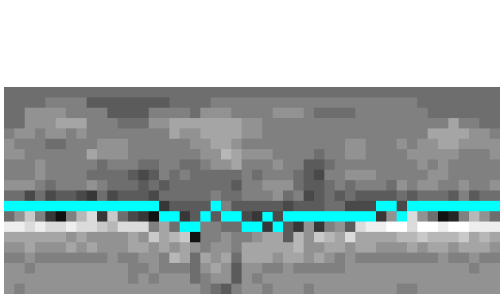
Hough transfrom followed by grey-weighted distance transform method; i.e. *HDT*, and the second is the *Hough* transform followed by circular shortest path method; i.e. *HCSP*. However, since microscope imaging can be very delicate and edges have an inherently fuzzy nature, a refinement of the initial estimate is sometimes required. For example, the objects to be segmented have contours that are described by more than one pixel thick contour line. This is typically true for the yeast image dataset that we show as a case study in this section. This problem is observed when high resolution images using high numerical aperture (NA) lenses are used with image sizes larger than 512x512 pixels. Further analysis of the results demonstrates that the circular minimal path method has a bias toward the inner part of the object as a result of the fact that some inner pixel values are lower than the edge pixel values. We state that, in these cases, the minimal path is no longer the ultimate valid representation of the contour. In this section, we use a yeast dataset for which this is typically the case.

As a possible solution to this problem, we developed an expansion algorithm that directly evaluates the polar image I used to extract the minimal path. We refer to I as a polar image, which is a polar representation of each object O_i in the original image. With our additional heuristic, we are able to achieve the necessary refinement of contours. This contributes to the accomplishment of precise measurements of the objects so that machine learning techniques can better recognize the subtle patterns within the data.

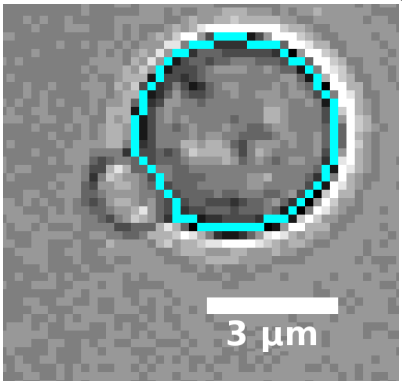
In the following subsections, we provide a background and subsequently the control parameters are described. In the last subsection we discuss the contour expansion (*CE*) algorithm.

3.5.1 Background

Our goal is to obtain the exact contours of ovoid objects; here applied to yeast cells. The initial contour detected by *HCSP* is expanded out toward the background surrounding the object until it meets certain criteria. The expansion is realized using dynamic programming in the polar image I. Starting from the initial contour detected by *HCSP* [Tle14], the contour is expanded in such a way that $r' \geq r$, where r' is the radial coordinate of the new contour pixel in image I and r is the initial radial coordinate. Every contour pixel considers a number of factors that decides



(a) Extracted minimal path from the resampled image.



(b) Contour backprojected on the original image.

Figure 3.9: *Sample application of circular shortest path on S. cerevisiae cell.*

whether it should be expanded to the next level or if it is a *final* contour point. To decide if the current contour point is a *final* contour point, we have identified three parameters; i.e. the *resistance*, *limit* and *convergence*; these parameters are discussed hereafter.

3.5.2 Control parameters

The *resistance* parameter res is a float variable in the range $res \in [0,1]$. It is the key player in this algorithm. It decides if the current contour point is a *final* point by evaluating the value of the point at the next radial coordinate. When the value of res is 0 the expansion is blocked, while at the value of 1 the pixels in I would not block the expansion. At a value of 0.5 we threshold the image I at the mean of its histogram and consequently the background pixels above that threshold value will block the expansion of the contour. At res values below and above 0.5, we would threshold I according to Eq. 3.8 and Eq. 3.9 respectively.

$$t = h_{min} + 2(h_{\mu} - h_{min}) \times res \quad \forall\ res\ \in\ (0, 0.5] \tag{3.8}$$

$$t = h_{\mu} + 2(h_{max} - h_{\mu}) \times (res - 0.5) \quad \forall\ res\ \in\ (0.5, 1] \tag{3.9}$$

where h_{μ} is the mean of the histogram of I, h_{min} is the minimum value of I histogram, and h_{max} is its maximum histogram value, res is the value of *resistance*, and t is the returned threshold value. The pseudocode of the threshold process is presented in **Box 3.2**.

The *limit* control parameter is an integer variable specifying a constraint of the maximally allowed radial coordinate for a new contour point. Specifically, it is the number of radial positions after the largest radial coordinate in the initial contour. After that level, the contour expansion is blocked.

Input : A polar image I of size $w \times l$
An integer $resistance \in [0,1]$
Output : A binary image B_i of size $w \times l$

```
if resistance ≤ 0.5 then
    t ← h_min + 2(h_μ − h_min) × resistance
end
else
    t ← h_μ + 2(h_max − h_μ) × (resistance − 0.5)
end
B_i ← threshold I at t.
return B_i
```

Box 3.2: *Polar image threshold based on* $resistance$.

