UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Jaan Tohver

Optical Character Recognition for Extremely Low Quality Images

Master's Thesis (30 ECTS)

Supervisor: Gholamreza Anbarjafari, PhD

Tartu 2019

Optical Character Recognition for Extremely Low Quality Images

Abstract:

Optical character recognition (OCR) from printed and handwritten documents is virtually a solved problem for all practical purposes. Modern OCR systems are able to achieve a 99.9% detection rate which is on par with human capabilities. Where most OCR systems fall short is when the the input is of low quality, such as containing large amounts of noise or motion blur, or being of low resolution.

A real world use-case of this problem is detecting car licence plates from a security camera feed. Security cameras are often of low resolution, use high levels of compression, and have low framerate since the video footage needs to be stored for a long period of time and storage cost is paramount. In addition, cars can move unpredictably causing motion blur during the capture of a video frame.

The goal of this project is to test various methods of improving OCR accuracy with low quality input. Namely,

- Training a neural network (NN) on low quality images. And testing results on low quality images.
- Training a NN on high quality images and testing on digitally enhanced low quality images.
- Training a NN on digitally enhanced low quality images and testing on digitally enhanced images.
- Using multiframe registration of frames from a video feed to improve detection quality.

The tests will be conducted using frames from a blurry, noisy, and low resolution video feed containing text. Third party solutions are used to extract areas containing text from those frames.

Some of the training and test data is gathered specifically for this task by filming, and extracting frames from, an intentionally blurry video. Additional synthetic motion blur is added to some of the images. Some data is gathered from available open source databases, such as images containing vehicle licence plates.

Keywords:

OCR, Deblurring, Low Quality, Low Resolution

CERCS:

CERCS code and name: https://www.etis.ee/Portal/Classifiers/Details/d3717f7b-bec8-4cd9-8ea4-c89cd56ca46e

Contents

Unsolved issues

1 Introduction

Optical character recognition (OCR) is the conversion of text in analog form, such as images or printed documents, into digital text. OCR is not a new concept by any means, a patent for a system which can be called the predecessor of modern OCR was already granted in 1931. Modern OCR systems are of course far more powerful.

1.1 Motivation

For all intents and purposes OCR is basically a solved problem for detecting text from clear and high resolution images containing printed or handwritten text. Modern systems perform on par with human capabilities in this regard, boasting a 99.9% detection rate. A problem still being researched, however, is OCR from low quality images. Low quality can mean low resolution, high levels of compression, containing artefacts generated by scanning for example, containing motion blur caused by movement of the object of the camera during image capture, or any combination thereof.

Accurate OCR from low quality images and video could solve several problems for automating processes where the system needs to detect some text. For example an autonomous parking garage, where a camera is filming approaching cars, a system analyses the video feed and opens the gate or barrier for cars with licence plate numbers matching authorized users.

1.2 Contributions

The goal of this thesis was test different approaches for such a system. Namely,

- Training a neural network (NN) on low quality images. And testing results on low quality images.
- Training a NN on high quality images and testing on digitally enhanced low quality images.
- Training a NN on digitally enhanced low quality images and testing on digitally enhanced images.
- Using multiframe registration of frames from a video feed to improve detection quality.

1.3 Outline

Chapter 2 describes the state of the art in OCR and low quality image enhancement and discusses the advantages and drwabacks of each of the approaches.

Chapter 3 describes the research problem and the necessity of finding a solution to the problem.
Chapter 4 describes in detail the approach taken to solve the problem.
Chapter 5 describes the evaluation process of the results as well as comparison to related works.
Chapter 5 concludes the thesis with a summary and research directions for the future

2 State of the Art

This section gives an overview of current commertial OCR and image enhancement solutions as well as research related to the topic.

2.1 History

The history of OCR can be traced back to the late 1920s when inventor Emanuel Goldberg started developing a system he called a "Statistical Machine" which searched microfilm archives using an optical code recognition system. He was granted a patent for his invention in 1931, which was later acquired by IBM. [1]

Most of modern OCR systems are based on artificial neural networks, often called just neural networks (NNs). The theoretical base for NNs dates back to the end of the 1800s and is based on the structure of the human brain in which neurons interact with eachother and the bonds between neurons can strengthen and weaken over time.

