



ISSN NO. 2320-5407

Journal homepage: <http://www.journalijar.com>

INTERNATIONAL JOURNAL  
OF ADVANCED RESEARCH

## RESEARCH ARTICLE

## Automation of resolving CAPTCHAs for web crawling.

Renuka Sakhare<sup>1</sup>, Abhay Bhagat<sup>2</sup> and Anil Bhadagle<sup>3</sup>.

1. Student, PVG's college of engineering, Pune.
2. Student, PVG's college of engineering, Pune.
3. Assistant professor, department of computer engineering, PVG's college of engineering, Pune.

**Manuscript Info****Manuscript History:**

Received: 14 December 2015  
Final Accepted: 26 January 2016  
Published Online: February 2016

**Key words:**

Web crawling, CAPTCHAs, text area detection, image segmentation, character recognition.

**\*Corresponding Author**

Renuka Sakhare.

**Abstract**

A web crawler is an automated computer program used by search engine to collect data of web pages from World Wide Web and the web crawler perform this by process called web crawling. To keep data updated crawler need frequent caching of web pages. But performance of web server gets affected as crawler retrieve data frequently in greater depth than human searchers. Thus to balance load and for authentication server asks crawler to verify themselves against CAPTCHAs. It is not feasible for human to solve and enter CAPTCHAs for more than two billion web pages exist on www every now and then. Thus to automate CAPTCHA solving, we describe a system for text recognition from CAPTCHA images. Our particular focus is reliable text extraction, recognition, feeding resolved CAPTCHA characters to crawler system in order to continue with crawling process without human involvement.

Copy Right, IJAR, 2016.. All rights reserved.

**Introduction:-**

The World Wide Web has grown from a few thousand pages in 1993 to more than two billion pages at present [1]. Due to the abundance of data on the web and different user perspective, information retrieval becomes a challenge. When a data is searched, hundreds and thousands of results appear. So for user's convenience the search engines have a bigger job of sorting out the results, in the order of interestingness of the user within the first page of appearance and a quick summary of the information provided on a page.

Web crawlers are programs which browse www in methodical and automated way. Web crawler creates copy of all the visited pages for later processing by a search engine. This process is iterative, as long the results are in close proximity of user's interest. Search engines use this algorithms which sorts and ranks the result in the order of authority that is closer to the user's query. Many algorithms are in use - Breadth first search, Best first search, Page Rank algorithm, Genetic algorithm, Naïve Bayes classification algorithm to mention a few [6]. There are important characteristics of the Web such as its large volume, dynamic page generation that make crawling very difficult. The high rate of change implies that by the time the crawler is downloading the last pages from a site, it is very likely that new pages have been added to the site, or that pages have already been updated or even deleted. Performance of web crawler based on freshness and age. When the same copy exists in the local as well as the remote sources, then it is considered to be the "fresh" page. Cho and Garcia [6] calculated the freshness of a page as shown in figure 1,

$$F(e_i; t) = \begin{cases} 1 & \text{if } e_i \text{ is up-to-date at time } t \\ 0 & \text{otherwise.} \end{cases}$$

Figure 1

where  $e_i$  is the element of database. Freshness focus on whether or not the local copy is the current copy of the resource. And the age of a page can be given as shown in figure 2,

$$A(e_i; t) = \begin{cases} 0 & \text{if } e_i \text{ is up-to-date at time } t \\ t - t_m(e_i) & \text{otherwise.} \end{cases}$$

**Figure 2**

Where  $t_m(e_i)$  is the time of first modification of  $e_i$  after the most recent synchronization [6]. Age focus on how long ago the local copy was updated. The freshness drops to zero when the real-world element changes and the age increase linearly from that point on. When the local element is synchronized to the real-world element, its freshness recovers to one, and its age drops to zero.

