Arrow Based Region of Interest Segmentation in Biomedical Images

A DISSERTATION
submitted towards the fulfillment of the
requirement for the award of the degree of
MASTER OF TECHNOLOGY
in
Computer Science and Engineering

By

NAVED ALAM

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE
Roorkee - 247667, India

JUNE 2016

AUTHOR'S DECLARATION

I declare that the work presented in this dissertation with title **"Arrow Based Region of Interest Segmentation in Biomedical Images"** towards the fulfillment of the requirement for the award of the degree of **Master of Technology** in **Computer Science & Engineering** submitted in the **Department of Computer Science & Engineering, Indian Institute of Technology Roorkee, India** is an authentic record of my own work carried out during the period from **June 2015 to May 2016** under the supervision of **Dr. Partha Pratim Roy**, Assistant Professor, Department of Computer Science and Engineering, Indian Institutes of Technology, Roorkee and **Dr. K.C. Santosh**, Assistant Professor, Department of Computer Science, University of Dakota, USA. The content of this dissertation has not been submitted by me for the award of any other degree of this or any other institute.

DATE: ..... SIGNED: .....

PLACE: ..... (NAVED ALAM)

CERTIFICATE

This is to certify that the statement made by the candidate is correct to the best of my knowledge and belief.

DATE: ..... SIGNED: .....

(DR. PARTHA PRATIM ROY)
Assistant Professor
Indian Institutes of Technology, Roorkee

PUBLICATION

Part of the work presented in this dissertation has been published in the following conference and journal-

Santosh, K. C., Naved Alam, Partha Pratim Roy, Laurent Wendling, Sameer Antani, and George R. Thoma. **"A Simple and Efficient Arrowhead Detection Technique in Biomedical Images."** International Journal of Pattern Recognition and Artificial Intelligence (2016).

Santosh, K. C., Naved Alam, Partha Pratim Roy, Laurent Wendling, Sameer Antani, and George R. Thoma. **"Arrowhead detection in biomedical images."** Electronic Imaging 2016, Document Recognition and Retrieval XXIII, no. 17 (2016): 1-7.

Naved Alam, Santosh, K. C. and Partha Pratim Roy. **"Arrow Detection in Biomedical Images using Sequential Classifier."** International Journal of Machine Learning and Cybernetics, 2016 (submitted).

Abstract

Biomedical images are often complex, and contain several regions that are annotated using arrows. Annotated arrow detection is a critical precursor to region-of-interest (ROI) labelling, which is useful in content-based image retrieval (CBIR). Different image layers are first segmented via fuzzy binarization. Candidate regions are then checked whether they are arrows by using BLSTM classifier, where Npen++ features are used. In case of low confidence score (i.e., BLSTM classifier score), we take convexity defect-based arrowhead detection technique into account. The detected arrow are then used to segment the region-of-interest (ROI). Our test results on biomedical images from imageCLEF 2010 collection outperforms the existing state-of-the-art arrow detection techniques. Our region segmentation techniques is preliminary approach to segment regions from detected arrows.

DEDICATION AND ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Dr. Partha Pratim Roy for the continuous support of my study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my study.

My co-advisor, Dr. K.C. Santosh, has been always there to listen and give advice. His insightful comments and constructive criticisms at different stages of my research were thought-provoking and they helped me focus my ideas. I am grateful to him for holding me to a high research standard and enforcing strict validations for each research result, and thus teaching me how to do research.

I am also grateful to the Dept. of Computer Science, IIT Roorkee for providing valuable resources to aid my research.

Finally, hearty thanks to my parents and siblings, who encouraged me in good times, and motivated me in the bad times, without which this dissertation would not have been possible.

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

In today's age, volume of images with diverse content are generated every day. It is important to store them and retrieve them efficiently for different application. Visual Contents and Images are very often used for searching images from large image databases. Content Based Image Retrieval (CBIR) can be used in various professional area like architecture, geography, artwork etc. However, it has got a strong application in biomedical imaging. Medical concepts and special medical cases are often illustrated by images. Medical images can play very important role in diagnosis of medical conditions and clinical decision support system (CDS). It will also help in implementing best practices and acquiring technical skill in area of medicines.

Many general purpose image retrieval system have developed for this purpose. They are primarily classified as:

1. **Text based Image Retrieval** - Keywords are used in order to search for interesting images in case of text based image retrieval system. Text information is extracted from the journal article using language processing tool. Journal's title and abstract can provide some details about the figure present in the journal. More detail information can be extracted from the figure's caption and discussion where the figure is mentioned in the journal text. Text based retrieval can provide good result but the relevance of the retrieved images and article are often not satisfactory. Hence, there was a need for retrieval based on query image content.

2. **Content Based Image Retrieval** - With new technologies, the focus have move to

the use of low-level features instead of keywords. In content based image retrieval (CBIR), we analyse content of the image to check for the similar image in database. The focus is to find the images which are most similar to the query image instead of trying to finding the image through keywords. This is due to the fact that the keywords in caption and discussion may not be summing the medical concept associated with the image. Besides, the processing of the article to find the relevant keyword for image do not always provide us with optimal result.

Biomedical journal article retrieval is primarily based on text based retrieval with no significant importance given to the use images in the articles until recently due to the drawback of text based retrieval. There have been quite a progress in this area in past year. Still, there are unsolved problem in this area which continue to attract researcher from various domain.

Authors frequently use annotation and pointers (like arrows, special symbols etc.) superimposed on figures and images to highlight the Region of Interest (ROI). In this case, the ROI is more important than whole of the image and provide us with more relevant information. These pointers are then referenced in the article text and figure caption to portray the medical concept which it describes. Thus, in CBIR, a lot of research are being focused on finding out these pointers and annotation in the images, followed by segmenting of the ROI to which these annotation points and then mapping these ROI to certain medical concepts. During retrieval, these ROIs can be matched with the ROIs in the query image and the retrieval can be thus performed based on the percentage of match.

1.1 Annotation and Pointers

Medical images are generally complex by nature and mostly comprises of several region. Among these region, only few region are relevant to the condition discussed in article text or caption. As said earlier, authors frequently use annotation and pointers (like arrows, special symbols etc.) superimposed on figures and images to highlight the Region of Interest (ROI). These pointers are then referenced in the article text and figure caption highlighting some medical concepts. Research are being done to develop methods for recognizing such pointers on medical images and illustrations. Identifying these pointer can aid in extracting the relevant region in the image that are likely to be more relevant to the article text than the whole image. Once, the relevant region is identified, these region can be annotated with biomedical concept which they signifies and can be mapped

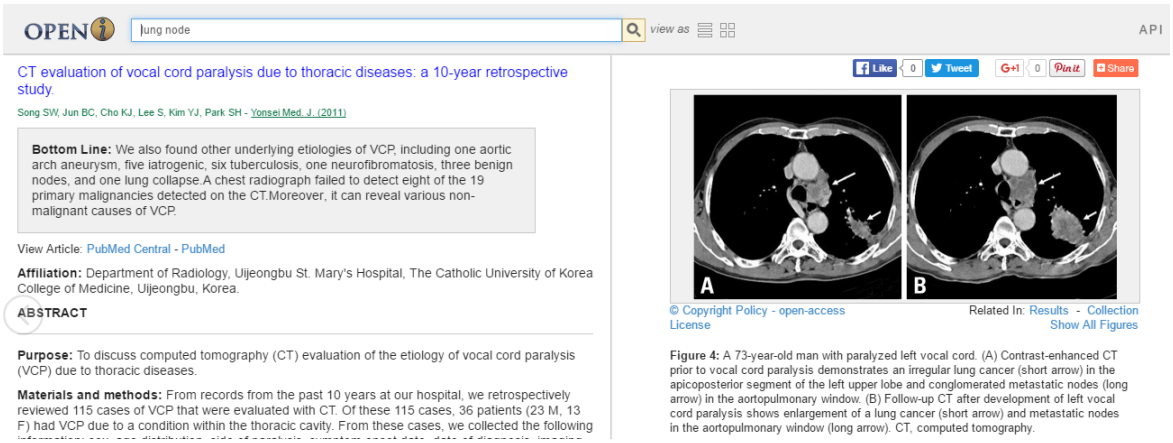
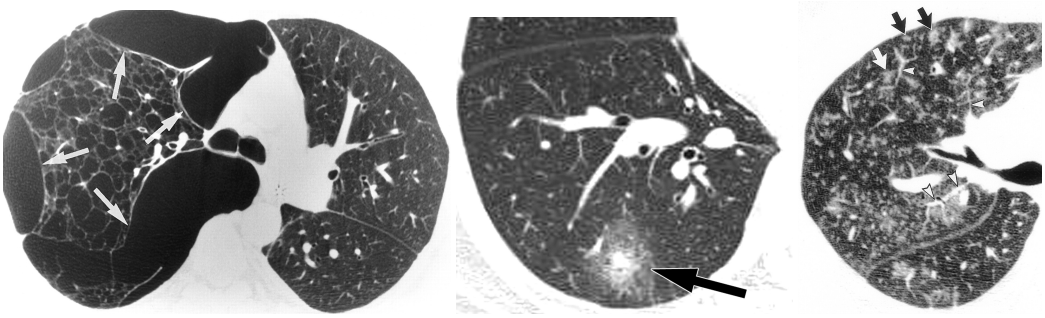


Figure 1.1: NLM's open-i image retrieval search engine (url: https://openi.nlm.nih.gov), the illustration highlights the importance of using arrow in biomedical images (i.e. its location pointing ROI and relationship between the texts and ROI).



(a) (b) (c)

Figure 1.2: Examples showing different types of arrows pointing specific image regions. These are taken from published biomedical articles.

into the image database. Therefore, detecting these pointers and annotation (which primarily consist of arrows) can help us to identify the meaningful region and improve CBIR performance [4, 5].

CHAPTER 2

RELATED WORKS

In CBIR, few techniques are reported in literature for detection of arrows overlaid in biomedical images. These techniques depend upon segmenting text like ad symbol like objects, sparse pixel vectorization and local or global thresholding.

In [6], Dori et al, proposed a technique to detect arrows based on sparse pixel vectorization [7]. It relies on the cross sectional runs concept (or width runs concept) of black image regions (assuming that the arrows are in black). An interesting application is utilizes by the technique but, it is restricted to machine printed line images. Other techniques used features such as eccentricity, convex area and solidity [8]. These features can define regular arrows (i.e., straight arrows showing left, right, top and bottom). Cheng et al used text-like and arrow-like objects separation, assuming that arrows are overlaid in either white or black color with respect to the background [9]. From the binary image, arrow-like object separation employs a fixed sized mask (after removing the small objects and noise as in [8]), which are then used for feature computation such as axis ratio, minor and major axis lengths, Euler number, area, and solidity. A recent study uses a boundary detection and pointer region to handle arrows which are distorted [10], which is followed by edge detection techniques and fixed thresholds as reported in [1, 11]. Overlapping region are computed using these candidates and are then binarized to extract the expected pointers's boundary. Fundamentally, edge-based arrow detection techniques are limited by the weak-edge problem [8–10]. Besides, the techniques rely on hard thresholding (either global or local) cannot be applied in general. This means that a hard threshold cue often weakens the decision. For edge detection in

binary or grayscale images, most state-of-the-art methods use classical algorithms like Roberts, Sobel and Canny edge detection. Template-based methods are limited since they require new templates to train new images. Also, it may be necessary to re-evaluate the threshold values when new images are used. Edge-based techniques are still considered since sampling points can be remarkably compact compared to solid regions, especially when broken boundaries are recoverable [12]. In biomedical images, one of the major issues for a broken boundary is non-homogeneous intensity distribution, where pointers overlap with content.

Detection of arrow based on the arrow edge are limited by the weak edges. This is discussed in [13], [14] and [10]. Good binarization techniques also tend to miss weak edges. The primary reason being local or global hard thresholding. Most of the method rely upon famous algorithm like Sobel, Robert and Canny edge detection. Template based method also do not perform well as they require a new templates for each of the new arrow images. Also, the threshold value needs to be re-evaluated whenever a new template is added to the system.