An artificial neural network consists of many simple connected processors, called artificial neurons, which produce activations based on an activation fuction producing real numbers in a predefined range. The simplest neural network consists of an input layer and an output layer. The neurons in the input layer produce activations based on the input. The output layer produces an output based on the activations of input layers. [2]

Most NNs are not as simple as that and contain any number of hidden layers between the input and output layers. These hidden layers each consist of any number of neurons which each produce an activation based on the output of another layer. [2]

maybe a graphic

Does not cover all cases? http://www.asimovinstitute.org/neural-network-zoo/

There are various architectures used for NNs, some are combinations of other types of networks. This work mostly focuses on convolutional neural networks (CNNs) and recurrent neural networks (RNNs). [2]

CNNs are primarily used for image processing. CNNs are feedforward networks (FFNs), which means neurons do no back-propagate any information and only pass it forward to the next layer. A distinct feature of CNNs is that they consist of convolutional layers in which not all neurons are connected to all neaurons in the previous and next layers. [2]

RNNs are primarily used for text and speech processing. RNNs are FFNs like CNNs, however unlike CNNs they are stateful, meaning in addition to receiving data from the previous layer, each neuron retains data from previous passes. RNNs are also able to handle arbitrary input and output lengths unlike CNNs [2]

1979: convolution + weight replication + subsampling (Neocognitron)

2.2 State of the Art

CNNs and RNNs are not mutually exclusive, however, and Baoguang Shi et'al devised one of the most promising novel OCR concepts in recent history in the form of a convolutional recurrent neural network (CRNN) in their 2015 paper "An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition".

Their reasoning is that popular models like CNN cannot be applied to sequence prediction, since they operate on fixed size inputs and outputs. CRNN then behaves like an RNN in that it can accept inputs and return outputs of arbitrary size, while still retaining properties of CNNs which make them invaluable in image processing.

2.2.1 Commertial OCR systems

Tesseract is probably the most well known OCR engine. It started as proprietary software developed by Hewlett Packard in 1985, but was open sourced in 2005. Since then its development is continued by the community. Development has been sponsored by Google starting from 2006.

citations

Tesseract supports over 100 languages by default and can be trained to work with any language or script.

2.2.2 Title of Subsubsection 2

Some text...

2.3 Title of Subsection 2

Rule: If you divide the text into subsections (or subsubsections) then there has to be at least two of them, otherwise do not create any.

Tip: You can also use paragraphs, e.g.

Type rules for integers. Some text ...

Type rules for rational numbers. Some text here too...

2.4 How to use references

Cross-references to figures, tables and other document elements. LaTeX internally numbers all kind of objects that have sequence numbers:

- chapters, sections, subsections;
- figures, tables, algorithms;
- equations, equation arrays.

To reference them automatically, you have to generate a label using `\label{some-name}` just after the object that has the number inside. Usually, labels of different objects are split into different namespaces by adding dedicated prefix, such as `sec:`, `fig:`. To use the corresponding reference, you must use command `\ref` or `\eqref`. For instance, we can reference this subsection by calling Section 2.4. Note that there should be a nonbreakable space ~ between the name of the object and the reference so that they would not appear on different lines (does not work in Estonian).

Citations. Usually, you also want to reference articles, webpages, tools or programs or books. For that you should use citations and references. The system is similar to the cross-referencing system in LaTeX. For each reference you must assign a unique label. Again, there are many naming schemes for labels. However, as you have a short document anything works. To reference to a particular source you must use `\cite{label}` or `\cite[page]{label}`.

References themselves can be part of a LaTeX source file. For that you need to define a bibliography section. However, this approach is really uncommon. It is much more easier to use BibTeX to synthesise the right reference form for you. For that you must use two commands in the LaTeX source

- \bibliographystyle{alpha} or \bibliographystyle{plain}
- \bibliography{file-name}

The first command determines whether the references are numbered by letter-number combinations or by cryptic numbers. It is more common to use `alpha` style. The second command determines the file containing the bibliographic entries. The file should end with `bib` extension. Each reference there is in specific form. The simplest way to avoid all technicalities is to use graphical frontend Jabref (`http://jabref.sourceforge.net/`) to manage references. Another alternative is to use DBLP database of references and copy BibTeX entries directly form there.