Internet browsers and Big Data companies use web crawling software to keep their database update where ultimate goal is to provide best services to internet users. For this crawler need more frequent crawling to reduce age and improve freshness of page. On the other side web servers may be overloaded as it has to handle the requests of the viewers of the site as well as the web crawler. The politeness policy of crawler is used so that the performance of a site is not heavily affected while the web crawler downloads a portion of the site. Crawlers can retrieve data much quicker and in greater depth than human searchers, so they can have a crippling impact on the performance of a site. Needless to say, if a single crawler is performing multiple requests per second and/or downloading large files, a server would have a hard time keeping up with requests from multiple crawlers. As the speed of crawler is very fast considering multiple requests per second server often get overwhelmed. Thus in order to slow down requests from same machine and to check whether that machine is not a malicious trap who generally attack systems by sending multiple request until it get crash(also called as DOS attack), system asks machine to solve CAPTCHAs. In the regular crawler to pass CAPTCHA authentication phase human intervention is requires as crawlers are not embedded with automatic CAPTCHA resolving programs. If human involve in whole web crawling process to resolve CAPTCHA one also need to consider the delay while crawling. By taking into account tremendous size of internet sites it is not feasible for human being to solve and enter CAPTCHA for each web server very often. This will significantly reduce the efficiency of crawling software. Also it will be more prone to error while considering human nature.

To address this problem web crawling process can be made automated by adding application of resolving CAPTCHA in already existing web crawling system. Thus our problem can be defined as to develop a software application to solve the issue of server authentication requirement (usually through CAPTCHAs) after repeatedly sending a request for access / download data from same IP address to server in web crawling. Hence to reducing the human interaction and making the whole process of crawling automated with improved efficiency and less delay.

### **Related Work:-**

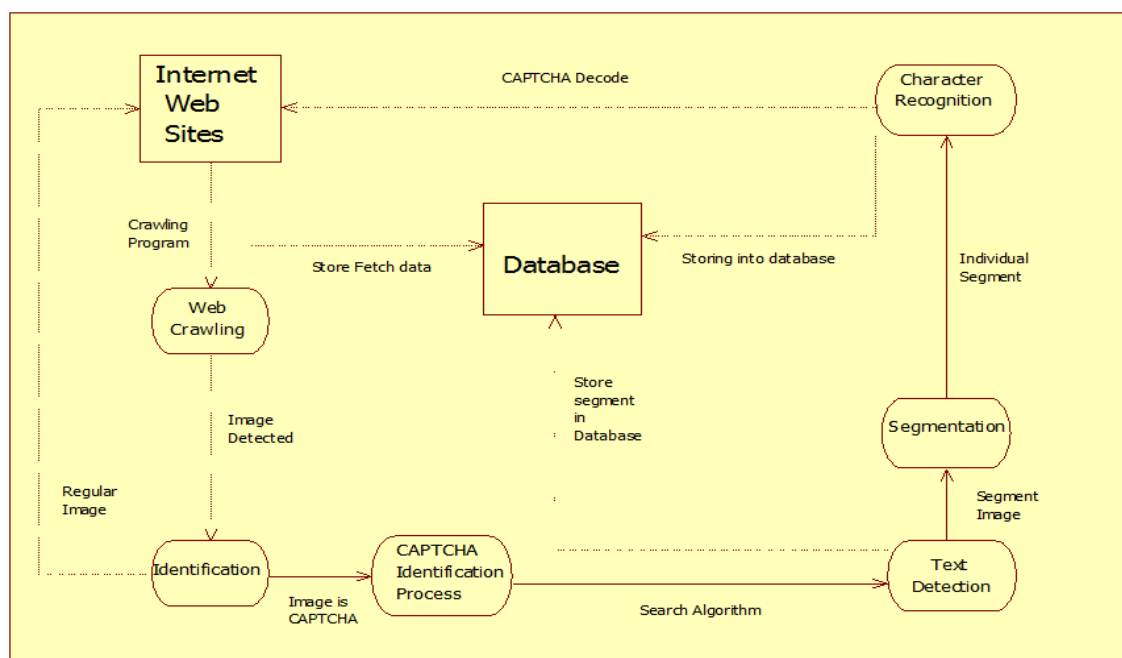
There have been a different methods dealing with text detection and reorganization in images. CAPTCHAs are nothing but image embedded with text thus in order to solve these CAPTCHAs one need to consider text detection and recognition using image processing. Basic approaches to text detection classified into three categories: texture-based methods, region based methods, and hybrid methods [2]. Texture-based methods involve texture properties of text such as style, orientation, and wavelet coefficients, the construction of gray-level co-occurrence matrix. Methods like Gabor lettering and spatial variance are used to automatically locate text regions [3]. Such approaches do not perform well with different character font sizes, and furthermore, they are computationally intensive. Region based methods use the properties of the color or gray scale or alignment in a text region or their differences in properties of the background. First extract candidate text regions through segmentation or clustering and then remove non-text regions. The third category, hybrid methods, is a fusion of region-based and texture-based methods. Traditional text recognition methods are based on sequential character classification by either sliding windows [7] or connected components, after which a word prediction is made by grouping character classifier predictions in a left-to-right manner. The sliding window classifiers include random ferns in Wang et al. [7], and CNNs [7].

