Image Segmentation of Multi-shaped Overlapping Objects

Kumar Abhinav[†], Jaideep Singh Chauhan[†] and Debasis Sarkar
Department of Chemical Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India

Keywords: Image Segmentation, Overlapping Objects, Multiple Shaped Objects, Contour Grouping, Concave Points.

Abstract: In this work, we propose a new segmentation algorithm for images containing convex objects present in multiple shapes with a high degree of overlap. The proposed algorithm is carried out in two steps, first we identify the visible contours, segment them using concave points and finally group the segments belonging to the same object. The next step is to assign a shape identity to these grouped contour segments. For images containing objects in multiple shapes we begin first by identifying shape classes of the contours followed by assigning a shape entity to these classes. We provide a comprehensive experimentation of our algorithm on two crystal image datasets. One dataset comprises of images containing objects in multiple shapes overlapping each other and the other dataset contains standard images with objects present in a single shape. We test our algorithm against two baselines, with our proposed algorithm outperforming both the baselines.

1 INTRODUCTION

A number of vision applications like Medical Imaging, Recognition Tasks, Object Detection, etc. require segmentation of objects in images. Overlapping objects in such images pose a major challenge in this task. Identifying hidden boundaries, grouping contours belonging to same objects and efficiently estimating the dimensions of partially visible objects are some of the complexities that overlap introduces. Further regions of high concentration of objects with a high degree of overlap introduces computational challenges for segmenting such images. In our work we introduce a method for tackling these issues for heterogeneous and homogeneous images. Heterogeneous images are defined as images containing multi-shaped overlapping objects, eg. a rod over a sphere. Homogeneous images are defined as images with similarly shaped overlapping objects. Segmentation is particularly challenging for a heterogeneous image because assigning a single shape entity to all the objects is bound to create inaccuracies. We test our work on two Datasets, Dataset-I (Fig.1.a) is a representative of the heterogeneous group of images, whereas Dataset-II (Fig.1.b) is a standard dataset containing homogeneous images.

To tackle the discussed challenges we break down the problem statement into the following three sub-problems: (1) *Contour extraction*, this corresponds to identifying the visible and hidden contours (2) *Contour mapping*, this is for grouping the identified contours which belong to the same object and mapping them to it (3) *Shape identification* for fitting a shape entity to the grouped contour segments, this step implicitly estimates the hidden parts.

To address these issues we propose a two step algorithm. First, we identify the contours and break them into segments. This cleaving happens at all points where contours of two different objects intersect. These cleave points are identified using a procedure called concave point extraction described in Sec.3.2. Next we stitch the broken segments together which belong to the same objects. This is done by fitting an ellipse over all the broken segments, and stitching them together depending on their proximity and orientation. This concludes the first step and solves the first two sub-problems. The next step of the algorithm is to assign a shape entity to the mapped contours. This can be easily done for homogeneous images, a single shape entity like an ellipse fits all. It becomes tricky for heterogeneous image. The intuition of our work is that we need to treat each contour based on its shape class. So, first we define shape classes for the dataset. Followed by defining a specific shape entity for each class which can be applied to a contour that falls in it. The images in Dataset-I contains two kinds of objects, needle(or rod) shaped and spherical shaped, overlapping each other. We use

† denotes equal contribution

Abhinav, K., Chauhan, J. and Sarkar, D.
Image Segmentation of Multi-shaped Overlapping Objects.
DOI: 10.5220/0006628404100418
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 4: VISAPP*, pages 410-418
ISBN: 978-989-758-290-5

these two shapes as the defined classes for assigning a shape entity. We fit an ellipse over the mapped segments and use its aspect ratio to determine the class to which the segment belongs to. Once classified each segment is treated according to the defined heuristic for the class. This step solves the third sub-problem and is followed by a post-processing step to reject the wrongly segmented objects.

We summarize our major contributions as follows:

- We propose a new algorithm to segment heterogeneous images with regions of high object concentration and a high degree of overlap.
- We test our work on both heterogeneous and homogeneous datasets, beating the baselines for both, demonstrating the general applicability of our work.

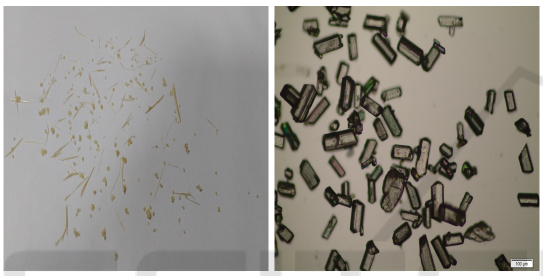

(a) Dataset-I Crystal. (b) Dataset-II Crystal.

Figure 1: Overlapping Crystals Image.

