Rough Set Approach for Efficient Solutions of OCR and Related Problems

Ushasi Chaudhuri

Roll No.: 15AT71P01

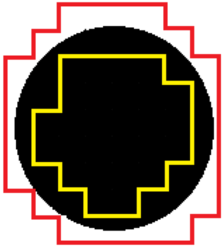
under the guidance of
Dr. Partha Bhowmick and Dr. Jayanta Mukhopadhyay
ATDC
IIT Kharagpur

September 13, 2021

Outlines

Introduction : Rough Set

- Rough set is an approximation of a crisp set in terms of a pair of sets for lower and upper approximations of the original set.



- It is easier to handle each of the approximating sets compared to the original crisp set.

Prior Work: Rough-Set

- Rough set — set the basic fundamentals Pawlak[1]
- Thangavel[2] tackled a dimensionality reduction problem in data mining.
- Rough sets used for feature selection for unsupervised clustering by Questier[3]

[1] Pawlak & Zdzislaw,"Rough sets", IJCIS-1982

[2] Thangavel,"Dimensionality reduction based on rough set theory", ASC-2009

[3] Thangavel,"Application of rough set theory to feature selection for unsupervised clustering", Chemometrics and Intelligent Laboratory Systems-2002

Prior Work: OCR using Rough Set

- Czajewski[4] use a discernibility matrix built on a reduced database for classification algorithms; average accuracy of 66.6%.
- Telugu[5] and Thai [6] character recognition machines, using rough sets.

[4]Czajewski,"Rough Sets in Optical Character Recognition", RSCTC-1998
[5]Srinivas,"Telugu Font and Character Recognition Using Rough Sets",ICCR-2008
[6]Kasemsiri,"Printed Thai character recognition using fuzzy-rough sets"

Prior Work: Unsupervised OCR

- Supervised Algorthms — near-human performance on handwritten digit recognition[7].
- Attains very high accuracy on the task of recognizing English characters[8]
- Unsupervised Algorithms — restricted Boltzmann machines (RBMs)[9], sparse autoencoders [10], sparse coding[11], and K-means; computational complexity and scalability.

[7] Ciresan,"Text detection and character recognition in scene images", IDSIA-12

[8] Coates,"Multi-column deep neural networks for image classification", ICDAR, 2011

[9] Hinton," A fast learning algorithm for deep belief nets", Neural Computation,2006

[10] "Measuring invariances in deep networks", NIPS,2009

[11] Gregor,"Learning fast approximations of sparse coding",ICML,2010

Introduction : Why OCR??

- Demanding subject in the field of document digitization.
- Editing, searching, and formatting of text
- Characters scripted using atypical and complex font styles: datasets huge in volume and diversity.
- Many supervised classification models, (immense time tenacity for training, resulting in slowdown)
- Google Tesseract[12] uses various geometric features; tedious training process to improve the efficiency; also reduces the speed of the OCR engine.
- To circumvent the pitfalls of training and supervised classification — rough set.

[12]"http://tesseract-ocr.repairfaq.org/"

Objectives:

- To develop an appropriate set of rough-set reduct.
- To apply the defined reduct to develop font independent unsupervised OCR system.
- To extend the individual OCR system to develop a complete PDF reader.
- To further explore if the rough set reduct can also be used for other applications such as logo retrieval.

Motivation

- Rough set provides a computationally efficient way of analysing spatial binary data set (image).
- Ease of defining the rough-set attributes(reduct).
- OCR is an important problem in modern day interface for text-to-speech and digital data storage.
- Reduced computation and font independence are of paramount importance, apart from accuracy.
- Can we develop an unsupervised OCR system?

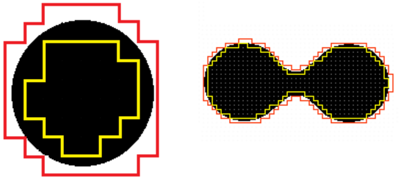
Motivation— Unsupervised OCR System

- Very difficult to develop a font independent supervised system.
- Requires re-training to incorporate newer fonts for supervised learning.
- Supervised system is computation and memory intensive.
- However, supervised system is potentially more accurate.
- We attempt to develop an efficient and accurate unsupervised OCR system.