The following paragraph shows how references can be used. Game-based proving is a way to analyse security of a cryptographic protocol [3, 4]. There are automatic provers, such as CertiCrypt [5] and ProVerif [6].

3 How to add figures and pictures to your thesis

Here are a few examples of how to add figures or pictures to your thesis (see Figures 1, 2, 3).

Rule: All the figures, tables and extras in the thesis have to be referred to somewhere in the text.

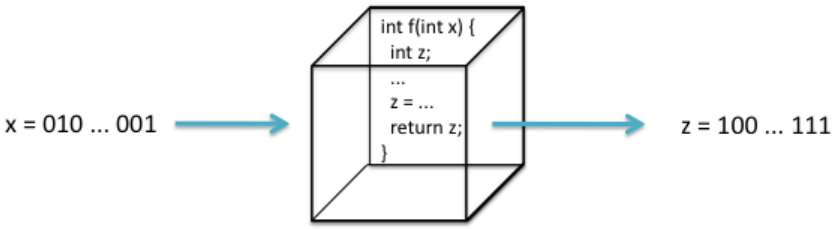


Figure 1. The title of the Figure.

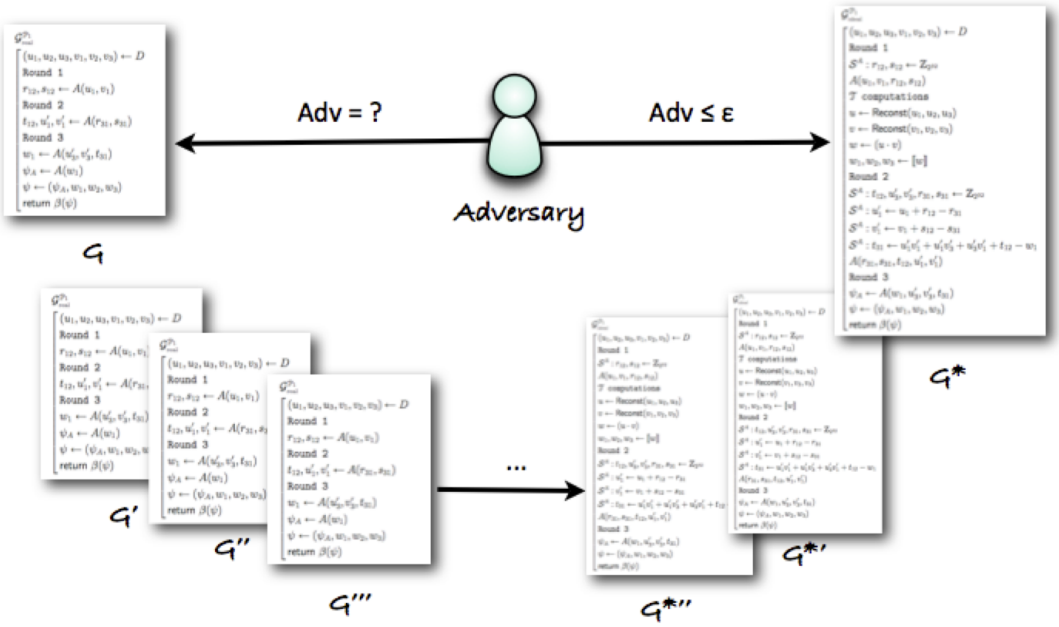


Figure 2. Refer if the figure is not yours [7].

Tip: If you add a screenshot then labeling the parts might help make the text more understandable (panel C vs bottom left part), e.g.