More recent works make use of over-segmentation methods, guided by a supervised classifier, to generate candidate proposals which are subsequently classified as characters or false positives. Some system are built on recent machine learning approaches [8] for improved classifier performance, combined with large training sets and

distributed language modeling. The system achieves record performance on all major text recognition benchmarks, and high quality text extraction from typical smartphone imagery with sub-second latency. Another impressive method makes use of an OCR system which use histogram equalization to extract images [9]. The OCR recognizing process includes several complex algorithms and previously loaded templates and dictionary which are crosschecked with the characters in the document and the corresponding machine editable ASCII characters. In [9] discuss how artificial neural network, genetic algorithm and fuzzy logic can be used in optical character recognition for the use of character recognition. Paper [5] has propose a modified dictionary search method, based on the Levenshtein edit distance, to correct errors in recognition dictionary based correction methods are able to handle partial missing and spurious characters, which often occur for scene text.

### System Overview:-

Our proposed system takes conventional multistage approach to text extraction, similar to designs as shown in figure 3.



**Figure 3: Schematic overview of the proposed framework**

Our system consists of web crawling block, which is a continuous process for copying data from given URLs. Whenever crawler stuck at CAPTCHA authentication, control shifts to CAPTCHA identification process which will break CAPTCHA with help of following stages. Text detection stage will take CAPTCHA image as input and separate textual area from background. Segmentation block is use to divide text area into individual segment where one segment will contain only one character. Segmented character will be given to character recognition stage. The output of this stage is character in electronic format i.e. text from image which is further stored in computer readable form. Now by entering solved CAPTCHA to respective web server, web crawling process will be resumed to keep search engines database updated.

### Web Crawling:-

To collect the web pages from www a search engine uses web crawler and the web crawler collects this by web crawling. Due to limitations of network bandwidth, time-consuming and hardware's a Web crawler cannot download all the pages, it is important to select the most important ones as early as possible during the crawling process and avoid downloading and visiting many irrelevant pages. Due to large size of web induces low coverage and search engine indexing not cover one third of the publicly available web.

Many web. Crawlers are simple program with seed URL. Data collected from first URL will get stored in database and other URLs collected will get append into url-frontier to crawl further. Thus crawling process continue in automated and incremental fashion. Many crawling algorithms are available such as breadth First Search, depth First Crawling, page Rank, online page importance calculation, crawling the large sites first, crawling through URL Ordering. One can use them according to convenience. Simplest crawling process algorithm can be given as,

Step 1:

```
// Start with given Seed URLs as input
  A_Star_Algo(Initial seed)
// Insert Seed URLs into the Frontier
  Insert_Frontier (Initial seed)
// Crawling by picking new link from the Frontier
while (Frontier! = Empty) do
  Link: = Remove_Frontier (URL)
  Webpage: = Fetch (Link)
  for (each URL found Webpage)
    Value=Calculate_Relevancy_Value(URL)
    If (Value > Threshold) do
      Insert_Frontier( URL )
  end for
if (CAPTCHA) do
goto CAPTCHA resolver
end while
```

### **CAPTCHA Solving:-**

By analyzing various log files of different web site they found that maximum web request is generated by web crawler and it is on an average 50%. This is the reason why we need to implant crawling with CAPTCHA solving stage. CAPTCHA is nothing but text in image format which is difficult for computer to understand due to many reasons. The reason for this difficulty is the font characteristics of the characters in CAPTCHA images are different to font of the characters in computer system. As a result, computer is unable to recognize the characters while reading them. Also text characters contained in images can be multicolor or any gray-scale value, variable size, low resolution and embedded in noisy backgrounds, bad quality of images. Many experiments have done on text recognition by applying conventional technology. Text recognition system extracts relevant information and enters it automatically. In this section we briefly describe the overall architecture of Text recognition system. A Text recognition system receives an input in the form of image which contains some text information. The Text recognition system can be divided in following module: (A) Pre-processing Module, (B) Text area detection Module, (C) Segmentation Module (D) Character recognition.