Definition

Definition (Rough-set)

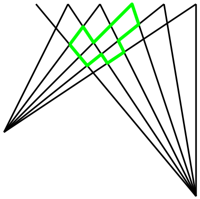
A *Rough-set* is an approximation of a crisp set in terms of a pair of polygonal sets for lower and upper approximations of the original set.



Definition

Definition (Rough-set Polygon)

A *Rough-set polygon* is a polygon which is constructed by two parametric families of straight lines with their centers at two points. Rectilinear polygons are a type of isothetic polygons with their centers lying at infinity.



For computational advantage, we consider only axes parallel polygons.

Definition

Definition (Rough-set Reduct)

A *Rough-set reduct* is considered to be a subset of definable attributes which may be considered as sufficient for a given purpose.

Person	Eyes	Beard	Height	Complexion
o_1	Blue	None	Short	Dark
o_2	Black	Moustache	Short	Fair
o_3	Black	None	Tall	Fair
o_4	Black	None	Tall	Dark
o_5	Black	None	Short	Dark

{Eyes, Height, Complexion}

Approximation Space

- Pawlak[13][14] was the first person to briefly set the basic fundamentals of rough-set.

  $\mathbb{A} = (U, R) \Rightarrow$ *approximation space*;
  where, U is the *universe*
  R is an equivalence relation or *indiscernibility relation*

- If points $x, y \in U$ and $(x, y) \in R$ then, x and y are indistinguishable in $\mathbb{A}$.

- We use X as a subset of U.
- An empty set is denoted as 0, while an universal set U is denoted as 1.

[13]Pawlak & Zdzislaw,"Rough sets", IJCIS-1982
[14]Pawlak & Zdzislaw,"Rough Sets: Theoretical Aspects of Reasoning About Data", Kluwer Academic Publishing-1991

Object Approximation

Definition (Tight Upper Approximation)

A finite and minimal union of elementary sets, also called a composed set, containing X is called the *best tight upper approximation* of X in $\mathbb{A}$, denoted by $\overline{Apr}_{\mathbb{A}}(X)$.

Definition (Tight Lower Approximation)

The greatest composed set in $\mathbb{A}$ contained in X is called the *best tight lower approximation* of X in $\mathbb{A}$, denoted by $\underline{Apr}_{\mathbb{A}}(X)$.

Object Approximation

The following topological relations [15] hold true with the above symbols:

$$\overline{Apr}_{\mathbb{A}}(X) \supset X \supset \underline{Apr}_{\mathbb{A}}(X) \tag{1}$$

$$\underline{Apr}_{\mathbb{A}}(1) = \overline{Apr}_{\mathbb{A}}(1) = 1 \tag{2}$$

$$\underline{Apr}_{\mathbb{A}}(0) = \overline{Apr}_{\mathbb{A}}(0) = 0 \tag{3}$$

$$\underline{Apr}_{\mathbb{A}}(\underline{Apr}_{\mathbb{A}}(X)) = \overline{Apr}_{\mathbb{A}}(\underline{Apr}_{\mathbb{A}}(X)) = \underline{Apr}_{\mathbb{A}}(X) \tag{4}$$

$$\overline{Apr}_{\mathbb{A}}(\overline{Apr}_{\mathbb{A}}(X)) = \underline{Apr}_{\mathbb{A}}(\overline{Apr}_{\mathbb{A}}(X)) = \overline{Apr}_{\mathbb{A}}(X) \tag{5}$$

$$\overline{Apr}_{\mathbb{A}}(X) = U - \underline{Apr}_{\mathbb{A}}(U - X) \tag{6}$$

$$\underline{Apr}_{\mathbb{A}}(X) = U - \overline{Apr}_{\mathbb{A}}(U - X) \tag{7}$$

[15] Moshkov *et al.*,"Partial Covers, Reducts and Decision Rules in Rough Sets: Theory and Applications", Springer Publishing Company-2009

Object Approximation

Upper and Lower boundaries for an object X.