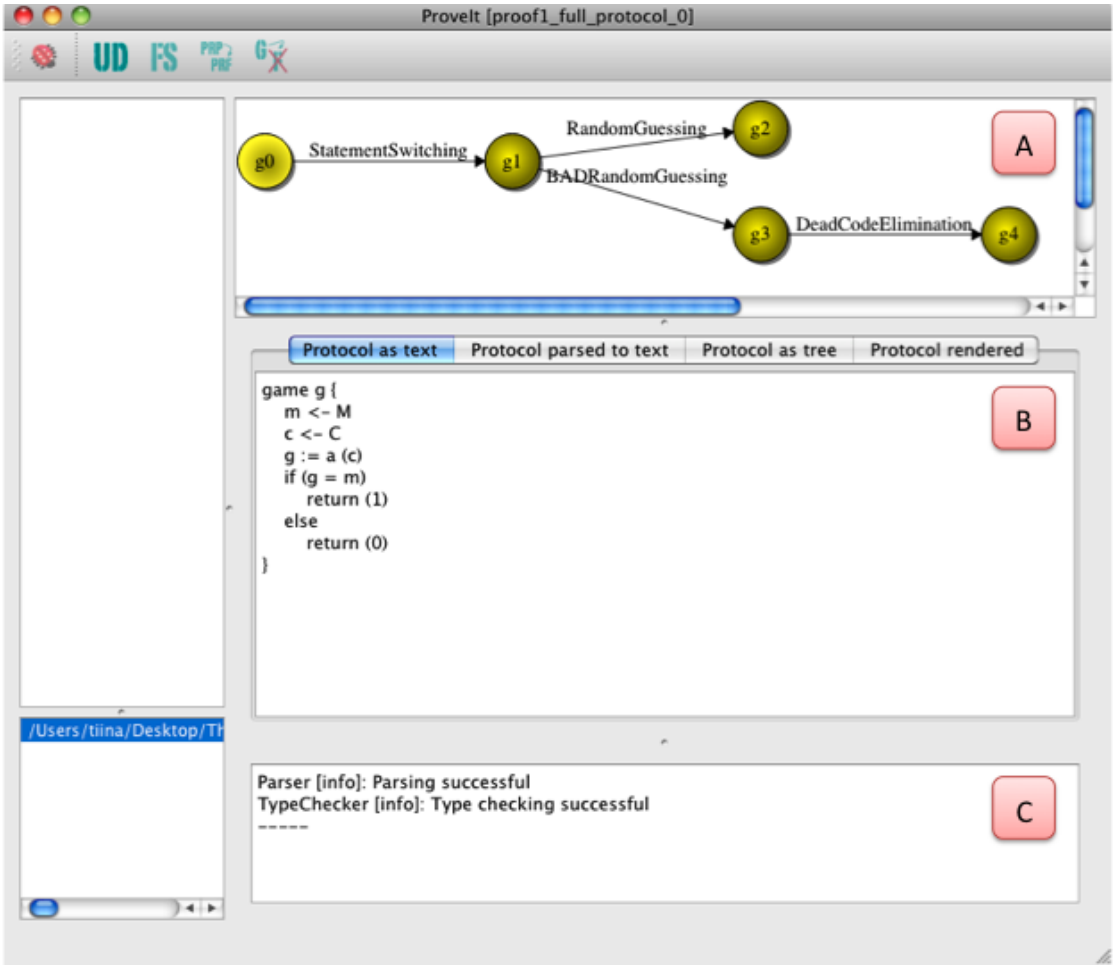
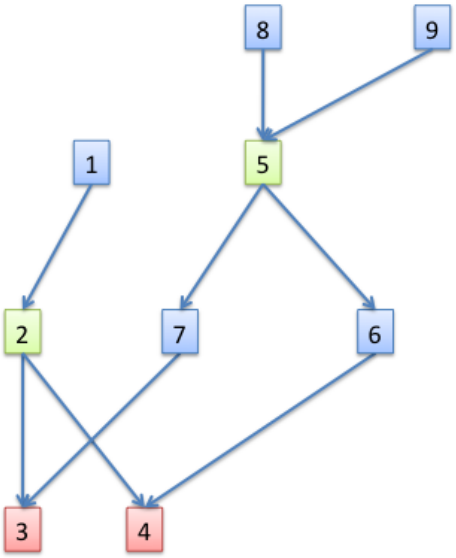


Figure 3. Screenshot of ProveIt.



Node	Decendants
1	2, 3, 4
2	3, 4
3	
4	
5	3, 4, 6, 7
6	4
7	3
8	3, 4, 5, 6, 7
9	3, 4, 5, 6, 7

Figure 4. Example how to put two figures parallel to each other.

