Indian Journal of Science and Technology, Vol 9(21), DOI: 10.17485/ijst/2016/v9i21/89259, June 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

# Dimensional Arrow Detection from CAD Drawings

**Aditya Mukesh Intwala[1*], Kiran Kharade[2], Ravindra Chaugule[2] and Atul Magikar[1]**

[1]Symbiosis Institute of Technology, Pune - 412115, Maharashtra, India;
adityaintwala@yahoo.co.in, atul.magikar@sitpune.edu.in
[2]Renishaw India; kiran.kharade@renishaw.com, Ravindra.Chaugule@renishaw.com

## Abstract

**Objectives:** This paper proposes an approach to effectively identify dimensional arrow heads from the CAD drawing images. **Methods/Analysis:** The proposed approach implements a multi-level thresholding and morphological operations in order to detect arrow like entity and some additional morphological steps in order to filter out non-arrow like entities from CAD drawing images. A histogram based multi-level thresholding is implemented. Morphological Black hat is used to detect solid arrow heads while morphological white hat is used to detect line arrow heads. Properties of arrow heads are used to filter out unwanted entities. **Findings:** CAD drawings contain arrows to depict dimensions of the models. In many situations, users do not have access to the CAD file for the drawing or do not have suitable software to visualize the CAD file. In such cases, an image of 2D drawing can be processed in automated or semi-automated way to extract dimensional relationships between entities. One step in such extraction is to detect dimensional arrow heads. The approach was tested on variety of images of CAD drawings with increasing difficulties with 96.655% success rate. The F1 score for each image was calculated and the cumulative average F1 score was found out to be 0.9596, precision rate is 95.95%, recall rate is 96.83% which suggests acceptable accuracy. **Applications/Improvement:** The proposed approach helps to detect dimensional arrow heads from CAD drawing images using image processing.

**Keywords:** Arrow Detection, CAD Drawings, Computer Vision, Feature Extraction, Image Processing, Pattern Recognition

## 1. Introduction

Sometimes it becomes necessary to process the dimensional information stored in CAD drawings in automated or semi-automated way. In many situations, the users of CAD drawings can access only the paper copy or an image of the drawing. They do not have an access to CAD file of the drawing or necessary software tools to process information stored in the CAD drawings. A typical CAD drawing contains GD&T symbols and dimension lines with arrow heads. One of the major steps in automated processing of drawing is to identify the dimensional lines and associate them with the relevant features. To identify dimensional lines, one first needs to detect the dimensional arrow heads. This paper mainly focuses on this step of automated processing of CAD drawing. The algorithm presented here can also be useful with some modifications in medical imaging to identify overlaid marking arrows and pointers.

In Dori and Wenyin[1-3] proposed a technique to detect arrows based on sparse pixel vectorization. They represented line at intervals along the tracking direction and recorded the middle points of these sections. These points were used to construct vectors. Such a vectorization process resulted in many thick, short bars from the arrow heads that were then used to make a decision. The use of this technique is limited to machine printed line images. Template based methods have limitations since they require new templates to train images with new types of arrows.

In[4] used text-like and arrow-like object separation. From the binary image, arrow-like object separation employs a fixed sized mask followed by edge detection techniques and fixed thresholds. These candidates are used to compute overlapping regions, which are then binarized to extract the boundary of the expected arrows. Edge-based techniques are simple and compact com-

*Author for correspondence

pared to solid regions based techniques, especially when a broken boundary does not affect the performance[5].

In Wendling et al.[6] also proposed a new method to detect arrows from line drawings. They provided a set of criteria defined from the geometric properties of an arrow, which were then aggregated using the Choquet integral. Mai[7] proposed a Real-time detection of arrow markings using curve-based prototype fitting.

In biomedical images, the major issue of broken boundary occurs. K. C. Santosh, et al.[8-10] considers the geometrical definition of an arrow, to efficiently handle arrow detection. This method can handle several different types of arrows.

Our approach is based on hard Thresholding and detection of contours from it. We have implemented multi-level thresholding and morphological approaches to get the arrow like features from the images. Contours are then found from this feature in order to detect arrow like features from the images. Our test images are CAD drawings with increasing level of details. We have mainly used Open CV 3.0 library for the implementation of our algorithm.