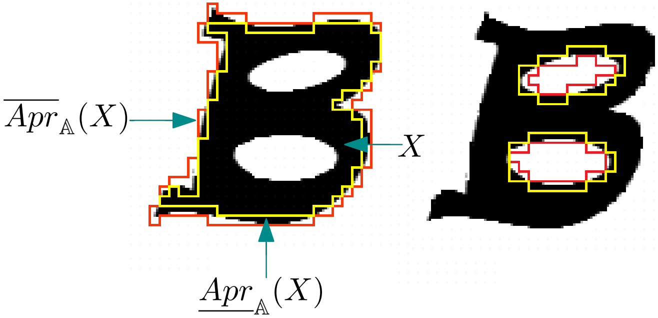
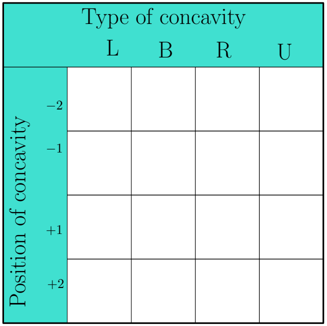

$\overline{Apr}_{\mathbb{A}}(X)$
X
$\underline{Apr}_{\mathbb{A}}(X)$

Figure: Boundaries of object X

Feature Approximation



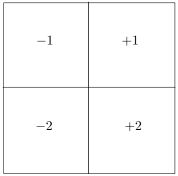


Figure: Approximations of concavity feature

Tight Upper and Lower Approximations

- S is a 2D digital object and $\mathbb{G}$ a cellular grid.
- *Tight upper approximation* is given by $\overline{\mathcal{P}}_{\mathbb{G}}(S)$, and *tight lower approximation* is $\underline{\mathcal{P}}_{\mathbb{G}}(S)$ of S (induced by $\mathbb{G}$) where $\underline{\mathcal{P}}_{\mathbb{G}}(S)$ and $\underline{\mathcal{P}}_{\mathbb{G}}(S)$ can have multiple polygons.
- *Accuracy*, $\alpha_{\mathbb{G}}(S)$ of S is given by

$$\alpha_{\mathbb{G}}(S) = \frac{\mathrm{area}\left(\underline{\mathcal{P}}_{\mathbb{G}}(\mathrm{S})\right)}{\mathrm{area}\left(\overline{\mathcal{P}}_{\mathbb{G}}(\mathrm{S})\right)} \tag{8}$$

Computation of Upper and Lower Approximation

Start vertex, $v_s = (i_s, j_s)$: To make the *tight approximations* of the rough set cover, we start by finding the first object point in the image.

$$i_s = \left(\left\lceil \frac{i_o}{g} \right\rceil - 1 \right) \cdot g, \tag{9}$$

$$j_s = \left(\left\lfloor \frac{j_o}{g} \right\rfloor + 1 \right) \cdot g. \tag{10}$$

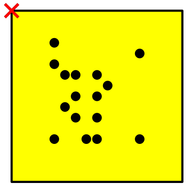
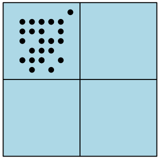
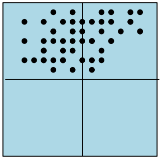


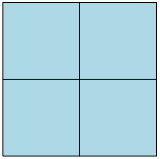
Figure: Top-left vertex

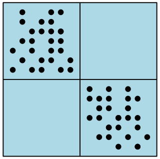
Computation of Upper and Lower Approximation I

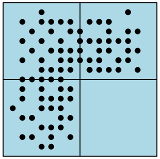
Class, C_i: Total of five possibilities (C_i) of object occupancies for any vertex v_i (i^{th} vertex).