2 RELATED WORK

Image segmentation has been in the research domain for a while, but the nature of problem is such that no approach generalizes well for a wide range of image datasets. Certain approaches extract edge information using the sharp changes in the pixel intensities in images (Pal and Pal, 1993; Rastgarpour and Shanbehzadeh, 2011). Object boundaries tend to introduce peaks in gradients of pixel intensity, thus these methods are effective in identifying the edges. Multi-scale analysis (Gudla et al., 2008) is one of the edge based methods which is robust enough to handle weak edges. Others use heuristic based grouping of pixels for identifying regions with similar properties in images. Region Growing (Adams and Bischof, 1994) and Region Splitting-Merging (Haralick and Shapiro, 1985) are two of the common approaches applied in these set of methods. However, these approaches don't segment images with overlapping objects. Such images lack crucial edge and pixel information for the hidden parts which is required for the effectiveness of these approaches.

Certain approaches use dynamic programming for the task (Baggett et al., 2005), they determine the most optimal boundaries for the hidden objects by defining a path of the highest average intensity along boundaries. Graph-cut methods segment objects by creating a graph out of pixels treated as nodes and difference between their intensities as a weight for edges, followed by finding the minimum cut for the graph (Shi and Malik, 2000; Felzenszwalb and Huttenlocher, 2004). Both of these approaches require prominent gradients at boundaries to be effective in segmenting objects.

One of the popular approaches for segmentation of overlapping objects is the watershed algorithm (Vincent and Soille, 1991) which in its classical form often results in over-segmentation. A potential solution to over-segmentation is region merging, which does solve the problem but the method efficiency varies depending upon the object size and object distribution concentration in the image. Marker-controlled watershed (Malpica et al., 1997; Yang et al., 2006) is also used for the purpose but the efficiency of the method depends highly on the accurate identification of the markers.

(Zhang and Pless, 2006) formulates the problem as a combination of level set functions and uses similarity transforms to predict the hidden parts of partially visible objects. The drawback of this method is that its performance is dependent on initialization and is computationally expensive. (Álvarez et al., 2010) proposes a morphological extension to the level based methods, it introduces morphological operations in traditional snake algorithms. This method outperforms earlier active contour methods as the curve evolution is simpler and faster. It however fails to converge when large number of closely spaced overlapped objects are present in the image. In (Zhang et al., 2012), the problem of segmenting overlapping elliptical bubbles is solved by segmenting the extracted contours into curves. These curves are further segmented into groups belonging to the same bubble followed by ellipse fitting. This approach works well for images having similar shaped objects, but fail for images containing multi-shaped overlapping objects.

Our proposed segmentation algorithm is inspired from concave point based overlapping object segmentation method (Zafari et al., 2015b). In (Zafari et al., 2015b), the idea of grouping the contours belonging to the same object present in an overlap cluster is carried out by fitting ellipses over the segmented contours. The grouped segments are further fitted with ellipses to separate overlapping objects. The efficiency of this approach declines when applied to images with multi-shaped objects.

Our approach is built for images containing objects with high degree of overlap and we attempt to solve the following issues present in earlier discussed approaches: (1) We attempt to segment images with regions of high object concentration, which becomes a problem for approaches which are computationally expensive, (2) Our approach does not rely on strong gradients at boundaries to be effective and (3) Most approaches don't take into account images with overlapping multi-shaped objects, we use adaptive shape fitting to address this problem.

3 PROPOSED METHODOLOGY

We propose a segmentation algorithm consisting of the following two steps: Contour Region Detection (CRD) and Shape Fitting. Fig.2 describes the methodology.

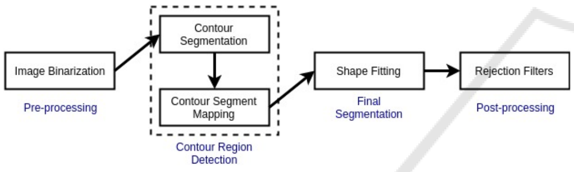


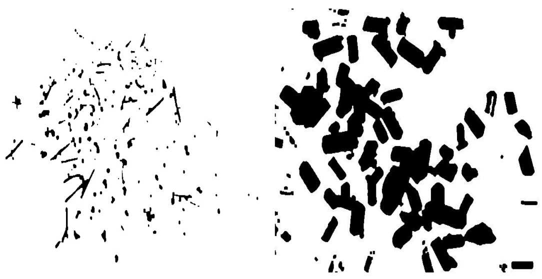
Figure 2: Overview of the proposed approach.

We take in RGB images as input and convert them to grayscale before further processing. After conversion, median blur is applied to remove noise from the image, followed by Image binarization. This is carried out using Otsu's method (Otsu, 1979) from which we get a binary image B_O. Finally, morphological operations are used to filter noisy regions and fill any small holes in the binary cell regions of B_O. Fig.3.a and 3.b shows the binarized image of Fig.1.a and Fig.1.b respectively.