#### **A. Preprocessing Module:-**

Generally image has low resolution and while reading them in computer they get embedded with noisy backgrounds. Thus qualities of images need to improve before actual identification. Using open computer vision library and functions one can easily convert image into gray scale computer readable matrix form. This process may introduce unwanted changes in matrix thus preprocessing is done as follows:

##### **1. Binarization:-**

For computer a picture is the combinations of picture elements which are also known as pixels. For each height and for each width of the image, convert image to grayscale by calculating average of RGB channels of the image pixel. Grayscale image matrix is further converting it into binary image. This process is called Digitization of image. Binary image pixel values are obtained using the characteristic function [2] as shown below.

$$B(x, y) = 1 \text{ if } g(x, y) < T$$

$$= 0 \text{ if } g(x, y) > T$$

Where T= Intensity mean

## 2. Noise Removal:-

Noise removal is one of the most important processes. Due to this quality of the image will increase and it will affect recognition process for better text recognition in images. Gaussian noise and blur are applied to each synthetic image [5]. There may be noise pixels that are introduced due to scanning of the image. Besides, same font and size may also have bold face character as well as normal one. Thus, width of the stroke is also a factor that affects recognition. Therefore, a good character recognition approach must eliminate the noise after reading binary image data, smooth the image for better recognition and extract features efficiently.

## 3. Gap Removal:-

Gap Removal will detect white pixel if that pixel is surrounded by two black pixels in horizontal, vertical or diagonal direction, then convert that white pixel to black. Similar process can be repeated with black pixels. By using it we can remove all gaps which are occurred within image it helps in character recognition of discontinuous character efficiently.

## 4. Normalization:-

Normalization is one of the important pre-processing operations for text recognition. The normalization is applied to obtain characters of uniform size, slant and rotation. Character Normalization It is necessary to normalize the character, letters and numbers to standard size. We can normalize the characters of different size into one fixed size to make our task easy with the help of template from classifier. Thus algorithmic steps for overall preprocessing module are,

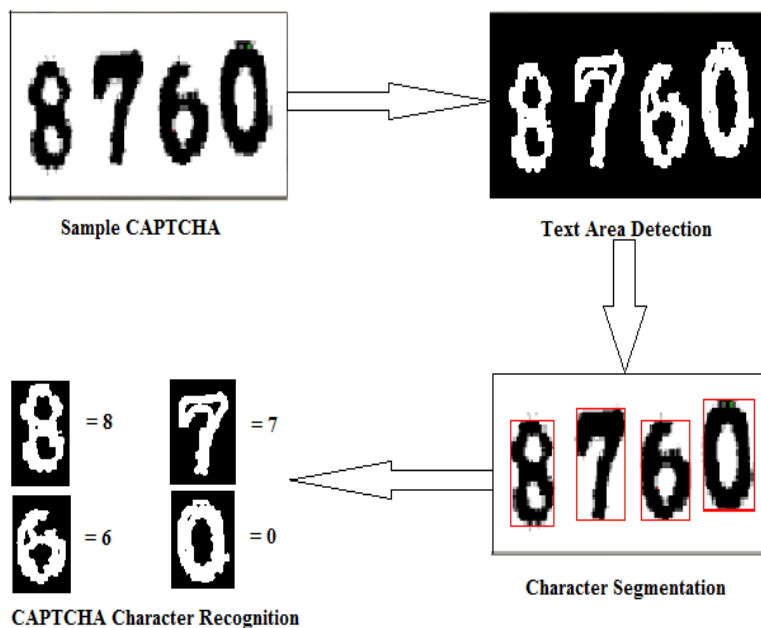


Figure 4: CAPTCHA solving processing

Step 2:

```

// Filter image into black and white using threshold
Img_mat=Binary(captcha.png)
Img_mat=GrayScale(Img_mat)
Calculate(Threshold T)
for each Pixel(x,y) ∈ Img_mat do
if(colour_intensity[x,y] > T) then
    colour_intensity[x,y]=white
  