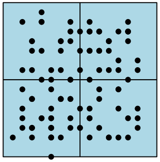

C_1


C_2


C_0


C_2

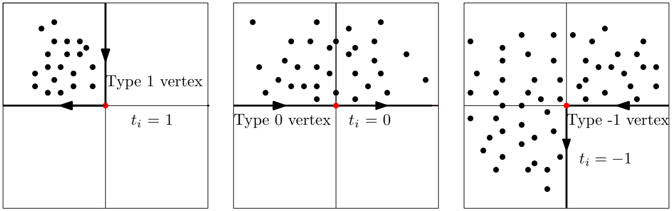

C_3


C_4

Computation of Upper and Lower Approximation I

Type, t_i: There can be 3 types of vertex:

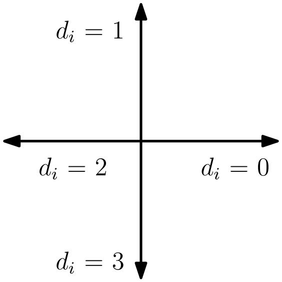
- 90° vertex — Type ‘1’ : $(1 \times 90° = 90°)$
- 270° vertex — Type ‘−1’ : $(-1 \times 90° = -90°)$
- straight 180° edge — Type ‘0’



Computation of Upper and Lower Approximation I

Direction, d_i: To assign the directions (d_i from the i^{th} vertex) as 0, 1, 2 and 3 for right, top, left and bottom respectively. A current direction from a vertex v_i uses the previous direction as well for its next direction calculation and is given by

$$d_i = (d_{i-1} + t) \mod 4 \tag{11}$$



Rought-Set: Dependance on cell size

We use rough-sets in two stages[16]—

- For construction of upper and lower approximations of a 2D digital object.
- Defining the approximations of their attributes comprising the reduct.

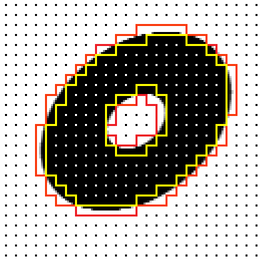
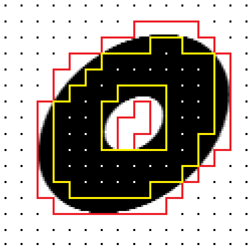



Figure: Cell-size 6×6 and 9×9.

[16] Pawlak & Zdzislaw,"Rough sets", IJCIS-1982

Attributes for the Rough-Set

With this we have got list of grid points with information about:

- vertices types
- x and y coordinates
- Direction of propagation

This list represents the basic structure of the character enveloped in the cover. From this list, we calculate various features of the cover in order to characterize the object covers individually.

Attributes for the Rough-Set Reduct

From the vertex sequence of each polygon in $\overline{\mathcal{P}}_{\mathbb{G}}(S)$ or $\underline{\mathcal{P}}_{\mathbb{G}}(S)$, we compute the values of the reduct attributes(i.e., features) like-

1. Approximate Euler Number
2. Relative Hole Positions
3. Edge Ratio
4. Directional Changes
5. Concavity Features
6. Hull Concavities

Approximate Euler Number

Definition (Hole polygon)

Some of the characters cover the entire region inside the cover, while some characters have voids inside them. These void covers are called as the lower boundary or a hole polygon.

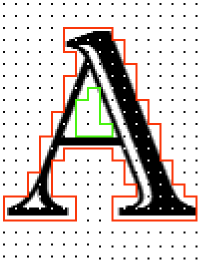
The Euler number(EN) is calculated as—

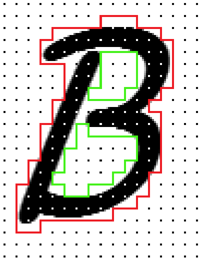
$$EN = 2 - n \tag{12}$$

where $n =$ total number of polygons in $\overline{\mathcal{P}}_{\mathbb{G}}(S)$.

Characters like `A` or `D` have Euler number 0, while `E`, `F`, etc., have Euler number 1. This is used as the primary attribute in our classification system.

Approximate Euler Number





A — $n = 2$; $\text{EN} = 0$

B — $n = 3$; $\text{EN} = -1$

Since we are using rough set lower boundaries, we are able to get 2 holes despite having internal connectivities.