Few of the State-of-art method to detect arrows are explained in this chapter.

2.1 MRF-HMM based pointer recognition

In [1], author proposed a Markov Random field followed by hidden Markov model based arrow detection method.

2.1.1 Pointer Model

Pointer boundary was modeled as a random field consisting of a number of line segment. Also, there exist some contextual dependency among line segment present in the arrow. Instead of defining the pointer model based on all the line segment present in the pointer, several boundary part found in the pointers and made up of three consecutive line segments were identified and a unique labeled was created for each of them. Contextual dependency between each boundary part were also established. Labeled boundary parts and their contextual dependency was used to train the MRF model. Each boundary part (segment) was denoted by 2-tuple feature vector formed by the angle measurement between three lines.

Since, line segment were labeled based on it neighborhood, an unknown boundary (labeling configuration) can never be identified directly by MRF model. Thus, it was

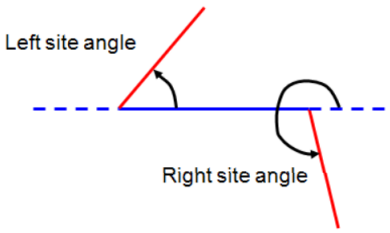


Figure 2.1: Angle Measurement used to create 2-tuple feature vector. [1]

needed to classify a labeling configuration into proper pointer class. A hidden Markov model based classifier is used to classify these labeling configuration into proper classes.

2.1.2 Pointer Recognition Method

Edge detection algorithm was applied on the query image followed by polygon approximation to extract the line segment. Line segment in the segmented query image is labeled by identifying the boundary parts present in the query images and the relationship of the boundary parts with it neighboring segment through MRF model developed before. For finding the boundary part in query image, a neighboring system was used. An ordered sequence of line segments from a boundary were obtained. Initially, two neighbor segment i.e. segment of indices i-1 and i+1 were used. However, considering two direct neighbor can cause some error in labeling as a noise segment can be present. This was solved by considering two immediate neighbors i.e. for each of the segment, four pairs of neighbor were possible, viz., (i-1, i+1), (i-1, i+2), (i-2, i+1), and (i-2, i+2). The Algorithm uses a belief propagation function to determine the best set of label for pointer boundary from the all combination of four possible boundary parts. Dynamic programming is used to find the best labeling configuration which maximizes the Belief Propagation function value.

Once the best labeling configuration for the query image is available, hidden Markov model based classifier is used to classify a pointer boundary into three classes viz, Curved Arrow, Straight Arrows and Arrowheads.

2.1.3 Evaluation

ImageCLEF08 data Set was used for evaluation. 82% of boundaries were correctly labeled and classified. 18% failure were due to i) Incorrect labeling by MRF, ii) Unknown boundary part, iii) Improper line segment due to weak edges.

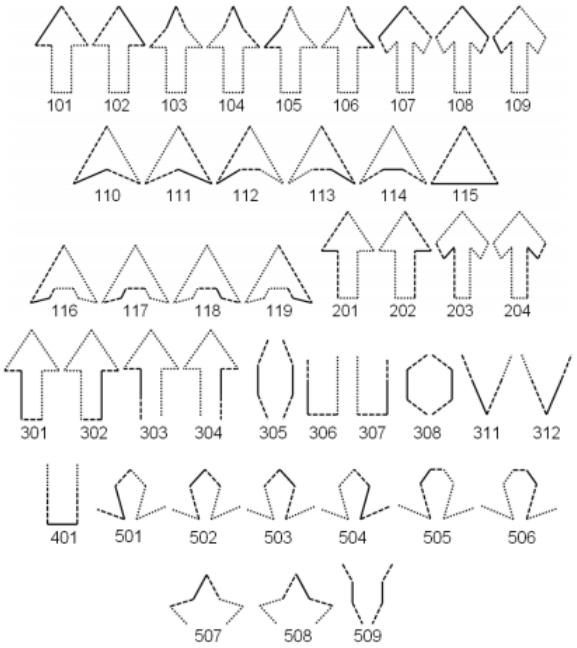


Figure 2.2: 43 label set defined for pointers. A label is assigned to the solid line segment. The dashed segment are the neighboring segment. [1]

2.2 Active Shape Model Based Pointer Recognizer

This was proposed in [2] as an improvement over MRF-HMM based pointer recognizer. It proposed a new pointer segmentation algorithm and an ASM model overlaid on previous MRF-HMM recognizer.

2.2.1 Pointer Segmentation

MRF-HMM based pointer recognizer was using the edge based segmentation. Edge based segmentation has a major drawback. Due to smoothing effect of the edge detection algorithm, small pointers tends to lose their tail information. Thus, correct label were not identified by MRF model for this pointer leading to miss. A Canny edge detection and region growing based segmentation algorithm was proposed. The algorithm has two phases. In first phase known as histogram analysis phase, interior region of the pointer is analyzed to get min and max of the intensity values. Also, a seed point is detected inside the pointer boundary. In the second phase, region growing is used to segment out the entire region of the target pointer.

In [10], an updated method for pointer segmentation has been proposed by the same author in order to overcome the weak-edge problem. Figure 4.4 depict the proposed

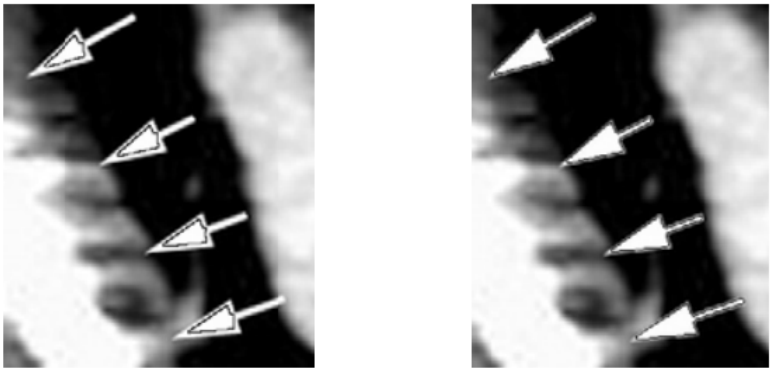


Figure 2.3: (a) Boundary Extraction by Canny Edge Detection, (b) Pointer boundary after applying Region Growing. [2]

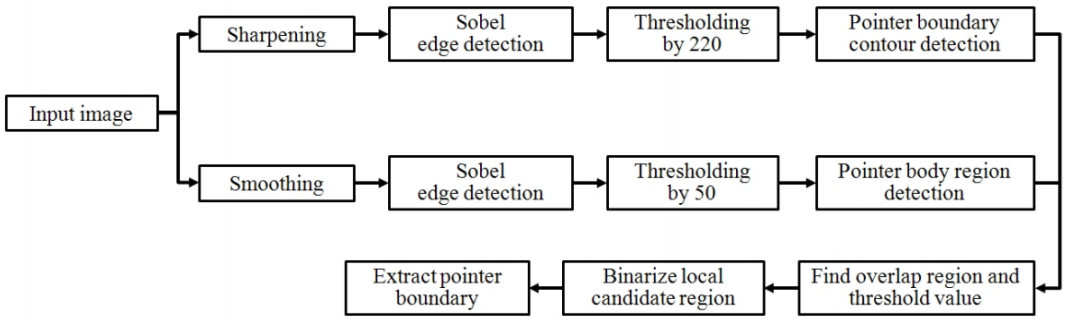


Figure 2.4: Proposed pointer segmentation method as in [2]

methods. It follows two different path, one to detect pointer boundary and other to detect pointer body region. A concrete candidate is found out by merging the two results. From this region, final threshold value for pointer segmentation is calculated.

2.2.2 Pointer Recognition

MRF based recognizer are not able to recognize pointers with some level of distortion even though the pointers is well separated from the background. Thus, an ASM based recognizer was developed to recognize pointers with some distortion. Unlike other approaches, we need to consider the pointer recognition explicitly in case of ASM. A segmented pointer is thus therefore rotated and mirrored several time and each of these rotated pointers are matched with the model pointers. Similarity is measured by finding the count of pixels with same intensity level from overlapped pointer regions, and dividing it by area (width x height) of the segmented pointer image. Pointer are first recognized by

Figure 2.5: Type of Pointers recognized by ASM. [2]

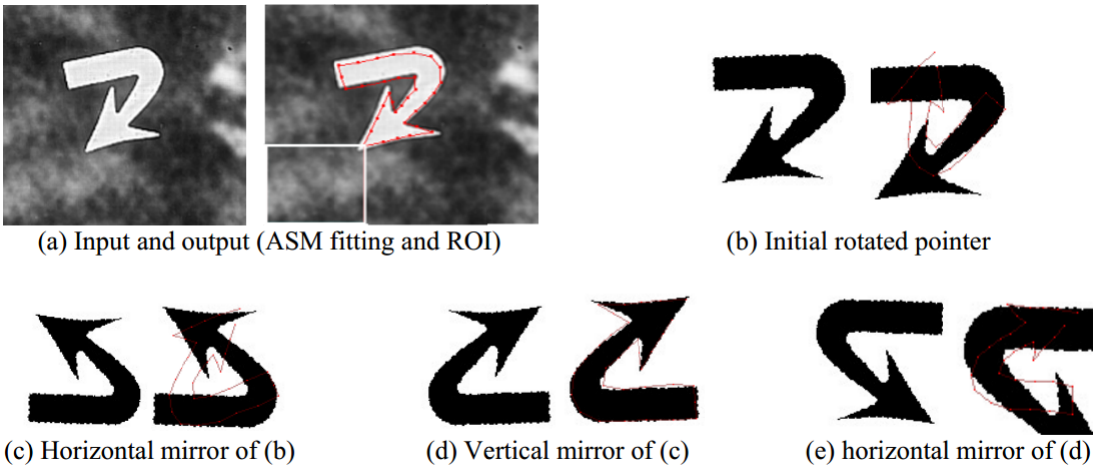


Figure 2.6: Example of ASM Recognition (Left: Input to ASM, Right: final fitting result with overlaid fitted model). [2]

the MRF model and ASM recognizer is only used when MRF model doesn't detect it as pointer.

2.3 Arrow Detection through Discreet Arrow Signature Matching

In [3], the author used geometric property of the arrow and introduced an arrow signature from certain key points associated with arrow. Images are binarized using the fuzzy binarization and segmented into different region. Signature is generated for each of these region and is compared with the theoretical signature to a match. The method is explained in detail below.

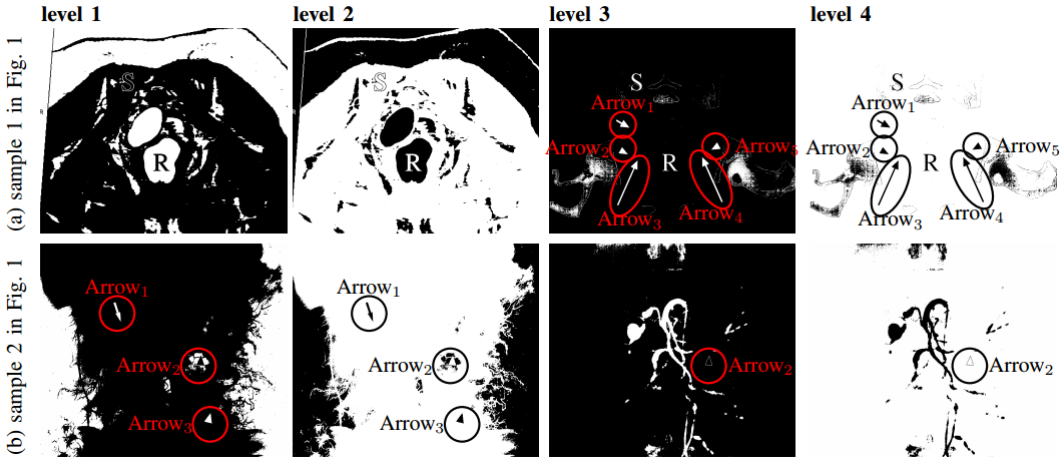


Figure 2.7: Four different binarized images from fuzzy binarization. It is to be noted, that in Sample 2, arrow 2 is not clearly visible at first two level but is clearly visible in third level. [3]

2.3.1 Binarization

Arrow can be present at either low intensity or high intensity. Also, it can overlapped or surrounded by texture area. Hence, fixed threshold based binarization tool will not be able to extract this arrow perfectly. An adaptive binarization tool known as fuzzy binarization is proposed. The fuzzy binarization is based on the fuzzy partitioning of the 2D histogram considering local variations and gray level intensities. Through this fuzzy binarization, four different threshold level are used and connected component are extracted from each of this four binarized images.