```

```

else
    colour_intensity[x,y]=black
end for

// Gap removal from black and white image matrix
for each Pixel(x,y) ∈ Img_mat do
    if(more than 4 neighbor pixels are white) then
        Pixel(x,y)=white
    else if(more than 4 neighbor pixels are black) then
        Pixel(x,y)=black
    end for
end for

```

### B. Text Area Detection Module:-

This step focuses the attention to areas where text may occur. An image is stored in the form of a two dimensional array in computer where a black pixel is represented by 1 and a white pixel by a 0. Here we simply convert binarize image into edge image where all character pixels as well as some non-character pixels which also show high local color contrast are registered in the edge image. In this image, the value of each pixel of the original image is replaced by the largest difference between itself and its neighbors (in horizontal, vertical and diagonal direction) [3]. To locate the Header Line the array is scanned row by row and the number of black pixels is recorded for each row resulting in horizontal histogram. Since text regions show high contrast values, it is expected that they produce high peaks in horizontal projection. The row with the maximum number of black pixels is the position of the header line called as Shirerekha. So by calculating Y-coordinates in later step, to find the x-coordinates of the leftmost and rightmost, top and bottom point of the text region we make a vertical histogram of the image starting from the top row of the image. Finally, the exact coordinates for each of the detected areas are used to create bounding boxes as shown in figure 4.

### C. Segmentation Module:-

Segmentation is done to make the separation between the individual characters of an image as shown in figure 4. Very often, adjacent characters are touching, and may exist in an overlapped. Therefore, it is a complex task to segment a given word correctly into its character components. For segmentation, all the templates of the alphabets that are pre-designed are loaded into the system. In segmentation, the position of the object i.e., the separate character area in the image is found using following steps,

Step 3:

```

//Retrieving each character from Img_mat in separate block
for each character with Pixel cluster (x,y) ∈ Img_mat && colour_intensity[x,y]=white from Img_mat do
    Upper Left(x, y) =min(x), min(y)
    Upper Right(x, y) =max(x), min(y)
    Lower Left(x, y) =min(x), max(y)
    Lower Right(x, y) =max(x), max(y)
    DrawRectangle (Upper Left, Upper Right, Lower Left Lower Right)
    Resize rectangle to default fix dimensions
end for

```

Step 4:

```

//Generating training data for future use
for each rectangle in Img_mat do
    classification_char ←appropriate character keyboard
    Open (Training_data.txt, WRITE)
    Append in Training_data.txt (classification_char and Pixel cluster of character block (x,y) ∈ Img_mat )
    Close (Training_data.txt)
end for

```

As mentioned in algorithm, size of the image is cropped to that of the template size. Many methods are available for this which are Flood Fill Segmenting, Block Segmenter, Fill and Shrink segments etc.[2].

## **D. Character Recognition:-**

### **1. Feature Extraction:-**

Feature extraction is the process to retrieve the most important data from the raw data. The most important data means that's on the basis of that's the characters can be represented accurately.

