IMAGE PROCESSING APPLICATIONS FOR INTEGRATED STEEL PLANTS

M. Tech. Thesis

By

MOHIT JAGDALE

DISCIPLINE OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JUNE 2020

IMAGE PROCESSING APPLICATIONS FOR INTEGRATED STEEL PLANTS

A THESIS

Submitted in partial fulfillment of the
requirements for the award of the degree
of
Master of Technology

by

MOHIT JAGDALE

DISCIPLINE OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
JUNE 2020

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **IMAGE PROCESSING APPLICATIONS IN INTEGRATED STEEL PLANTS** in the partial fulfillment of the requirements for the award of the degree of **MASTER OF TECHNOLOGY** and submitted in the **DISCIPLINE OF ELECTRICAL ENGINEERING, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from JULY 2018 to JUNE 2020 under the supervision of Dr. Prabhat Kumar Upadhyay, Associate Professor - Electrical Engineering, IIT Indore and Dr. Jose Korath, Chief - Intelligent Systems & Mathematical Modelling, Tata Steel, Jamshedpur.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

18.06.2020

Signature of the student with date
(MOHIT JAGDALE)

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge.

18-06-2020

Signature of the Supervisor of
M.Tech. thesis #1 (with date)
Dr. Prabhat Kumar Upadhyay

18.06.2020

Signature of the Supervisor of
M.Tech. thesis #2 (with date)
Dr. Jose Martin Korath

MOHIT JAGDALE has successfully given his M.Tech. Oral Examination held on **<Date of M.Tech. Oral Examination>**.

Signature(s) of Supervisor(s) of M.Tech. thesis
Date: 22-6-2020

Convener, DPGC
Date: 22-6-2020

Signature of PSPC Member #1
Date: 22-06-2020.

Signature of PSPC Member #2
Date: 22/06/2020

ACKNOWLEDGMENTS

I take this opportunity to express my sincere gratitude to my project guide **Dr. Prabhat Kumar Upadhyay**, Associate Professor, Discipline of Electrical Engineering, IIT Indore for his constant encouragement and guidance during my Master's study and thesis work.

I'd like to express my deepest gratitude to my guides at Tata Steel, **Dr. Jose Martin Korath,** and **Mr. Chandan Kumar Lal Das** for their precious guidance, support, and motivation at all the stages during the entire work. The valuable time spent with them has undoubtedly helped in supplementing my thoughts in the right direction for attaining the desired objective.

I also express my sincere gratitude to my PSPC members **Dr. Surya Prakash** and **Dr. Swaminathan Ramabadran** for their valuable suggestions and feedback.

MOHIT JAGDALE

MT1802102002

MTech (Communication & Signal Processing)

Discipline of Electrical Engineering, IIT Indore

DEDICATION

I dedicate this thesis and my project work to the God Almighty, who guides me through my Teachers and supports me through my Family…

Abstract

Manufacturing processes in an integrated steel plant involve handling, tracking and characterization of bulk raw material & products at different stages of operations. Computer vision is a promising and cost-effective technology which is finding application at an ever-increasing pace in many of the solutions used in the integrated steel plant. Imaging solutions are used for applications like material identification, size estimation, product tracking & quantification and quality inspection etc. in steel manufacturing process. This dissertation work applied three different flavors of image processing techniques to address three business cases across raw material and product handling in the steel plant. The entire project work can be arranged in three different phases as described below.

In the first phase of the project, automatic estimation of the size distribution of coke particles using image analysis was attempted. This is done using the watershed algorithm for image segmentation to replace the existing technique of manual size determination using Sieve Analysis. The advantage of this is better accuracy, more reliability and productivity.

In the second phase, we worked on the problem of Bar Counting. The bars are currently dispatched by their weights after being manufactured from the Bar Mill. The main problem with this is the pilferage taking place during transport. The objective of this project is to accurately count the bars using Image Processing algorithms. This time, we scaled up from the conventional algorithms to use machine learning techniques for object detection purposes. More than 97% accuracy is achieved until now.

In the third and the last phase, we take up the problem of finding out the sizes of the stone fragments. Calculating the size distribution of rock particles is very crucial in getting to know whether the blasting done at the mining site is optimal or not. The segmentation task was very challenging as there was no particular shape or texture present for a specific stone fragment. Finally, this is done using the deep learning technique of mask R-CNN.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

PSD	Particle Size Distribution
ML	Machine Learning
HOG	Histogram of Oriented Gradients
LBP	Local Binary Patterns
CNN	Convolutional Neural Network
AI	Artificial Intelligence
DL	Deep Learning
ROI	Region Of Interest
RPN	Region Proposal Network
FPN	Feature Pyramid Network
COCO	Common Objects in Context
VIA	VGG Image Annotator

Chapter 1

Introduction

An integrated steel plant consists of various mills and plants operating together with a specified function assigned to each of them. Many of them are there to process the raw particles such as iron ore and coal particles. Also, various plants are there for the purpose of conversion of iron to steel to produce different products out of it. Tata Steel is one of such very early established plants. Many of the processes are working in a conventional way and many systems are needed to be modernized. This challenge of formulating such problems of revolutionizing a particular system is done by the Automation Division in the Tata steel.

Image processing is a branch of signal processing dealing with different algorithms related to images to extract useful information out of it. Due to the current advancement in computer processing powers the image processing is becoming much real-time. Thanks to stronger computers, not only large data such as images can be processed in a much faster way but many machine learning and deep learning algorithms are becoming popular in solving real problems.

This project deals with applying the concepts and algorithms in the field of image processing to solve some of such industrial problems so that the age-old systems can be replaced with much reliable and much smarter systems. In the early phase of the project, we begin our work by applying the conventional image processing algorithms to estimate the coke particles' size distribution. Then in the second phase, we use a little more advanced machine learning algorithm to solve the problem of Bar-counting. In the third and the last phase, we use the most advanced technique of instance segmentation which is originally developed at FAIR(Facebook AI Research) to solve the problem of estimation of stone fragments' size distribution.

1.1 Business problems implemented using image processing

The three business problems at Tata Steel which were approached to be solved using image processing/computer vision algorithms are as follows:

1. Estimation of coke particles' size distribution
2. Bar-counting
3. Stone fragments counting and size estimation

The coke particles are used in the blast furnace for the reduction of iron ore. The permeability inside the furnace depends upon the size distribution of the coke particles which is currently calculated using a manual method called as sieve analysis. It is essential to modernize this method considering the low accuracy and heavy labor work that it involves. We try to approach this problem using continuous imaging and parallelly estimating the size distribution using image segmentation techniques. For this purpose, we've used the marker based technique of Watershed segmentation. We've achieved the size distribution very close to the ground truth distribution. Still, accuracy can be improved using a better method for the marker selection.

Then in the second phase of the project, we try to solve the problem of Bar-counting. Advancement in this problem is necessary to avoid the pilferage of bars taking place during the transport. This time, we tried to use a much advanced approach of machine learning combined with image processing for the object detection and classification purpose. We try to extract the features of the bar objects in the image that can define the texture and the shape. We developed a unique logic for actual classification of the object within the image as being a bar or a non-bar and by that actually counting the total number of bars in the image. We've achieved the accuracy of more than 97% which can still be improved.

In the third and the last phase, we've approached the problem of counting the stone fragments in the image along with finding their size measurements. Blasting is done at coal mining site to remove the upper layers to reach to the coal. From the stone fragments' size, it is possible to conclude if the blasting is done optimally or not. Currently this problem is solved manually using some sketching tools which is extremely time-consuming. Initially we approached this problem using conventional techniques that failed as there is no specific texture or shape or color present that can define a specific stone fragment. Ultimately, we used the very advanced technique of instance segmentation using the Mask R-CNN model to get the desired results.

1.2 Conventional image segmentation techniques

Image segmentation is a very essential and fundamental step in image analysis. Image segmentation basically partitions the image into different parts which are in some way or other hold similar

characteristics such as features or various attributes[1]. Image segmentation can be basically done in two ways. The first is semantic segmentation and second is instance segmentation.

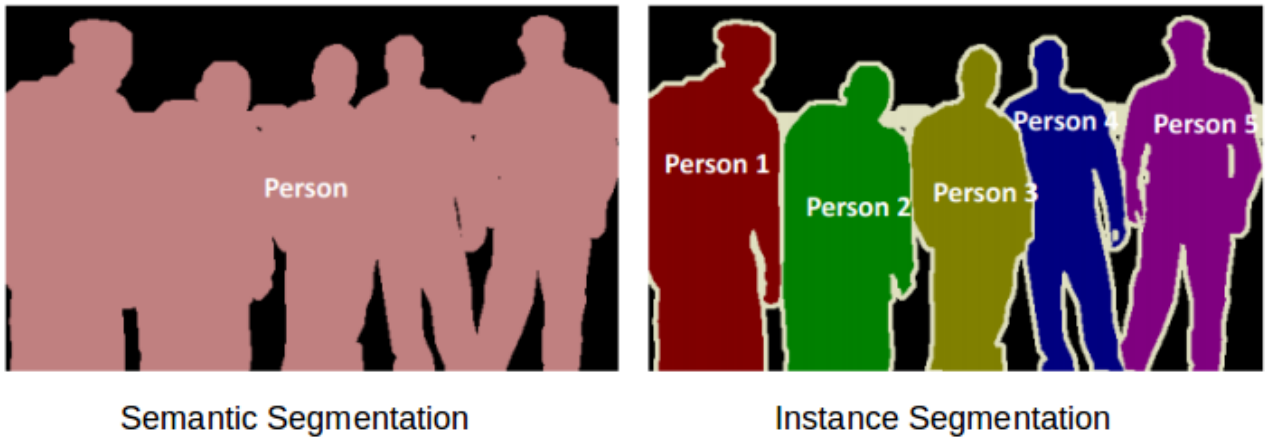


Fig 1.1 Semantic and instance segmentation [2]

The semantic segmentation can only separate the object pixels from the background but the Instance segmentation separates objects from each other at a pixel level.

There are many conventional techniques for performing image segmentation. The most common among them is thresholding and edge-based segmentation. In the thresholding technique, the image is binarized using a particular set threshold. The edge-based technique works by detecting sudden changes in the intensity levels of the pixels. These techniques mainly use first and second derivative operators such as Sobel and Robert's operator for this purpose. The region-based methods are based on finding the regions which have similar characteristics. Region growing and region splitting are the techniques used in this method. We've tried to approach the problem of coke particle size distribution using classical techniques such as the Canny edge detector and Watershed algorithm.

1.3 Object detection using machine learning

Machine learning is a branch of Artificial Intelligence that includes algorithms to make a computer to learn from its own experiences without being explicitly programmed. Popular machine learning methods include supervised ML algorithms in which a labeled data is used to make a computer

making a hypothesis about the function for predicting the output for future unseen data. Whereas in the unsupervised algorithms, the data used to train the model is not labeled and the objective of the algorithm is to find the pattern in the data to conclude the right inference. There are two more varieties of algorithms also present namely Semi-supervised algorithms and Reinforcement learning algorithms.

In the problem of bar counting, we first extracted the useful feature vectors describing the texture and the shape of the objects to be classified. Hence we created a labeled dataset matrix. So, we used the supervised learning algorithm, Support Vector Machine(SVM) to classify the data by creating a hyperplane in the n-dimensional hyperspace.