CRD step is then carried out on the binary image, B_O to extract the contour information. It identifies the contours corresponding to both visible and hidden boundaries and return contour segments grouped together on basis of them belonging to the same object. Shape Fitting is the final segmentation step where a shape entity like an ellipse or a rod is assigned to the grouped contour segments. Rejection Filters are applied in the post-processing step for identifying and eliminating wrongly segmented objects.

3.1 Contour Region Detection

CRD is the first step of our proposed methodology. In this step, segmentation of overlapping objects is carried out using the boundaries of the visible objects. We begin first by extracting the contour points corresponding to each of the object's contour present in the



(a) Dataset-I image. (b) Dataset-II image.

Figure 3: Binarized Image for both the Datasets.

image. For this we use the border following algorithm described in (Suzuki et al., 1985). Next we describe some notations that we'll be using from hereon, let the detected object's contour C is described using a set $C_{contour}$ with N contour points such that, $C_{contour}$ = $\{p_i(x_i,y_i);\ i=1,2,...,N\}$, where N is the number of contour points and p_i is the i^{th} contour point with co-ordinates (x_i, y_i). p_{i+1} and p_{i-1} are the neighbouring contour points of p_i. The contour region detection next involves two distinctive tasks: contour segmentation and contour segment mapping.

3.1.1 Contour Segmentation

The contour points extracted previously represents the contour of a single object or multiple overlapped objects being detected as a single entity. Fig.4 displays the detected contour points on a sample contour of the binarized crystal image shown in Fig.3.a. The overlapping of objects forms an irregular closed shape. The intersections of the object boundaries in a overlapped object contour are represented using break points or concave points. Concave points are thus used to segment the contour of overlapping objects into separate contour segments.

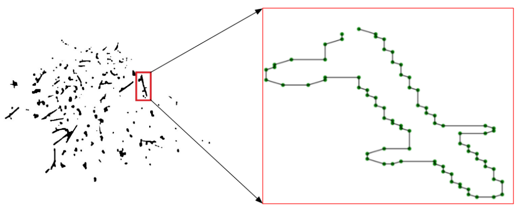


Figure 4: Plotting a contour from Dataset-I to show its Contour points, which are represented using Green points.

The concave points are determined by a two step procedure: (1) Determining the corner points using RDP algorithm and (2) Identifying concave points from corner points using convexity check.

The corner points are the points of local minima or maxima of the curve drawn by plotting the predetermined contour points. Thus, for each of the object's contour governed by its contour points, the corner points are computed using Ramer-Douglas-Peucker (RDP) algorithm (Douglas and Peucker, 1973). RDP is a path simplification algorithm which reduces the number of contour tracing points. The RDP algorithm takes in an ordered sequence of contour points represented using set $C_{contour}$ as input. The algorithm returns a set of corner points extracted from the set $C_{contour}$. Let this set of corner points be represented using a set C_{corner}, $C_{corner} = \{p_{cor,i}(x_i,y_i) \mid p_{cor,i} \in C_{contour};\ i = 1,2,...,M\}$, where $p_{cor,i}$ is the i^{th} corner point and M be number of corner points extracted such that $M \leq N$. Fig.5.a shows a set of corner points being extracted from contour points of a sample contour as displayed in Fig.4.

Convexity check is then carried out on the set of corner points contained in C_{corner} to obtain a set of concave points. The algorithm to compute concave points is described in Algorithm 1.

In Algorithm 1, detected corner points from C_{corner} are the inputs. Initially vectors are created by joining two adjacent corner points in clockwise direction along direction of second corner point. Thus, each corner point will be contained in two vectors, one originating from the concerned point and the other terminating at it. For every corner point, the sign of the cross product of the two vectors it is a part of, is computed and stored as the orientation of that corner point. Net orientation of the contour is then computed by taking sign of sum of all the orientations, which basically gives effective sign over all orientations. If the net orientation is positive it means that the overall contour traversal along the corner points is in anti-clockwise direction else in clockwise direction. Wherever the corner point's orientation is different from the net orientation i.e. there is a break in the traversal direction, that corner point is marked as a concave point.

Let the set of concave points computed from Algorithm 1 be represented using a set $C_{concave}$, $C_{concave} = \{p_{concave,i}(x_i,y_i) \mid p_{concave,i} \in C_{corner};\ i = 1,2,...,P\}$, where $p_{concave,i}$ is the i^{th} concave point and P be number of concave points computed such that $P <= M <= N$. Fig.5.b displays the concave points being extracted from the corner points of a sample contour displayed in Fig.5.a.