### **2. Classifier:-**

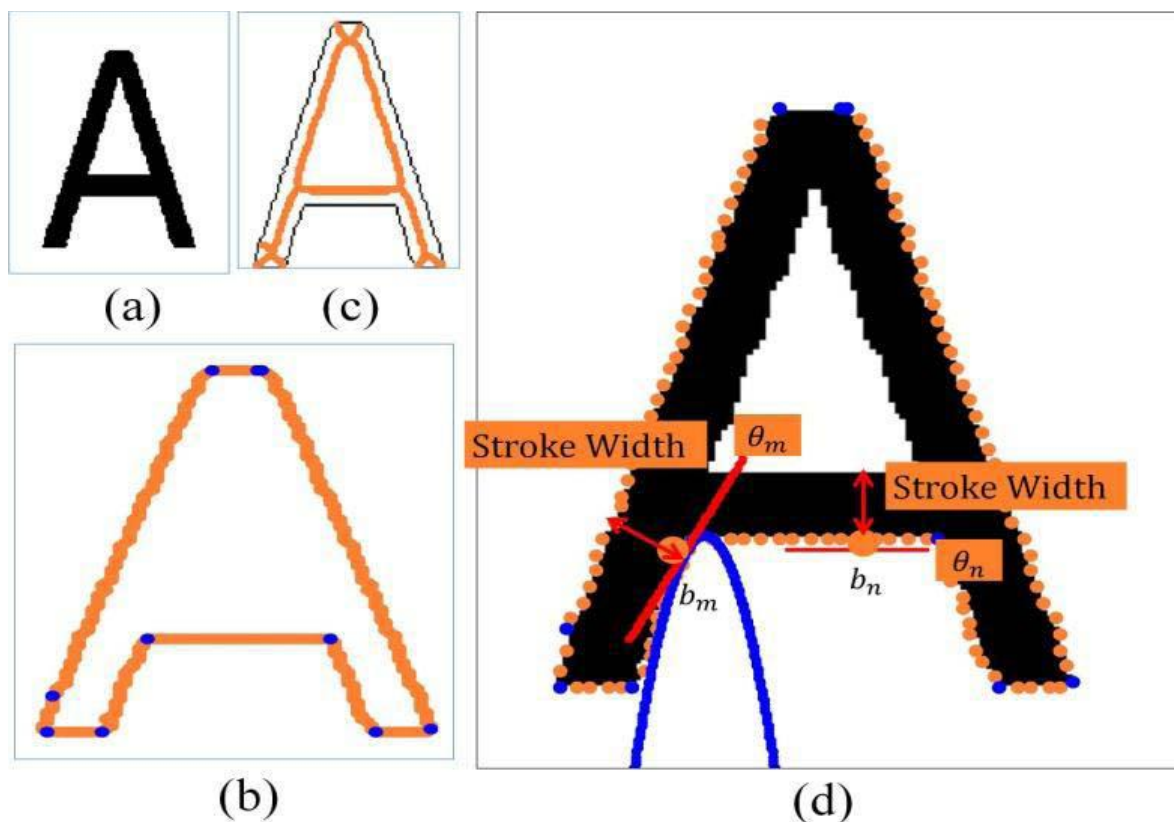
In our current work, scene text characters include 10 digits [0-9] and 26 English letters in upper case [A-Z] and lower case [a-z], 62 character classes in total. We are considering three public datasets are employed for training character recognizer and evaluating its performance, and these datasets contain image templates of complete and regular text characters cropped from scene images. We train a character recognizer to classify the 62 classes of characters. In text retrieval, character recognition is a binary classification problem. Classifiers compare the input feature with stored pattern and find out best matching class for input. There are many technique used for classification such as Artificial Neural Network (ANN), Template Matching, Support Vector Matching (SVM) *etc.* The classification is the process of identifying each character and assigning to it the correct character class, so that texts in images are converted in to computer understandable form. Each character image is mapped to a textual representation. The image from the segmented stage is correlated with all the templates which are preloaded into the system. Once the correlation is completed, the template with the maximum correlated value is declared as the character present in the image. For each of the 62 character classes, we train a binary classifier to distinguish a character class from the other classes or non-text outliers. For example, we train a binary classifier for character class 'A', then this classifier will predict a patch containing 'A' as positive, and predict a patch containing other character classes or non-text outliers as negative. Algorithm can be given as,

Step 5:

```
//Character recognition from CAPTCHA
//classifier will compare current character block with stored //data and will return matching value
Perform Step 2, 3
Open (Training_data.txt, READ)
for each rectangle in Img_mat do
    for each data in Training_data.txt do
        matching= Classifier()
        if(maximum_matching < matching) then
            maximum_matching= matching
        end for
    string_append (classification_char)
end for
Return string
```

### **3. Character Stroke Configuration:-**

From the pixel-level perspective, a stroke of printed text is defined as a region bounded by two parallel boundary segments. Their orientation is regarded as stroke orientation and the distance between them is regarded as stroke width. Here, the structure map of strokes is defined as stroke configuration. In a character class, although the character instances appear in different fonts, styles, and sizes, the stroke configurations is always consistent. For example, character 'B' is always a vertical stroke with two arc strokes in any pattern. Therefore for each of the 62 character classes, we can estimate a stroke configuration from training patches to describe its basic structure. We estimate stroke configuration by synthesized characters generated from computer software rather than scene characters cropped from scene images, because synthesized character can provide accurate boundary and skeleton related to character structure. The Synthetic Font Training Dataset proposed by [4] is employed to obtain stroke configuration. This dataset contains about 67400 character patches of synthetic English letters and digits in various fonts and styles, and we select 20000 patches to generate character patches. It covers all the 62 classes of characters. Each character image is normalized into  $n \times n$  pixels with no antialiasing. In estimating stroke configuration, character boundaries and skeletons are generated to extract stroke-related features, which are used to compose stroke configuration.



