# Optical Character Recognition

**SABER SHEYBANI**[1] **AND SUSHMITA SIVAPRASAD**[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: sheybani@umail.iu.edu,sushsiva@umail.iu.edu*

*project-000, May 1, 2017*

**Optical Character Recognition is a technology for converting images of texts, into machine encoded text format. In this project, the input data is in standard image formats and our goal is to recognize the words/letters in the image as accurately as possible and convert the dataset into TXT format. The heart of OCR as a pattern recognition system is a classifier. The images will go through a pre-processing phase which will convert them into a form that is compatible for feeding to the classifier. Then the classifier will determine the class of each glyph in the image. The whole recognition system is installed and operated on the cloud.**



**Keywords:** OCR, Ansible, K-Nearest Neighbor

https://github.com/cloudmesh/sp17-i524/blob/master/project/S17-IR-P012/report/report.pdf

## INTRODUCTION

The age of digitalization has made it quintessential to store documents in a digital form for the purpose of allowing valuable information to be stored, edited and searchable in the middle of billions of records. OCR technology is used quite frequently by every industry that has handwritten, scanned or photographic data. This project proposal gives a detailed view on how the OCR technology works, the preprocessing done to remove the noise and the algorithms used to recognize the images. We have delved deep into some of the basic concepts used in the implementation process. We have also discussed some important applications of this technology in the real world and created a benchmark.

## EXECUTION PLAN

This execution plan is to show the distribution of work over the time period of the course. The steps followed to achieve the final results.

1. **6 March - 12 March 2017** To implement the OCR, we first looked for a dataset to train the algorithm and test it. The desired library to use to write the codes and the complexity of the final goal of the project was discussed.

2. **13 March -19 March 2017** Looking for preprocessing steps in order to cleanse noise from the image and convert the image into a standard form which makes it adequate for later processing.

3. **20 March - 26 March 2017** Starting to work with Ansible and Cloudmesh Client for running tasks on virtual nodes.

4. **27 March- 02 April 2017** starting to deploy individual modules for preprocessing, to the virtual machines on Chameleon Cloud service, using Ansible. Cloudmesh Client was used to reserve the virtual machines on the cloud services.

5. **03 April- 09 April 2017** Executing a preliminary form of character recognition using K Nearest Neighbour classification and a large dataset.

6. **10 April - 16 April 2017** Implementing segmentation of image into lines, words and letters.

7. **17 April - 23 April 2017** Executing benchmark and finalizing the project report.

## BACKGROUND

### OCR Technology

Optical Character Recognition is a technology which is used to convert different types of documents that can be in the form of scanned documents, including handwritten or machine-written into an editable and searchable form [1]. The images can be in either the basic black & white or colored. The technology analyzes the structure of the document and divides it into small refined segments, so that each segment contains one character. Finally, individual characters are singled out one by one and fed to a classification algorithm which will return the closest letter that the individual character could possibly be identified with.

## SYSTEM CONFIGURATION AND TECHNOLOGIES USED

### System Configuration

The python codes and the libraries are run on a Ubuntu 16.04 LTS.

### Technologies Used

1. **Python Programming Language** : An object oriented programming language with an open-source license. We chose Python for this project as there are many libraries available in Python that helps in creating a code for OCR easier. Packages such as PIL, Pillow and OpenCV are one of the few libaries that helps in image processing.

2. **OpenCV** : OpenCV is an open source computer vision library. It is used for building computer vision applications. It consists of more than 2500 algorithms including both machine learning and computer vision algorithms [2]. These algorithms are devised for facial and image recognition, track any moving objects etc. We have used OpenCV 2 for creating a KNN classifier and a few of our preprocessing techniques.

3. **Ansible** : Ansible is an IT automation tool. It uses YAML in order to issue the state of the server [1]. Ansible implements the internal command that is required to reach that state which depends on the operating system. The ansible playbook which consists of these internal commands can be applied to any server or service. There is no requirement to instal an additional software on the target system as the commands are run over an SSH session.