3.1.2 Contour Segment Mapping

The detected concave points in set $C_{concave}$ are used to split the earlier detected object's contour C into contour segments L_i's. Let $L_i = \{p_{i_1}, p_{i_2}, ..., p_{i_s}\}$ represent a contour segment of the contour C where s is the number of contour points on L_i. The set L_i contains the set of contour points from $C_{contour}$ present between two concave points, p_{i_1} and p_{i_s}. The start and end points of the corresponding segment, L_i are represented by p_{i_1} and p_{i_s}, such that $p_{i_1}, p_{i_s} \in C_{concave}$. Since total number of concave points are P, we have

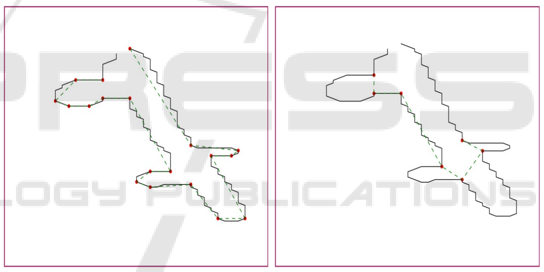
$$C = L_1 + L_2 + ... + L_P \tag{1}$$

From now on, the detected contours C are referred to as contour clusters. A contour cluster is a general term for detected contours which may contain multiple overlapped objects or just a single object. Our proposed segmentation algorithm segments the overlapped objects present in a contour cluster, if overlapping exists otherwise it just segments the single object present. The sample contour present in Fig.4 can now be referred as a contour cluster.

Overlapping of objects and their shape irregulari-

Algorithm 1: Concave Point Extraction.

```
1:  net_orient = 0        ▷ Overall orientation of contour
2:  orientation = []  ▷ Orientation at each corner point
3:  FOR $i$ = 1 to $M$ :        ▷ $M$ is # Corner Points
4:      GET $p_{cor,i}$ , $p_{cor,i-1}$ , $p_{cor,i+1} \in C_{corner}$
5:      IF $i$ equals 1 :
6:          $p_{cor,i-1} = p_{cor,M}$
7:      ENDIF
8:      IF $i$ equals $M$ :
9:          $p_{cor,i+1} = p_{cor,1}$
10:     ENDIF
11:     ▷ Vector initialization from given two points
12:     $\boldsymbol{V}_i = (p_{cor,i-1}, p_{cor,i})$
13:     $\boldsymbol{V}_{i+1} = (p_{cor,i}, p_{cor,i+1})$
14:     orientation[$i$] ← SIGN($\boldsymbol{V}_i \times \boldsymbol{V}_{i+1}$)
15:     net_orient ← net_orient + orientation[$i$]
16: ENDFOR
17: net_orient ← SIGN(net_orient)
18:   ▷ If net_orient is "+ve" means anti-clockwise if
    "-ve" means clockwise
19: FOR $i$ = 1 to $M$ :
20:     IF orientation[$i$] not equals net_orient :
21:                         ▷ Break in traversal direction
22:         INSERT $p_{cor,i}$ in $C_{concave}$
23:     ENDIF
24: ENDFOR
```



(a) Corner Points Detection (Red points). (b) Concave Points Detection (Red points).

Figure 5: Contour segmentation points of a sample contour in Dataset-I.

ties in a contour cluster lead to a presence of multiple contour segment of a single object. Contour segment mapping is thus needed to map together all the contour segments belonging to a same object in a contour cluster.

The segment mapping algorithm iterates over each pair of the contour segment(L_i's), checking if they could be mapped (grouped) together. The algorithm follows orientation based mapping, where an ellipse is first fitted on each of the contour segment using classical least square fitting. The fitted ellipse on a contour segment is then mapped with other ellipses based on proximity and orientation. The algorithm for segment mapping is discussed in Algorithm 2.

Algorithm 2: Contour Segment Mapping.

```
1:      ▷ $L^s_i$ : $i^{th}$ contour segment of contour cluster $C$
2:  FOR $i$ = 1 to $P$ :              ▷ $P$ is # Concave Points
3:      $fit\ ellipse\ E_i\ to\ segment\ L_i$
4:      $major\_axis_i, minor\_axis_i, orient\_angle_i \leftarrow E_i$
5:      $aspect\_ratio_i \leftarrow \frac{major\_axis_i}{minor\_axis_i}$
6:      IF $aspect\_ratio_i < 2$ :
7:                    ▷ Fitted ellipse close to a circle
8:            ▷ Mark as unique segment, isn't mapped
9:          MAP $L_i$ $alone\ to$ Set $G$
10:         CONTINUE
11:     ENDIF
12:     FOR $j = i + 1$ to $P$ :
13:         $fit\ ellipse\ E_j\ to\ segment\ L_j$
14:         $major\_axis_j,\ minor\_axis_j \leftarrow E_j$
15:         $orient\_angle_j \leftarrow E_j$
16:         $aspect\_ratio_j \leftarrow \frac{major\_axis_j}{minor\_axis_j}$
17:         IF $|$ $orient\_angle_i$ - $orient\_angle_j$ $| < \varepsilon$:
18:             MAP $together\ L_i\ and\ L_j\ to$ Set $G$
19:         ENDIF
20:     ENDFOR
21: ENDFOR
```

In Algorithm 2, the contour segment with fitted ellipse close to a circle (aspect ratio ≤ 2) isn't mapped with any other contour segment. It's assumed that only elongated ellipses, with larger aspect ratio (> 2) have more than one contour segment present in the contour cluster C.