Relative Hole Positions

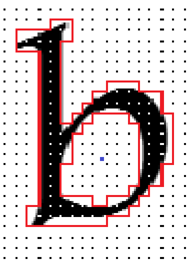
Definition (Hole position)

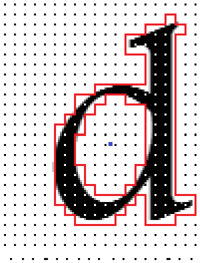
A hole position is given by the centroid of the hole w.r.t to the centroid of the containing polygon.

The hole centroid (c), is found in the local coordinate with the top left vertex of the character as the reference point.

The relative position (PoH) of each hole polygon is determined by comparing its center c with the top-left vertex v_0 of the outer polygon in $\overline{\mathcal{P}}_{\mathbb{G}}(S)$.

Relative Hole Positions





'$-$' and '$+$' to denote left and right lateral halves
'1' and '2' for upper and lower halves;
b has PoH $= +2$ and d has PoH $= -2$.

	EN	PoH	VDC	Concavity	ER
b	0	$+2$	8	-	2
d	0	-2	8	-	2

Edge Ratio

Definition (Horizontal Perimeter Component (HPC))

For each polygon in $\overline{\mathcal{P}}_{\mathbb{G}}(S)$, we define horizontal perimeter component (HPC) as the sum of the lengths of its horizontal edges.

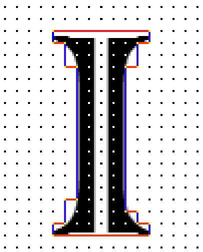
Definition (Vertical Perimeter Component (VPC))

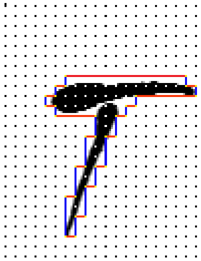
For each polygon in $\overline{\mathcal{P}}_{\mathbb{G}}(S)$, we define vertical perimeter component (VPC) as the sum of the lengths of its vertical edges.

Definition (Edge Ratio (ER))

The ratio VPC:HPC, discretized to the nearest value in $\{\frac{1}{2}, 1, 2\}$, is called edge ratio (ER).

Edge Ratio





By ER conventions, we get an ER of 2 and 1 for I and T

	EN	PoH	VDC	Concavity	ER
I	+1	-	2	-	2
T	+1	-	2	-	1

Directional Changes

Definition (U-turn)

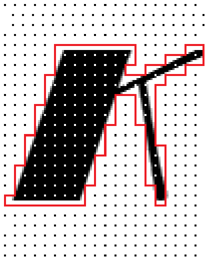
U-turn is defined by a vertex sequence where two consecutive vertices are of type $\langle +, + \rangle$; and for each such U-turn.

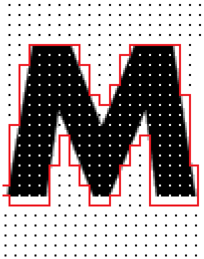
Definition (Vertical Direction Change (VDC))

When we start traversing along the upper boundary of the rough set perimeter of the object from the top left corner, the number of times we encounter a directional 'U-turns' in the vertical direction (ex 1 to 3, or 3 to 1) is defined as vertical direction change (VDC).

Similarly we have Horizontal direction change (HDC).
e.g. `I` has VDC 2; `A` has VDC 4.

Directional Changes





VDC(K)=6 and VDC(M)=8 The two challenges of using VDC —

- Variations due to different fonts — solved using error function.
- For characters with same VDC, like (M/W), we also use their VDC position informations.

Concavity Features

Definition (Concavity)

Concavity is defined as two consecutive vertices of type $\langle -, - \rangle$ (without considering the type intermediate 0 vertices).