## 2. Proposed Algorithm

Our arrow head detection approach is based on concepts of multi-level thresholding and finding contours from that image. The input to our algorithm may be a multi-channel colored or single-channel gray scale image of CAD drawing.

We have tested our approach for the following 3 types of arrow heads as shown in Figure 1.

### 1.1 Gray Scaling

For arrow head detection, our algorithm requires a gray-scale image, thus if the input image is multi-channel colored image, then it must be converted to grayscale single channel image[11]. Let's consider R, G and B are color components of original colored image. The final grayscale pixel intensity can be obtained as

$$f_g = 0.2126\ R + 0.7152\ G + 0.0722\ B \quad (1)$$

### 1.2 Multilevel Thresholding

Multilevel thresholding of the grayscale image is carried out in the next step. In multilevel thresholding, cluster-

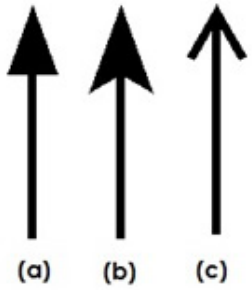


**Figure 1.** Types of arrows, (a) & (b) are solid arrows and (c) is line arrow.

ing based image thresholding takes place[12]. We have used OTSU's method of multilevel thresholding[13-15] to convert the grayscale image to a binary image. In this method, a histogram of an image is calculated and probabilities for intensities are calculated.

Let $t$ be the threshold value, $q_i$ be the probabilities of two histogram bins separated by threshold $t$; $\mu_i$ be the mean of histogram bin and $\sigma_i^2$ be the variances of two bins and $x(i)$ be the value of center intensity of histogram bin $i$,

$$q_i(t) \quad \sum f_g(i) \quad (2)$$

$$\mu_i(t) = \frac{\left[\sum_0^t f_g(i) * x(i)\right]}{q_i} \quad (3)$$

The variances between two histogram bins can be calculated as follows

$$\sigma_b^2(t) = \sigma^2 - \sigma_q^2(t) \quad (4)$$

$$\sigma_b^2(t) = q_1(t) * q_2(t) * \left[\mu_1(t) - \mu_2(t)\right]^2 \quad (5)$$

Threshold Value = Maximum $\sigma_b^2(t)$

The required threshold value would be maximum of all the calculated values of $\sigma_b^2(t)$ for the image. This value is then used for thresholding the image. The output image will be in binary form.

$$f_t = T(f_g) = \begin{cases} 1, \text{if } f_g(i,j) \geq \text{ Threshold Value} \\ 0, \text{if } f_g(i,j) < \text{ Threshold Value} \end{cases} \quad (6)$$

### 2.3 Morphological Operation

After thresholding, morphological operations are applied to the binary image. A morphological Black Top Hat operation is applied[16] in order to detect solid arrows as shown in Figure 1a and b and morphological White Top Hat operation[17] is applied in order to detect line-based arrow heads as shown in Figure 1c.

The Black Hat is the result obtained by subtracting the original image from the image obtained by a morphological closing operation. This generates the dark spots on the image which are smaller than structuring element used in the operation. Morphological closing operation first dilates the image to connect the broken connections and then erodes it[18,19]. Thus, by subtracting the original image from this would return only the dark areas.

Consider $f_t$ be the threshold image and s(x) be structuring element for closing operation, then the resultant image after applying Black Hat can be obtained as below

$$T_{bh}(f_t) = f_t \bullet s(x) - f_t \quad (7)$$

Here, $\bullet$ is closing operator, $T_{bh}$ is Black Hat function. We have used, $s(x)$ as 2 x 2 kernel of ones.

The White Top Hat is the result obtained by subtracting the morphological opening image from the original image. This returns bright spots on the images which are smaller than the structuring element. Morphological opening operation first erodes the image to break connections and then dilates it[18,19]. Thus, by subtracting morphologically opened image from original image would return only the bright areas.

Consider $f_t$ be the threshold image and s$(x)$ be structuring element for opening operation, then the resultant image after applying White Top Hat can be obtained as below

$$T_{wh}(f_t) = f_t - f_t \circ s(x) \quad (8)$$

Where, $\circ$ is opening operator, $T_{wh}$ is White Top Hat function. We have used, $s(x)$ as 5 x 5 kernel of ones.