The *convergence* parameter is an integer variable specifying a constraint of the maximum number of iterations before the expansion process is terminated. Subsequently, the contour at the last iteration is considered as a final contour. In general, the contour is saturated and reaches its final status after a few iterations. This will be shown in an example in the following sub-section.

3.5.3 The Expansion Algorithm

The expansion algorithm is applied after the initial contour processing. The flowchart in Fig. 3.10 explains how the expansion algorithm works and how the pixels are processed in the polar image I. The pseudocode of this algorithm is referred to in **Box 3.3**. The algorithm iterates until the *convergence* criterion is met, or until, in an iteration, no point is expanded anymore (cf. line 2). In each iteration, the *expanded* boolean variable is set to false (cf. line 3), and all the contour points are processed (cf. line 4). If any point is expanded at that iteration, then *expanded* variable is set to true, and the algorithm keeps running as long as the *convergence* criteria is not reached.

The processing procedure of the contour points at each iteration is shown in a separate block in the flowchart (*Process Contour Pixels*), and its pseudocode is presented separately as **Box 3.4**. This procedure evaluates each point in the contour and considers whether to expand it to the next radial coordinate or not. As long as the current contour point is not marked as a *final* contour point (cf. line 2), it is marked as *final* if the *limit* parameter is reached (cf. line 4); otherwise, the previous and the next neighbour contour points are checked if any of them is at a radial position lower than that of the current point (cf. line 7), in which case the point is skipped in this iteration. If none of these neighbour contour points is at a previous radial coordinate, these neighbour points are checked whether they are both marked as *final* points, forcing the current point to be *final* contour point as well (cf. line 8). If both neighbours are not marked as *final* points, the current point is marked as *final* if the point at the next radial position is blocked by the *resistance* parameter (cf. line 11). When the *resistance* control parameter does not block the expansion, the point is expanded to the next radial position unless one of the neighbours is marked as *final* and the current point has higher radial coordinate than that neighbour (cf. line 13), then the current point is marked as a *final* contour point (cf. line 14). Other than that, the point is expanded to the next radial position (cf. line 16), and normally this expanded contour point will be marked as a *final* contour point in the next iteration or when the *convergence* criteria is met before the next iteration.

Figure 3.11 illustrates an example showing the extracted initial estimate by *HCSP* algorithm and the final expanded contour extracted by *CE* method in a pathological cost image function. The image function shown in Fig. 3.11(a) is a sample of a polar image I having ten angular and ten radial coordinates. The first and last angular coordinates θ_A and θ_B are actually for the same point since as the path from θ_A to θ_B is a circular path. The *black* squares in I represent pixels with an intensity value of 0, the *grey* squares represent pixels with an intensity value of 1, and the *white* squares represent pixels with an intensity value of 11. In Fig. 3.11(b), the initial contour points extracted by applying *HCSP* are displayed by *green* spots.