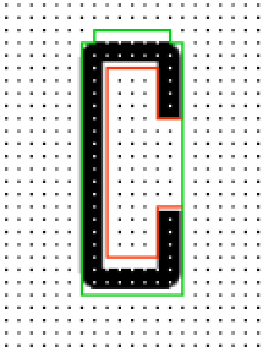
If more than two type -1 vertices occur one after another, then we get a nested concavity. Types of concavities—

- left (L)
- right (R)
- upward (U)
- downward (B)

Concavity Features

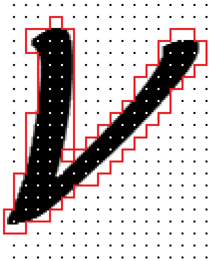
The two challenges of using concavity features —

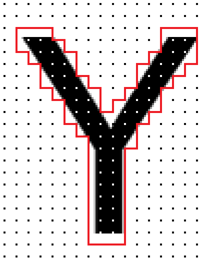
- Different grid size ⇒ different covers ⇒ differently oriented concavities; Solution — hull concavity.



U, V and Y, having similar cover, we use *depth* information. We discretize the relative depth in 1, 2, 3.

Concavity Features





⟨concavity direction, region, depth⟩

v=(U,+1, 2) and Y=(U,+1, 1)

	EN	PoH	VDC	Concavity	ER
V	+1	-	8	(U,+1, 2)	2
Y	+1	-	8	(U,+1, 1)	2

Hull Concavities

Definition (Orthogonal Hull)

An orthogonal hull is a polygon with some degenerate edges connecting extreme vertices in each coordinate direction.

Useful for finding the main concavity direction and ignore the rest of the concavities. To construct an orthogonal hull from a rough-set cover, we use retracing algorithm [17]. Wherever we get any concavities, we retrace the cover back and get the hull. Again, the four types of concavities depending upon orientation —

- left (L)
- right (R)
- upward (U)
- downward (B)

[17] Arindam Biswas *et al.*,"A linear-time combinatorial algorithm to find the orthogonal hull of an object on the digital plane", Information Sciences-2012

Hull Concavities

In this type of concavity, all the spare noise concavities are more or less removed.

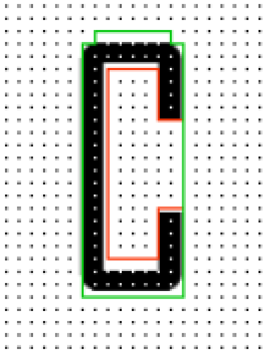


Figure: Hull concavity of C at L at +2

Partial Illustration of Attributes

	EN	PoH	VDC	Concavity	ER
B	-1	$+1, +2$	2	(L,$+1, -$)	2
E	$+1$	-	2	(L,$+1, -$),(L,$+2, -$)	$\frac{1}{2}$
I	$+1$	-	2	-	2
M	$+1$	-	8	(D,$-2, -$),(D,$+2, -$),(U,$+1, -$)	1
T	$+1$	-	2	-	1
V	$+1$	-	8	(U,$+1, 2$)	2
Y	$+1$	-	8	(U,$+1, 1$)	2
b	0	$+2$	8	-	2
d	0	-2	8	-	2
3	$+1$	-	8	(R,$+1, -$),(R,$+2, -$)	1

Table: Information table (shown partial)

Illustration of Classifiability Based on Attributes

The attribute space classification of the characters in two stages of the pipeline —

Position of concavity	Type of concavity L	B	R	U
−2		N	35S	
−1			3	
+1	G5S			HKNY
+2	CHKhn			

Position of hole	
−2	d
−1	q4
0	o
+1	P
+2	bd

Figure: Decomposition of equivalent class CGHKNSXhn35 based on rough-set attributes

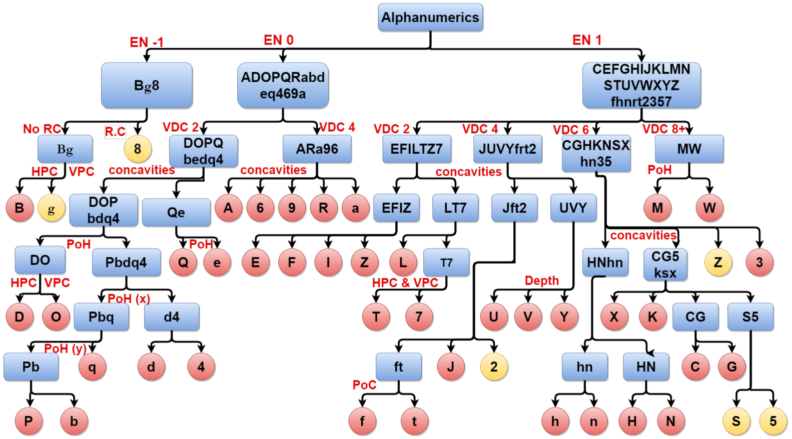
Limited Interface to Supervised Classifier