This image is then subtracted from the inverted threshold image to obtain only the area of interest of arrows from the image.

$$T_{arrow} f(t) = f(t)^{-1} - T_{bh} f(t) \quad (9)$$

Where, $(f_t)^{-1}$ is an inverted thresholded image.

### 2.4 Additional Morphological Operations

This step is needed for the images where arrows are connected to other features. To avoid false detections of whole connected set of entities as arrows, we have to perform this separation step to break the linking connections between arrow heads and other entities. This can be done by an erosion operation followed by a dilation operation[20].

Erosion operation can be expressed as

$$T_e = T_{arrow}(f_t) \ominus s(x) = \{(i,j): s(i,j) \subset T_{arrow}(f_t)\} \quad (10)$$

Where, $\ominus$ represents erosion operator, $T_e$ is eroded output image.

$T_e$ is then further subjected to a dilation operation as below

$$T_d(T_e) = T_e \oplus s(x) = ((T_{arrow}(f_t))^c \ominus s(x))^c \quad (11)$$

Where, $\oplus$ is dilation operator and $c$ is compliment of function, $T_d$ is dilated output image.

### 2.5 Contour Detection

As a next step, the resultant image is subjected to a contour detection operation[21]. The detected contours are closed contours which are basically arrows. Each contour is then subjected to the enclosed area check. If the enclosed area is within the prescribed limit then they are labeled as arrows. This limit can be obtained from training images. For the test images we have used, an area range from 3 to 70 is used to mark contour as an arrow head. A bounding box is drawn in the labeled region in the source image.

## 3. Pseudo-Code

The pseudo-code for the proposed algorithm is as given below.

* Input image
* Check if grayscale or not
* If not grayscale:
    * Convert to grayscale
* Multilevel thresholding
* Bottom Hat / Top Hat morphological operation
* Inverted threshold Image – Bottom Hat image
* Apply erosion
* Apply dilation
* Find contours
* For all contours:

* Calculate contour enclosed area
* Check if (LL < contour enclosed area < UL):
    * Label contour as an arrow
    * Draw bounding box for detected arrow
* Display Detected Arrow Image

## 4. Flowchart

The flowchart of the proposed algorithm for arrow detection from CAD drawing images is given in the Figure 2.

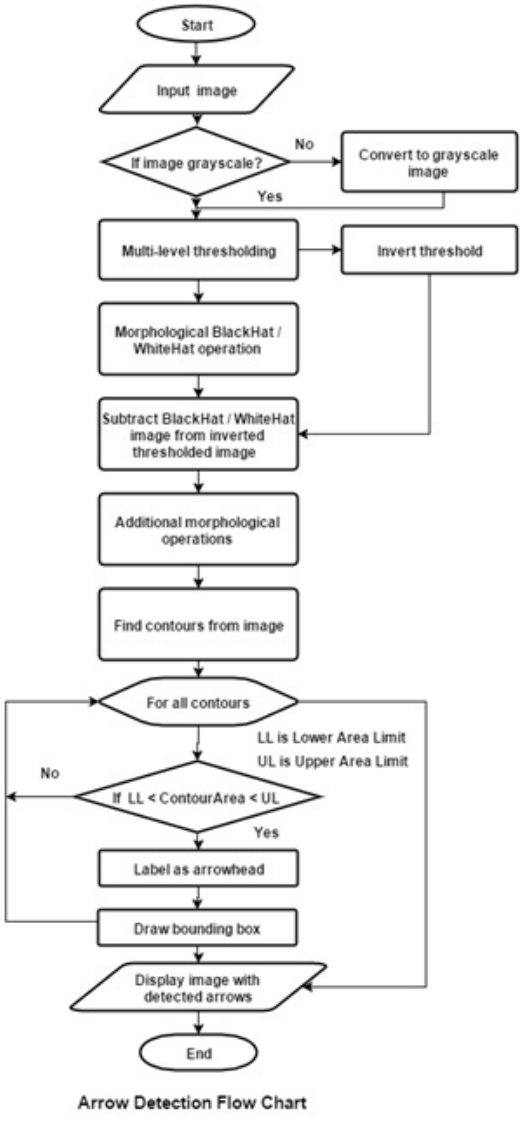


**Figure 2.** Flowchart for proposed algorithm.

## 5. Experimentation and Discussion

We have tested this algorithm on different test images of CAD drawings of increasing difficulties taken from book on machine drawing. Our algorithm was implemented in Python 3.4 using Open CV 3 library. Output results of the implemented proposed algorithm are shown in the following Figures 3, 4, 5 and 6.