2.3.2 Signature Computation

For each of the CC, we need to find the arrowhead point. Potential candidate can be found out by four orthogonal scan. In each scan, we need to capture the first pixel in the arrow. Four orthogonal scan will return four arrowhead candidate. One of them will be the actual arrowhead. Once, we have the arrowhead point, set of lines passing through this point is projected in all direction and the segment of this line which lies inside the region is found out. The length of this line at different directions gives us the arrow signature. This signature is then matched with the theoretical signature for similarities.

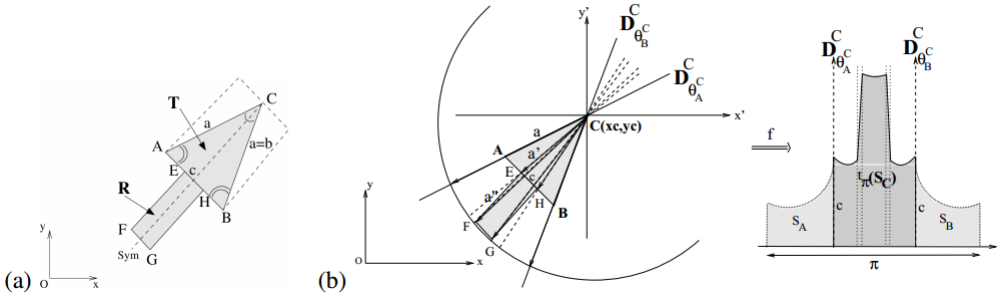


Figure 2.8: (a) An arrow, (b) Signature of arrow as computed from point C (Arrowhead point). [3]

2.3.3 Evaluation

Using signature matching, a precision value of 93% was achieved with a recall value of 86%. This method doesn't work very well for curved arrows as the signature would be different for different curved tails.

CHAPTER 3

ARROW DETECTION USING CONVEXITY DEFECT APPROACH

As discussed in previous chapter, arrowhead detection is a critical precursor to region-of-interest (ROI) labeling and image content analysis. In this chapter, we proposed an arrowhead cropping technique for arrow detection. To detect arrowheads, images are first binarized using fuzzy binarization technique to segment a set of candidates based on connected component principle. To select arrow candidates, we use convexity defect-based cropping, which is followed by template matching via dynamic programming. The similarity score via dynamic time warping (DTW) confirms the presence of arrows in the image. Our test on biomedical images from imageCLEF 2010 collection shows the interest of the technique.

3.1 Contribution outline

Our method can be summarized as shown in Fig. 4.1. It relies on a grayscale fuzzy binarization process at different levels. Similar to previously reported work [3], candidates are segmented based on connected component (CC) principle. These candidates are filtered using hull convexity defect-based technique. This step helps prune artefacts (or unwanted noisy connected components) and store arrowhead-like candidates. Next we perform template matching using dynamic programming to confirm whether the candidate is an arrowhead. In our assessment, An arrow is said to be detected if their

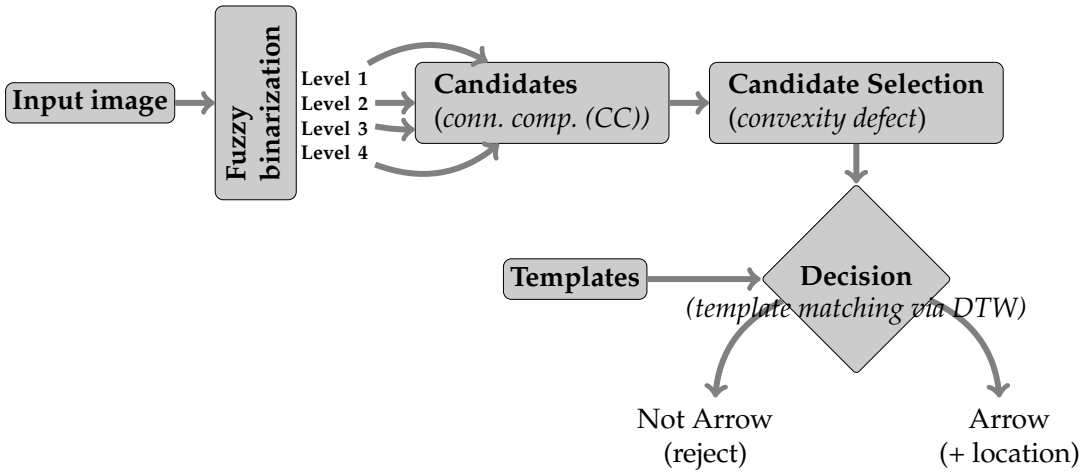


Figure 3.1: Overall system workflow in block format. Block-wise explanation can be found in Section 3.2

matching score exceeds an empirically set threshold.

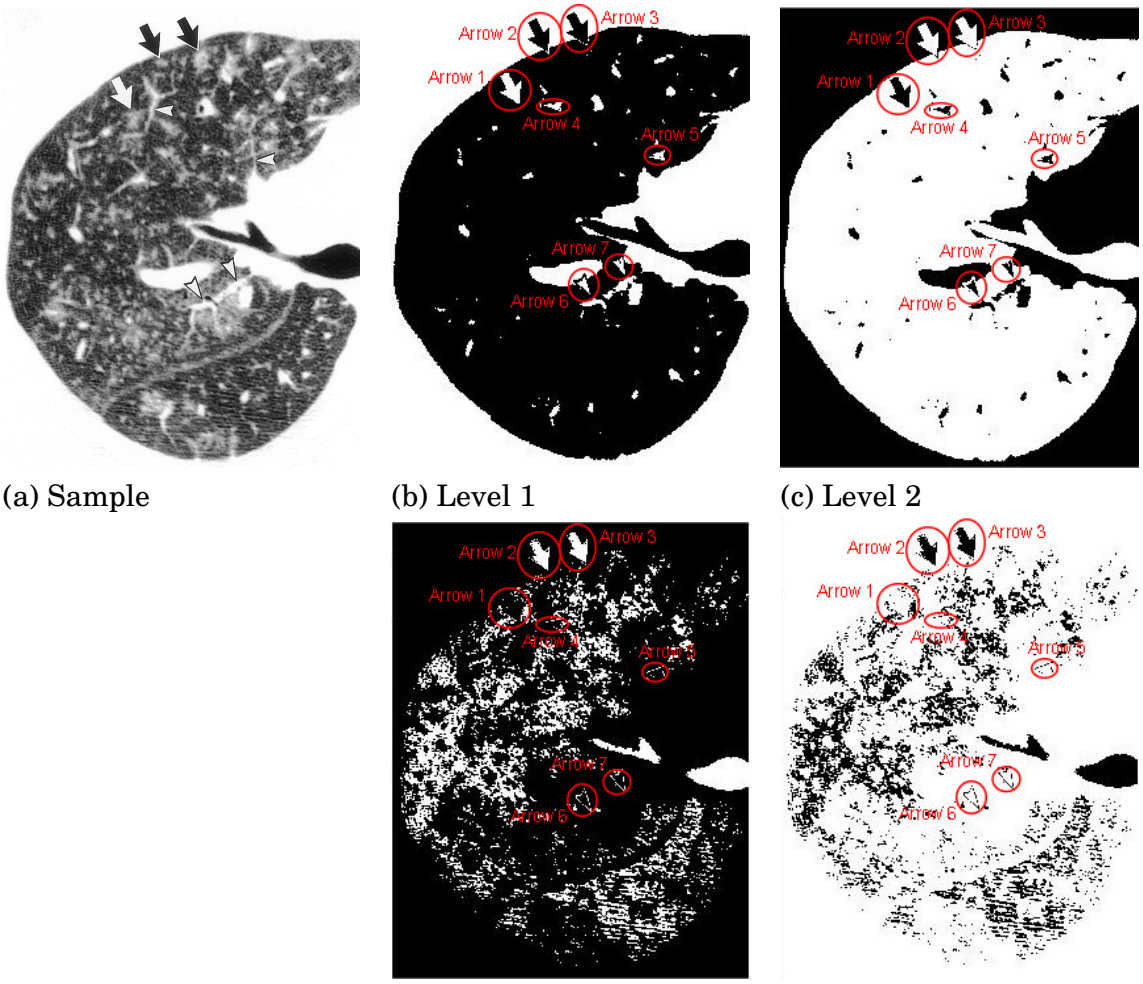
Unlike the common state-of-the-art methods, our method uses four different levels of fuzzy binarization. This ensures that overlaid arrow candidates are not missed. however, it may result in repetitions. We note that the primary variation in an arrow appearance is due to its tail (shape and size). Therefore, our method limits itself to just detecting arrowheads.

Rest of the chapter is organized as follows. In Section 3.2, we explain our method in detail, where it mainly includes binarization process (Section 3.2.1) and candidate selection (Section 3.2.2). Results are reported in Section 4.3.

3.2 Method

3.2.1 Multilayer image segmentation

In medical images (see Fig. 1.2), arrow can appear ta both low intensity or high intensity with respect to the background to increase their visibility. Additionally, arrow can be blurred, overlapped or surrounded by artefact similar in intensity with the arrow. In such situation, fixed threshold based typical binarization tools failed to segment the arrow candidate. Hence, we proposed an adaptive binarization tool. It rely on the concept of fuzzy partition of 2D histogram of image by considering the local variation and gray level intensities [15]. Threshold is then automatically set through the computation of 2D Z-function criteria. It is based on the optimization of fuzzy entropy and employs two



(a) Sample (b) Level 1 (c) Level 2

(d) Level 3 (e) Level 4

Figure 3.2: Fuzzy binarization (of Fig. 1.2 (c)): four different levels (level 1 to level 4), where the arrows are circled in red. Arrows can appear both in white and black with respect to the background color.

kernels: high level cuts and low level cuts. We also consider their inversion and thus work on four different binarized levels as shown in fig. 3.2. Arrow candidate are circled in red with respect to the background color in fig. 3.2. The main idea of using four different levels of binarization is not to miss the overlaid arrows. Furthermore, since the arrows are repeated in other levels of binarization, deformed and/or distorted arrows can be discarded . Connected components (CCs) are then segmented as candidate regions. From a pool of several candidates, we are required to select arrow-like candidates.

Figure 3.3: Examples showing the changes in tail structure. Further, an absence of the tail is also possible.

3.2.2 Candidate selection

Our candidate selection process is based on the characteristics of the arrowhead, which can typically be represented by a triangle. Unlike the previously reported work [3], we do not take tail information into account. One of the primary reasons is that it may vary geometric signatures computed from extreme points of a triangle (i.e., triplet) because tail structures tend to vary from time to time. Such a change will affect overall appearance of the arrow (Fig. 3.3). After we detect arrowhead, we will take the corresponding tail into account since both came from the same CC.

To detect an arrowhead, the following steps are carried out: 1) convexity defect-based arrowhead candidate cropping; and 2) arrowhead candidate matching with the templates.