- Classification is context-free;
  `(0/o/O), (i/l/I/1), (C/c), (J/j), (K/k), (M/m), (P/p), (S/s), (U/u), (V/v), (W/w), (X/x), (Y/y), (Z/z).`
- Characters which bear structural resemblance with each other over a varied font style;
  `B(z/2), (9/g), (s/5), and (g/8).`
  These four character pairs are included in the indefinite class.

Tesseract Reinforced Model

- Few characters with high confusion are not well-discernible by the combinatorial attributes in the reduct.
- Certain atypical or idiosyncratic font styles creating confusion in the discretized topological space defined for the rough set.
- We design a hybrid model of character classification wherein we run the Tesseract for all the indefinite class characters.
- Improves the overall performance of the system, significantly gains in the average runtime.

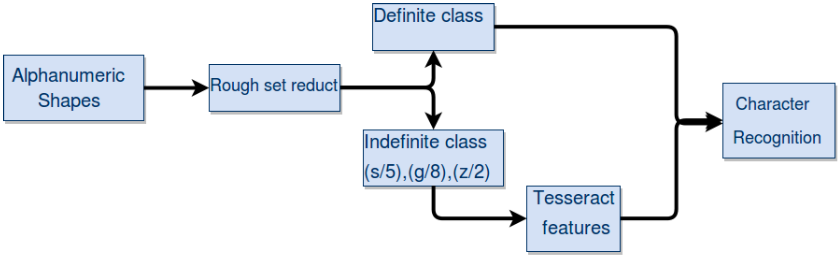
Classification Criteria



Red— Unsupervised, Yellow— Supervised

Block Diagram of the Overall Method

Proposed algorithm with Tesseract reinforcement.



Data Set

For testing, we have used the Chars74k dataset of optical characters [18]. This dataset contains

- images of 26 capital letters
- 26 small letters
- ten numeric digits.

They are written with 1016 different font styles.
Each image has a resolution of 128 $\times$ 128 pixels.

[18] "http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/"

Results

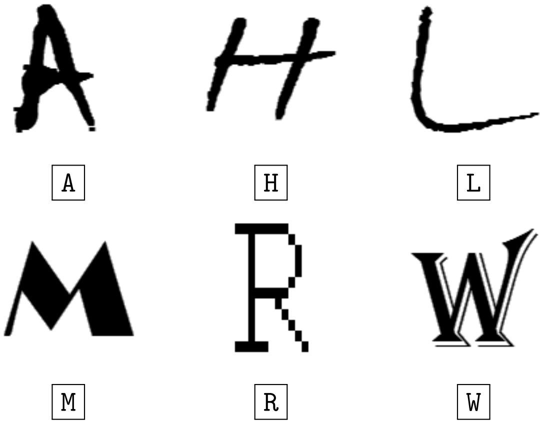


Figure: Reduct and Tesseract are independently successful.

Results

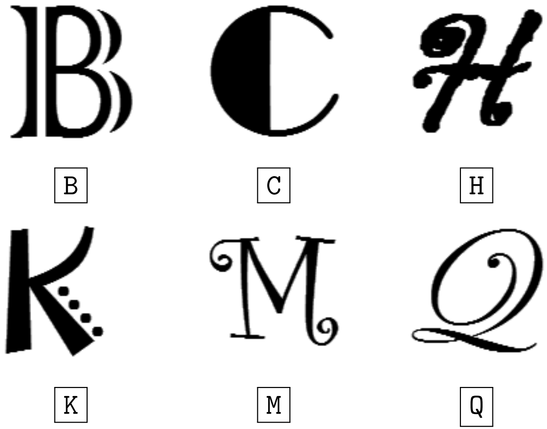


Figure: Reduct is successful; Tesseract fails.