Blue boxes in output images represent detected solid arrows similar to Figure 1a and b, green boxes represent line arrows similar to Figure 1c and red boxes represent false positives detected.

## 6. Results

The statistical accuracy evaluation using $F_1$ score of the proposed approach is performed as follows.

The positive predictive value ($p$) also called as precision is calculated as shown in Equation (10).

$$p = \frac{number\ of\ correct\ positive\ results}{number\ of\ all\ positive\ results} \quad (10)$$

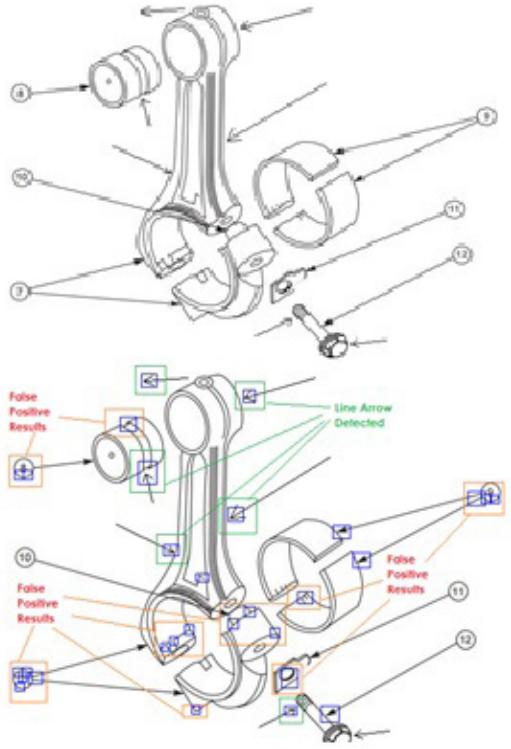


**Figure 3.** Arrow detection on sample image with line arrow head.

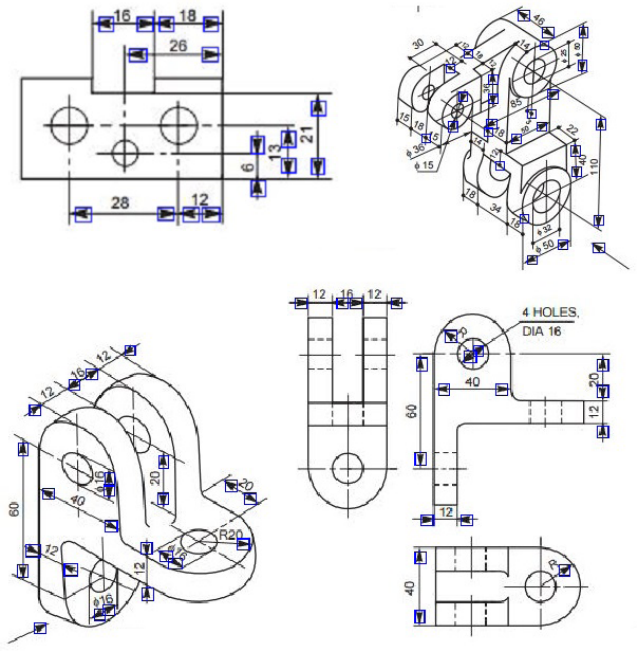


**Figure 4.** Arrow detection on sample images.

**Table 1.** Algorithm summary

| No. of Arrows to Detect 'N' | Detected Arrows 'A' | False Positive 'B' | Missed Arrows 'C' | Total Detected |
|---|---|---|---|---|
| 1199 | 1156 | 40 | 43 | 1196 |

**Table 2.** Precision - Recall summary

| Precession $P = \frac{A}{(A+B)}$ | Recall $R = \frac{A}{N}$ | $F_1$ score $F_1 = 2 * (\frac{p * r}{p + r})$ |
|---|---|---|
| 0.9595 | 0.9683 | 0.9596 |

The recall ($r$) also called as sensitivity is calculated as show in equation (11).

$$r = \frac{number\ of\ correct\ positive\ results}{number\ of\ positive\ results\ that\ should\ have\ been\ returned} \quad (11)$$

The $F_1$ scores for each level of image are calculated along with average $F_1$ scores.