3.2.2.1 Convexity defect-based arrowhead candidate cropping

To select arrow-like candidates, we apply hull convexity defect concept (see Fig. 3.4). A set of points along the contour of the binary CC are said to be convex if the line segments joining each pair of its points is contained within within the CC. In convex combination, a weight or coefficient w_i is accredited to every point x_i in the set S. The coefficients assigned are all positive and the sum of all coefficients is one. A weighted average of the points is then computed through these coefficients. For each combination of coefficients, the convex combination thus formed is a convex hull point. Choosing coefficients in all possible ways would give us the whole convex hull. In single formula, convex hull can be written as following set: $\left\{\sum_{i=1}^{|S|} w_i x_i | (\forall i : w_i \geq 0) \wedge \sum_{i=1}^{|S|} w_i = 1\right\}$. This indicate that the convex hull of a finite point set $S \in \mathbb{R}^n$ is a convex polygon when n = 2. In Fig. 3.4 (b), an example is shown. Using such a convex hull, we attempt to remove tail since their exists convex shaped silhouettes in both sides (see Fig. 3.4 (c)), which is computed by subtracting an original candidate from the convex hull. In Fig. 3.4 (d), the convexity defect region is shown, which is just a convex hull of both convex shaped silhouettes. At the end, in Fig. 3.4 (e), arrowhead candidate(s) is(are) selected by subtracting an original

(a) (b) (c) (d) (e)

Figure 3.4: Arrowhead candidate cropping: (a) an arrow, (b) convex hull, (c) convexity defect, (d) a complete convexity defect region, and (e) arrowhead candidates.

image with the convexity defect region.

3.2.2.2 Arrowhead candidate matching with template

To confirm arrowhead candidates (see Fig. 3.4), we apply a template matching technique. We extract a feature along the contour and match with the predefined templates using dynamic time warping (DTW) technique. The arrowhead candidate is confirmed when the similarity score crosses the empirically designed threshold.

Feature extraction. Along the contour, we have a set of coordinate points, $P = \{p_i\}_{i=1,\ldots,n}$. To extract feature vector (f), we compute the change in angle with respect to x-axis from any consecutive pair, $f = \{\alpha_i\}_{i=1,\ldots,n}$, where $\alpha_i = \arctan\left(\frac{y_l - y_{l-1}}{x_l - x_{l-1}}\right)$. This goes in a cyclic order either clock wise or anti-clock wise. In our feature vector, continuous redundancy of α_i can be possible, $\alpha_i = \alpha_{i+j}, j = 1,\ldots,m$, where $m \leq n$. Therefore, few representative pixels (called the dominant pixel) are desired to express the shape of the contours. Through polygonal approximation [16–18], the contour is represented using fewer points such that properties of contours are retained. Next the geometrical characteristics like concavities or inflexion points can be evaluated. In our implementation, to make it simple and effective, we compute the difference between the angles and check whether it crosses the threshold, ϵ. The choice of ϵ is usually user-defined. This means we take α_i if $|\alpha_i - \alpha i + 1| \leq \epsilon$. Like most line fitting/polygonal approximation (or dominant point detection) methods, it can be made non-parametric by using the error bound due to digitization as a termination condition. Fig. 3.5 shows three examples, where the changes in angles are shown at all dominant points. To make the feature vector rotation invariant, one needs to follow either clock-wise or anticlock-wise to compute changes in angles.

Dynamic time warping (DTW). DTW help to find the differences between two different length non-linear sequences [19, 20]. In Fig. 3.5, one can notice the variations in feature vector from one arrowhead to another. Let us consider two feature sequences: $f_1 = \{\alpha_i\}_{1=1,\ldots,n}$ and $f_2 = \{\beta_j\}_{j=1,\ldots,m}$ of size n and m, respectively. It is desired to get the

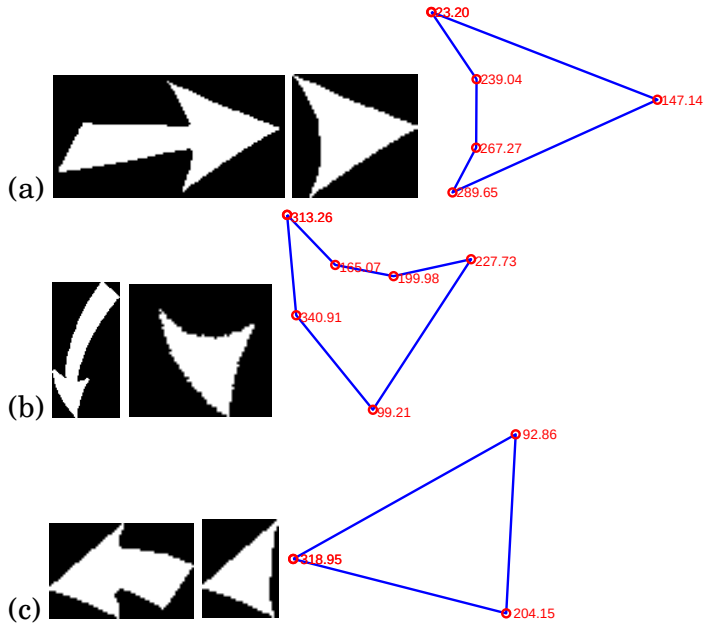


Figure 3.5: Three examples showing a complete process (from left to right) starting from an original candidate (resulting from fuzzy binarization - see Fig. 3.2), arrowhead cropping (see Fig. 3.4) to feature extraction after polygonal approximation.

optimal alignment between two sequences through this algorithm. At first, an $n \times m$ matrix is constructed. Next for each events, local distance $\delta(i,j)$ between the elements, e_i and e_j is calculated i.e., $\delta(i,j) = (e_i - e_j)^2$. Let $D(i,j)$ be the global distance up to (i,j),

$$D(k,l) \quad = \quad \min \begin{bmatrix} D(k-1,l-1), \\ D(k-1,l), \\ D(k,l-1) \end{bmatrix} + \delta(k,l)$$

with an initial condition $D(1,1) = \delta(1,1)$. Tis is to allow warping path going diagonally from starting node $(1,1)$ to end (n,m). The objective is to find the least cost path. The warping path therefore represent the difference in the cost of the compared signatures. Mathematically, the warping path is, $W = \{w_k\}_{k=1\ldots l}$, where $\max(i,j) \leq l < i+j-1$ and k^{th} element of W is $w(i,j)_k \in [1:n] \times [1:m]$ for $k \in [1:l]$. The optimised warping path W satisfies the following three conditions: boundary condition, monotonocity condition and continuity condition. We then define the global distance between f_1 and f_2 as,

$$\Delta(f_1, f_2) = \frac{D(n,m)}{l}.$$

The last element of the $n \times m$ matrix gives the DTW-distance between f_1 and f_2, which is normalised by l i.e., the number of discrete warping steps along the diagonal DTW-matrix.

Overall, DTW measures the similarity between two sequences, and can be summarized as follows.

1) If the cropped candidate is not actually an arrowhead, DTW results in high cost. The results are opposite to those arrowhead candidates.

2) Thanks to DTW, noise in arrowhead (along the contour) does not let the cost to go beyond the threshold. This means that some of the arrowheads with noisy artefacts connected to them are still detected.

3) Feature extraction and DTW matching techniques provide robustness to rotation and scale changes. As an example, $\Delta($  $,$  $) = 0.00$ and $\Delta($  $,$  $) = 0.00$

3.3 Experiments

3.3.1 Datasets, ground-truth and evaluation protocol

Testing is done on the well-known dataset of imageCLEF[21]. It contain 298 CT scan of the chest. At least, one arrow is present in each of these images. There are altogether 1049 arrows in the dataset. Groundtruths were created for all the pointers in all of the images in dataset. Information like arrow type, location, color and pointing direction were the information included in the groundtruth. For validation of our results, we used presicion, recall and F1-score as our evaluation criteria.

$$\text{precision} = \frac{m_1}{M},\ \text{recall} = \frac{m_1}{N}\ \text{ and } F_1\ \text{score} = 2\left(\frac{(m_1/M)\times(m_1/N)}{(m_1/M)+(m_1/N)}\right),$$

where m1 is the total number of arrow correctly detected by our approach, M is the total number of candidate (including false postive) detected as arrow and N is the total number of arrows present in dataset.

3.3.2 Result and analysis

3.3.2.1 Our results

Table 4.5 shows the performance evaluation scores in terms of precision, recall and F1-score. In the reported results, we prioritize the recall measure since we do not like to miss arrow candidates. The method achieved F_1 score of 91.09%.

Table 3.1: Our Result and Performance comparison (in %) with previously reported techniques.

		Previously reported methods			
Metric	Our method	M1 [9]	M2 [11]	M3 [10]	M4 [3]
Precision	88.50	81.10	22.80	84.20	93.14
Recall	93.80	74.10	77.80	81.60	86.92
F_1-score	91.09	77.00	35.00	83.00	89.94

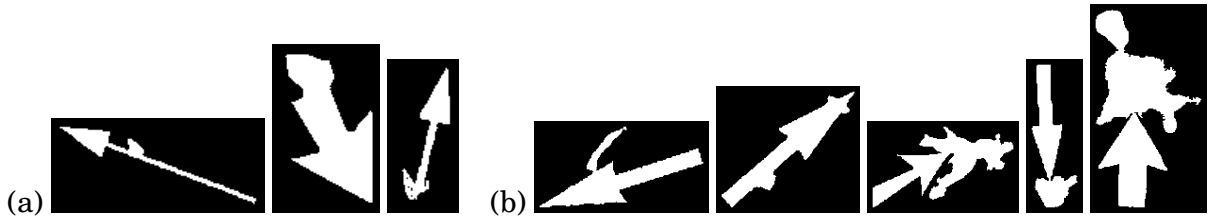


Figure 3.6: Examples of when the proposed method succeeds and fails: (a) noisy artefacts connected along the tail does not affect the method, and (b) they do largely affect when connected with arrowhead.

Our method is able to detect arrowheads regardless their tail structure. But, if the shape of the arrowhead is affected by noisy artefacts, the proposed method fails. Fig. 4.7 shows both examples: noisy artefacts that are connected along the tail, and noisy artefacts that are connected with arrowhead. Also, the method does not detect highly curved arrows since convexity defect-based arrowhead cropping does not yield expected arrowhead candidates.

CHAPTER 4

ARROW DETECTION USING SEQUENTIAL CLASSIFIER

In this chapter, we explore a machine learning approach for detections of arrows in Biomedical images. This approach too uses fuzzy binarization technique for segmentation as described in previous chapter. Candidate regions are then checked whether they are arrows by using BLSTM classifier, where Npen++ features are used. In case of high error value (i.e., BLSTM classifier score) in classification, we take convexity defect-based arrowhead detection technique into account. Our test on medical images from 2010 imageCLEF lung database surpass the existing state-of-the-art techniques for arrow detection .