4. **Cloudmesh Client** : It is an open source client interface tool that provides us with an easy-to-use interface for accessing cloud services, creating single and multiple VMs, clusters and workstations.We can manage the resources we would like to use and customize them as per our requirement to run the projects.
   It provides an interface to execute jobs on High Performance Computing clusters [3].The users can use just one platform to manage all of the cloud resources. Cloudmesh client creates a local copy of the data which results in clouds with similar configurations to be created as well. The default features of the Cloudmesh Client allows easy control of the cloud as well. Cloudmesh includes an API, commandline client and a commandline shell.

5. **Chameleon Cloud** :Chameleon cloud provides OpenStack Cloud (kilo) using the KVM virtualization technology [4]. It is an Infrastructure as a Service(IaaS) platform that allows us to create as well as manage the virtual environment. The virtual machine that we use here is compatible with KVM. Chameleon Cloud also gives access to the bare-metal computing resources, which allows administrative rights to use cloud for computing experiments [5].

## METHOD SURVEY

Optical Character Recognition have already been developed in numerous ways, focusing on different goals. We did a survey on the possible approaches for character recognition.

The main components of every OCR system can be enumerated as feature extraction component, and the classifier.

Feature extraction methods can be separated into two groups: Template matching, and Structural classification [6]. In **Template matching**, the individual pixels of the image are directly used as features. For each possible character, there is one class and one template feature set, associated with it. In classification, a similarity metric will evaluate the distance of the input character to each of the templates. And thus, the input will be represented with the class of the most similar template.

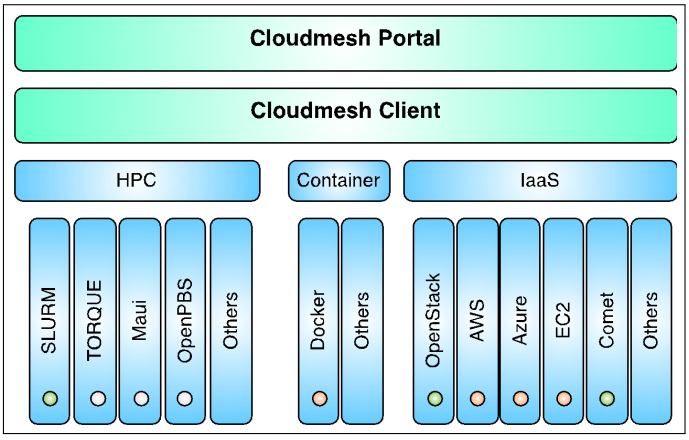


**Fig. 1.** Architecture of Cloudmesh Client [3]

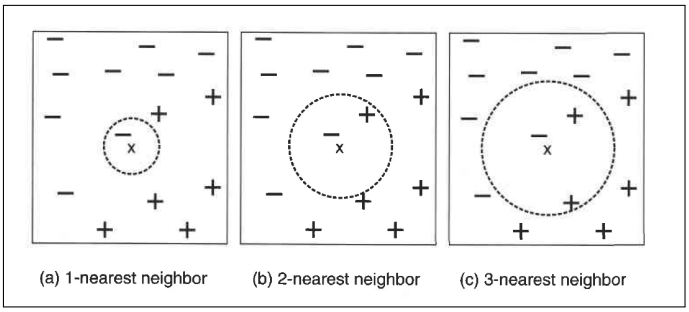


**Fig. 2.** Example of k-NN classification [7]

In **structural classification** structural features of every character, such as loops and curves, are used as features.

There are various types of classification methods. A number of methods that have been used for OCR are Nearest-Neighbor, Artificial Neural Networks, and Support Vector Machines. K-Nearest Neighbor and Artificial Neural Networks are discussed further as the candidates for our job.

### K Nearest Neighbour

It is a non parametric algorithm where each of the training data is considered to have a set of vectors and a class label associated with each vectors. The training set will have the labels for the classes , given the test set it calculates the distance between the training set and the test set and calculate the nearest points. A single value of 'K' is given which allows us to decide how many neighbours influence the classification. Figure 2 displays a schema of KNN classification.