1.4 Instance segmentation using deep learning

Deep learning is a subset of machine learning. The motivation behind deep learning-based networks is to imitate the human brain to process the data. The deep learning networks are proven to be very efficient in processing many forms of data such as texts, sound, and images. Deep learning networks consist of multiple layers starting from input layers to hidden layers and finally the output layer. Each layer consists of many nodes called neurons. Each of the neurons can be said to be a mathematical function that holds its own weights and bias to produce the output using the activation.

Though there are machine learning algorithms available, we can understand the need for ANN(Artificial Neural Network) with a simple example. Suppose there's a highly non-linear, 100-dimensional data. To classify this data, even if we extract just the 2^{nd} order features, it'll create around 5000 features which will be a hard problem for an ML algorithm. Hence we need deep learning which will automatically select which features are actually important for the classification purpose. It does so by automatically adjusting all the weights and biases of each of the neurons in the network using the gradient descent and back-propagation so that the cost between the result and the desired output is reduced.

CNN(Convolutional Neural Networks) has proven to be very efficient in processing the image data either for the object detection purpose or for the image classification purpose. There's a complete family of deep learning algorithms for the image processing purpose which include R-CNN, then Fast R-CNN, and the most advance Mask R-CNN.

We used the mask R-CNN algorithm in the problem of stone fragments' size distribution to finally reach the solution.

1.5 Organization of thesis

The rest of the thesis is organized as follows:

Chapter 2 presents the details of the work done in the first phase of the project. This includes the problem of estimation of coke particles' size distribution using a Watershed algorithm.

Chapter 3 elaborates the problem of Bar-counting tried to be solved in the second phase of the project. This is done using machine learning algorithms.

Chapter 4 includes the discussion on the problem of stone fragments size calculation using initially the conventional algorithms and then with the deep learning approach.

Chapter 5 presents the conclusion and future work.

Chapter 2

Estimation of coke particles' size distribution

2. 1 Problem objective

To automate the measurement of raw material particle size distribution in an Integrated Steel Plant using image processing techniques.

2. 2 Background

In an Integrated Steel Plant, the reduction of iron ore takes place at Blast Furnace. Permeability inside this Blast Furnace determines the efficacy of the reduction taking place in which coke particles are mostly used for the reduction. So, the permeability inside the furnace depends upon the size distribution of the coke particles used in the reduction reaction. If the size of the particles is too small, there won't be enough space for the gas to pass through. And if the size is too large, coke combustion won't be optimum. So, coke particles of optimum size are used in the furnace. Hence, to estimate the distribution of coke particles is essential before they're used inside the furnace.

2.3 Present practice

Presently, coke particle size distribution is estimated using a conventional method known as 'Sieve Analysis'.

Fig 2.1 shows a sieve that is used for this method of Sieve Analysis. In this method, distribution is calculated manually. Workers first collect a sample of around 200 kg. Keeping sieves of various sizes on top of each other in descending size order, workers distribute the coke particles on these sieves. In this way, coke particles of varying sizes get segregated. Size-wise weighing is done. And, according to that, the final distribution is found out.

Fig 2.1 Sieve used in Sieve analysis

There are many drawbacks to using this method. Some of them are as follows:

1. Intensive labor work is required to carry out the entire process.
2. The time required is also very high.
3. If the aspect ratio of the particle is high, then PSD from the sieve analysis could be misleading.
   Hence erroneous size classification takes place.

2.4 Proposed approach

To replace this conventional method with an automated one, we proposed the following approach.

As shown in the above Fig 2.2, in the proposed approach, coke particles will be continuously spread over a conveyor belt. Continuous imaging of those particles will be done from a camera device from a fixed height.

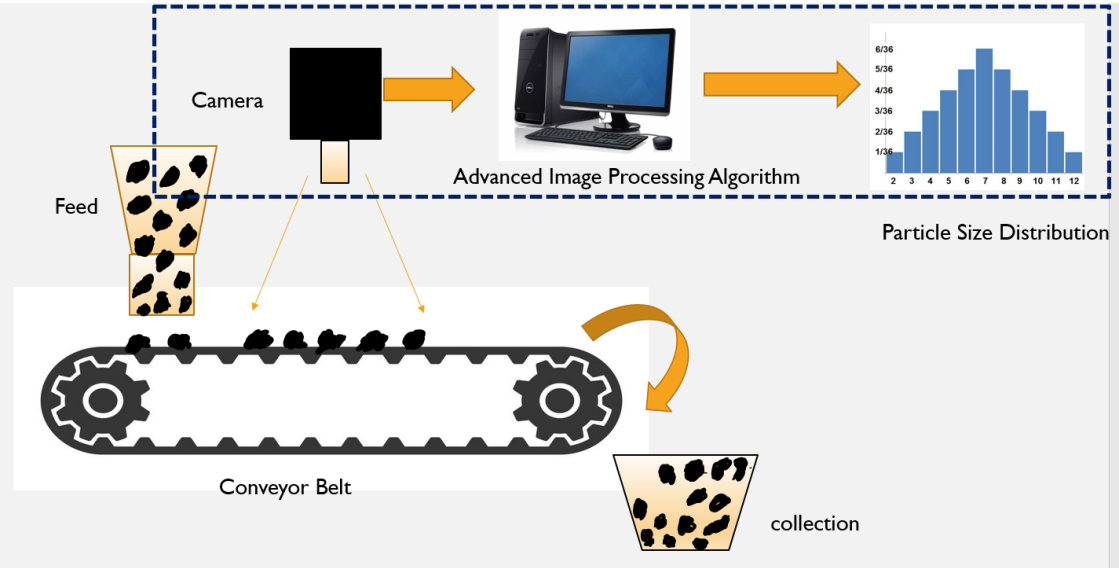


Fig 2.2 Representation of the proposed approach

These images captured will be processed through some advanced image processing algorithm. The output of the algorithm will be particle size distribution.

This chapter further describes different image processing methods/algorithms used to get the required PSD(Particle Size Distribution).

2.5 Image segmentation using Morphological Watersheds

Image segmentation in digital image processing is a process of partitioning a digital image into multiple segments or sets of pixels. The objective of image segmentation is to simplify the image so that further analysis becomes easier. Image segmentation is typically used to locate objects or boundaries in the image. More precisely, image segmentation is the process of giving a label to every pixel in an image such that pixels with the same label have similar characteristics.

To begin with the entire process, imaging of coke particles was done. Coke particles from a single sample were spread on a very clean white background. And the imaging was done from a fixed height. So, the images in the experiment resemble with the images that are going to be processed as in the approach stated above.

One of the major challenges in the process of segmentation of the images of the coke particles was that in every image along with the coke particles the unnecessary dust was also present. So our preprocessing before the actual segmentation should be such that this dust gets eliminated. Also, one more challenge is that whenever the particles get spread on the conveyor belt as in Fig 2.2, they're most likely to spread in such a way that the particles are touching each other.

Keeping this in mind, choosing the right method of image segmentation was a crucial choice. More general and relatively easy techniques such as edge-based segmentation do not work here properly as it's in most cases unable to separate the coke objects from each other which are touching to each other and in which there is no background. So we began our experiment with image segmentation using morphological watersheds.

2.5.1 Background

To understand the idea of watersheds, we need to first visualize the image in a three-dimensional view also known as a topographic view. In this view, we have three types of points 1. Regional minimum points. 2. Watershed of a minimum where if a drop of water if placed certainly falls to a single minimum. This region is also called as 'catchment' of that minimum. 3. The 'watershed lines' which divide the topographic surface, at which water drop is equally likely to fall to any side of the minimum[3].

The goal of this segmentation algorithm is to find this watershed/divide lines.

A simple idea that is used in this algorithm can be described using a simple analogy. Consider an image in which the divide lines have to be found as shown in Fig 2.3

The topographical view of this image is shown in Fig 2.4. The topographic view means that the height at a particular pixel is proportional to its intensity.

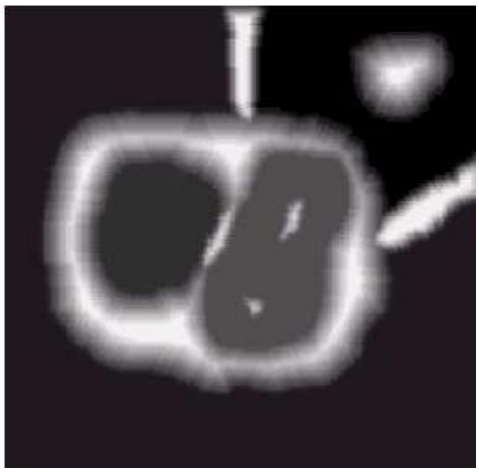


Fig 2.3 Original image

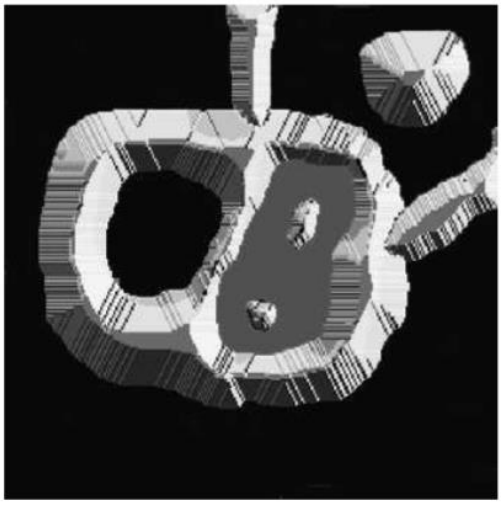


Fig 2.4 Topographic view

Now, in this topographic view, assume that from the bottom of each regional minima, a hole is punched. And water is allowed to flood in through this hole. Know that water gets inside this basin at a constant rate.

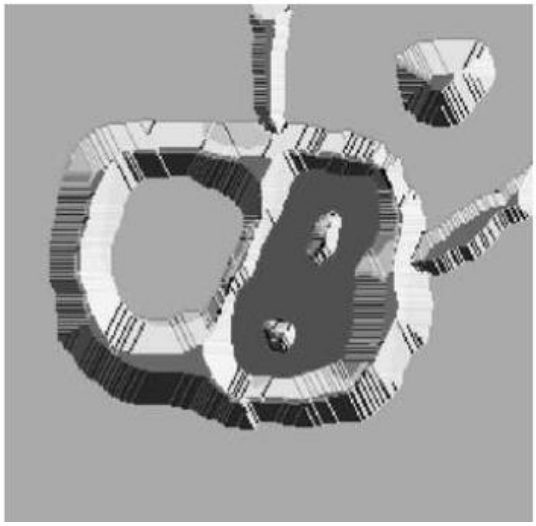


Fig 2.5 Water begins to rise inside the catchment basin

Slowly water begins to rise in each catchment basin as shown in Fig 2.5. Water is shown with a light gray color in the image.

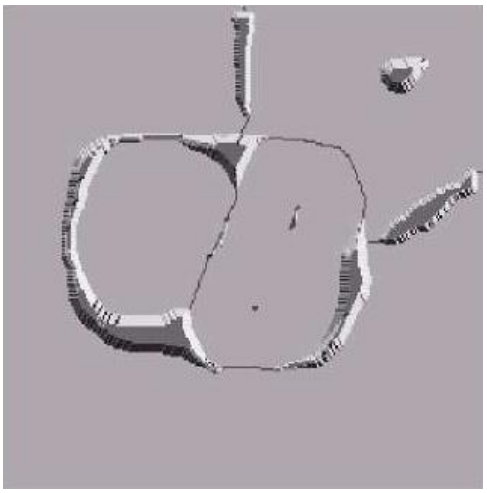


Fig 2.6 Water about to merge

And at the moment when water from different catchment basins is about to merge, the flooding of water is stopped. And a 1-pixel thick dam is built. This process is continued till max flooding is done. Ultimately, this dam gives the final watershed lines when superimposed over the original image[4].