4.1 Contribution outline

Our method can be summarized as shown in Fig. 4.1. It rely on a grayscale fuzzy binarization technique at different levels [3], where candidates are segmented based on connected component (CC) principle. Unlike the common state-of-the-art technique, we adopted four levels of fuzzy binarization. This ensures that overlaid arrow candidates are not missed. Npen++ and the Radon features are computed for these candidates and are tested through BLSTM-trained arrow model. BLSTM tries to classify the candidate into arrow or non-arrow class. If the cross entropy error value of the BLSTM classifier is below the threshold, the candidate is classified as an arrow. Otherwise, the candidate is passed through convexity defect-based technique, discussed in previous chapter. The latter step prunes artefacts (i.e., unwanted noisy object and/or image regions) and stores

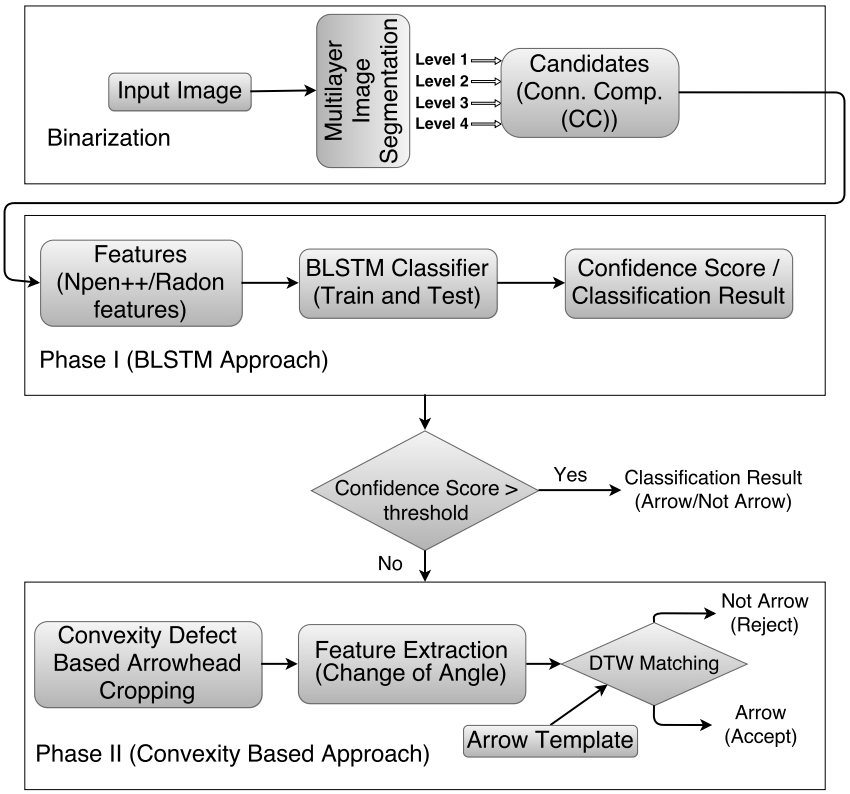


Figure 4.1: Overall system workflow in block format. Block-wise explanation can be found in Section 3.2.

arrowhead-like candidates, since it deals with just the arrowhead.

4.2 Method

Our arrow detection technique comprises of two classifiers analyzing the candidates in sequence to determine whether the candidate is an arrow. It consists of a neural network based BLSTM classifier followed by convexity-defect based arrowhead detection. In other words, BLSTM classifies each candidate as arrow (and non-arrow) candidates with some cross-entropy error score. This cross-entropy error defines how confident BLSTM is for classification. If cross entropy error value score is below the threshold, BLSTM classifies the arrow candidate. If not, candidates are then tested through convexity-defect based arrowhead detector.

4.2.1 Features

The performance of any neural network classifier depends on the features that are used to represent the candidates. In our approach, we have tested the performance with two different feature descriptors: Npen++ [22] and the Radon feature. Npen++ features are expected to be worked for well defined geometric patterns (such as arrows), including curvature, curliness and orientation. Similarly, the Radon feature suits well suited for the patterns since projection in the Radon space changes 2D arrow image into 1D signal that can considered as strokes. BLSTM classifier used below have been known very well for strokes and gesture recognition [23]. Therefore, their integration is relevant.

4.2.1.1 Npen++ feature descriptor

Npen++ features were originally introduced for handwriting recognition. It actually comprises a number of features computed along the handwriting trajectory. The normalized sequence of the captured coordinates $(x(i), y(i))$ forms the input to the system. A sequence of features is computed along this trajectory. But, not all of these features are relevant for our arrow detection approach. Most of the features of Npen++ depend on the baseline. In original Npen++ recognizer, the baseline $b(x(i))$ coincide with the original writing line on which a text or word was written. However, in our approach, we consider the lowest line parallel to x-axis passing through the contour of arrow as baseline.

Vertical Position. The vertical position of the point $(x(i), y(i))$ is the vertical distance between $b(x(i))$ and $y(i)$ where $b(x(i))$ is the baseline's y-value for ith point on the contour.

Orientation. At point $(x(i),(i)y)$, the local writing direction is described as:

$$cos\alpha(i) = \frac{\Delta x(i)}{\Delta s(i)} \tag{4.1}$$

$$sin\alpha(i) = \frac{\Delta y(i)}{\Delta s(i)} \tag{4.2}$$

where $\Delta x, \Delta y$ and Δs are computed as follows:

$$\Delta s(i) = \sqrt{\Delta x^2 + \Delta y^2} \tag{4.3}$$

$$\Delta x(i) = x(i-1) - x(i+1) \tag{4.4}$$

$$\Delta y(i) = y(i-1) - y(i+1) \tag{4.5}$$

Curvature. The computation of curvature at a point $(x(i), y(i))$ consider the previous and next point to that point and is describe as follows:

$$cos\beta(i) = cos\alpha(i-1) * cos\alpha(i+1) + sin\alpha(i-1) * sin\alpha(i+1) \tag{4.6}$$

$$sin\beta(i) = cos\alpha(i-1) * sin\alpha(i+1) + sin\alpha(i-1) * cos\alpha(i+1) \tag{4.7}$$

Note that this sequence does not actually compute curvature but compute angular difference which suffice in our case.

Aspect. The ratio of the height to the width of the bounding box accommodating the succeeding and preceding points of $(x(i), y(i))$ is the aspect $A(i)$ of the contour at point i. It is computed as:

$$A(i) = \frac{\Delta y(i) - \Delta x(i)}{\Delta y(i) + \Delta x(i)} \tag{4.8}$$

Curliness. The deviation from a straight line in the neighborhood of $(x(i), y(i))$ describe Curliness $C(i)$ feature. It is computed as the ratio of length of the contour and larger side of the bounding box.

$$C(i) = \frac{L(i)}{max(\Delta x, \Delta y)} - 2 \tag{4.9}$$

where $L(i)$ is the length of contour in the neighborhood of the point computed as the sum of all line segments in the neighborhood of the point. Δx and Δy are width and height of the bounding box.

We have applied Npenn++ feature for selection of arrow-like candidates. Since the features of Npenn++ are rotation dependent, the orientation of the arrow is calculated using minor axis and major axis calculation. Each of the arrow is rotated along it orientation such that it lay flat along the x-axis.

For training, each of the arrow is rotated along it orientation using the above method and is then rotated perpendicularly and mirrored several time so as to consider all of it flat orientation into consideration.

4.2.1.2 Radon Transform

The radon transform computes projections of an image matrix along specified directions. A projection of a two-dimensional function $f(x, y)$ is a set of line integrals. It is computed

by calculating the length of line integrals from multiple sources along parallel paths, or beams, in a certain direction. In general, the Radon transform of $f(x,y)$ is the line integral of f parallel to the y-axis is

$$R_\theta(x') = \int_{\infty}^{-\infty} f(x'cos\theta - y'sin\theta, x'sin\theta - y'cos\theta)dy' \tag{4.10}$$

where

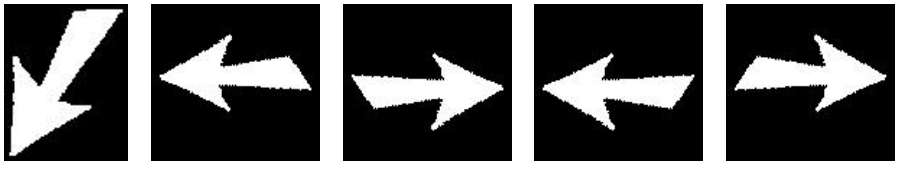
$$\begin{bmatrix} \text{x'} \\ \text{y'} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ \text{-sin}\,\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \text{x} \\ \text{y} \end{bmatrix}.$$

The consecutive beams are projected at a distance of 1 pixel. The image is rotated several time around it center. The beams are projected multiple times for each rotation of the image. The set of projections for all the rotation of the image is used to represent the image after the transformation. Since arrow is a regular geometric shape, the radon transform of arrow tend to have regularities.

4.2.2 BLSTM Classifier

Again, we are motivated by the use of the recently introduced bidirectional long short-term memory(BLSTM) [23]. In simple words, the BLSTM is a form of recurrent neural network having connections between the nodes forming a directed cycle. It, thus, provide a 'memory' of earlier states of network.

Long short-term memory (LSTM) layer. The LSTM network node follow a distinct architecture known as memory block. A memory cell forms the most important component of memory block and it interact with network with the help of three gates, viz., an input gate, an output gate and a forgot gate. This helps memory cells retain it state for a long time which help in modeling the context at feature level. The input signal is processed in both directions: forward and backward by two different layers, and improve the ID sequence recognition. Next layer is combined the output of both the layer to form



(a) (b) (c) (d) (e)

Figure 4.2: Arrows are rotated and mirrored.

the feature map. Like convolutional neural network, LSTM layer can have multiple forward and backward layer and output layer can have multiple feature maps. Stacking of different LSTM layers is also possible using max-pooling subsampling.

Candidate selection. Using the above feature, a BLSTM model for the arrow is created. A candidate is passed tested through the trained BLSTM model. BLSTM classifier provides yields fairly significant amount of false positives. It is primarily because of the presence of artefacts. Like the state-of-the-art works, the candidate classified by BLSTM with high confidence score (low cross entropy error) is accepted. If not, remaining candidates are passed though second phase of screening (i.e., convexity defect-based arrowhead detection). The latter phase aims to eliminate the false positives that are coming from BLSTM).

4.3 Experiments

4.3.1 Datasets, ground-truth and evaluation protocol

Testing is done on the well-known dataset of imageCLEF[21]. It contain 298 CT scan of the chest. At least, one arrow is present in each of these images. There are altogether 1049 arrows in the dataset. Groundtruths were created for all the pointers in all of the images in dataset. Information like arrow type, location, color and pointing direction were the information included in the groundtruth. For validation of our results, we used presicion, recall and F1-score as our evaluation criteria.

$$\text{precision} = \frac{m_1}{M},\ \text{recall} = \frac{m_1}{N}\ \text{ and } F_1\ \text{score} = 2\left(\frac{(m_1/M)\times(m_1/N)}{(m_1/M)+(m_1/N)}\right),$$

where m1 is the total number of arrow correctly detected by our approach, M is the total number of candidate (including false postive) detected as arrow and N is the total number of arrows present in dataset.

4.3.2 Our results and analysis

Table 4.1 shows the performance evaluation scores in terms of precision, recall and F_1 score, where BLSTM alone has a very good recall value but, with large large false positives. This way, F_1 score is still satisfactory. It holds true for both features: Npen++

Table 4.1: Performance of the proposed system (in %).

Metric	BLSTM classifier with features: Npen++	Radon	Convexity defect-based approach	BLSTM classifier followed by Convexity defect-based Npen++	Radon
Precision	13.96	6.14	88.50	95.39	92.69
Recall	98.43	97.31	93.80	99.21	99.10
F_1-score	24.45	11.55	90.09	97.26	95.79

and the Radon that are applied separately. False positives are produced when we receive low confidence score (high cross entropy error). The candidates having low confidence scores (via BLSTM) are then correctly be detected/classified at the latter phase, thanks to convexity defect-based arrowhead cropping (see Chapter 3). On the whole, the sequential classifier that combines BLSTM and convexity defect-based arrowhead cropping provides better results.

As reminder, our BLSTM classifier uses a threshold on the cross entropy error value to decide whether we should further test the candidate with convexity defect-based approach. If not, we accept as it is classified by BLSTM classifier. Graph in Fig. 4.3 shows the change in the accuracy of the BLSTM with change in this threshold value. We observe that we get better results for the threshold values that are in the range: [0.2 – 0.7].

Processing times, on average, for both Npen++ and the Radon features are almost identical. For each candidate image, Npen++ feature took approximatively 12.7 ms and 14.4 ms, respectively without and with convexity defect-based approaches. On the other hand, with radon feature, each candidate image took 15.5 ms and 16.2 ms , respectively without and with convexity defect-based approaches.

4.3.2.1 Combining Npen++ and Radon feature

We also tried combining both the classifier trained with Npen++ and Radon feature. We used two different setup for combining these two classifier 1) Weighted average of score of Npen++ and Radon feature; and 2) Classification using the feature which gives high confidence among the two.