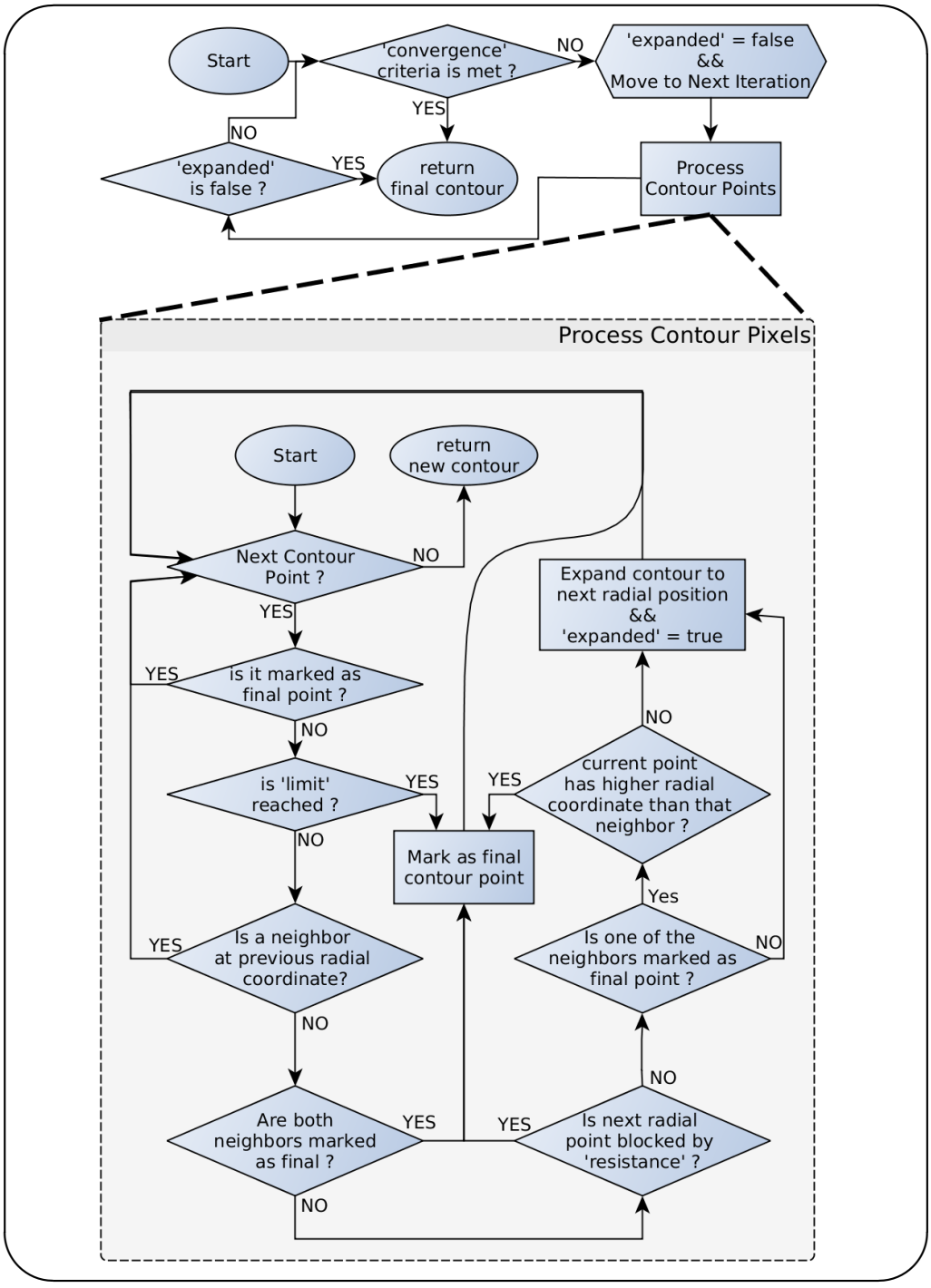


Figure 3.10: *Flow chart explaining the expansion algorithm.*

To illustrate how the *CE* algorithm works, we apply contour expansion on image I with the parameters set as follows: *resistance = 0.5*, *limit = 1*, and *convergence = 5*. After the first iteration, the contour evolves to take the position illustrated in Fig. 3.11(c). The points that were not considered for expansion in this iteration are left as *green* spots. Those expanded are displayed by *blue* spots and the *red* spot is the contour point marked as final. The point at the eighth angular coordinate is marked as *final* because the point at the next radial position was blocked by the *resistance* parameter. Figure 3.11(d) shows the contour points after the second iteration. Note here that the point at the seventh angular coordinate is marked as *final* because both its connected neighbours are marked *final* as well. Figure 3.11(e) shows the path after the third iteration. In the fourth iteration, the first four points and consequently the last contour point are marked as *final* points by the *limit* parameter (cf. Fig. 3.11(f)). In addition, the points at the fifth and ninth angular coordinates were marked *final* because their connected neighbours are marked *final* contour points as well. Since no contour point was expanded in this iteration, the contour expansion process is terminated at this point, even before the *convergence* criteria is met.

To demonstrate the application of this algorithm on a real sample, a *RoI* containing two yeast cells is depicted in Fig. 3.12. The initial estimate is shown as well as the expanded contour for two yeast cells from a sample image acquired by a Zeiss LSM5 confocal microscope [Inc12]. The *yellow* contours represent the initial estimate extracted after the application of *Hough* transform and circular minimal path algorithm, while the *red* contours represent the final expanded version of the contours acquired ensuing the application of our new contour

Input : A binary image B_i of size $w \times l$.
Integer array CsP of initial contour, of size w.
Boolean array isFinal of size w.
Boolean variable expanded.
Integer limit $\in [0, l]$.
Integer convergence ≥ 0.
Output : Integer array finalContour of size w.

```
1  initialize isFinal and expanded to false, i ← 0.
2  while (expanded ≠ false && i < convergence) do
3  |  i ← i + 1
4  |  expanded ← false
5  |  CsP ← ProcessContour(CsP) (Procedure in Box 3.4)
6  end
   /* Current contour is final                                    */
7  for each element p in CsP do
8  |  finalContour[p] ← CsP[p].
9  endfor
10 return finalContour
```

Box 3.3: *Expand Contour Algorithm.*

expansion algorithm. Figure 3.12(a) shows the contours on the bright-field channel used for initial estimation and expansion of the contours, while Fig. 3.12(b) shows the same contours superimposed on the fluorescent counterpart of the cells acquired as a second image channel by the same microscope. The latter channel is used to obtain the meaningful measurements and pattern recognition regarding protein levels and gene expression in the yeast cells. It is clear from this image, how much of the green signal would not be included in the cell measurement if contour expansion would not have been applied. This is even more quintessential when measuring proteins that are bound to the membrane of the cell.

3.6 Validation

In order to validate the new segmentation methods described in this chapter, i.e. the *Hough* transform followed by minimal path algorithms, we create an artificial dataset of 1000 images representing the yeast cells in an approach similar to that implemented in [Yan12]. Such dataset

```
  /* Process contour pixels                                          */
 1 for each point c in CsP with radial coordinate r do
 2     if isFinal[c] ← false then
 3         if r ← limit then
 4             isFinal[c] ← true
 5         else
 6             if (CsP[c − 1] ≠ (r − 1)) and (CsP[c + 1] ≠ (r − 1)) then
 7                 if CsP[c − 1] ← true and CsP[c + 1] ← true then
 8                     isFinal[c] ← true
 9                 else
10                     if (B_i[c, r + 1] ← Background pixel then
11                         isFinal[c] ← true
12                     else
13                         if (isFinal[c − 1] ← true and c has higher r than c − 1) or
                            isFinal[c + 1] ← true and c has higher r than c + 1 then
14                             isFinal[c] ← true
15                         else
16                             Expand c to next radial position (r + 1).
17                             expanded ← true
18                         end
19                     end
20                 end
21             end
22         end
23     end
24 end
```