2.5.2 Watershed segmentation algorithm

Let p(x, y) be a gradient image. In this image, consider 'R' different set of points belonging to regional minima which are denoted by S_1, S_2,…, S_R. Set of points in the catchment basin associated with a particular S_i be denoted by $C(S_i)$. Let L[n] represent the set of coordinates (j, k) for which p(j, k) < n. That is,

$$L[n] = \{(j, k) \mid p(j, k) < n\} \quad \ldots\ldots\ldots (2.1)$$

So, L[n] is a set of points in the image p(x, y) lying below the plane p(j, k) = n.

Flooding in the topographic region is done in the integer increment. That is from n=min+1 to n=max+1. Let at a particular flooding stage n and for a minimum S_i, $C_n(S_i)$ is a set of points in the catchment basin which are flooded. Therefore,

$$C_n(Si) = C(S_i) \cap L[n] \quad \ldots\ldots\ldots(2.2)$$

Let, at stage n of flooding, C[n] be the logical union of the catchment basins which are flooded:

$$C[n] = \bigcup_{i=1}^{R} Cn(Si) \quad \ldots\ldots\ldots(2.3)$$

It can be shown that the elements in these two sets $C_n(S_i)$ and L[n], with each flooding stage, either increase or remains constant.

From equations 2.2 and 2.3, it can be shown that C[n] is a subset of L[n] and C[n-1] is also a subset of L[n]. So, we can conclude that each connected component of C[n] belongs to a unique connected component in L[n]. The Watershed algorithm calculates C[n] from C[n-1] proceeding recursively. Let M denote the set of connected components in L[n]. Then for each connected component m ∈ M[n], there are three possibilities:[5]

1. m ∩ C[n-1] is a null set, which implies that a new minimum is found.
2. m ∩ C[n-1] has unique connected component of C[n-1].
3. m ∩ C[n-1] has greater than one connected component of C[n-1]. This condition occurs when a boundary separating two catchment basins is found. When this occurs, a dam must be built.

Due to the noise present in the image and distortions present in the gradient of the image, the algorithm stated above, in most cases leads to over-segmentation. The concept of 'marker' is often used to control this problem. The approach used here is to control a large number of minima where the algorithm is applied.

2.6 Implementation

The implementation consists of two main stages. First is preprocessing and second, being the actual algorithm stages. The implementation is explained with following Fig 2.7.

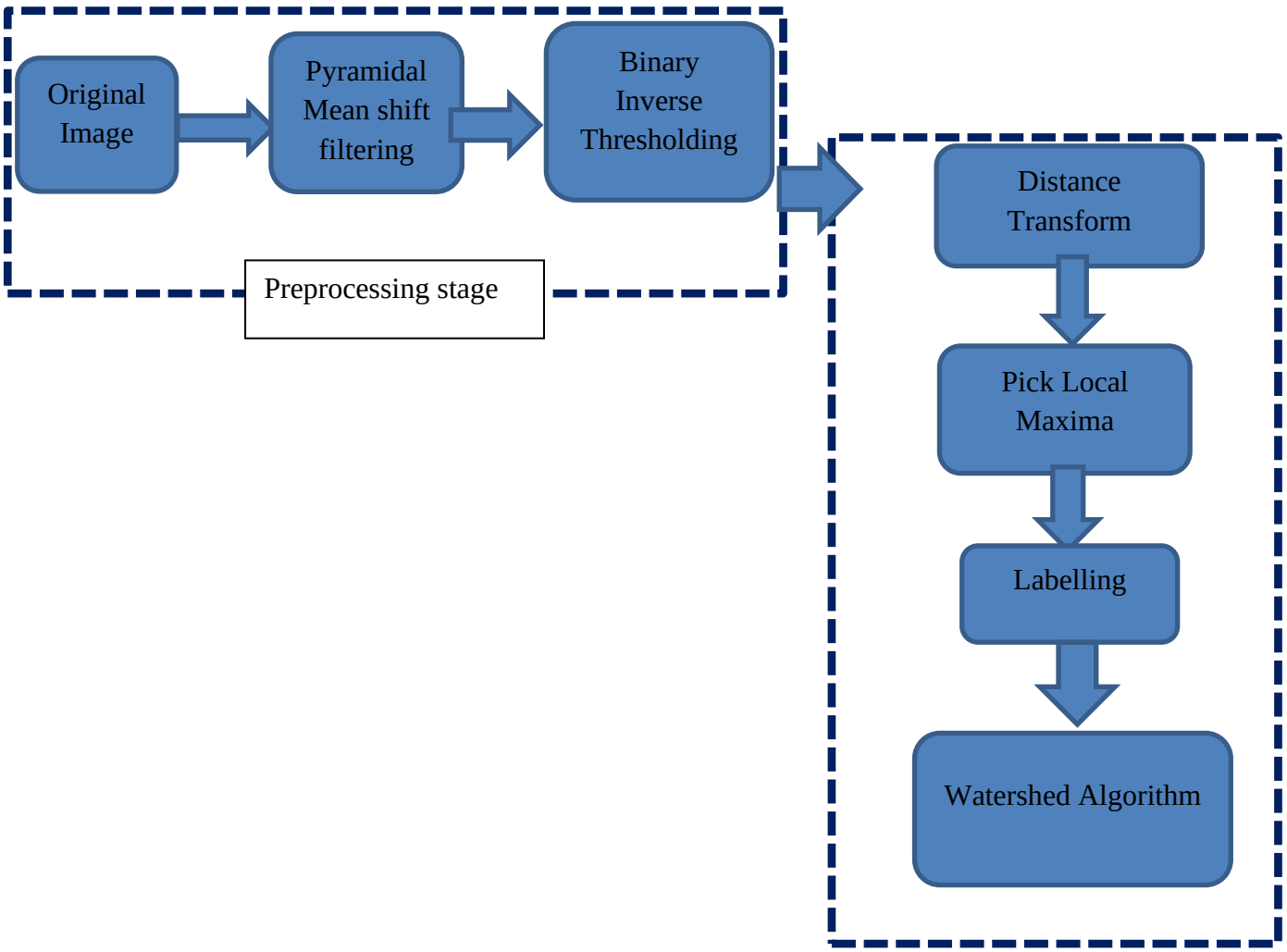


Fig 2.7 Implementation stages

In the implementation process, an image captured at CRMT(coke and Coal Raw Material Testing Lab) is used as the input to the algorithm, as shown in Fig 2.8.

Fig 2.8 A sample coke particles' image

The images for the experiment purpose were taken from a fixed height. The coke particles are spread over the white background. To clean the unnecessary dust before feeding it to the main algorithm is one of the important challenges.

So, we first process the image using pyramidal mean shift filtering. This function is a preprocessing stage for a completely different type of segmentation technique called a mean shift segmentation. This function helps in thresholding operation making the output of the threshold smoother. The mean shift filtered image of the above image is shown in Fig 2.9.

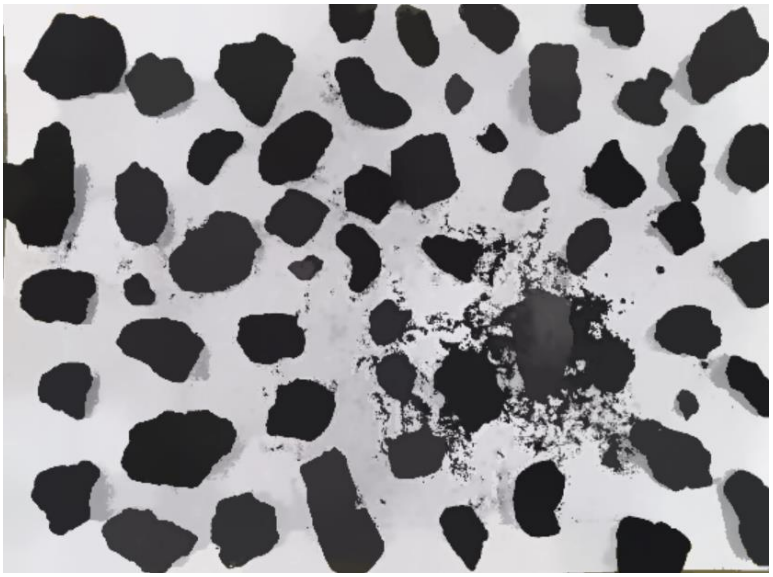


Fig 2.9 Mean shift filtering applied to input image

This image is then fed to thresholding operation where the ‘binary inverse thresholding’ technique is used as the background is white. The result is shown in Fig 2.10.

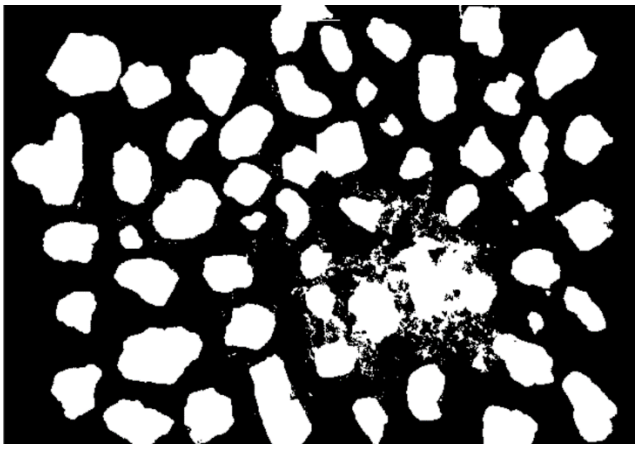


Fig 2.10 Binary inverse thresholding

The preprocessing operation gets completed at this stage.

Next, the Euclidean distance transform is applied to the image in the further stage. To each pixel in the image, this transform assigns a value that equals the Euclidean distance from that pixel to the nearest background/white pixel[6]. This gives us the gradient image needed for the algorithm.

Let's call the output of the distance transform as the distance map D. The next stage is to find the peaks i.e. the local maxima present in the distance map making sure that the maxima's are 20 pixels apart [7]. This is done using the peak_local_maxima function in the OpenCV library. These peaks form our markers provided for the watershed algorithm. The final function i.e. watershed function returns a matrix of labels which is of the same size as the input image. Each pixel in it has a particular label value. The pixels which are given the same label value belong to the same object. The final task is to collect these objects using unique labels. The result after applying the watershed and extracting object with a unique label is as shown in Fig 2.11.