Results

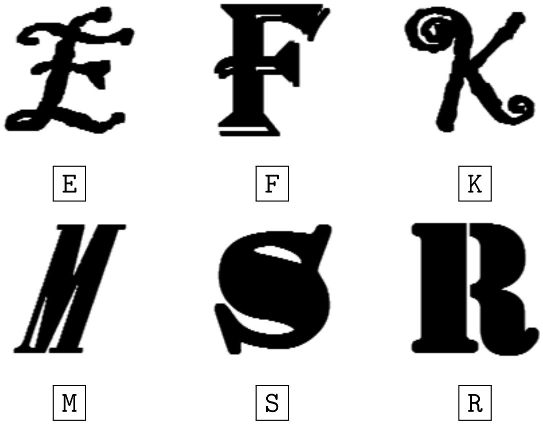


Figure: Reduct combined with Tesseract is successful.

Results

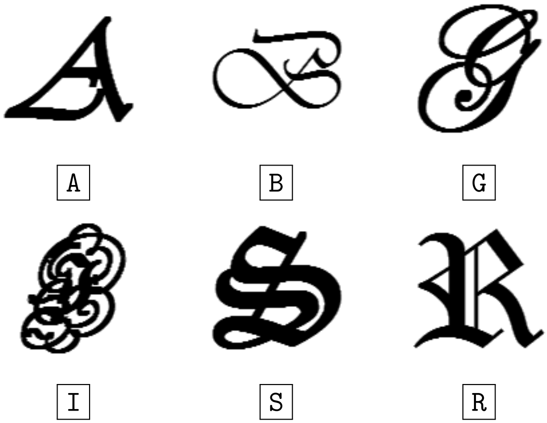


Figure: None is successful.

Results

To improve the accuracy, we group the characters into two classes, definite class and indefinite class. The indefinite class characters are those characters which are either unclassified or the characters having lower accuracy in classification.

- We get a result of about 88.98% accuracy
- The standard Google Tesseract, version 3.02.02, gives 64.79% with the inbuilt *eng.trainneddata* training.

Results

Accuracy		
Letters	Our Algorithm	Tesseract
CEIJKLMSVXYZf83	Above 90%	75.49 - 90.84%
ABDFHNOPQRTUW	80 - 90%	4.90 - 87.00%
Gbdem247	70 - 80%	5.01 - 80.70%
anqrt56	60 - 70%	0.78 - 52.85%
gh9	below 60%	1.08 - 60.33%
Average 88.98%	-	64.79%

Comparison by accuracy

Results

- Average execution time of 0.203 secs per character for the Google Tesseract engine
- 0.093 secs for the recognition using our engine based on reduct attributes.

This CPU time is achieved on a 64-bit Intel® 2-Core™ i5 processor, with 4GB RAM, DELL machine.

Conclusion

- Main advantage of the grid polygon approximation method is its reduced time complexity.
- When used for some large scale classification process, it saves us a huge amount of time.
- Also memory efficient as only classification criteria to be stored.
- The Tesseract used here is the untrained Tesseract. It's accuracy can be improved with training but at the cost of reduced speed.
- Reduced number of invocation of Tesseract in hybrid model for computational saving.

Scope for Future Work

- To extend the work to develop a PDF reader. .
- These geometrical features can be further used for various other applications like electrical symbol classification, logo retrieval, etc.
- To investigate the effect of skew on OCR accuracy.
- Possibility of application in Indian language OCR.
- To study robustness of the attributes in presence of noise.

Publication

- U. Chaudhuri, P. Bhowmick and J. Mukhopadhyay, “A Novel OCR System based on RoughSet Reduct”, *Pattern Recognition and Machine Intelligence* (PReMI) Conference, ISI Kolkata,Dec 2017.(communicated)

Thank You