Box 3.4: *Process Contour Points Procedure.*

has a random number of elliptical shapes ranging from $2\,\mathrm{to}\,100$ per image with a random Gaussian values for the background. The Gaussian values for the background is created to resemble that of the real bright-field microscope images. The eccentricity of the ellipses ranges randomly with $\frac{\mathrm{semi-minor}}{\mathrm{semi-major}}$ factor between $0.75\,\mathrm{and}\,1$ to resemble the actual cell shape. As a case study, an experiment was performed and 8 images were acquired. These images depict 110 cells in total. Our methods were applied on this dataset.

To evaluate our segmentation methods, we considered well-known metrics; i.e. the Pratt Figure of Merit (*FoM*) and *F1-measure*. Moreover, the tuning of parameters for the different segmentation methods and inspection of the segmented results were performed using *Paramorama* [Pre11], a prototype developed to optimize parameters based on parameter sampling and interactive visual exploration. *Paramorama* is implemented as a plug-in for the *CellProfiler* biomedical image analysis framework [Car06].

The following subsections provide definitions for the Pratt *FoM* and that of *F1-Measure*. The last subsection shows the result comparing *HCSP* and *HDT* with state of the art segmentation package, i.e. *CellStat*. In addition, it shows the result comparing the extension algorithm *HCSP-CE* with *CellStat*, *CellSerpent* and the initial *HCSP*.

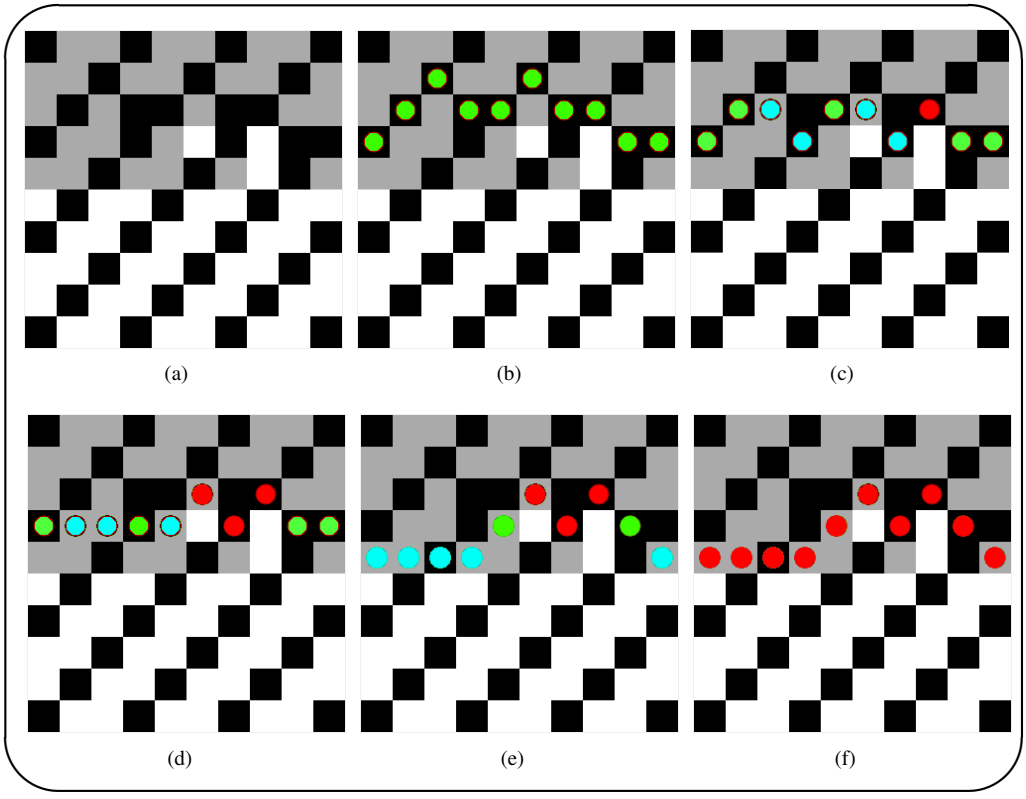
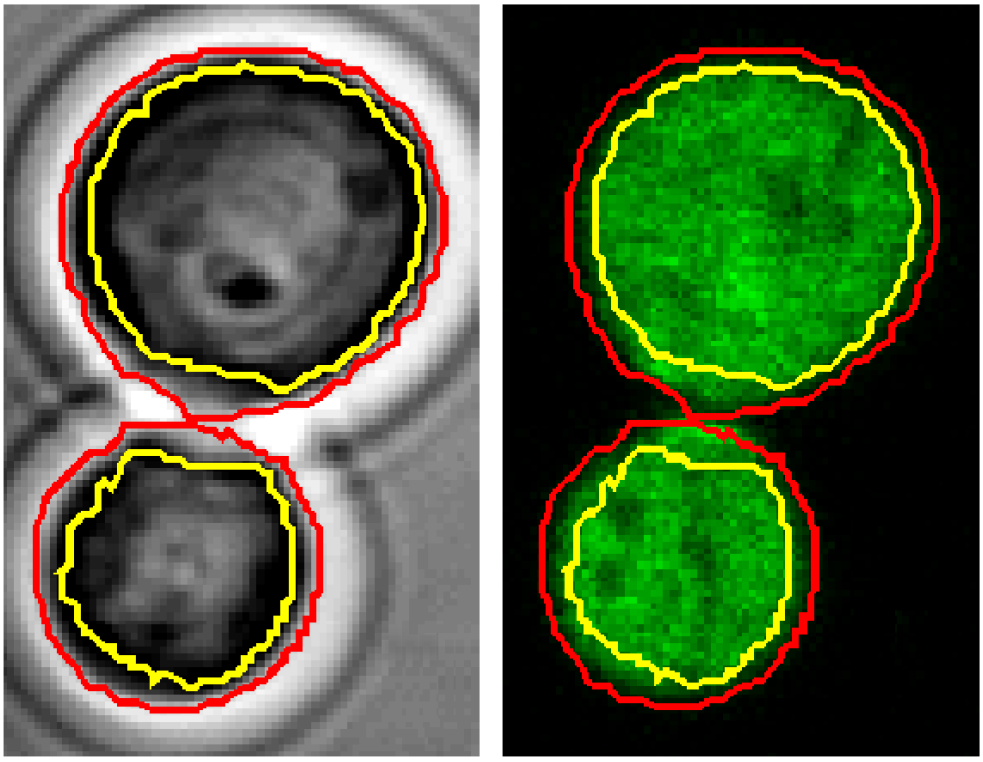