For the contour cluster C containing P contour segments, let these segments are mapped to form K groups. This new unique contour set containing groups of mapped together contour segments is represented using a set G with K elements such that :

$$G = \{G_1, G_2, ..., G_K\} \tag{2}$$

$$\bigcup_{i=1}^{K} G_i = \{L_1, L_2, ..., L_P\} \tag{3}$$

$$\bigcap_{i=1}^{K} G_i = \Phi \tag{4}$$

An element in this unique set of grouped contour segments G, represents the entire visible contour of an unique object present in the overlapping object contour cluster.

Fig.6 displays the result of the Contour Segment Mapping using color visualization. The sample contour cluster is the one shown in Fig.4. Fig.6.a shows the result before segment mapping, where each colored segment represents a contour segment L_i of the above mentioned sample contour cluster, C. These segments are split using concave points as displayed in Fig.5.b. Fig.6.b shows the result after segment mapping. It can be seen that earlier present contour segments denoted by yellow, green and pink colors respectively now gets mapped together and is represented using pink color. Each of the unique colored segments in Fig.6.b represents an element of the set G.

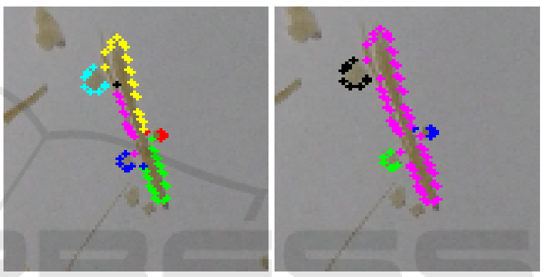

(a) Before Contour Segment Mapping. (b) After Contour Segment Mapping.

Figure 6: Contour Segment Mapping result on a sample contour cluster from Dataset-I(colors are used for illustration to visualize contour segment mapping, images are magnified thus continuous color segments looks broken).

3.2 Shape Fitting

The final step of the algorithm is to assign a shape entity to the object contours present in a contour cluster. Here we present a heuristic based approach for assigning this entity. We begin with modeling the contours with ellipse fitting. The classic least square fitting algorithm is used for assigning the ellipses. An ellipse is fitted on each element of the set G governed by Eq.2-4(representative of all the contours belonging to the same object). Fig.7 displays result of the segmentation on some detected contour clusters of the binarized crystal image(Fig.3.a).

For Dataset-II, ellipse fitting suffices the segmentation process, as it contains rectangular shaped objects whose dimensions are well represented by the major and minor axes of the fitted ellipse. However, the heuristics are required for Dataset-I, where the contours are needed to be classified according to their shape representation. The aspect ratio of the fitted el-

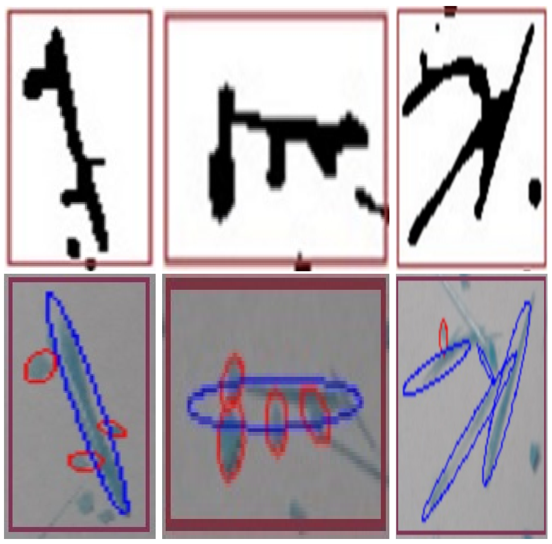


Figure 7: Overlapped object segmentation in Dataset-I. First row contains three different detected contour clusters, second row shows segmentation of corresponding clusters.

lipse determines its closeness to a circle (aspect ratio < 2) or a rod. The threshold value for the aspect ratio is set to be 2 by analyzing the data manually. We cannot represent the rod shaped crystals with ellipse due to a problem of overestimation of length. This is due to the fact that the rod shaped crystals have very small width and the fitted ellipse couldn't take account of this. As a solution to this, we propose a new bounding box algorithm as a shape estimator for rod shaped crystals. It takes into account the small width of the crystal. The results after applying this algorithm are shown in Fig.8. The procedure for the bounding box algorithm is discussed below:

1. The orientation angle, center coordinates and the major and minor axis of the fitted ellipse on the rod shaped crystal's contour are computed.
2. A pair of parallel line is used with their slope equal to the ellipse's orientation angle and their initial position coinciding with the major axis.
3. The parallel lines are separated equidistantly from the major axis along the direction of minor axis in opposite directions.
4. The separation distance is increased until a far end of the crystal contour is achieved at the both side.
5. The distance between parallel lines gives the width of the rod shaped crystal.
6. The same algorithm is extended to calculate rod's length using a new parallel line pair with initial slope along normal to orientation angle and initial position coinciding with minor axis.
7. The bounding box generated by the intersection of these two pairs of parallel lines at their final positions gives the desired shape estimator for the segmentation of rod shaped crystals.