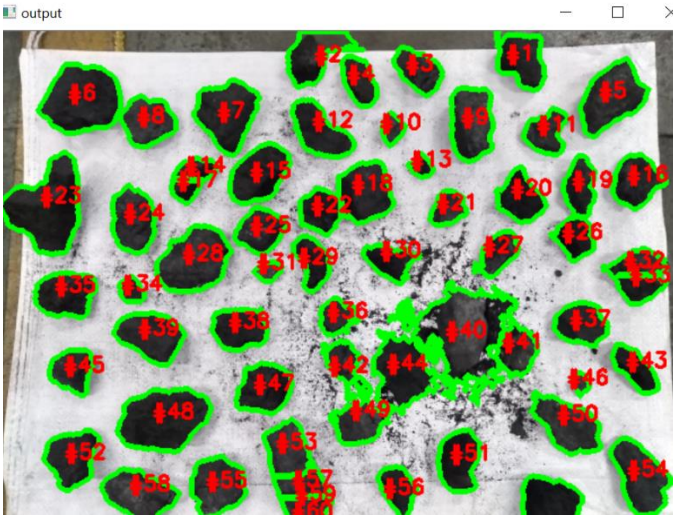


Fig 2.11 Segmentation result

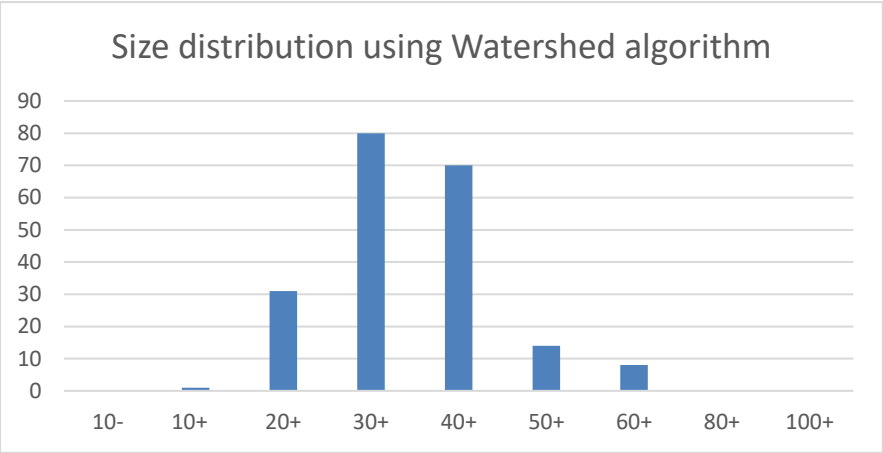
2.7 Result

First of all, a calibration image was captured from the same height as that of experimental images as shown in Fig 2.12. This image consists of a calibration box of known dimensions. We've to get these dimensions from the image in terms of the pixel so that we can find millimeters per pixel in the actual images.

Fig 2.12 Calibration image

Then from a coke sample of 120kg, all the coke particles were distributed as in Fig 2.8. The watershed segmentation algorithm was applied to them. And from the same coke sample, the ground truth distribution is also collected. The ground truth here is calculated using the sieve analysis method. And finally, the two distributions are compared using their respective mean and standard deviation as shown in the following figures, Fig 2.13 and Fig 2.14.

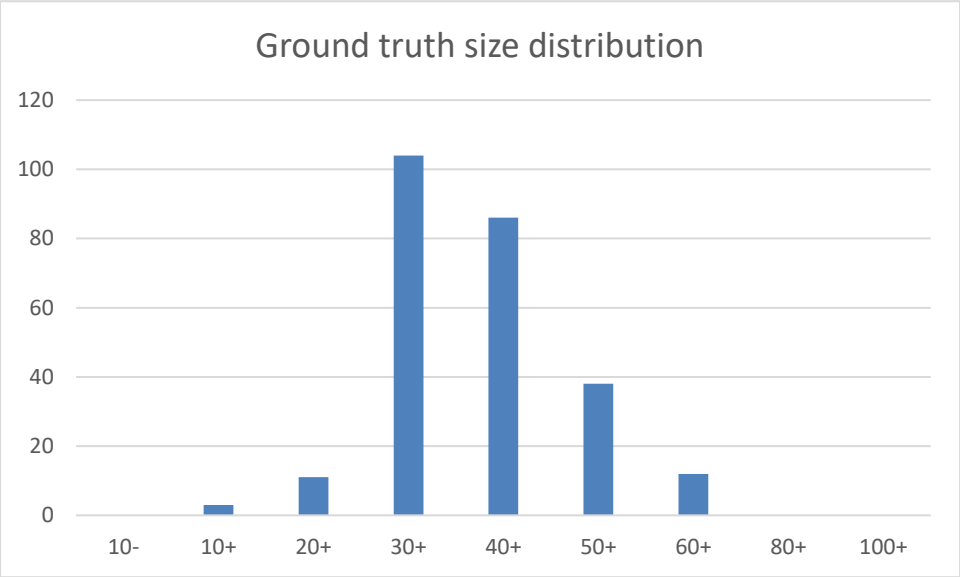
In the following graphs, on the horizontal axis is the size of the particles in millimeters(mm) and on the vertical axis, we have the count of particles corresponding to respective sizes.



Mean = 31.42

Standard deviation = 22.66

Fig 2.13



Mean = 39.93

Standard deviation = 28.22

Fig 2.14

Chapter 3

Bar-counting using machine learning

3.1 Objective and challenges

Steel bars when manufactured from 'Bar Mill' in a steel plant, get dispatched based on weight. But this process of dispatch is vulnerable to pilferage that takes place during transportation. To avoid this, dispatch of bars based on count is proposed to take place. The mechanical counting solutions are available but they are costly and need extensive maintenance.

So, the objective of this project is to accurately count the number of bars using imaging.

A sample image of bars is shown in Fig 3.1.

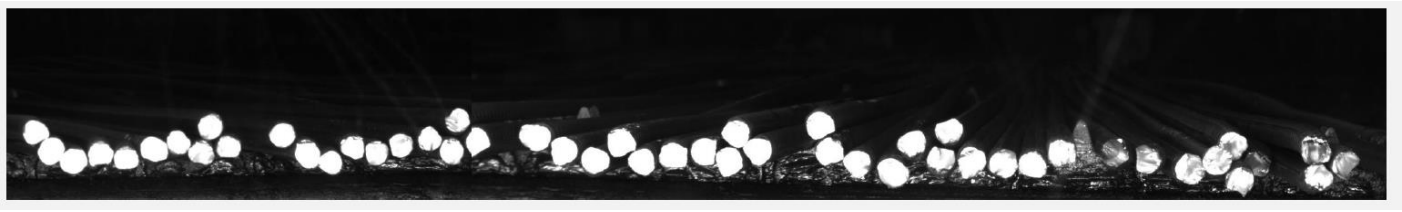


Fig 3.1 A sample bar image

Fig 3.2 Irregular bar tip

Fig 3.3 Dark bar tips

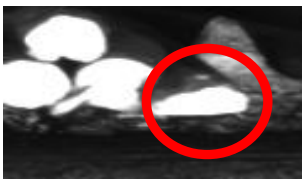


Fig 3.4 Misaligned bars

But, there are various challenges in counting the number of bars in such images as described below.

1. Bars are counted by detecting the tip of the bar in the image. But these tips are often irregular in shape as in Fig 3.2.
2. Sometimes the tip of the bar is very dark as in Fig 3.3.
3. Often the bars are misaligned as in Fig 3.4.

3.2 Proposed approach

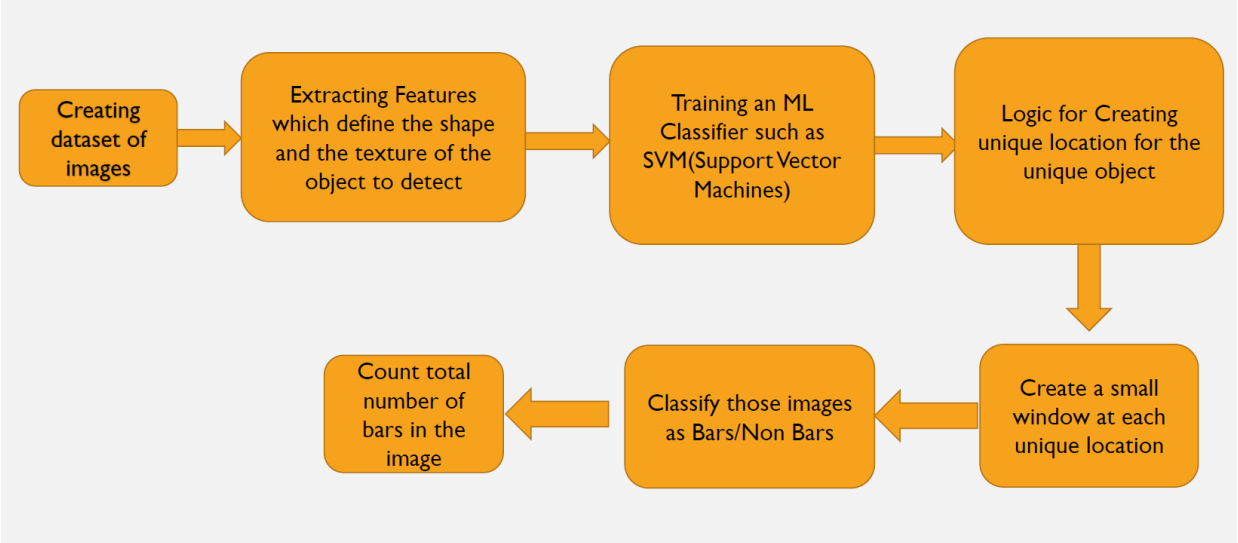


Fig 3.5 Proposed approach for the Bar-counting problem

The proposed approach for the bar counting using the image processing problem is as shown in Fig 3.5.

In this first of all, from the images, a dataset of images is prepared consisting of bar images and non-bar images. Then from the images present in the dataset, features are extracted which represent the shape and the texture of the object in the image. Using these features, a machine learning classifier such as Support Vector Machine(SVM) is trained.

The next task is to find a unique location for each of the objects in the image using some specific logic. Then a small window gets created around that location to classify it as the bar or non-bar using the trained ML classifier. This gives the final count of the bars.

3.3 Dataset creation

The images used for this experiment consist of mainly two types of objects. Either the objects are bars or more specifically, the tip of the bar or they are not the bars. So we need to create a dataset consisting of two classes only. We call it class 1 and class 0, for bars and non-bars respectively.

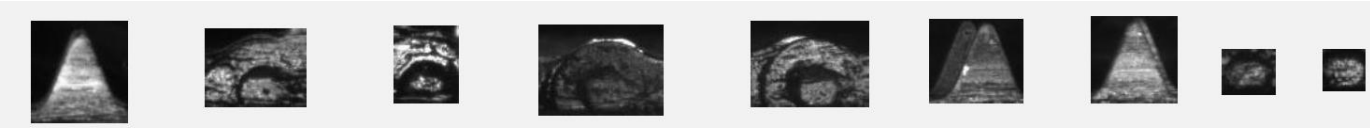


Fig 3.6 Class 0 images for non-bar objects

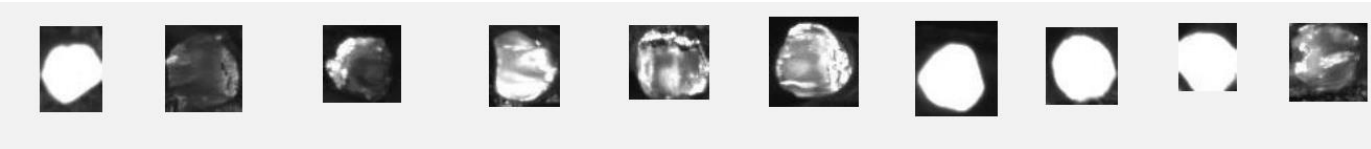


Fig 3.7 Class 1 images for bar objects

Above Fig 3.6 is a collective representation of the small cropped images belonging to class 0 which is for non-bar objects. Around 200 of such images were collected in the dataset. Similarly, Fig 3.7 is a collective representation of cropped images of small size (approximately 50 * 50 pixels) which belong to class 1 which is for bar objects in the dataset. Around 300 of such images were collected.