Figure 3.11: *Expanding the Initial estimate of the contour. Black, grey and white squares represent pixels with intensity values of* 0, 1 *and* 11 *respectively.*

3.6.1 Pratt Score

The Pratt Figure of Merit (*FoM*) is considered because it has been reported as a very good criterion to detect the differences in segmentation results [Cha06]. Pratt's *FoM* provides a quantitative comparison of the results of the different contour detection methods by measuring the deviation of the output contours from a ground-truth. The ground-truth dataset were separately delineated using our dedicated structural annotation software (*TDR*) with a digitizer tablet (*WACOM, Cintiq LCD-tablet*) [Ver02]. The Pratt measure is computed according to Eq. 3.10.

$$P = \frac{1}{\max(I_A, I_I)} \sum_{i=1}^{I_A} \frac{1}{1 + \alpha.d^2(i)} \tag{3.10}$$



(a) Bright-Field view of yeast cells.

(b) Contours superimposed on a fluorescent channel of the same yeast cells.

Figure 3.12: *The expanded contour is depicted in red, while the yellow contour is the initial estimate extracted by the Hough transform and Circular Shortest Path algorithm (HCSP).*

where I_A being the number of detected edge points, I_I the edge points in the ideal ground-truth image, α a scaling factor or an empirical calibration constant set to the optimal value established by Pratt, $\alpha = 1 \div 9$ [Abd79], $d(i)$ is the distance between the detected edge point and the nearest edge point in the ideal ground-truth [The10]. Pratt's *FoM* is an indicator of the quality of edge and reflects the overall behaviour of the distances between the edges. Pratt's *FoM* is a relative measure, which varies in the range [0,1], where 1 represents the optimal value, i.e., the detected edges coincide with the ground truth [Boa09].

3.6.2 F1-Measure

The *F1-measure*, also known as *F1-score*, is a measure of accuracy used in many domains including the comparison of segmentation algorithms. It considers the precision and recall in its computation [Hou13]. The precision is defined as the number of correctly detected objects divided by the number of detected objects by the segmentation method. The recall is defined as the number of correctly detected objects divided by the number of actual objects. We considered a yeast cell object correctly detected if the centroid of its detected binary mask also exists as a foreground pixel in the ground-truth binary mask. The F1-score can also be used to measure the accuracy of the extracted contour by considering the binary mask of the individual detected object. In this case, the precision is defined as the correctly detected pixels in the binary mask divided by the number of detected pixels. The recall is then defined as the number of correctly detected pixels divided by number of actual pixels in the ground-truth mask. When this later definition is used we will refer to the F1-score as $F1_p$-score to indicate its computation at the pixel level. The F1-score is computed according to Eq. 3.11.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{3.11}$$

3.6.3 Results

In this subsection, we present the validation results of the methods we prooposed earlier including the *HDT* and *HCSP* methods proposed in Section 3.2 and Section 3.4; in addition to the optimization method applied after *HCSP* as discussed in Section 3.5, i.e. the *HCSP-CE* method.

HDT and *HCSP* validation

The *HDT* method uses *Hough* transform followed by the grey-weighted distance transform as the minimal path algorithm. The *HCSP* method uses *Hough* transform followed by our proposed circular shortest path algorithm. To evaluate these methods, we compare them with state of the art yeast segmentation software, i.e. *CellStat*.

CellStat also uses dynamic programming and cost functions and it is implemented in *Matlab*. It is designed to segment bright-field images of *S. cerevisiae*, but it has a constraint on the cells to be detected, as they can not be clumped within other cells [Kva08].