The calculated dimensions are converted from pixels to micrometres using scaling factor of the microscope and the camera.

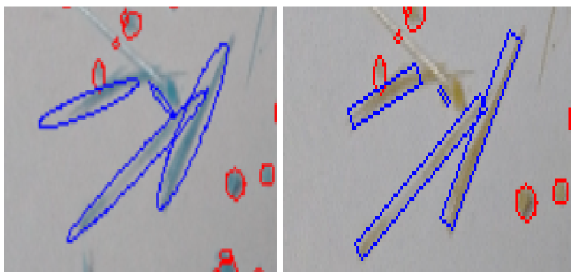

(a) Normal ellipse fitting. (b) Bounding box algorithm.

Figure 8: Rod shape crystal estimation using bounding box.

3.3 Rejection Filters

This is the post-processing step of our proposed methodology. It reduces the false positive segmentation which may be introduced due to incorrect ellipse fitting in the shape fitting step. The two filters being used are discussed accordingly.

The first filter is a *masking rejection* filter, which checks whether the fitted ellipse covers at least 75% black pixels within its boundaries in the binary image. It makes sure that the fitted ellipse is not empty. Finally, an *area overlap* filter is used, which on encountering a high degree of overlap between neighbouring ellipses rejects the neighbouring smaller ellipse. A high degree of overlap is defined by the larger ellipse encompassing more than 50% of the area of the smaller ellipse. The output of applying different rejection filters is displayed in Fig.9.

4 EXPERIMENTS

4.1 Dataset

We choose two datasets to test our proposed algorithm. The first dataset (Dataset-I, Fig.1.a), comprise images taken from a hand-held camera of crystals present in rod and circular shapes. It consists of 2 subsets of 10 images, each image has an average of 200 crystals. Each of these two subsets contains different percentages of rod and circular shaped crystals. The second dataset (Dataset-II, Fig.1.b), consists of microscopic images of similarly shaped

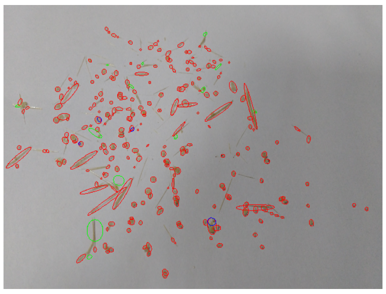


Figure 9: Effect of Rejection Filter on Dataset-I. Red ellipses are final fitted ellipse after rejection filters, Green ellipses are rejected due to *masking rejection* filter, Blue ellipses are rejected due to *area overlap* filter.

rectangular crystals. It consists of 15 images, each image has an average of 100 crystals. The images in both datasets are of dimension 640 x 480 pixels. The ground truth is generated by manually marking crystal's center in all the images contained in both datasets. Dataset-II represents a generic homogeneous dataset that most existing algorithms work with, such datasets contain similarly shaped crystals overlapping each other. On the other hand Dataset-I is a heterogeneous dataset, containing multi-shaped crystals overlapping each other. We demonstrate the novelty of our proposed algorithm on this dataset.

4.2 Performance Measures

To evaluate our algorithm performance, first we compare our results with two baselines followed by verification of results with the experimental crystal mixture data. For evaluation we use precision and recall as follows:

$$Precision = \frac{TP}{TP+FP} \tag{5}$$

$$Recall = \frac{TP}{TP+FN} \tag{6}$$

where TP is True Positive and it is defined the number of correctly segmented crystals, False Negative (FN) is the number of unsegmented crystals and False Positive (FP) is the number of incorrectly segmented crystals. The values of TP, FN and FP are calculated manually and are taken average over each of the datasets.

As a sanity check we use Jaccard Similarity Coefficient (JSC) to check the accuracy of our segmentation (Choi et al., 2010). A binary map of the segmented object O_s and the ground truth particle O_g is created. Distance between the ground truth particle's center and the fitted ellipse's (or rod's) center for a segmented particle is used as the parameter for binary map generation. The JSC is computed as follows:

$$JSC = \frac{n(O_s \cap O_g)}{n(O_s \cup O_g)} \tag{7}$$

The distance threshold value (JSC threshold, ρ) for the binary map is set to 8 pixels. It means that if the distance between the ground truth crystal's center and fitted ellipse's center on segmented crystal is less than 8 pixels then they are mapped together in the binary map i.e. added to the $O_s \cap O_g$ set.