Weighted average. In this setup, One of the classifier is given some weight, α. The weight of other classifier is calculeted as $(1 - \alpha)$. The output of both the classifier are

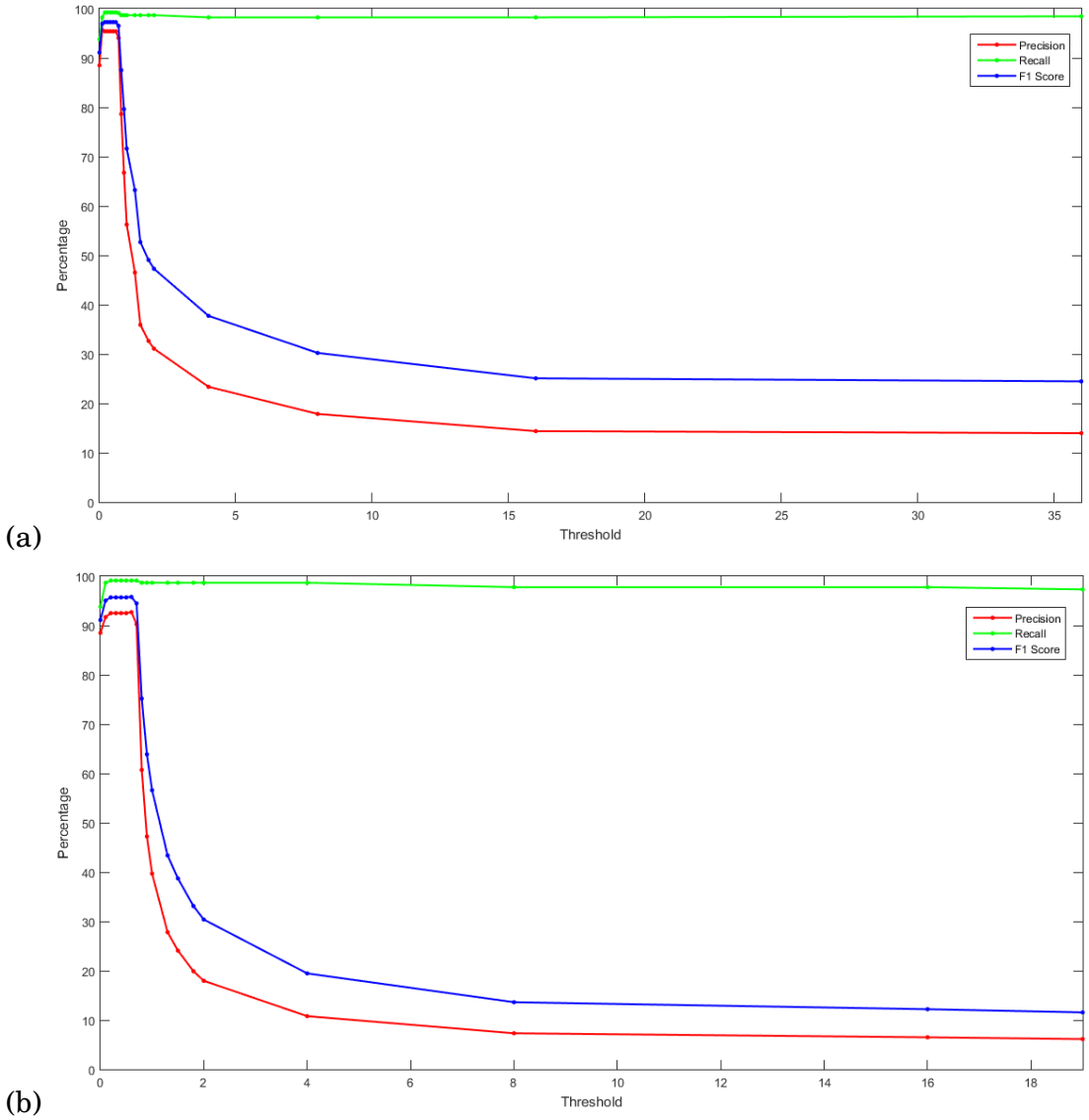


Figure 4.3: Graph showing the change in accuracies with change in the threshold value of Cross Entropy Error for (a) Npen++ feature and (b) Radon feature.

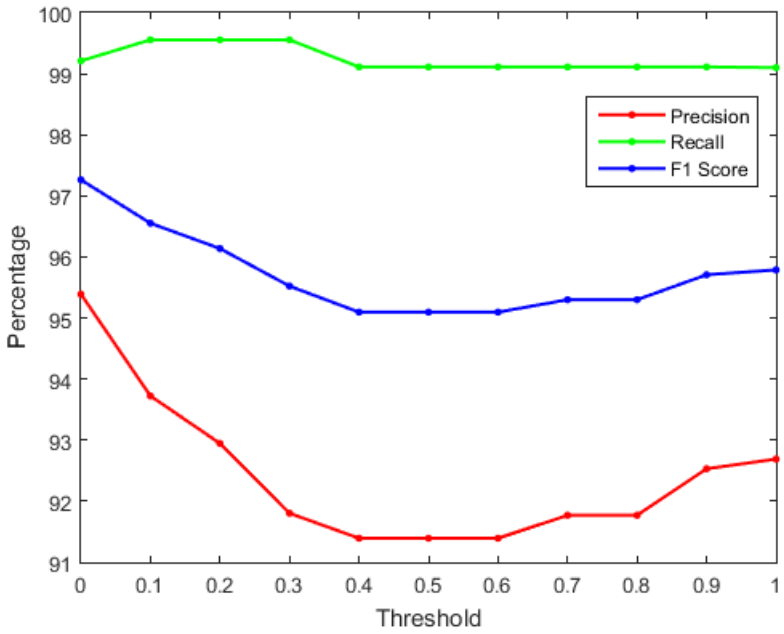


Figure 4.4: Graph showing the change in accuracies with different value of α.

multiplied with this weight and are then added to get the final classification score. This score is then used for classifying the candidate.

$$x_{avg} = \alpha \times x_{Npen++} + (1 - \alpha) \times x_{Radon} \tag{4.11}$$

Graph 4.4 show the change in accuracies with change in the value of α. Accuracies is maximum for Npen++ feature alone and tend to decrease with increase in the weight of Radon's score. This show that combining Radon trained classifier with Npen++ feature trained classifier does not add to the accuracies.

Parallel Combination. In this setup, each of the candidate image is passed through both Npen++ trained clasifier as well as Radon trained classifier in parallel. The result of the classifier classifying the candidate with high confidence score (low cross entropy error) is accepted as final. If both the classifier fails to classify the candidate with good confidence score, the candidate is passed through convexity defect based algorithm for final classification. This setup gave a boost of 1% in previously obtained accuracy.

The results of both the combination is shown in the table 4.2.

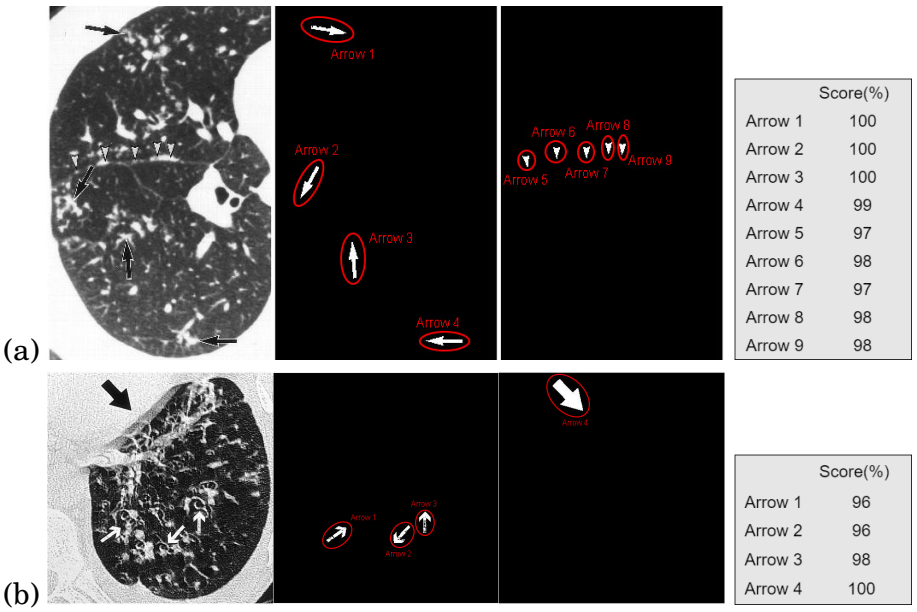


Figure 4.5: Examples showing different binarization levels are used to detect arrows. This demonstrates the idea of image inversion used in binarization since arrows are not just black-filled.

4.3.3 State-of-the-art comparison

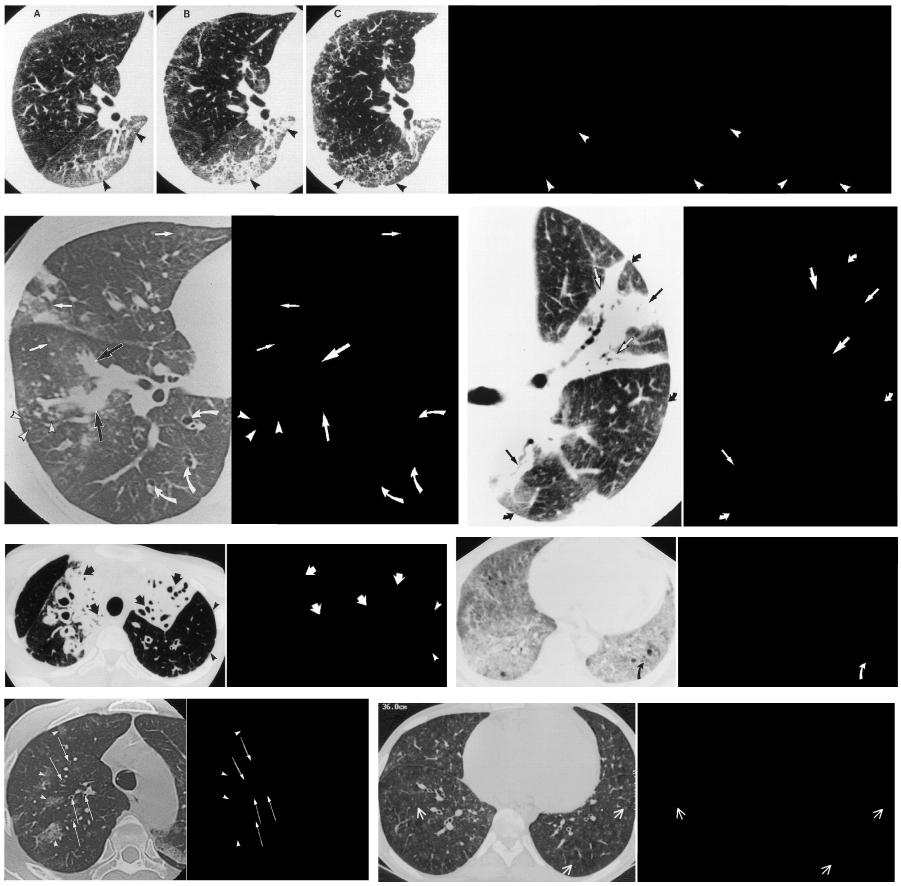
Further, the comparative study with state-of-the-art methods has been made. In this comparison, our benchmarking methods are categorized into two groups: 1) state-of-the-art methods that are specially designed for arrow detection; and 2) common template-based method by using well-known state-of-the-art shape descriptors.

4.3.3.1 Recent arrow detection methods

Four well-known methods from the state-of-the-art that are specially designed for arrow detection are used: 1) global thresholding-based method (M1) [9], 2) two edge-based methods (M2:M3) [10, 11], and 3) a template-free geometric signature-based method [3]. The results are provided in Table 4.5, where method 5 (M5) performs the best with

Table 4.2: Accuracies obtained by combining Npen++ trained classifier and Radon trained classifier.

Metric	Parallel Combination	Weighted Average
Precision	96.75	95.39
Recall	99.88	99.21
F_1-score	98.29	97.26



Continued on next page

Table 4.3: Performance comparison (in %) of previously reported methods.