3.4 Feature extraction

A general idea behind image classification using the machine learning technique is to extract the features from the image i.e. to create an n-dimensional feature vector out of it so that in the n-

dimensional hyperspace, we can separate using some hyperplane the images and their corresponding feature vectors.

In the case of this problem of bar counting, it is necessary to find features that will define the shape as well as the texture of the object in the image.

So, the combined use of two such features is done here which are Histogram of Oriented Gradients (HOG) and Local Binary Patterns(LBP).

3.4.1 Histogram of Oriented Gradients(HOG)

The very effective method of HOG was first used by N. Dalal and B. Triggs in their work regarding human detection or more specifically pedestrian detection in an image. The idea behind this method is to calculate the normalized local histogram of gradient orientation in the image. This is supported by the idea that even in the absence of precise knowledge of gradient or edge position, the object shape and it's appearance can be characterized by the direction of edge and the corresponding gradient[8].

A simplified step by step process of calculating HOG features as described below:

1. For the localized processing of the image, the image needs to be subdivided into smaller regions. So, the first step is to resize the input image into an aspect ratio of 1:2 say 64*128 so that subdivision of image into 8*8 pixel set becomes easy.
2. For each pixel in the image, G_x - gradient in the x-direction, and G_y - gradient in the y-direction is calculated.
3. Using this, gradient magnitude and gradient direction are calculated for each pixel.

   Total gradient magnitude = $\sqrt{Gx^2 + Gy^2}$ &

   Gradient direction = $\tan^{-1}\frac{Gy}{Gx}$
4. Then, there are multiple ways using which histogram can be created using gradient magnitude and direction. One of them is that the range of angle i.e. 0 rad to π rad is divided

into say 20 bins. For each pixel value, it's corresponding gradient magnitude is filled in the bin where the gradient orientation of the same pixel lies.

5. This process gives a 9-dimensional vector for a 16*16 portion of the image. Finally, to get the feature vector for the entire image, all such small feature vectors are combined.

3.4.2 Local Binary Patterns(LBP)

Analysis of two-dimensional texture features of an image has potential applications in the field of pattern recognition. But a major challenge in the real world problem is that the textures are often non-uniform due to variation in scale, orientation, and other visual appearance[9].

So, local binary patterns(LBP) are texture features that are gray scale based and are rotation invariant. Also, the computational complexity is very low compared to other texture feature extraction methods. A core idea in finding the LBP is to compare the central pixel with the surrounding pixels at each pixel in the image.

A simplified step by step process for calculating LBP features is described below:

1. The image is converted from RGB to a gray-scale.
2. To choose two parameters P & R for variable neighborhood sizes, where P = number of points to consider in a circularly symmetric neighborhood and R = the radius of the circle to get the variable scale[9].
3. For example, if we set, P=8 and R=1, at any particular pixel location we can compare the current pixel value with that of surrounding pixels as shown in Fig 3.8.
   If the value of a pixel is greater than the central pixel, it'll be assigned a 0 and 1 if it's lesser.

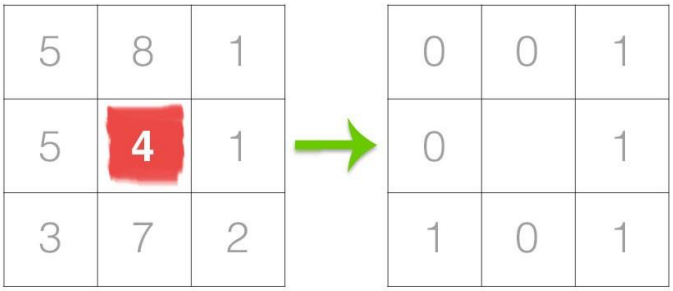


Fig 3.8 Creating a binary string for central pixel [9]

4. From this 3*3 neighborhood, we get a binary string that can be converted to a decimal value to get the code for that pixel location. For example, if we can get the binary string 00010111 from the above image, which when converted to decimal gives 23.
5. For each pixel, the same process is repeated until we get an entirely new image of the same dimension as that of the original image as shown in Fig 3.9.

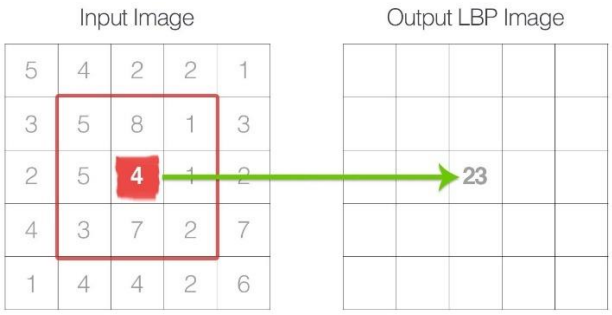


Fig 3.9 Creating LBP image of the same dimensions [8]

6. The last step is to get the histogram of this new image of texture codes acquired. This histogram serves as the LBP texture feature descriptor.

3.5 Logic for Bar-counting

The images in the dataset belong to either of the two classes of bar or non-bar image. For each of that image, the HOG and LBP features are calculated and stacked together. To create a uniform feature for each of the images, first, the image gets resized to 50*50 pixels. Then for that image, first the HOG feature vector is calculated using a cell size of 16*16 pixel. After that, for the same image, the LBP feature vector is calculated using 24 circularly symmetric neighborhood points and assuming a radius of 8 pixels. This LBP feature is stacked to the HOG feature to make a single vector out of both. In this way, the feature matrix gets created out of the dataset.

This is trained using machine learning algorithm of Support Vector Machine(SVM). SVM are supervised learning models associated with learning algorithms that analyze data used for classification and regression analysis[1]. This algorithm works on creating linear decision boundaries to classify multiple classes. The SVM is trained using the polynomial kernel.

Now, the real challenge was to calculate the number of bars in the test image. Generally, a sliding window is used along with the image pyramids. In this method, a sliding window of varying size moves along the entire image and each of that window keeps on classifying the current window. But this method could not be useful in the case of this problem as the bars were very closely spaced as well as they are highly misaligned. Hence a new logic needed to be developed which can accurately count the number of bars in the test image. So, logic is developed keeping in mind that no object other than the bar and non-bar will be present in the test image.

A flow diagram of the logic used can be given as below in Fig 3.10.

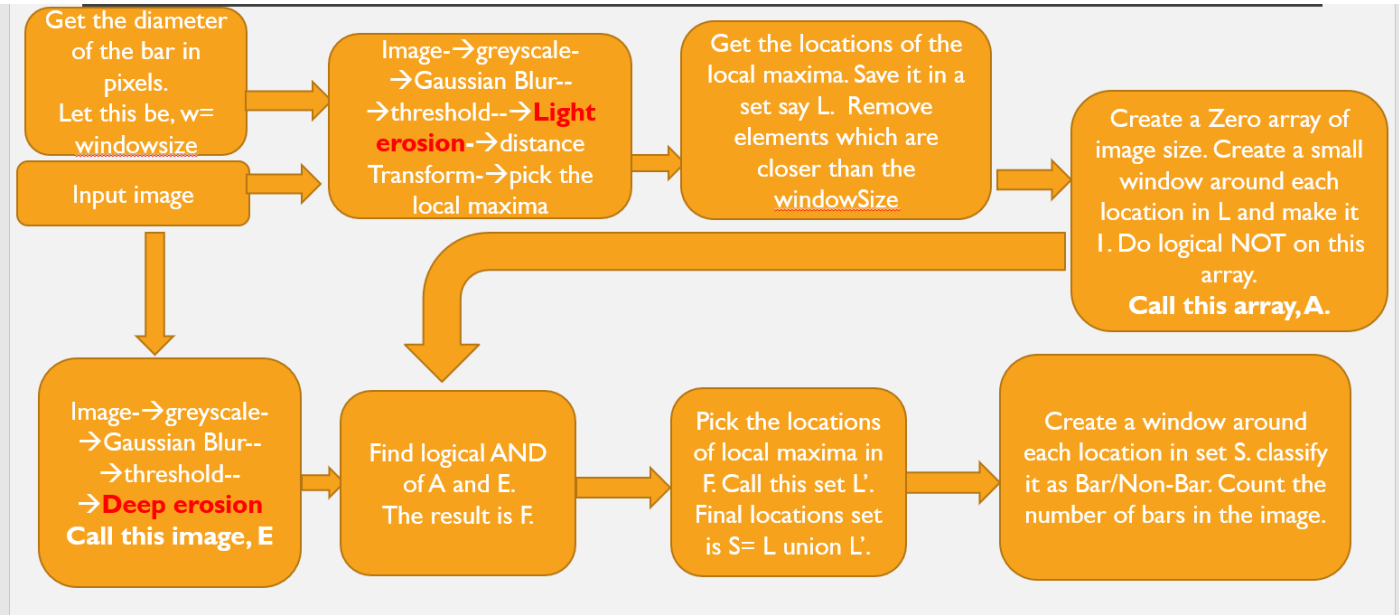


Fig 3.10 Logic for bar counting in the test image

The central idea used in this logic is to first find each of the objects present in the image and then create a unique location for the unique object.

As the first block says to get the diameter of the bar in pixels, which is called as 'W'. To get this w from the test image, another process is used which operates in the same program just as a function. In this function, first, the image is thresholded keeping the high threshold value. Then the connected components in the image are acquired. From all of these connected components, the median diameter of the first 30 components is taken as 'W'. This W is later used as the window-size while creating the window around locations.

The next task is to get a set of locations for the objects. For that, on the input image, some basic operations are performed such as converting to grayscale, Gaussian blurring, thresholding, morphological erosion. Note that the erosion done here is a light erosion i.e. with 4 iterations only. Then on this image, distance transform is applied to get the local maxima. Locations of all the local maxima are stored in a set L. Locations that are closer than the window-size W are removed from the set L. Now the problem is that the locations in L are not all the locations that we want.

So, An array of zero elements of the same size as that of the image is created. In this array, we create a small window of 1's around the locations in L. Logical NOT on this is performed and now call this array 'A'.

Again we perform some operations using this A so that we get all the unique locations for the unique object in the image.

So again the original image is taken. It's converted to grayscale, thresholded, and now performed deep erosion on this i.e. with around 20 iterations. We call this image, 'E'.

Do logical AND of A and E. The result is F. Then we get the final set of locations.

In this image F, get the locations of local maxima using the same process used. Keep this in set L-dash.

The final set of locations is the union of sets L and L-dash.

In this final set, we got a unique location for the unique object present in the image.

Now the final task is to create a window of size W*W pixels around each of these locations and to keep on classifying that window using the trained model of SVM whether it belongs to the bar object or the non-bar object.

3.6 Result

To get the result in this problem, the above-stated logic is implemented and the classification is done using the trained SVM classifier. The following Fig 3.11 is used here for the representational purpose. In practice, the actual images used to get the result are much broader in the width that cannot be shown here fully.