The average JSC (AJSC) value over each dataset is thus used as a third measure to evaluate the segmentation performance. Fig.10 presents the variation of AJSC value of our proposed algorithm with different values of threshold ρ for both the datasets. Note, here a higher value of ρ represents a higher leniency in measuring the accuracy for segmentation. Our algorithm achieves very similar AJSC values for both the datasets for low ρ values.

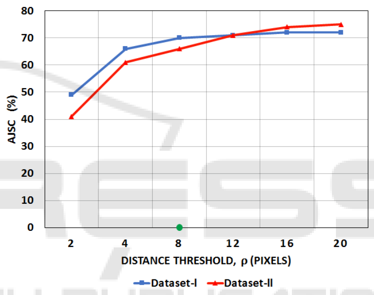


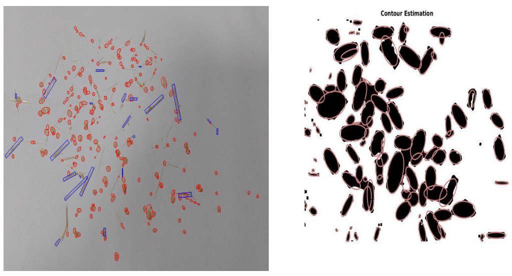
Figure 10: AJSC performance of the proposed segmentation method with different values of distance threshold, ρ.

4.3 Results

Fig.11.a and Fig.11.b show results of our proposed segmentation algorithm implemented on the two different set of crystal image samples, Fig.1.a and Fig.3.b respectively.

The performance of the proposed segmentation method is compared with two existing segmentation algorithms, Segmentation of Partially Overlapping Nanoparticles (SPON) (Zafari et al., 2015b) and Segmentation of Overlapping Elliptical Objects (SOEO) (Zafari et al., 2015a). The SPON and SOEO methods are particularly chosen as it is previously applied for segmentation of overlapping convex objects being represented by an elliptical shape. The implementation made by the corresponding authors is used for SPON (Zafari et al., 2015b) and SOEO (Zafari et al., 2015a).

Examples of visual comparison of the segmentation results for both the datasets are presented in the Fig.12 and 13. Distinctive segmentation efficiency of

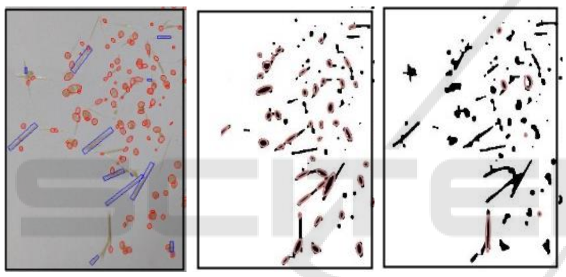


(a) Blue boxes represent rod crystals, Red ellipses represent circular crystals.

(b) Red ellipses represent similarly shaped crystals.

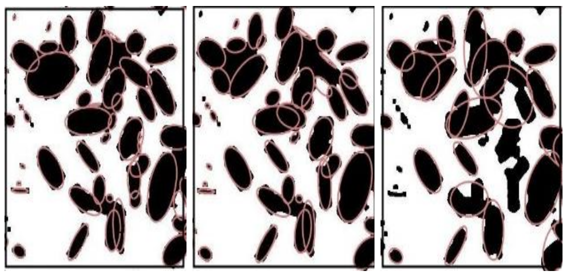
Figure 11: Final segmentation result of our proposed algorithm on sample image from Dataset-I(a) and Dataset-II(b).

our proposed algorithm could be observed as compared to SPON and SOEO. Binarized image after pre-processing is used as an input to the SPON and SOEO algorithm since the pre-processing step is dependent on the experimental setup.



(a) Proposed (b) SPON (c) SOEO

Figure 12: Dataset-I segmentation comparison. Results of proposed algorithm is displayed on original crystal image whereas SPON and SOEO on binarized crystal image.



(a) Proposed (b) SPON (c) SOEO

Figure 13: Dataset-II segmentation comparison. All the results are displayed on the binarised crystal image.

The corresponding performance statistics of the competing methods applied to the two datasets are presented in Table1 and Table2. Our algorithm outperforms both SPON and SOEO in terms of Recall and AJSC criterias for both the datasets whereas comparable results are obtained for Precision criteria. For Dataset-I the gain is substantial with our algorithm outperforming SPON by 36% on Recall. SOEO gives a very poor score for Dataset-I (2% Recall) reaffirming the challenges of segmenting images with multi-shaped objects. We also achieve substantially higher AJSC values for both the datasets indicating improved accuracy of placing segments in the image for our algorithm. ρ is taken to be 8 *pixels* for computing the AJSC value.

Table 1: Dataset-I performance comparison.

Methods	Recall %	Precision %	AJSC %
Proposed	**87**	85	**70**
SPON	51	**89**	28
SOEO	02	75	04

Table 2: Dataset-II performance comparison.