### Feed Forward Neural Networks

Artificial Neural Network is a paradigm in computing, inspired by the structure of biological nervous systems. It consists of a network of processing units, where the output of each unit is a nonlinear function of its weighted inputs that come from other units. Such network can be trained to solve different kinds of problems, including classification and clustering. A

feed forward neural networks is one in which the neurons are organized in a number of layers and each layer only feeds to the next one, but not to the previous one (no feedback). However, in a back-propagation process, the errors from one iteration of classification will be fed back from the output to the network, in order to modify and improve the network for next iterations.

Due to simplicity and relatively good performance of KNN, we choose it as the classifier for this project. There is an easy to use implementation of it in OpenCV library. Our KNN uses euclidean distance as the distance metric to calculate the nearest neighbours for the 'K' value.

Euclidean Distance

$d(p,q) = \sqrt{\sigma(p_i - q_i)^2}$

For the value of K, 5 is chosen so that it is neither too small and sensitive to the noise nor too large which might result in including the points from other classes.

## PREPROCESSING TECHNIQUES

The steps in an OCR full session are as follows: Preprocessing: The input images need to be segmented into units that each of them keep only one glyph (symbol). Also, the colored or grayscale images will be binarized. Feature extraction: The glyphs will be decomposed into features like lines, closed loops, line direction, and line intersections. Character recognition: The image features will be fed to the classifier and they will be compared with stored glyph features and the nearest match will be chosen.

Preprocessing is required on the raw images that we are using to filter out the required subject and distinguish from any other unwanted objects from the image such as watermarks, background subjects etc. We have conducted different preprocessing techniques in order to remove noise and convert the image into a grey scale format as color images requires more complex methods of processing

### Noise Reduction Techniques

Noise reduction is done for extracting out any unwanted bit-pattern, there are linear as well as non-linear techniques for this. Linear : In this method is used to remove any isolated pixel noise from the image. Here the required output filter is taken as a linear combination of the neighborhood pixels Non- Linear : These kind of filters are used to replace the value of a particular pixel in order to remove any kind of impulse noise

### Histogram Based Method

It gives a value to the intensity of the pixel and plot it on a histogram , where darker the image , more the data points would be on the left and center of the histogram . Lighter the image , more the data points would be on the right side of the histogram. Using a histogram equalization method the contrast on the image can be improved in this case. In the histogram equalization method , an image is divided into blocks of pixels and an histogram equalization is done. This allows us to distinguish the images we actually require from the other background images . It allows us to enhance the visibility of the characters' present on the image.

### Median Filter

It is a non-linear noise reduction technique , it is a low pass filter. In this case the pixel values are taken for an area on the image and an average of the pixel value is taken and assigned to the center pixel in that area. Figure 3 displays a schema of how median filter works. It is an effective means for removing the salt and pepper noise which are random lines occurring on the image due to poor quality of the picture or if the image wasn't scanned well [8]. Figure 4 shows the result of applying a median filter on a scanned image, we can see the reduction in dots and other marks on the image, making it more smooth and usable.

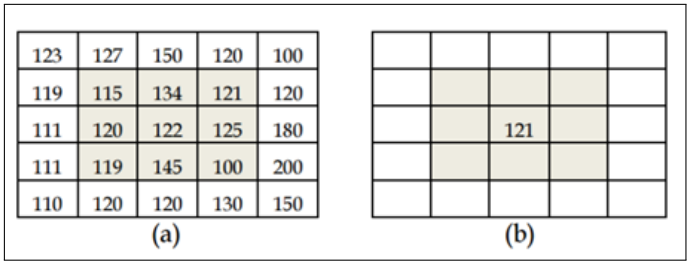


**Fig. 3.** Averaging of a pixel in median filter [8]

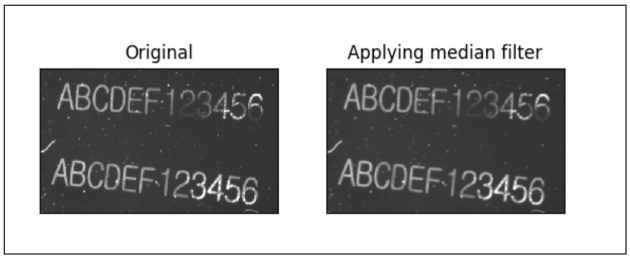


**Fig. 4.** Result of applying median filter on a scanned image

### Gaussian Blur

Gaussian Blur filtwe is a low pass filter which is used to eliminate isolated pixel noise. Image smoothing is done here using gaussian filters where the weighted average of the pixel values is computed with the gaussian coefficients as weight. The filter provides a smooth texture to the noisy image.