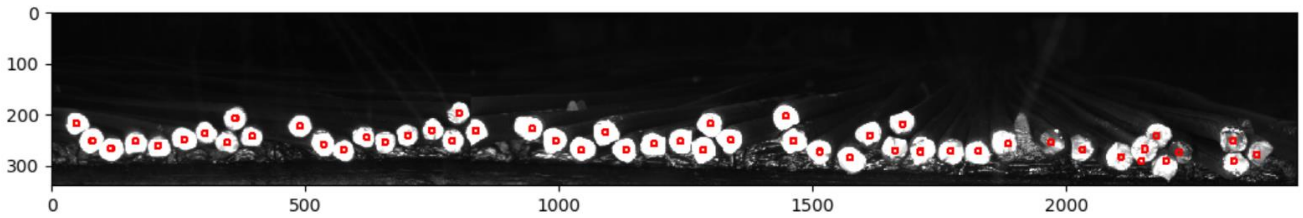


Fig 3.11 Result of object detection

The small red squares show the detected bars in that image.

To get the quantified result, the above bar counting algorithm is applied to the bar image such as Fig 3.12.

Fig 3.12 A sample input bar image

And the result of the algorithm is subjectively shown in Fig 3.13.

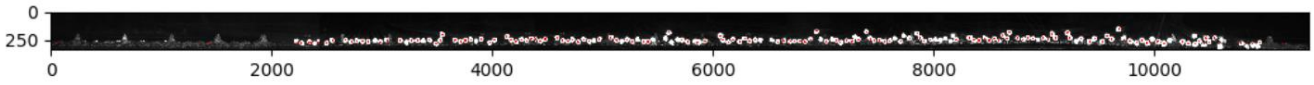


Fig 3.13 Result on the input image

But the results produced by the algorithm are also composed into a confusion matrix as it is generally done for a machine learning classification problem which is shown below in the Fig 3.14.

Total samples = 335	Predicted: Bars	Predicted: Non-Bars
Actual : Bars	168	5
Actual: Non-Bars	4	157

Fig 3.14 Confusion matrix

The total samples shown imply the total number of locations taken into consideration.

From the above confusion matrix, we can calculate the accuracy as well as the error rate.

$$\text{Accuracy} = \frac{(True\ Positives + True\ Negatives)}{Total\ samples}$$

$$= {}^{(168+157)}/_{335}$$

$$= 97.01\ \%$$

$$\text{Error Rate} = \frac{(False\ Positives + False\ Negatives)}{Total\ samples}$$

$$= \frac{(5+4)}{335}$$

$$= 2.98\ \%$$

Chapter 4

Stone fragments size estimation using deep learning

4.1 Objective

To determine the efficiency of fragmentation during blasting at the coal mining site. Quantitatively this is expressed as the size distribution of fragments through image analysis.

4.2 Background

In the process of coal extraction from coal mines, explosives are first used in the mining area to break through the stone layer. After the blasting is done, images of the stone fragments are captured as shown in Fig 4.1.

Fig 4.1 Stone fragments

From such captured images, the size distribution of the stone fragments is to be calculated. In each of these images, a white ball of known dimension is kept for the calibration purpose.

From the size distribution of these fragments, it is determined whether the blasting is done optimally or not.

Currently, the size distribution is calculated manually using sketching tools from these images. The objective of the project is to fully automate the process of finding the size distribution of the stone fragments.

4.3 General approach

Before applying some special or advanced method for the problem of stone image segmentation, we tried to solve the problem with the known methods such as canny edge detector and watershed segmentation. As it is seen in Fig 4.1, there is no very clear boundary or edge between the two stone objects to differentiate. Therefore these conventional methods fail here.

First, the result with the Canny edge algorithm is shown in Fig 4.2. As this very popular algorithm of image segmentation also depends upon gradient magnitude and gradient direction, it doesn't produce any good results in this problem.

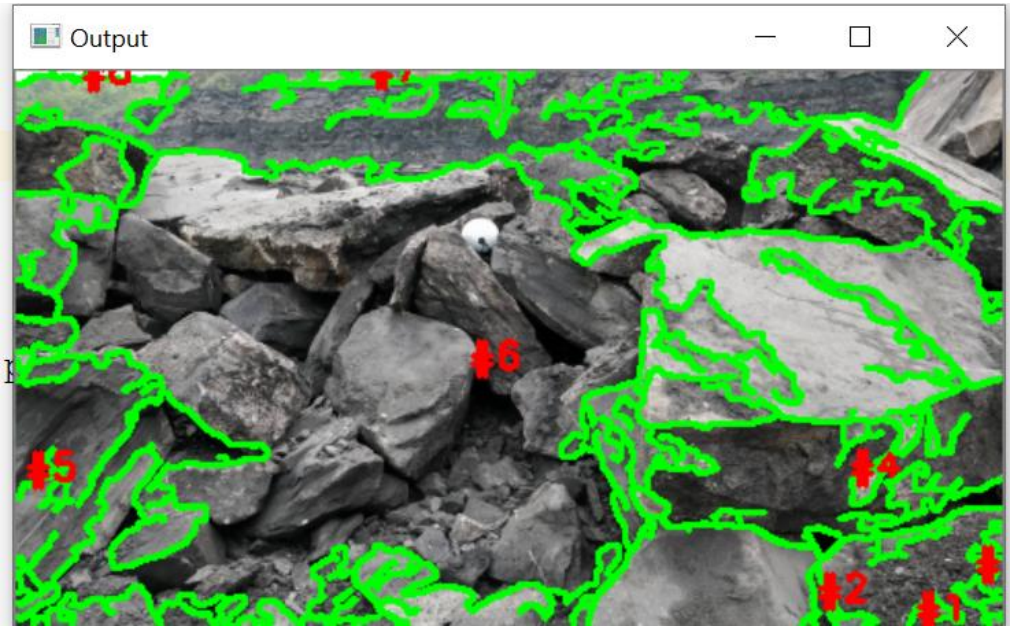


Fig 4.2 Result with Canny edge detector

Canny edge detection is applied after preprocessing with mean shift filtering. The two thresholds set for the canny detection are 100 and 200. We can see that this segmentation doesn't work here at all.

Next, we tried applying a Watershed algorithm on the same problem. The result of which is shown in Fig 4.3.

The entire process of applying the Watershed algorithm is the same as explained in chapter 3. Though some segmentation has been done and is very much unwanted and incorrect.

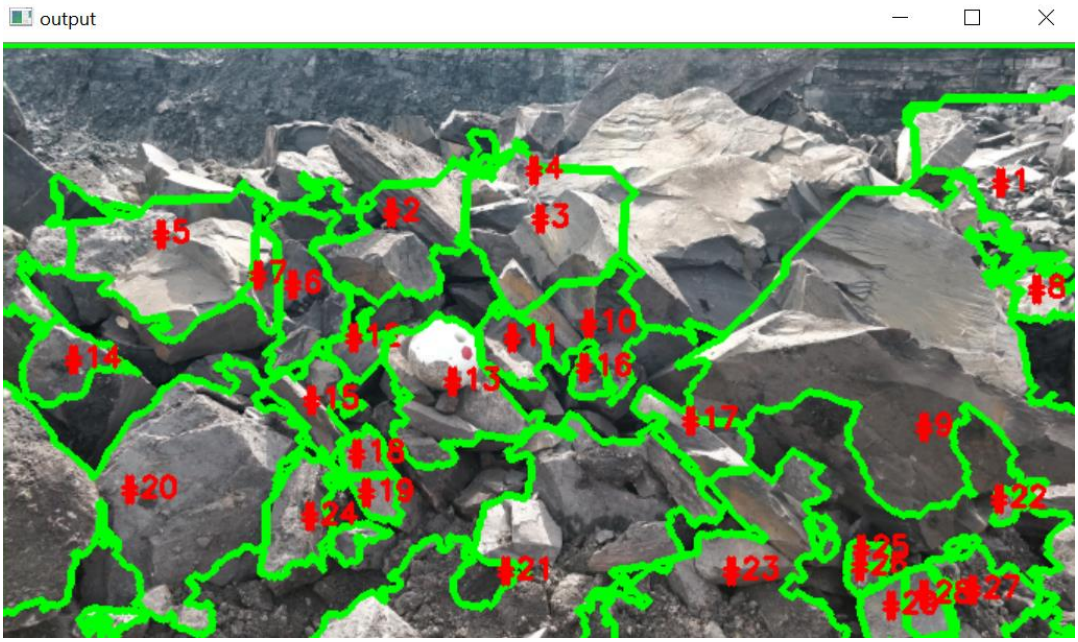


Fig 4.3 Result with the Watershed algorithm

Keeping in mind the two results with standard conventional techniques of image segmentation, we tried to perform the segmentation in a non-standard way with our logic as explained below.

Fig 4.4 A sample image to apply the segmentation

First of all, mean shift filtering is applied to the input image so that, the edge detection will be smoother as the unnecessary fluctuations in the pixel values are removed through this as shown in Fig 4.5.

Fig 4.5 Mean shift filtering applied to an input image

Then, for the edge detection purpose, a non-standard kernel matrix is selected as shown in Fig 4.6.

```
([0, 0, 0, -2, 0, 0, 0],
 [0, 0, 0, -2, 0, 0, 0],
 [0, 0, 0, -2, 0, 0, 0],
[-2, -2, -2, 24, -2, -2, -2],
 [0, 0, 0, -2, 0, 0, 0],
 [0, 0, 0, -2, 0, 0, 0],
 [0, 0, 0, -2, 0, 0, 0])
```

Fig 4.6 Kernel for edge detection

The result of the convolution of the image with the above kernel is shown below in Fig 4.7.

Then to fuse the small gaps between the edges, morphological closing is done. Morphological closing is the combination of morphological dilation followed by erosion. Then all the 4-connected components are found and the components having significant smaller areas are removed. And finally, dilation is performed to strengthen the important edges. Then the result looks as shown in Fig 4.8.