Methods	Recall %	Precision %	AJSC %
Proposed	**82**	**87**	**66**
SPON	81	82	31
SOEO	39	76	23

We provide a verification of our segmentation algorithm using composition analysis. Dataset-I contains images of the crystal p-Amino benzoic acid which is present in two shapes namely circular (prism) and the rod (needle). The circular shaped crystal is called the β-polymorph whereas the rod shaped crystal is called the α-polymorph. Using known dimensions of the two shapes from Section 3.2, the volumes are computed by estimating rods as cylinders and circular crystals as spheres. Weight percentage of each of these polymorphs are then computed using known density, 1.369 g/ml and 1.379 g/ml for α-polymorph and β-polymorph respectively.

Two sets of crystal samples are prepared by mixing different known weight percentage of each of the polymorph. Using our algorithm, for each of the set, average weight percentage of each shape is computed and checked against known weight percentage of the two samples. The corresponding performance statistics for the verification step is shown in Table 3. The algorithm has an average error of 1.95% over the actual weight percentages.

Finally, we do a complexity analysis. For K detected contour points, M corner points and P concave points the algorithm has the following complexity :

$$O(K*(M+P^2)) \tag{8}$$

The proposed method is implemented in Python, using a PC with a 2.60 GHz CPU and 6GB of RAM. The computational time of our proposed algorithm is around 0.5s per input image of size 640 x 480 pixels.

Table 3: Verification from weight analysis, Actual : known sample percentage; Image: calculated weight percentage from proposed algorithm.

Polymorphs	Sample A		Sample B	
	Actual	Image	Actual	Image
α wt.(%)	0.0	3.5	20.0	19.6
β wt.(%)	100.0	96.5	80.0	80.4

5 CONCLUSIONS

A new method for segmenting convex objects with high degree of overlap is developed in this paper. The algorithm uses concave point for segmentation and heuristic based approach for adaptive shape fitting. The key novelty of our work is to segment multi-shaped convex objects with high degree of overlap present in high density in an image. The proposed algorithm is tested on two different datasets, the first dataset contains multi-shaped crystals and the other contains similar shaped crystals. The algorithm performance is compared against two competing algorithms, SPON and SOEO, with our proposed algorithm outperforming both.

REFERENCES

Adams, R. and Bischof, L. (1994). Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence*, 16(6):641–647.

Álvarez, L., Baumela, L., Henríquez, P., and Márquez-Neila, P. (2010). Morphological snakes. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2197–2202. IEEE.

Baggett, D., Nakaya, M.-a., McAuliffe, M., Yamaguchi, T. P., and Lockett, S. (2005). Whole cell segmentation in solid tissue sections. *Cytometry Part A*, 67(2):137–143.

Choi, S.-S., Cha, S.-H., and Tappert, C. C. (2010). A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48.

Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181.

Gudla, P. R., Nandy, K., Collins, J., Meaburn, K., Misteli, T., and Lockett, S. (2008). A high-throughput system for segmenting nuclei using multiscale techniques. *Cytometry Part A*, 73(5):451–466.

Haralick, R. M. and Shapiro, L. G. (1985). Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132.

Malpica, N., Ortiz de Solórzano, C., Vaquero, J. J., Santos, A., Vallcorba, I., Garcia-Sagredo, J. M., and Pozo, F. d. (1997). Applying watershed algorithms to the segmentation of clustered nuclei.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66.

Pal, N. R. and Pal, S. K. (1993). A review on image segmentation techniques. *Pattern recognition*, 26(9):1277–1294.

Rastgarpour, M. and Shanbehzadeh, J. (2011). Application of ai techniques in medical image segmentation and novel categorization of available methods and. In *Tools, Proceedings of the International MultiConference of Engineers and Computer Scientists 2011 Vol I, IMECS 2011, March 16-18, 2011, Hong Kong*. Citeseer.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905.

Suzuki, S. et al. (1985). Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46.

Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):583–598.

Yang, X., Li, H., and Zhou, X. (2006). Nuclei segmentation using marker-controlled watershed, tracking using mean-shift, and kalman filter in time-lapse microscopy. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53(11):2405–2414.

Zafari, S., Eerola, T., Sampo, J., Kälviäinen, H., and Haario, H. (2015a). Segmentation of overlapping elliptical objects in silhouette images. *IEEE Transactions on Image Processing*, 24(12):5942–5952.

Zafari, S., Eerola, T., Sampo, J., Kälviäinen, H., and Haario, H. (2015b). Segmentation of partially overlapping nanoparticles using concave points. In *International Symposium on Visual Computing*, pages 187–197. Springer.

Zhang, Q. and Pless, R. (2006). Segmenting multiple familiar objects under mutual occlusion. In *Image Processing, 2006 IEEE International Conference on*, pages 197–200. IEEE.

Zhang, W.-H., Jiang, X., and Liu, Y.-M. (2012). A method for recognizing overlapping elliptical bubbles in bubble image. *Pattern Recognition Letters*, 33(12):1543–1548.