**Fig. 5.** After applying gaussian filter on the image preproccesed with median filter

### Binarization

Otsu's method [9]: Otsu's method concludes finding the best intensity threshold to separate two classes, often background vs foreground but not always. The algorithm tries to find a separation point that has the minimum weighted within class variance. If the input images are grayscale, the algorithm will simply find a threshold that any intensity below that will be considered as the background and the intensity of the corresponding pixels will be rounded to zero. Similarly, the intensities above the threshold will be rounded to 1. The resulting image (array) will be binary.

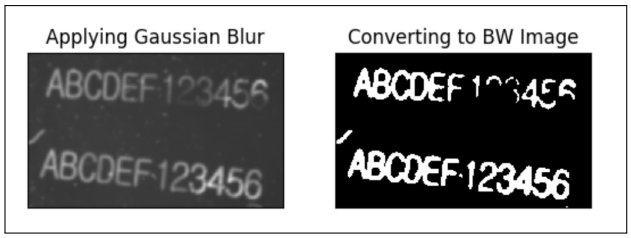


**Fig. 6.** Applying binarization after applying gaussian filter

### Segmentation

Segmentaion of the image happens in multiple levels, namely lines, words and letters. Application of all of these three will enable the conversion of whole pages of scanned documents into text format. In this project, segmentation at all levels is implemented using projection profile approach. Projection of the intensities on the vertical axis will differentiate the rows that contain some text, from the ones that only include the background. For word segmentation if the number of background columns are more than a threshold, the two letters will be considered as incorporating different words. The background columns are detected using projection on the horizontal axis, but only for the rows that are included in the current line. Thresholds are defined using expert views in typography [10].
The letter segmentation is simply done after one background column is found. The figure 7 displays the application of our segmentation algorithm on a sample image.

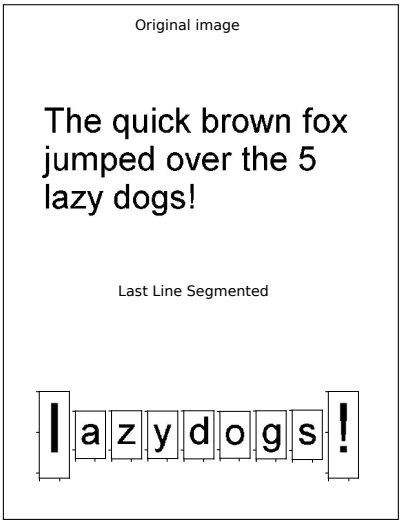


**Fig. 7.** Applying segmentation

## APPLICATION

OCR converts images to machine-readable text. That will make it the initial tool that needs to be used for processing any documents or simply any written material in a digital image, which has been captured by a camera [11]. It's output can be stored significantly more compact than scanned images. But beyond that, it enables us to process the output information for numerous applications. Examples of these applications include creating a narrator machine to help the visually impaired read nondigital documents and signs, or automatic recognition of automobile number plates.

## LICENSE

The project is developed under the open source Apache License 2.0. The license file is included in the Git repository of the project. The packages and softwares used in developing the project include OpenCV, Python 2.7, Ansible, Cloudmesh. All of these packages are open-source. The license file for OpenCV is included in the project repository (CV_LICENSE.txt).

## CLOUD DEPLOYMENT

Automatic deployment of program on virtual environment was done using Ansible [12] .The jobs were collected, organized in Playbooks [13] and run on virtual clusters provided by Chameleon Cloud [14]. The tasks include Installing the essential libraries on the remote machine and running the program. The VM reservations were done using Cloudmesh Client. Three Ansible playbooks are used, one for deploying the software stack, one for running the OCR codes on a standard dataset and another one for running it on an arbitrary image. The software stack required for running the program includes the following components:

1. **Git** The Git module is needed for cloning the OCR Python codes. It is installed using apt module.
2. **OpenCV** OpenCV, as discussed in the previous sections, has become the standard library of computer vision. It is used in this project for multiple purposes. As this library is extensive, its installation may take a significant time. Hence, we tried to examine all the possible ways to install it. Building from the source, which is the recommended way for installing it (and any other package) took more than 20 minutes. Even after that, connecting it to the existing Python installation on the VMs was problematic. The alternative ways include using PyPi, Conda (Anaconda package management system), and apt. After trial and error, Apt appeared to be the best solution with installation time of less than 5 minutes and hassle-free python integration. It can be easily installed on VMs using Ansible apt module. So we chose this method.
3. **OCR Code** The codes of the program were checked in the project repository. An Ansible role cloned these codes into each node of the cluster.
4. **Dataset** An Ansible role was used to download and extract the dataset. Unarchive module was used for this purpose.

## DATASET

The dataset used in our project for training the KNN is Chars74k [15]. This dataset provides an extensive collection of English letters and digits, in handwritten form or in various computer fonts

[16]. In this project, a subset of the data, including characters from computer fonts with 4 variations (combinations of italic, bold and normal) was used. The glyphs in this subset include 0-9, A-Z, and a-z. For each glyph, there are 1016 different files, each with one font. However, some of the fonts are similar to each other.
Before this dataset, the MNIST handwritten digits database were used for preliminary purposes [17].

## BENCHMARKING

After wrapping all the necessary material, the code was run in a role and the result was saved in text files, using another role. The accuracy of classification is printed in the console of the local machine. The program was first deployed on single VMs that were reserved manually using Cloudmesh Client. Later on, clusters of 2-4 nodes were used to measure the execution time of deployment. Note that for running the benchmark, there is a trade-off between the runtime and accuracy of classification. The more samples are used for training the dataset, the higher the accuracy will be achieved and of course the longer execution time is needed.

### Sample size analysis

Running a dataset of 50 images as the test and train data with K=5, takes around 20 seconds and yieds an accuracy of 65%. The time complexity of the algorithm is O(n2), thus, using 100 samples will multiply the runtime by 4. However, as the dataset used contains 74k images, there is still significant diversity among 100 images. As a result, using 100 images only improves the classification accuracy up to 75%. Changing the number of nodes does not affect the runtime, unless paralellization is exploited.

Each VM on Chameleon Cloud has only 1 core, 2050076 kB (2 GB) of RAM, and 20 GB of storage. Due to small size of RAM, we had to keep the number of training samples under 200.

### Cluster size analysis

In this section, the sample size is around 50 different fonts for training the classifier and another 50 for testing it.

For the cluster of **2 nodes**: It took 67 seconds to deploy (allocate) the cluster, 153 seconds to install the software stack and dataset on the nodes, and 169 seconds to run the benchmark on them. The accuracy of classification on one of the nodes was 72.23 and on the other, it was 70.84.

For the cluster of **3 nodes**: It took 98 seconds to deploy (allocate) the cluster, 170 seconds to install the stack and 180 seconds to run the benchmark.

For the cluster of **4 nodes**: It took 131 seconds to deploy (allocate) the cluster, 175 seconds to install the stack and 187 seconds to run the benchmark. As we can see, the higher number of nodes will only add to the time of cluster creation. As the nodes are quite similar, the execution time of the tasks which run in parallel is similar.

### OCR on arbitrary images

As the arbitrary images can be significantly different from the standard dataset, the sample size needs to be much higher. However, due to small size of RAM, for running OCR on arbitrary images, we only used a sample size of 200. That causes the accuracy of OCR to be very low, around 20%.

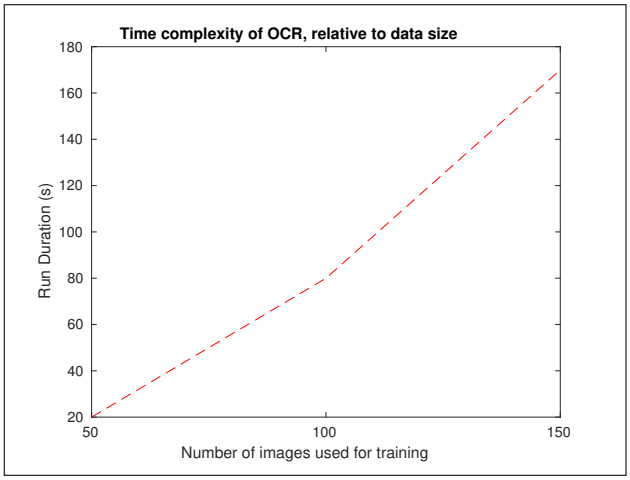


**Fig. 8.** Time complexity of running OCR on Chameleon Cloud

## REPRODUCIBILITY

Instructions on reproduction of the project are provided in the project repository [18].