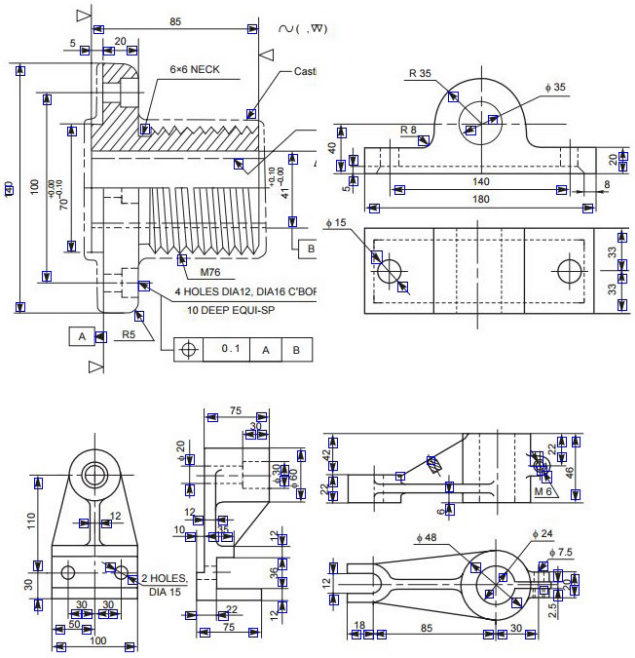


**Figure 5.** Arrow detection on sample images. All arrow heads from the images are detected successfully with our proposed approach 96.66% of time with 3.34% false positive detection.

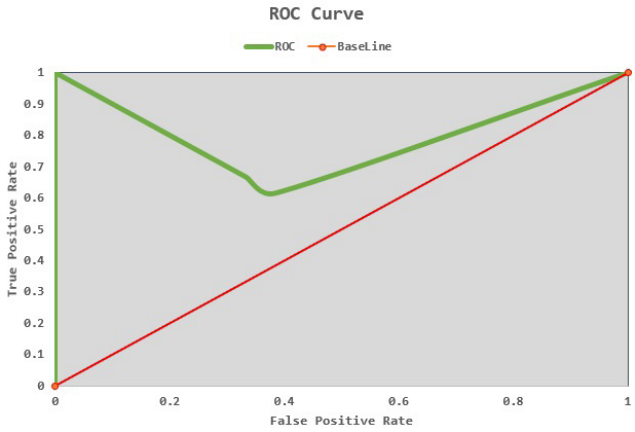


**Figure 6.** ROC curve of proposed algorithm. Our algorithm's precision rate is 95.95%, recall rate is 96.83%, average $F_1$ Score is 0.9596 and Area under Curve from ROC curve came out to be 0.80, which suggest highly accurate and acceptable results.

$$F_1 = 2 * (\frac{p * r}{p + r}) \quad (12)$$

If $F_1F_1$ score is near to 1, an algorithm can be considered as highly accurate and accuracy goes on reducing when $F_1F_1$ score moves towards zero.

## 7. Conclusion

The test CAD drawings image data which we have used is a fairly good representation of drawings which are encountered in practice. For such a test data, this approach gives fairly accurate and acceptable results for practical purposes.

The only drawback that this approach suffers is dependency on the arrow sizes in terms of pixels (which is captured in terms of enclosed contour area) in an input image. For the test data we have used, we found experimentally that specifying the enclosed contour area range from 3 to 70 works well.

It is recommended that, for solid arrow head detection, one should employ Black Hat morphological operation and for line-based arrow heads, one should employ White Top Hat morphological operation. Given the somewhat generic geometrical alignment of lines in line-based arrow heads, there may be a marginal increase in the false positive detection. This can be reduced by using slightly larger kernels in morphological operations.

## 8. Future Work

Our Future work would include detecting arrow heads from more regular images than CAD drawings, where the background of the arrows is not as clearly distinguishable as the CAD drawings.