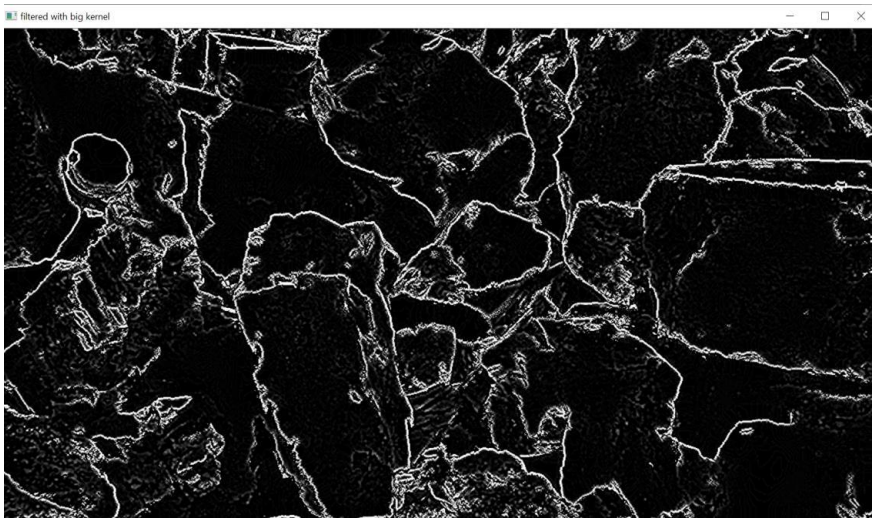


Fig 4.7 Result of the convolution

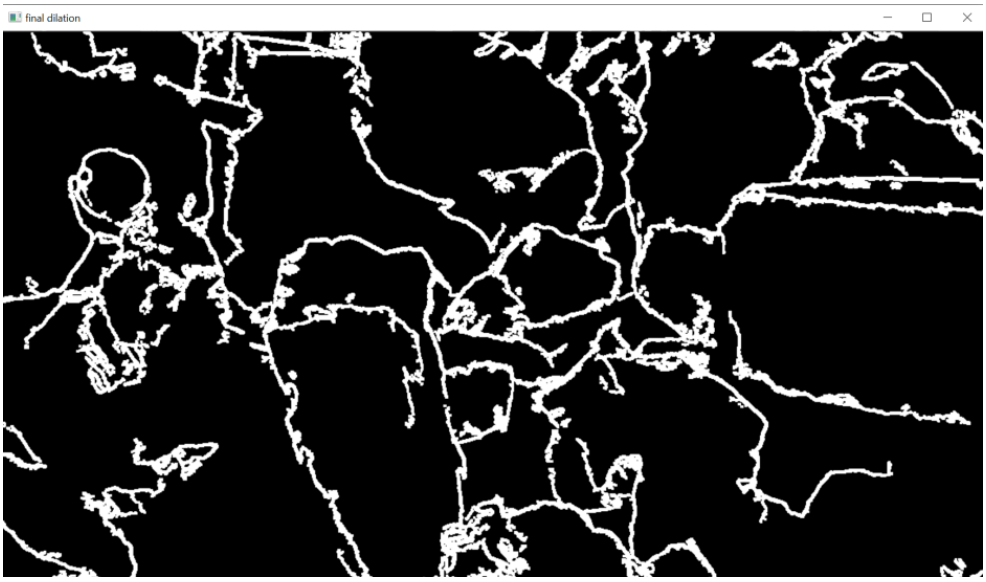


Fig 4.8 Final result

We can see from the above result that, the contours obtained after performing the entire process are not going to be very accurate. So, it is concluded that no conventional technique is capable enough to detect the stone objects present in these images to give the accurate size distribution. Hence, finally, the approach of deep learning is used to solve the problem.

4.4 Mask R-CNN approach:

Although there are various techniques available for object detection using deep learning, the best and the most advanced among them is Mask R-CNN for instance segmentation. The fundamental difference between semantic and instance segmentation is that in the semantic segmentation, every pixel in the image is labeled with a class of its enclosing object while in the instance segmentation, each object of each class is separated at the pixel level. Hence we needed a method for instance segmentation for which Mask RCNN suits the best. Mask R-CNN stands for Mask Region-Based Convolutional Neural Networks, which is developed at FAIR(Facebook AI Research) by Kaiming He and his team in 2018. This method detects objects in the image with high efficacy and for each of the objects, instance generates a high-quality segmentation mask[10]. So, we chose to use this technique of instance segmentation for our problem. To understand this, we need to first get to know some former techniques such as R-CNN, Fast R-CNN, and Faster R-CNN.

R-CNN works in two steps. Region proposal step and classification step. A selective search is used to choose around 2000 regions in the image which is called Region Proposal. In the classification step, feature vector extraction and a set of linear SVM's are included. Each of the proposed regions is fed to CNN that produces a 4096-dimensional feature vector as an output. Then the class label is produced by feeding this feature vector to SVM's. the drawback of the R-CNN is a large amount of training time.

Fast R-CNN do not feed all the 2000 regions to the CNN every time. It shares the computation of convolutional layers between different region proposals[11]. Here region proposals are acquired from convolutional feature maps and are processed through ROI pooling. Then this feature map is converted to feature vector using a Fully Connected Layer(FC). Finally, the class of the proposed region is determined by a softmax layer.

In faster R-CNN, the only difference is that, instead of using selective search for the region proposal, a completely new component called RPN-Region Proposal Network is used which is placed just after the last convolutional layer. Then these proposals are sent to ROI pooling.

Now let's talk about the Mask R-CNN technique which works in two stages. First is the region proposal and then the classification of those regions producing masks as well as bounding boxes for each of the classification. The final masks are generated by a different convolutional network whose output is a matrix having 1's at the object location and 0's elsewhere.

Mask R-CNN consists of a standard like ResNet50 as a backbone network. The primary stages of the network detect low-level features and the later stages of the network keep the high-level features. As an extension of the backbone network, the Feature Pyramid Network(FPN) is used so that objects at different scales can be represented in a better way. The Region Proposal Network(RPN) proposes all the possible regions by scanning the entire FPN. A set of boxes known as anchors which are individually used to get assigned to a ground-truth class and bounding boxes. The ROI pooling module is different in Mask R-CNN. As ROI pooling generates a slight inaccuracy by producing a little misaligned feature map. The correction is done by using ROI-Align in which the feature map is sampled at different points and then bilinear interpolation is used. The ROI classifier gives the selected regions to CNN which then produces masks for them. To compute the loss during training the masks are of low resolution. Finally, the masks are scaled up to the size of the ROI bounding box, and the mask for each object is created.

4.5 Training the model

The popular method of transfer learning is used to build a model in a time-efficient way. Transfer learning is useful in computer vision as it permits to develop an accurate model in a shorter time. The idea used in transfer learning is that instead of training the deep learning model from scratch, it just uses a pre-trained model which is trained for solving different but similar problem. In this way, a lot of time is saved as we don't start from scratch also the learning of the model is leveraged. As we know that the lower layers of a DL model correspond to the lower level features which are generally problem independent but the higher-level features are kept in latter layers which are problem specific[12]. Therefore, while training the model using the transfer learning, we freeze the lower-level layers of the model and we adjust the weights of the latter layers only.

Apart from requiring much lesser time for the training purpose, other advantages of using transfer learning are that we require a much smaller dataset while training and therefore the hardware required while training need not be of very high-end configuration.

Mask R-CNN uses the ResNet101 as the backbone architecture which is originally trained on the MS coco dataset. COCO stands for Common Objects in Context. There around 91 object categories in the coco dataset.

So we created a dataset consisting of 100 stone fragment images in which 80 were for training and 20 were kept for the testing purpose. We add our object named ‘Rock’ to the original dataset. It is necessary to annotate all the images in the dataset. The annotation is done using the VIA tool which is an abbreviation for the VGG Image Annotator tool.

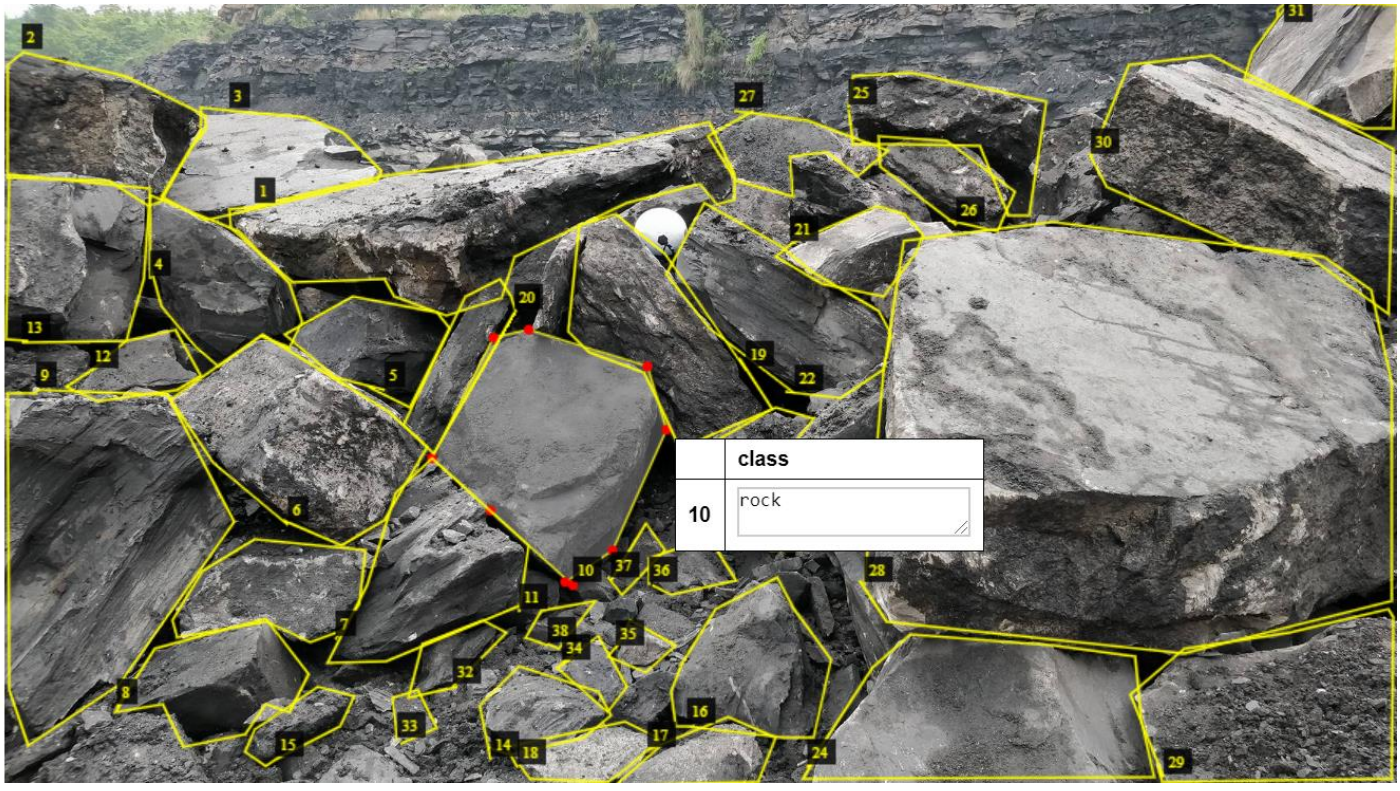


Fig 4.9 Image annotation using VIA

Each of the stone objects in the images is tried to be enclosed using polylines as shown in Fig 4.9. When the annotation on the entire dataset is done, a .json file is saved for the annotations.

This file consists of locations of the vertices of the enclosed polygons for each of the objects in each image. Using these vertex locations only the exact polygonal masks are recreated in the code.

Then while training, we load the originally trained coco weights excluding the four latter layers namely ‘mrcnn_class_logits’, ‘mrcnn_bbox_fc’, ‘mrcnn_bbox’, ‘mrcnn_mask’. And the training was done using Keras model in python3. The number of epochs used was 20. At each epoch in the training, the entire data is trained once. Multiple epochs are used to improve the accuracy of the weights. The number of steps used for each epoch was 100. It means that at each epoch, the entire data is divided into 100 batches/steps.

We collected few loss parameters at the initiation of the training and at the end of the training as shown in the following Table 4.1 and Table 4.2.

Type of loss	Value of loss
Rpn_class_loss	2.1329
Rpn_bbox_loss	0.2911
Mrcnn_class_loss	0.9479
Mrcnn_bbox_loss	1.5875
Mrcnn_mask_loss	1.9406
Total training loss	6.9000

Table 4.1 Training loss values at the beginning of training

Type of loss	Value of loss
Rpn_class_loss	0.0226
Rpn_bbox_loss	0.1040
Mrcnn_class_loss	0.1785
Mrcnn_bbox_loss	0.0939
Mrcnn_mask_loss	0.2502
Total training loss	0.6492

Table 4.2 Training loss values at the end of training

Along with the training loss calculated at each step at each epoch, the model also produces the validation loss that is calculated at the end of each epoch in the training process.