	Detection Rate	F1-score
HCSP	97.23%	0.972
HDT	97.23%	0.972
CellStat	94.97%	0.950

Table 3.1: *Detection Rates and F1-measure on artificial dataset.*

In Table 3.1 the detection rates and *F1-scores* [Hou13] are shown after applying the three methods on the artificial dataset. In the last step of the contour extraction process, the shape is checked for its circularity value. If this value is less than an empirically determined value, i.e. 0.614, the object is rejected. Checking the detection rate on the whole artificial dataset using *CellStat* is not feasible. Therefore, we manually checked on a sample drawn from the complete set. To verify whether this sample is representative for the whole dataset, an unpaired student t-test was performed on the results obtained from the *HCSP* method. The p-value of this test is 0.95.

As a case study, an experiment with *S. cerevisiae* yeast cells was performed with 8 representative images. In total, these images contain 110 cells. Our methods were applied on this dataset (both *HDT* and *HCSP*). The results are compared to that obtained from *CellStat* [Kva08]. The detection rates and F1 – measures are shown in Table 3.2.

	Detected	Correct	Precision	F1
HCSP	115	106	0.964	0.942
HDT	120	104	0.945	0.904
CellStat	112	101	0.918	0.910

Table 3.2: *Detection Rates and F1-measure on yeast images.*

For the comparison of the accuracy of the detected contours, Pratt's *FoM* is selected. Table 3.3 shows the results of the comparison between the proposed methods and that of the *CellStat* software on our *S. cerevisiae* dataset. The detection rate outperforms that of *CellStat* and the number of detected contours that gains better Pratt score (referred to as Wins in Table 3.3) is also higher than that of *CellStat* making it a suitable segmentation method to be used in analysis of yeast cells. Hence, this method is successfully used in our yeast analysis platform proposed in Chapter 2.

	HCSP	*CellStat*	*HDT*	*CellStat*
Detected	106	101	104	101
Precision	96.4%	91.8%	94.5%	91.8%
Wins	60	50	57	51

Table 3.3: *Comparing Detected Contours of yeast images.*

HCSP-CE validation

In order to validate the new *HCSP-CE* method, its performance is compared to the previous *HCSP*, *CellSerpent* and *CellStat*.

Figure 3.13 shows the number of detected cells that gained a higher Pratt score using *HCSP* method followed by contour expansion (*CE*) algorithm (*HSCP-CE*). These cells were obtained from a dataset of 312 *Saccharomyces cerevisiae* yeast cells distributed over 14 bright-field

images acquired by a *Zeiss LSM 5 Exciter-AxioImager M1* confocal microscope. Compared to *CellSerpent*, the *HCSP-CE* method has 163 cells with better Pratt scores, referred to as wins, while *CellSerpent* has only 138 wins. In addition, comparison with *CellStat* leads to 284 wins of cells with higher Pratt scores in the *HCSP-CE* method, with only 16 wins for *CellStat*. This result is expected as the method implemented in *CellStat* looks for the minimal path as the final contour of yeast cells in a similar way to that of *HCSP*, which scored 18 wins. On the other hand, *CellSerpent* performs better than *CellStat* because its method is based on Active Contour models for cell detection. However, Active Contour models couldn't outperform our algorithm.

In Table 3.4 a comparison between the detection rates of the four used segmentation methods on the dataset of 312 cells is listed. In Table 3.5, the accuracy of the detected contours is compared between the four methods using average Pratt scores and $F1_p$-scores for the individual objects.

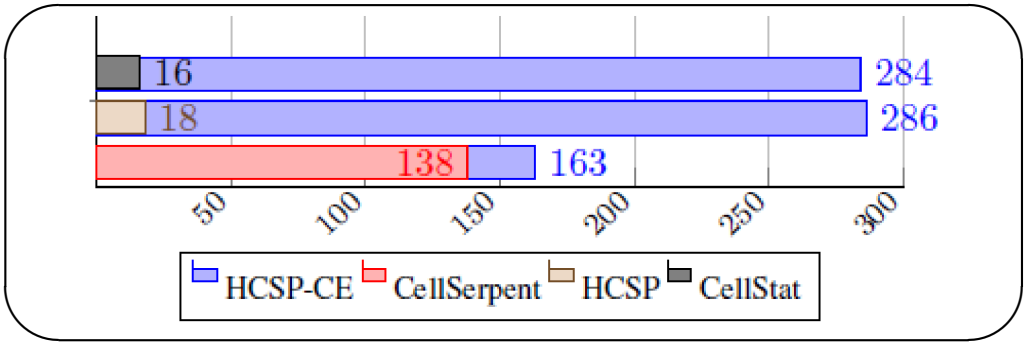


Figure 3.13: *Number of cells with higher Pratt score (wins).*

.

	HCSP-CE	CellSerpent	HCSP	CellStat
Detected	**322**	359	**322**	231
Correct	296	**302**	296	181
Precision	**0.92**	0.84	**0.92**	0.78
Recall	0.95	**0.98**	0.95	0.58
F1	**0.93**	0.90	**0.93**	0.67

Table 3.4: *F1-measure and average Pratt Score of different segmentation methods on a yeast dataset.*

.