	Previously reported methods				
Metric	M1 [9]	M2 [11]	M3 [10]	M4 [3]	M5 [24]
Precision	81.10	22.80	84.20	93.14	88.50
Recall	74.10	77.80	81.60	86.92	93.80
F_1-score	77.00	35.00	83.00	89.94	91.09

Table 4.4: Performance comparison (in %) of template based method.

	Template-based methods				
Metric	GFD [25]	SC [26]	ZM [27]	RT [28]	$\mathscr{D}$-Radon [29]
Precision	75.10	68.30	55.20	59.50	62.10
Recall	78.33	71.40	57.70	63.60	65.30
F_1-score	76.68	69.82	56.40	61.48	63.65

precision, recall and F_1 score values 95.39%, 99.21% and 97.26%, respectively.

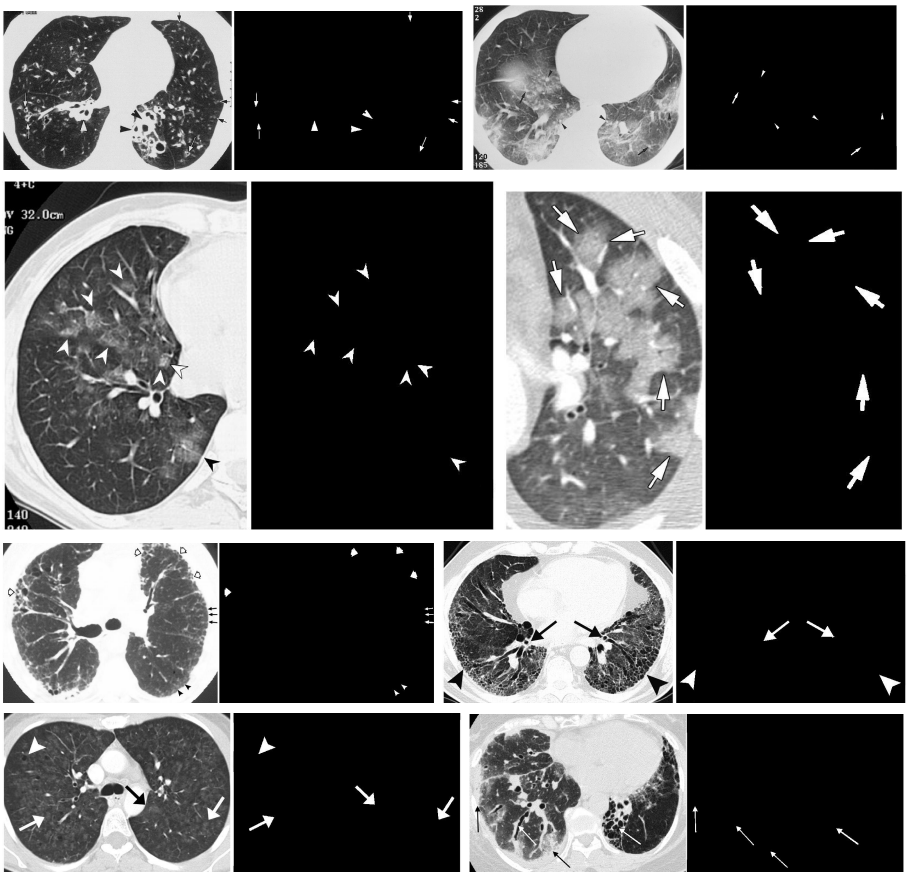


Figure 4.6: Examples illustrating arrow detection. Detected arrows (in white with black background) on the right, are the combined results of four different binarization levels.

(a) 

Figure 4.7: Examples of when the proposed method succeeds. Noisy artefacts connected along the tail does not affect.

Table 4.5: Performance comparison (in %) of our method with best of template best techniques and previously reported work.

Metric	Our method	GFD [25]	M5 [24]
Precision	96.75	75.10	88.50
Recall	99.88	78.33	93.80
F_1-score	98.29	76.68	91.09

4.3.3.2 Template-based methods

In case of template-based method, we created 11 templates (arrows) having different shapes (including sizes). The template size can further be extended in accordance with the dataset. To extract shape features, we took the most frequently used shape descriptors (in computer vision) from the state-of-the-art. They are 1) generic Fourier descriptor (GFD) [25], 2) shape context (SC) [26], 3) Zernike moment (ZM) [27], 4) R-transform (RT) [28] and 5) DTW-Radon [29]. As before, results (precision, recall and F_1-score) are provided in Table 4.5. Among all shape descriptors, GFD provides the best performance.

On the whole, considering such a dataset, the proposed method outperforms the best state-of-the-start arrow detection method by more than 6% F_1 score, and the template-based (shape descriptor) method by more than 20% F_1 score.

CHAPTER 5

REGION OF INTEREST EXTRACTION

As discussed earlier, arrow detection is precursor to region-of-interest (ROI) extraction and labeling. Our research interest lie in successful extraction and labeling of region-of-interest that is being pointed by the arrow. In this chapter, we propose a region growing based region-of-interest (RoI) extraction algorithm from arrows detected using algorithm from previous chapter. The detected arrow is first processed for arrowhead point detection. Seed point are generated using detected arrowhead point and region growing is used to finally segment out the ROI.

5.1 Method

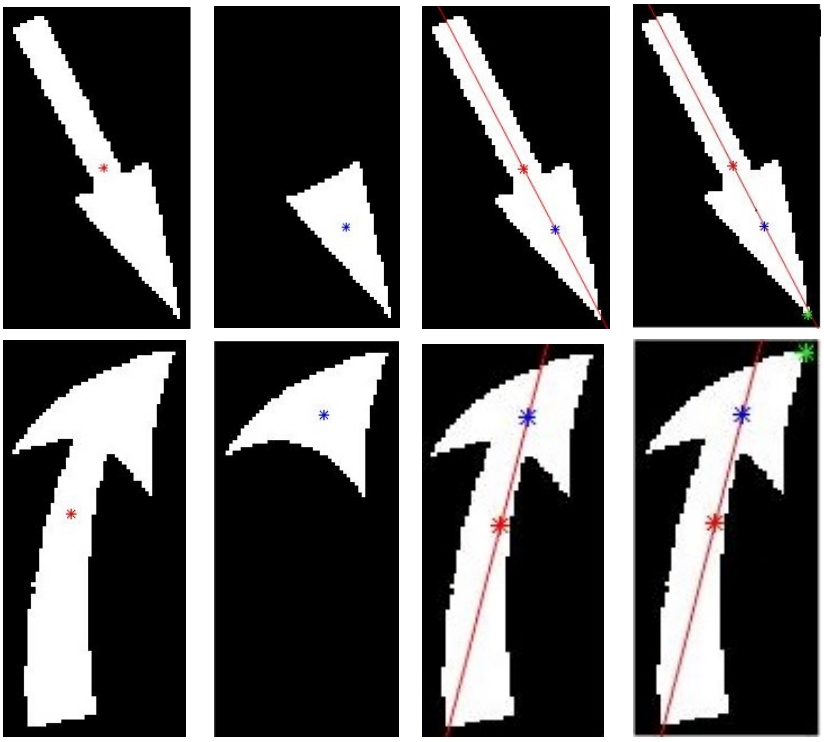
5.1.1 Arrowhead Detection

To detect arrowhead point, we calculated the centroid of the actual arrow and the centroid of the cropped arrow. The centroid of an two dimentional object is the center of mass point on which the object would balance when placed on a needle. Mathematically, the centroid $(\bar{x}, \bar{y})$ of a set of n-point (x_i, y_i) is given by,

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \tag{5.1}$$

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{5.2}$$

A vector originating from the centroid of the original arrow and passing through the centroid of the cropped arrowhead gives the direction of the location of the arrowhead.



(a) (b) (c) (d)

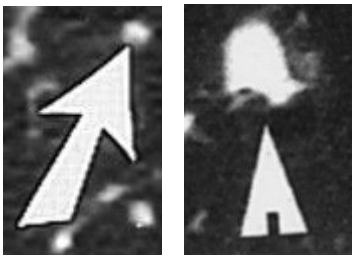
Figure 5.1: Arrowhead Detection: (a) Centroid of the original arrow (red asterisk), (b) Centroid of the cropped arrow (blue), (c) A reference line joining the two centroid, (d) Last positive pixel (green asterisk) in the direction of the line is the detected arrowhead.

The last pixel located on the arrow in the direction of vector give us the arrowhead point. Examples are shown in Figure 5.1.

5.1.2 Seed Point Generation and Region growing

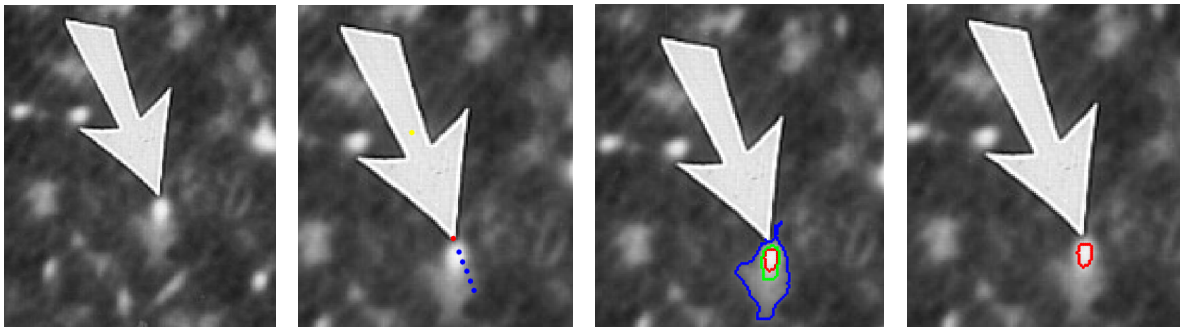
In order to use region growing to segment out the ROI, we need to have seed point in the region itself. One of the approach is to take the seed point at a particular distance from the arrow in the direction where arrow is pointing. However, this didn't work as the ROI can be located very close to the arrow or at quite a distance from the arrow. Figure 5.2 shows an example of the two different case. A point with fixed distance from the ROI would not work for all the case.

To overcome this problem, we took several seed points in the direction of arrow (shown in figure 5.3(b)). Few of these seed points would be lying on the ROI and would give us the ROI on region growing. While some other seed points would lie on the background. Applying region growing through these seed point would result in large region which we



(a) (b)

Figure 5.2: Example showing distance of ROI from Arrowhead. In first image, ROI is very close to the arrowhead while in second image, ROI is located at some distance.



(a) (b)

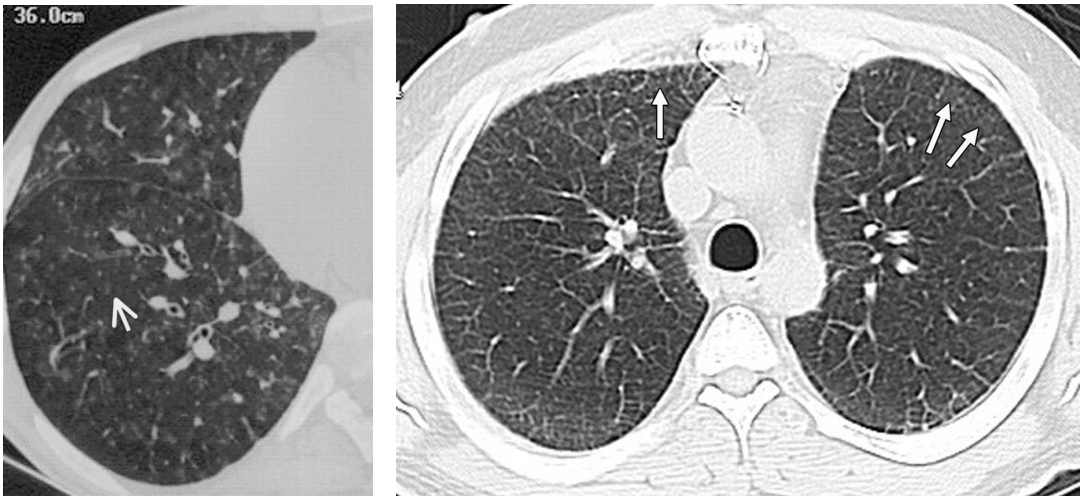
Figure 5.3: Roi Segmentation: (a) An arrow, (b) Seed points, (c) Different region detected near arrowhead, (d) Selected region

can filter out using the region area. In our algorithm, out of all the grown region through various seed point (see figure 5.3(c)), we consider the region with lowest area (see figure 5.3(d)) as the segmented ROI.