Example: A screenshot of ProveIt can be seen on Figure 3. The user first enters the pseudocode of the initial game in panel B. ProveIt also keeps track of all the previous games showing the progress on a graph seen in panel A.

There are two figures side by side on Figure 4.

4 Other Ways to Represent Data

4.1 Tables

Table 1. Statements in the ProveIt language.

Statement	**Typeset Example**
assignment	$a := 5 + b$
uniform choice	$m \leftarrow M$
function signature	$f : K \times M \rightarrow L$

4.2 Lists

Numbered list example:

1. item one;
2. item two;
3. item three.

4.3 Math mode

Example:

$$a + b = c + d \tag{1}$$

Aligning:

$$\begin{aligned} a &= 5 \\ b + c &= a \\ a - 2 * 3 &= 5/4 \end{aligned}$$

Hint: Variables or equations in text are separated with $ sign, e.g. a, $x - y$.

Inference Rules

$$\text{addition}\frac{\Gamma \vdash x : T \qquad \Gamma \vdash y : T}{\Gamma \vdash x + y : T}$$

Bigger example:

$$\text{assign}\frac{\Gamma \vdash c := a + b \qquad \text{addG}\ \dfrac{\Gamma \vdash a : \mathsf{Rat} \qquad \text{var}\ \dfrac{\Gamma \vdash b : \mathsf{Int} \qquad \Gamma \vdash \mathsf{Int} \subseteq \mathsf{Rat}}{\Gamma \vdash b : \mathsf{Rat}}}{\Gamma \vdash a + b : \mathsf{Rat}}}{\Gamma \vdash c : \mathsf{Rat}}$$

4.4 algorithm2e

Algorithm 1: typeChecking

Input: Abstract syntax tree

Result: Type checking result; In addition, type table $\text{type}_{\text{type\_G}}$ for global variables, $\text{type}_{\text{game}}$ for the main game and type_{fun} for each $fun \in F$

```
while something changed in last cycle do
    foreach global statement s do parseStatement(s, type_type_G);
    ;
    foreach function fun do
        foreach statement s in fun do parseStatement(s, type_fun);
        ;
    foreach statement s in game do parseStatement(s, type_game);
    ;
```

4.5 Pseudocode

```
expression
  : NUMBER
  | VARIABLE
  | '+' expression
  | expression '+' expression
  | expression '*' expression
  | function_name '(' parameters ')'
  | '(' expression ')'
```

Figure 5. Grammar of arithmetic expressions.

4.6 Frame Around Information

Tip: We can use minipage to create a frame around some important information.

1. integer division (\div) – only usable between Int types
2. remainder (%) – only usable between Int types

Figure 6. Arithmetic operations in ProveIt revisited.

5 Conclusion

what did you do?

What are the results?

future work?

References

[1] G. Emanuel, "Statistical machine," Patent 1 838 389, December, 1931. [Online]. Available: http://www.freepatentsonline.com/1838389.html

[2] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608014002135

[3] M. Bellare and P. Rogaway, "Code-based game-playing proofs and the security of triple encryption," Cryptology ePrint Archive, Report 2004/331, 2004, http://eprint.iacr.org/.

[4] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs," Cryptology ePrint Archive, Report 2004/332, 2004, http://eprint.iacr.org/.

[5] G. Barthe, B. Grégoire, and S. Zanella Béguelin, "Formal certification of code-based cryptographic proofs," in *36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009*. ACM, 2009, pp. 90–101. [Online]. Available: http://dx.doi.org/10.1145/1480881.1480894

[6] B. Blanchet, "Proverif: Cryptographic protocol verifier in the formal model," http://www.proverif.ens.fr/.

[7] L. Kamm, "ProveIt – How to make proving cryptographic protocols less tedious," Talk at the 21st Estonian Computer Science Theory Days at Kubija, January 2012. [Online]. Available: http://www.cs.ut.ee/~varmo/tday-kubija/kamm-slides.pdf

Appendix

I. Glossary

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Jaan Tohver**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   Optical Character Recognition for Extremely Low Quality Images

   supervised by Axel Rose and May Flower

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, dd.mm.yyyy