## FUTURE WORK

For improving the current OCR system, various kinds of sophisticated structural features can be added to feature extraction. It can also be extended to operate on languages other than English. In order to improve the performance, the power of multiple cores/VMs can be used for parallelization, so that after segmentation, each VM will operate on one line of the text. Thus, multiple lines can be processed at the same time. For improving the recognition accuracy, a lexicon can be used to evaluate the validity of the words that have been formed by OCR, and then possibly correct them to the closest word in the lexicon.

## ACKNOWLEDGEMENT

A very special thanks to Professor Gregor von Laszewski and the teaching assistants Miao Zhang and Dimitar Nikolov for all the support and guidance. This project proposal is written during the spring 2017 semester course I524: Big Data and Open Source Software Projects at Indiana University Bloomington.

## AUTHOR BIOGRAPHIES

**Saber Sheybani** received his B.S. (Electrical Engineering - Minor in Control Engineering) from University of Tehran. He is currently a PhD student of Intelligent Systems Engineering - Neuroengineering at Indiana University Bloomington.

**Sushmita Sivaprasad** is a graduate student in Data Science at Indiana University under the department of Informatics and Computing. She had completed her bachelors in Electronics and Communication from SRM University , India and her master's in International Business from Hult International Business School, UAE.

## REFERENCES


[1] "What is ocr and ocr technology," Web Page, 2017. [Online]. Available: https://www.abbyy.com/en-us/finereader/what-is-ocr/

[2] "About opencv," Web Page, accessed: 2017-4-6. [Online]. Available: http://opencv.org/about.html

[3] G. V. Laszewski, "Cloudmesh, an introduction," presentation, p. 31, 2015. [Online]. Available: https://drive.google.com/file/d/0Bx_sUfI4VkKSVG9KOE8xU05KREE/view

[4] "Openstack kvm user guide," Web Page. [Online]. Available: https://www.chameleoncloud.org/docs/user-guides/openstack-kvm-user-guide/

[5] "Bare metal user guide," Web Page. [Online]. Available: https://chameleoncloud.org/docs/bare-metal-user-guide/

[6] E. Borovikov, "A survey of modern optical character recognition techniques," *arXiv preprint arXiv:1412.4183*, 2014.

[7] V. K. Pang Ning Tan, Michael Steinbach, *Introduction to Data Mining*. Pearson Education, 2006.

[8] Y. Alginahi, "Preprocessing techniques in character recognition," pp. 9–10. [Online]. Available: http://cdn.intechopen.com/pdfs/11405.pdf

[9] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[10] G. Dowding, *Finer points in the spacing and arrangement of type*. Hartley & Marks Publishers, 1995.

[11] "Optical Character Recognition," Web Page, Mar. 2017. [Online]. Available: https://en.wikipedia.org/wiki/Optical_character_recognition

[12] "Ansible," Web Page. [Online]. Available: https://www.ansible.com/

[13] "Playbook," Web Page, Mar. 2017. [Online]. Available: http://docs.ansible.com/ansible/playbooks.html

[14] "A configurable experimental environment for large-scale cloud research," Web Page, Jan. 2017. [Online]. Available: https://www.chameleoncloud.org/

[15] "Character recognition in natural images," Web Page. [Online]. Available: http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/

[16] T. E. de Campos, B. R. Babu, and M. Varma, "Character recognition in natural images." in *VISAPP (2)*, 2009, pp. 273–280.

[17] C. J. B. Yann LeCun, Corinna Cortes, "The mnist database of handwritten digits," Web Page. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[18] "Readme," Web Page. [Online]. Available: https://github.com/cloudmesh/cloudmesh.ocr/tree/master/code