	HCSP-CE	CellSerpent	HCSP	CellStat
$F1_p$	**0.92**	0.89	0.76	0.47
Pratt	**0.58**	0.55	0.19	0.15

Table 3.5: *Accuracy of detected contours.*

The *HCSP-CE* segmentation method has an overall highest scores with an F1-score of 0.93, $F1_p$ of 0.92 and an average Pratt score of 0.58. The following method is *CellSerpent* with an F1-measure of 0.90, $F1_p$ of 0.89 and average Pratt of 0.55. *CellStat* follows with an F1-measure of 0.67, $F1_p$ of 0.47 and average Pratt of 0.15. *HCSP* scores similar F1-measure to that of *HCSP-CE* as they both use the same underlying method for detection, however the Pratt score of *HCSP* is 0.19 and the $F1_p$ is 0.76, which is noticeably worse than *HCSP-CE* and close to that of *CellStat* as they both look for minimal paths for contour extraction.

3.6.4 Robustness under Noise

The *HCSP-CE* method is further validated for its robustness under noise. We evaluate how the method performs under various noise levels. First we selected a random sample and prepared a groundtruth. The ground-truth is separately delineated using our dedicated structural annotation software (*TDR*) with a digitizer tablet (*WACOM, Cintiq LCD-tablet*) [Ver02]. Subsequently, we determined the optimal parameter values to perform the segmentation. This step involves an initial heuristic determination of parameter values, then quantitatively evaluate the segmentation accuracy after iterating each of the parameters individually while fixing values of all the other parameters. This procedure is repeated until parameter values are saturated. Figure 3.14 shows the optimal saturated parameter set values after three repetitions.

In order to evaluate the segmentation accuracy, we used F1-scores for the detected cells per image in addition to F1-scores of the extracted mask per cell as described in Section 3.6.2. We used the average of the two F1 scores to determine the optimal values.

In order to generate noisy images, we first determined three main types of noise. Gaussian noise, Poisson noise and Speckle noise. The principal sources of Gaussian noise in digital images arise during acquisition e.g. sensor noise caused by poor illumination and/or high temperature, and/or transmission [Cat13]. Poisson noise is known also as Shot noise, it occurs in photon counting in optical devices. Poisson noise describes the fluctuations of the number of photons detected (or simply counted in the abstract) due to their occurrence independent of each other [Bla00]. Speckle noise, also known as salt and pepper kind of noise, is caused usually by the scattering of light from a highly coherent source, such as a laser, and generates a random-intensity distribution of light that gives the image a granular appearance [fre16].

We vary the signal to noise ratio *SNR* for each type of noise, and perform the segmentation on the sample using the optimal parameters mentioned in Fig 3.14. Image noise can often be described by an additive noise model, where the resulted image $f(i,j)$ is the sum of the true image $s(i,j)$ and the noise image $n(i,j)$ as depicted in Eq. 3.12.

$$f(i,j) = s(i,j) + n(i,j) \tag{3.12}$$

We considered the noise $n(i,j)$ as zero-mean and we describe it by its variance σ_n^2. The impact of the noise on the image is described by the signal to noise ratio (SNR), which is given by Eq. 3.13, where σ_s^2 and σ_n^2 are the variances of the true image and the noise image respectively.

$$SNR = \frac{\sigma_s}{\sigma_n} = \sqrt{\frac{\sigma_s^2}{\sigma_n^2} - 1} \tag{3.13}$$