Validation loss is calculated using the same formula as that of the training loss but it is not used to update the weights in the back-propagation process. In fact, by comparing the values of training and validation losses, we can understand how perfectly our model is fitted. As, if the validation loss is greater than the training loss, we can call it overfitting. And if both are equal, we can say that the model is perfectly fitted.

The different validation losses collected at the end of last epoch are shown in following Table 4.3.

Type of loss	Value of loss
Val_rpn_class_loss	0.0325
Val_rpn_bbox_loss	0.2747
Val_mrcnn_class_loss	0.3994
Val_mrcnn_bbox_loss	0.2970
Val_mrcnn_mask_loss	0.2988
Total validation loss	1.3024

Table 4.3 Validation loss values at the end of last epoch

4.6 Result

New weights are saved as a .h5 file when the training is completed. Now the evaluation of the unseen images is done using these new weights. The image to evaluate the model performance is shown in Fig 4.10.

Fig 4.10 Test image

And the result of object detection using MR-CNN is shown in Fig 4.11 below. After extracting the mask the code produces a unique color for each of the object masks. From the calibration ball present in the input image, we input, the size in cm as well as the size in pixels of the ball, so that we get the length per pixel in the image which is used for converting the mask dimensions from pixels to cm.

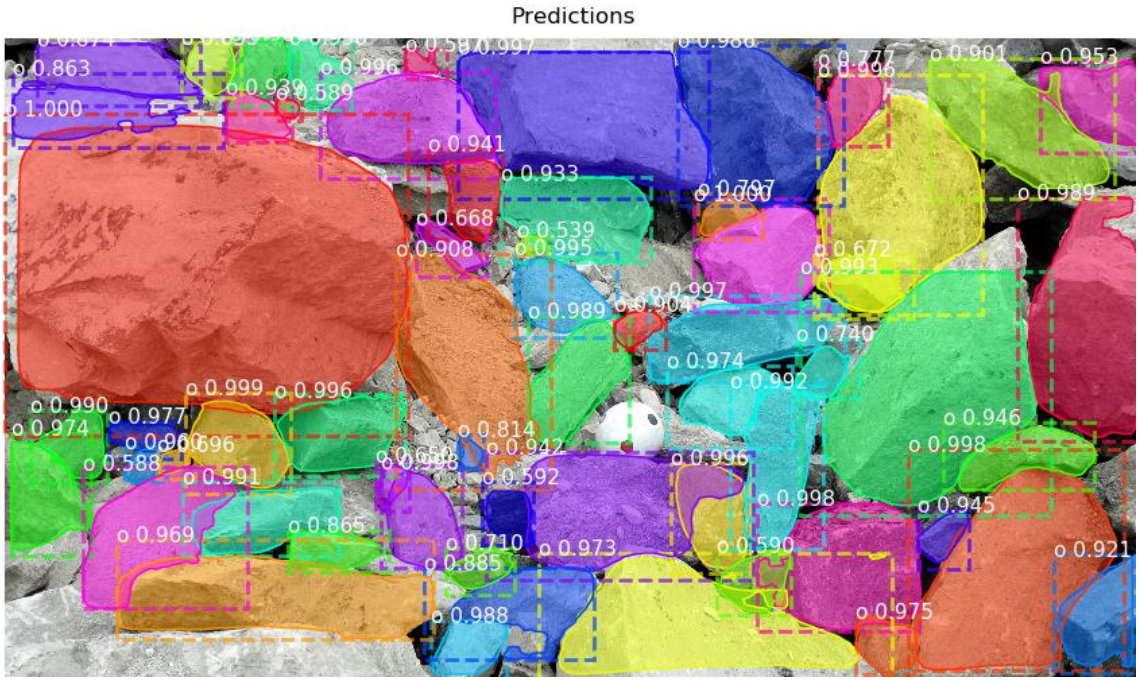


Fig 4.11 Object predictions on the test image

As it's seen in the prediction image, for each of the objects a unique mask is produced which segments that object from the background. Also, a bounding box along with the probability for the presence of the object is produced for each of the objects.

So, we can extract each of the individual masks from the predictions, and using the calibration, we can find the area in cm^2, length of a major and minor axis in cm. Following Fig 4.12 shows the size distribution of the stone fragments by stone count for respective sizes in cm^2.

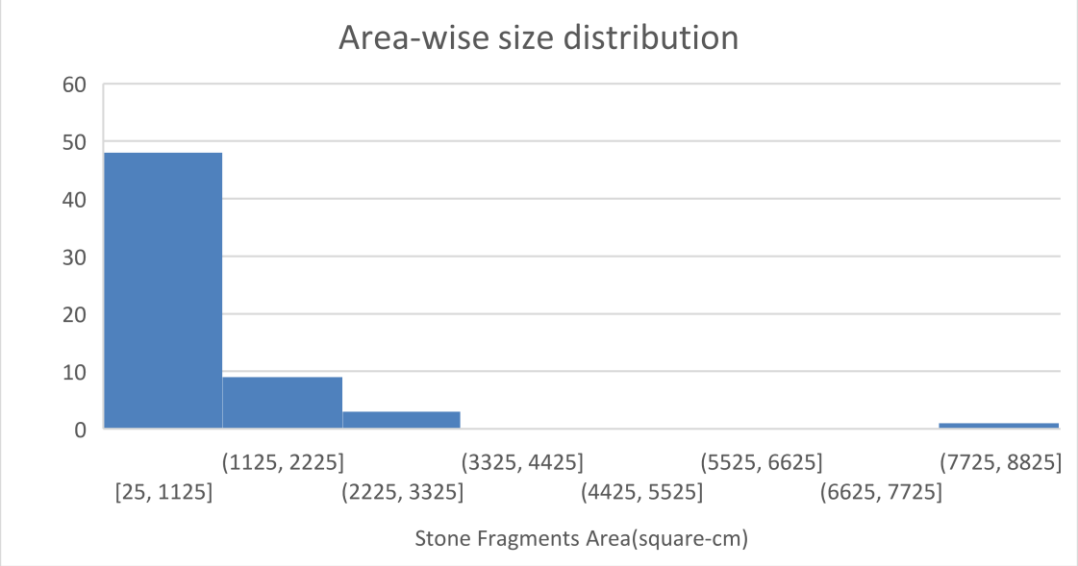


Fig 4.12

The configuration details of the Mask R-CNN model for the testing of this particular image is given in the following Table 4.4.

BACKBONE	ResNet101
BACKBONE_STRIDES	[4, 8, 16, 32, 64]
BATCH_SIZE	1
BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE	None
DETECTION_MAX_INSTANCES	100
DETECTION_MIN_CONFIDENCE	0.5
DETECTION_NMS_THRESHOLD	0.3
FPN_CLASSIF_FC_LAYERS_SIZE	1024
GPU_COUNT	1
GRADIENT_CLIP_NORM	5.0
IMAGES_PER_GPU	1

IMAGE_CHANNEL_COUNT	3
IMAGE_MAX_DIM	1024
IMAGE_META_SIZE	14
IMAGE_MIN_DIM	800
IMAGE_RESIZE_MODE	square
IMAGE_SHAPE	[1024 1024 3]
LEARNING_MOMENTUM	0.9
LEARNING_RATE	0.001
LOSS_WEIGHTS	{'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE	14
MINI_MASK_SHAPE	(56, 56)
NAME	rock
POOL_SIZE	7
POST_NMS_ROIS_INFERENCE	1000
POST_NMS_ROIS_TRAINING	2000
ROI_POSITIVE_RATIO	0.33
RPN_ANCHOR_SCALES	(32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE	1
RPN_NMS_THRESHOLD	0.7
RPN_TRAIN_ANCHORS_PER_IMAGE	256
STEPS_PER_EPOCH	100
TOP_DOWN_PYRAMID_SIZE	256
TRAIN_ROIS_PER_IMAGE	200
USE_MINI_MASK	True
USE_RPN_ROIS	True
VALIDATION_STEPS	50
WEIGHT_DECAY	0.0001

Table 4.4 Configuration details of MR-CNN model while testing

Chapter 5

Conclusion & future work

5.1 Conclusion

1. We conclude that all the work done helped in automating the related processes at the Steel plant.
2. The size distribution calculated using the watershed algorithm is very close to the ground truth distribution. So, estimation of coke particle size distribution using image processing algorithm will be implemented by removing the conventional method of Sieve analysis for as it's proved to be a much accurate method requiring lesser time and less labor work.
3. The Bar Counting algorithm can also replace the conventional mechanical approach for the same as the maintenance required can be reduced to the minimum and accuracy could be increased using the advanced algorithm.
4. The mask R-CNN approach for the size distribution estimation of stone fragments proved to be very efficient for detecting each of the stone fragments in the image. This saved a lot of labor for size calculation and improved the accuracy.

5.2 Future work

In the first project of coke particle size distribution, a better approach for finding the markers can be used. So, a Watershed algorithm can be applied at the right minima only which will improve the accuracy by reducing the over-segmentation which is sometimes taking place using the current method.

In the Bar-Counting problem, instead of using the machine learning technique in which we have to extract the features by ourselves, a deep learning approach can be used for much accurate object detection and counting.

REFERENCES

[1] Dilpreet Kaur and Yadvinder Kaur, “Various Image Segmentation Techniques: A Review”, International Journal on Computer Science and Mobile Computing, IJCSMC, Vol. 3, Issue. 5, May 2014, pg.809 – 814

[2] Pulkit Sharma , “Computer Vision Tutorial in Implementing Mask R-CNN”, 22nd July 2019,

https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/#:~:text=We%20will%20instead%20use%20the,model%20to%20perform%20instance%20segmentation.

[3] Digital image processing, R. Gonzalez and R. Woods, *Prentice-Hall, Upper Saddle River, N.J.,* (*2008*), pp. 769-770

[4] Digital image processing, R. Gonzalez and R. Woods, *Prentice-Hall, Upper Saddle River, N.J.,* (*2008*), pp. 770-771

[5] Digital image processing, R. Gonzalez and R. Woods, *Prentice-Hall, Upper Saddle River, N.J.,* (*2008*), pp. 775-776

[6] OpenCV, Open Source Computer Vision, “Image segmentation with a watershed algorithm”,

https://docs.opencv.org/master/d3/db4/tutorial_py_watershed.html

[7] Adrian Rosebrock, “Watershed OpenCV”, 2nd Nov 2015,

https://www.pyimagesearch.com/2015/11/02/watershed-opencv/

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 886-893 vol. 1, DOI: 10.1109/CVPR.2005.177.

[9] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 971-987, July 2002, DOI: 10.1109/TPAMI.2002.1017623.

[10] ArIES, IIT Roorkee, "Texture Analysis using LBP", 26th August 2018,

https://medium.com/@ariesiitr/texture-analysis-using-lbp-e61e87a9056d

[11] DataTurks: Data Annotations Made Super Easy, "Understanding SVM's for image classification", 10th August 2018,

https://medium.com/@dataturks/understanding-svms-for-image-classification-cf4f01232700

[12] *Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick*; The IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2961-2969

[13] Aditi Mittal, "Instance segmentation using Mask R-CNN", 17th June 2019,

https://towardsdatascience.com/instance-segmentation-using-mask-r-cnn-7f77bdd46abd

[14] Pedro Marcelino, "Transfer Learning from pre-trained models", 23rd October 2018,

https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751