5.2 Experiments

5.2.1 Datasets and ground-truth

Testing is done on the well-known dataset of imageCLEF[21]. It contain 298 CT scan of the chest. At least, one arrow is present in each of these images. There are altogether 1049 arrows in the dataset. However, it us very difficult to point out the region exactly pointed by the arrow in some of these images (See figure 5.4). It is not possible to create ground-truth for such images without any medical expert. We have not consider such images in our experimentation. Out of 298 images, we have consider 133 images for testing. These 133 images have altogether 427 region pointed by arrows.



(a) (b)

Figure 5.4: Example showing images where exact roi cannot be clearly demarcated

Table 5.1: Performance of the proposed approach

Number of region present	Number of region properly segmented	Number of region segmented with some artifacts	Number of region missed
427	358	28	41

5.2.2 Our result and analysis

Table 5.1 shows the performance evaluation of our approach. Out of 427 regions, our algorithm was able to segment out 358 region without any errors. It was also able to detect another 28 region but with some extra artifacts. Figure 5.5 show some example where our approach was able to perform very well.

The two primary reason for missing out on segmenting some of the region properly were, 1) The region having very little intesity difference from its background, and 2) The region is attached to some other region and there is no clear demarcation between the two. An example of both is shown in 5.6.

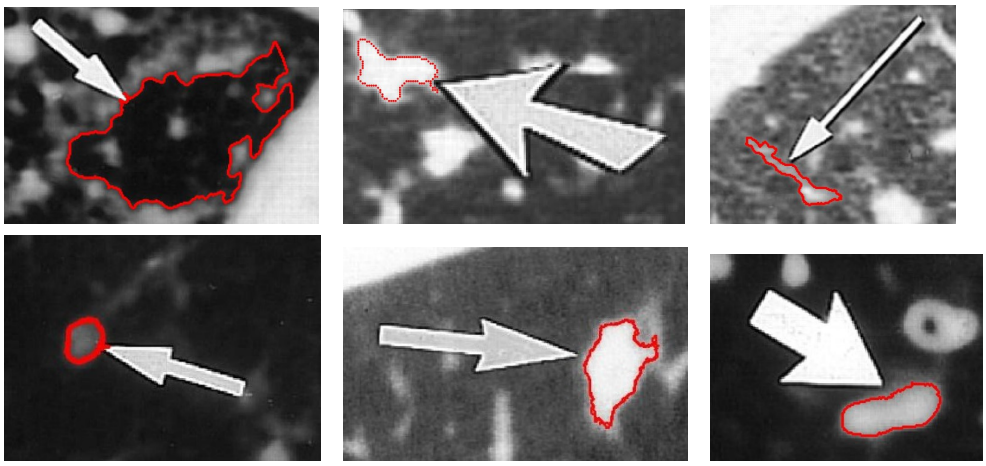
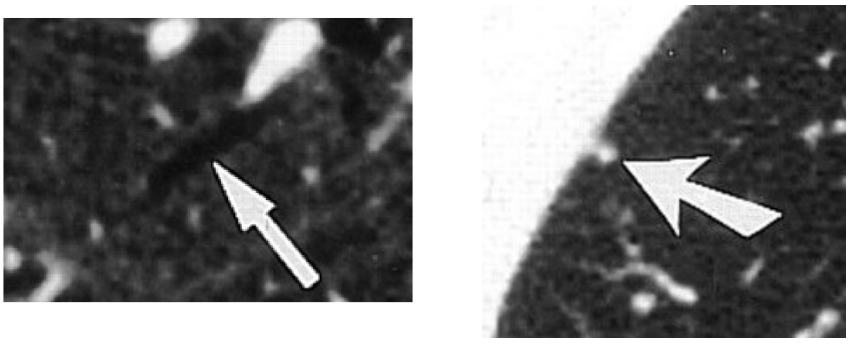


Figure 5.5: Examples illustrating region segmentation.



(a) (b)

Figure 5.6: Examples illustrating the case where our method failed: 1) The pixel intensity at ROI is not very different from the background, 2) ROI is attached to the wall and hence region growing technique failed to segment it.

CHAPTER 6

CONCLUSION AND FUTURE WORK

A new method to detect overlaid arrows in biomedical images is presented in this paper. Fuzzy binarization is used on image first to segment a set of candidates. To select arrow candidates, we first extract Npen++ features from arrow and test it through BLSTM classifier. The candidate are then conditionally passed through hull convexity defect-based arrowhead cropping, which is followed by template matching via dynamic programming. In our assessment, (using imageCLEF 2010 database), our results surpasses the state-of-the-art methods. These detected arrow are then search for the arrowhead points. Seed points are generated and are used for segmenting the region-of-interest using region growing technique.

To the best of our knowledge, this is first time arrow detection has been done without the use of tail information. Variation is the shape and size of the tail tend to change the overall shape of the arrow. Thus, cropping off the tail helped us to identify arrows.

Our aim was to identify the meaningful ROI and annotate them using concept appearing in the biomedical text. As next step, the segmented region can be labeled with keywords extracted from the journal article and image caption. These tagged ROI can be stored in the CDSS (clinical decision support system) database forming an efficient retrieval system.

REFERENCES

[1] You, D., Antani, S., Demner-Fushman, D., Rahman, M. M., Govindaraju, V., and Thoma, G. R., "Biomedical article retrieval using multimodal features and image annotations in region-based cbir," in [*IS&T/SPIE Electronic Imaging*], 75340V–75340V, International Society for Optics and Photonics (2010).

[2] You, D., Antani, S., Demner-Fushman, D., Rahman, M. M., Govindaraju, V., and Thoma, G. R., "Automatic identification of roi in figure images toward improving hybrid (text and image) biomedical document retrieval," in [*IS&T/SPIE Electronic Imaging*], 78740K–78740K, International Society for Optics and Photonics (2011).

[3] Santosh, K. C., Wendling, L., Antani, S., and Thoma, G., "Scalable arrow detection in biomedical images," 3257–3262, IEEE Computer Society, Stockholm (Sweden) (August 2014).

[4] Demner-Fushman, D., Antani, S., Simpson, M., and Rahman, M., "Combining text and visual features for biomedical information retrieval and ontologies," tech. rep., LHNCBC Board of Scientific Counselors, National Institutes of Health, Bethesda, MD (September 2010).

[5] Demner-Fushman, D., Antani, S., Simpson, M. S., and Thoma, G. R., "Design and development of a multimodal biomedical information retrieval system," *Journal of Computing Science and Engineering* **6**(2), 168–177 (2012).

[6] Dori, D. and Wenyin, L., "Automated cad conversion with the machine drawing understanding system: Concepts, algorithms, and performance," *IEEE Transactions on System, Man, and Cybernetics-part A: System and Humans* **29**, 411–416 (1999).

[7] Dori, D., Member, S., and Liu, W., "Sparse pixel vectorization: An algorithm and its performance evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**, 202–215 (1999).

[8] Park, J., Rasheed, W., and Beak, J., "Robot navigation using camera by identifying arrow signs," in [*International Conference on Grid and Pervasive Computing - Workshops*], 382–386, IEEE Computer Society (2008).

[9] Cheng, B., Stanley, R. J., De, S., Antani, S., and Thoma, G. R., "Automatic detection of arrow annotation overlays in biomedical images," *Int. J. Healthc. Inf. Syst. Inform.* **6**, 23–41 (Oct. 2011).

[10] You, D., Simpson, M., Antani, S., Demner-Fushman, D., and Thoma, G. R., "A robust pointer segmentation in biomedical images toward building a visual ontology for biomedical article retrieval," in [*IS&T/SPIE Electronic Imaging*], 86580Q–86580Q, International Society for Optics and Photonics (2013).

[11] You, D., Apostolova, E., Antani, S., Demner-Fushman, D., and Thoma, G. R., "Figure content analysis for improved biomedical article retrieval," in [*IS&T/SPIE Electronic Imaging*], 72470V–72470V, International Society for Optics and Photonics (2009).

[12] Hori, O. and Doermann, D. S., "Robust table-form structure analysis based on box-driven reasoning," in [*International Conference on Document Analysis and Recognition - Volume 1*], 218–, IEEE Computer Society, Washington, DC, USA (1995).

[13] Park, J., Rasheed, W., and Beak, J., "Robot navigation using camera by identifying arrow signs," in [*Grid and Pervasive Computing Workshops, 2008. GPC Workshops' 08. The 3rd International Conference on*], 382–386, IEEE (2008).

[14] Chenga, B., Stanleya, R. J., Dea, S., Antanib, S., and Thomab, G. R., "Automatic detection of arrow annotation overlays in biomedical images," *Healthcare Information Technology Innovation and Sustainability: Frontiers and Adoption: Frontiers and Adoption* , 219 (2013).

[15] Cheng, H. and Chen, Y.-H., "Fuzzy partition of two dimensional histogram and its application to thresholding," *Pattern Recognition* **32**, 825–843 (1999).

[16] Ramer, U., "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing* **1**(3), 244 – 256 (1972).

[17] Douglas, D. H. and Peucker, T. K., "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The Canadian Cartographer* **10**(2), 112–122 (1973).

[18] Prasad, D. K., Leung, M. K., Quek, C., and Cho, S.-Y., "A novel framework for making dominant point detection methods non-parametric," *Image and Vision Computing* **30**(11), 843 – 859 (2012).

[19] Sakoe, H., "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing* **26**, 43–49 (1978).

[20] Keogh, E. J. and Pazzani, M. J., "Scaling up dynamic time warping to massive dataset," in [*European PKDD*], 1–11 (1999).

[21] Müller, H., Kalpathy-Cramer, J., Eggel, I., Bedrick, S., Radhouani, S., Bakke, B., Kahn Jr, C. E., and Hersh, W., "Overview of the clef 2009 medical image retrieval track," in [*Multilingual Information Access Evaluation II. Multimedia Experiments*], 72–84, Springer (2010).

[22] Jaeger, S., Manke, S., Reichert, J., and Waibel, A., "Online handwriting recognition: the npen++ recognizer," *International Journal on Document Analysis and Recognition* **3**(3), 169–180 (2001).

[23] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J., "A novel connectionist system for unconstrained handwriting recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **31**(5), 855–868 (2009).

[24] Santosh, K., Alam, N., Roy, P. P., Wendling, L., Antani, S., and Thoma, G. R., "A simple and efficient arrowhead detection technique in biomedical images," *International Journal of Pattern Recognition and Artificial Intelligence* (2016).

[25] Zhang, D. and Lu, G., "Shape-based image retrieval using generic fourier descriptor," *Signal Processing: Image Communication* **17**, 825–848 (2002).

[26] Belongie, S., Malik, J., and Puzicha, J., "Shape matching and object recognition using shape contexts," **24**(4), 509–522 (2002).

[27] Kim, W.-Y. and Kim, Y.-S., "A region-based shape descriptor using zernike moments," *Signal Processing: Image Communication* **16**(1-2), 95 – 102 (2000).

[28] Hoang, T. V. and Tabbone, S., "The generalization of the r-transform for invariant pattern representation," **45**(6), 2145–2163 (2012).

[29] Santosh, K. C., Lamiroy, B., and Wendling, L., "Dtw-radon-based shape descriptor for pattern recognition," **27**(3) (2013).