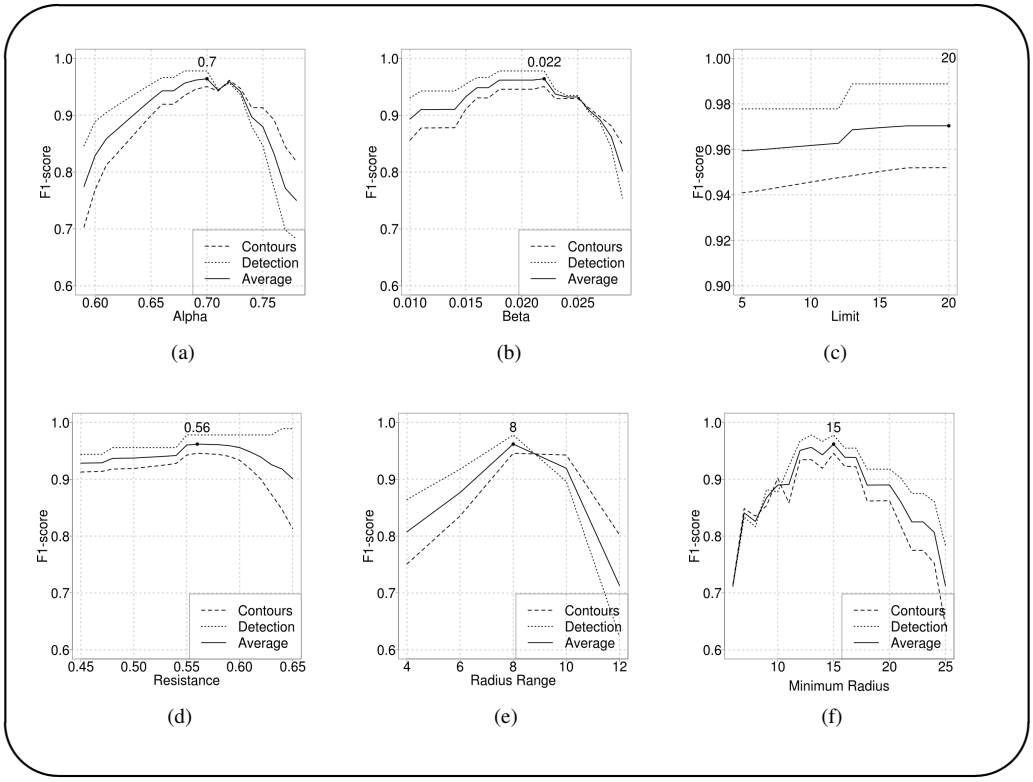


Figure 3.14: *Optimal Segmentation Parameters. (a) - The optimal value of the alpha parameter for the sample dataset is* 0.7*. This parameter is part of the thresholding equation described in 3.2.2. (b) - Beta parameter associated with alpha has an optimal value at* 0.022*. (c) - The limit parameter used in the optimization of contours was described in 3.5.2. It has an optimal value at 20. (d) - The resistance is also a contour optimization parameter described in 3.5.2. Its optimal value is 0.56. (e) - The radius range used in filling the Hough accumulator as described in 3.2.1. The optimal value is 8. (f) - The minimum radius to consider during the acuumualtor filling has an optimum at 15.*

Figure 3.15 shows the segmentation performance under the noisy conditions with an *SNR* ranging from 1 to 35. For Gaussian noise, the final average F1-score is 0.926 and the standard deviation is 0.009. For Poisson noise the final average F1-score is 0.934 and the standard deviation is 0.012. For Speckle noise the final average F1-score is 0.928 and the standard deviation is 0.010. These numbers along with the visual inspection of the plots show that the *HCSP-CE* segmentation method is very robust under various noise conditions.

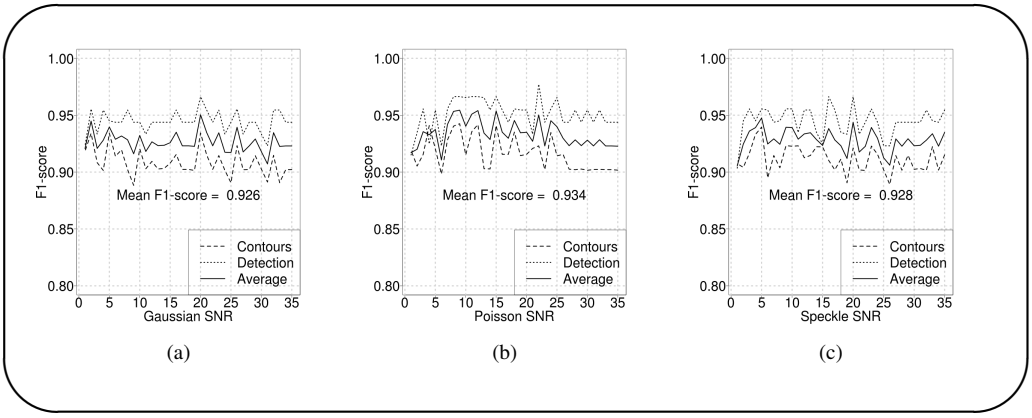


Figure 3.15: *F1-score at various noise levels. (a) - Assessing segmentation performance under various levels of Gaussian noise. (b) - Assessing segmentation performance under various levels of Poisson noise. (c) - Assessing segmentation performance under various levels of Speckle noise.*

3.7 Conclusion

In this chapter, we proposed new methods to detect oval-shaped objects, such as yeast cells in bright-field microscopy images. We started by defining *Hough* transform and introducing our proposed equations to threshold the *Hough* accumulator array. Subsequently, we introduced the definition of polar transformation, the concept of polar coordinate system and how transformation from cartesian to polar system is achieved. We also demonstrated the polar representation of a sample yeast image. The polar representation of an object image is required to extract the minimal path corresponding to the contour of the object. The minimal path concept is also discussed in this chapter and two algorithms to extract this minimal path are introduced. These algorithms are the *Grey-weighted distance transform* and *Circular shortest path*. Hence, we proposed two segmentation methods that uses *Hough* transform and minimal path algorithms; i.e. *HDT* and *HCSP* methods. Subsequently, we introduced our additional expansion algorithm that operates on the polar representation of object images; i.e. *HCSP-CE*. To validate the *HDT* and *HCSP* new methods, they have been applied on two datasets along with state of the art software packages common to yeast biology. The first dataset is a large artificial dataset of ovoid shapes, and the second is a actual small dataset of *S. cerevisiae* yeast cells. For evaluation, we considered two metrics namely the Pratt Figure of Merit (*FoM*) and *F1-Measure*. The results show higher Pratt score and *F1-Measure* compared to the method from the *CellStat* software. On the other hand, the *HSCP-CE* is applied to a different dataset of 312 *S. cerevisiae* yeast cells distributed over 14 bright-field images. *HCSP-CE* was compared to methods from *CellStat* and *CellSerpent*.