## 9. Acknowledgment

We express our deepest gratitude to Renishaw Metrology Systems Ltd, Pune, India for allowing and facilitating our work. Our special thanks to the Director of Software Division, Renishaw Metrology Systems Ltd, Pune, Mr. Vikas Saxena. We express our deepest thanks to Mr. Kiran Kharade, Software Consultant at Renishaw Pune, who took time out to hear, guide and keep us on the correct path and Mr. Gaurav Saxena, Manager, Software Development, for his support and encouragement towards this project. We choose this moment to acknowledge their contribution gratefully.

## 10. References

1. Dori D, Liu W. Sparse pixel vectorization: An algorithm and its performance evaluation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1999; 21(3):202-15.
2. Dori D, Wenyin L. Vector-based segmentation of text connected to graphics in engineering drawings. In Advances in structural and syntactical pattern recognition. Berlin Heidelberg: Springer; 1996. p. 322-31.
3. Dori D, Wenyin L. Automated CAD conversion with the machine drawing understanding system: Concepts, algorithms and performance. IEEE Transactions on Systems, Man and Cybernetics. Part A: Systems and Humans. 1999; 29(4):411-16.
4. Beibei C, Joe Stanleya R, Soumya D, Antani SK, Thoma GR. Automatic detection of arrow annotation overlays in biomedical images. Healthcare Information Technology Innovation and Sustainability: Frontiers and Adoption: Frontiers and Adoption. 2011; 6(4):23-41.
5. Wang N, Liu W, Zhang C, Yuan H, Liu J. The detection and recognition of arrow markings recognition based on monocular vision. Chinese Control and Decision Conference, CCDC'09; Guilin. 2009. p. 4380-6.
6. Wendling L, Tabbone S. A new way to detect arrows in line drawings. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2004; 26(7):935-41.
7. Maier G, Pangerl S, Schindler A. Real-time detection and classification of arrow markings using curve-based prototype fitting. IEEE Intelligent Vehicles Symposium IV; 2011. p. 442-7.
8. Santosh KC, Wendling L, Antani SK, Thoma GR. Scalable arrow detection in biomedical images. 22nd International Conference on Pattern Recognition (ICPR); 2014. p. 3257-62.
9. Santosh KC, Lamiroy B, Wendling L. DTW–Radon-based shape descriptor for pattern recognition. International Journal of Pattern Recognition and Artificial Intelligence. 2013; 27 (03):30.
10. Santosh KC, Wendling L. Graphical symbol recognition. Wiley Encyclopedia of Electrical and Electronics Engineering. 2015.
11. Gooch AA, Olsen SC, Tumblin J, Gooch B. Color2gray: Salience-preserving color removal. Proceedings of ACM SIGGRAPH ACM Transactions on Graphics (TOG); 2005. p. 634-9.
12. Hammouche K, Diaf M, Siarry P. A multilevel automatic thresholding method based on a genetic algorithm for fast image segmentation. Computer Vision and Image Understanding. 2008; 109(2):163-75.
13. Huang D-Y, Wang CH. Optimal multi-level thresholding using a two-stage Otsu optimization approach. Pattern Recognition Letters. 2009; 30(3):275-84.

14. Tsai D-M, Chen Y-H. A fast histogram-clustering approach for multi-level thresholding. Pattern Recognition Letters. 1992; 13(4):245-52.
15. Otsu N. A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, Cybernetics. 1979: 9(1):62-6.
16. Dougherty ER, Lotufo RA. Hands-on morphological image processing. The International Society for Optical Engineering SPIE. Bellingham: SPIE Press; 2003. p. 71.
17. Jalba AC, Wilkinson MHF, Roerdink JBTM. Morphological hat-transform scale spaces and their use in pattern classification. Pattern Recognition. 2004; 37(5):901-15.
18. Jalba AC, Wilkinson MHF, Roerdink JBTM. Shape representation and recognition through morphological curvature scale spaces. IEEE Transactions on Image Processing. 2006; 15(2):331-41.
19. Serra J. Image analysis and mathematical morphology. USA: Academic Press, Inc.; 1983.
20. Arbelaez P, Maire M, Fowlkes C, Malik J. Contour detection and hierarchical image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2011; 33(5):1-20.
21. Arumugadevi S, Seenivasagam V. Comparison of clustering methods for segmenting color images. Indian Journal of Science and Technology. 2015 Apr; 8(7):670-7.