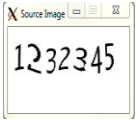

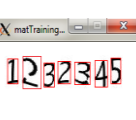
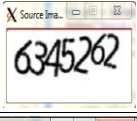

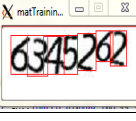
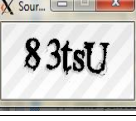

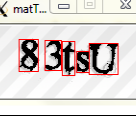
**Figure 5: Character Stroke configuration**

Finally, we can conclude that to support faster web crawling CAPTCHAs are breakable using this approach. Because more accurate CAPTCHA characters detection helps search engine to make crawling more effective and accurate.

### Result:-

As discuss above text detection and recognition approach for web crawling is simple and less time consuming. Few of the samples of CAPTCHA that we have tested are listed in table 1. Our approach successfully detects text region embedded in CAPTCHA image and characters as shown with higher accuracy.

CAPTCHA images	Text area recognition	Segmentation	Success rate
			85%
			70%

			90%
			85%
			90%

**Table 1: Table captions should be placed above the table**

### Conclusion:-

Embedding web crawlers with CAPTCHA resolver will improve efficient and speedy performance for uninterrupted web crawling. Character recognition technique will automate crawling process by reducing human involvement. Text recognition technique which includes text area separation from background, image segmentation, and character recognition can be combined for very less false positive rates in CAPTCHA breaking. Thus text resolving techniques can be efficiently utilized in resolving CAPTCHAs for web crawling, which ultimately results in improving the crawling process performance.

### Acknowledgement:-

We have taken efforts in development of this system. However, it would not have been possible without the kind support and help in every aspect. We would like to extend our sincere thanks to all teachers from our college PVG's COET Pune who supported us. We are highly indebted to sponsors of this work, Mr. Prashant Patil from Company Innoplexus consulting services Pvt. Ltd. for their guidance and constant supervision as well as for providing necessary information regarding the system. Our thanks and appreciations also go to my colleagues in developing the system and people who willingly helped us out with their abilities.

### References:-

1. **Shalini Sharma** (Department of Computer Science Shoolini University). "Web Crawler", April 2014 volume 4 IJARCSSE.
2. **Miss. Poonam B. Kadam, Mrs. Latika R. Desai** (Computer Department, D.Y.P.I.E.T). "A Hybrid Approach to Detect and Recognize Texts in Images" July-2013 Volume 2 IJARCSSE.
3. **Julinda Gllavata, Ralph Ewerth and Bernd Freisleben**. "A Robust Algorithm for Text Detection in Images", November 2010.
4. **Chuai Yi, Yingli Tian**, (Senior Member, IEEE). "Scene Text Recognition in Mobile Applications by Character Descriptor and Structure Configuration", IEEE Transactions on image processing, volume 23, no. 7, July 2014.
5. **Cong Yao, Xiang Bai, Wenyu Liu** (Member, IEEE). "A Unified Framework for Multioriented Text Detection and Recognition", IEEE Transactions on image processing, volume 23, no. 11, November 2014.
6. **Pavalam S M, S V Kashmir Raja, Felix K Akorl3 and Jawahar M** (National University of Rwanda). "A Survey of Web Crawler Algorithms", International Journal of Computer Science Issues, Vol. 8, Issue 6, No 1, November 2011.
7. **Max Jaderberg Karen Simonyan Andrea Vedaldi Andrew Zisserman** (University of Oxford). "Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition", cs.CV 9 December 2014
8. **Alessandro Bissacco, Mark Cummins, Yuval Netzer, Hartmut Neven**. "PhotoOCR: Reading Text in Uncontrolled Conditions", Volume 2 IJARCSSE 2013.
9. **Karishma Tyagi, Vedant Rastogi** (Rajsthan, INDIA). "Survey on Character Recognition using OCR Techniques", Vol. 3 Issue 2, February – 2014 IJERT
10. **Pratik Madhukar Manwatkar, Dr. Kavita R. Singh** (YCCA India). "Text recognition from images: A review", Volume 4, Issue 11, November 2014 IJARCSSE