



Thesis report

Course: Advanced Course Military Technology, Independent Project 2023	
Course code: 1OF011	Credits: 15 ETCS
Supervisor: Commander Thomas Svensson	Date: April 11, 2023
Examiner: Associate Professor Hans Liwång	Number of words: 14093

Automatic compilation and summarization of documented Russian equipment losses in Ukraine – A method development

Abstract

Since the Russian invasion of Ukraine on the 24th of February 2022 – most of the United Nations have, in one way or another, participated in the most significant war of many decades. The war is characterized by Russia's atrocious war crimes, illegal annexations, terror, propaganda, and complete disrespect for international law. On the other hand, the war has also been characterized by Ukrainian resilience, a united Europe, and a new dimension of intelligence gathering through social media.

Due to the internet, social media, the accessibility of mobile devices, and Ukraine's military and civilian effort in documenting Russian equipment – its whereabouts, status, and quantity, Open-Source Intelligence possibilities have reached new levels for both professionals and amateurs. Despite these improved possibilities, gathering such a vast amount of data is still a Herculean effort.

Hence, this study contributes a starting point for anyone wanting to compile equipment losses by providing a process specialized in automatic data extraction and summarization from an existing database. The database in question is the image collection from the military analysis group Oryxspioenkop. To further complement the information provided by Oryxspioenkop, the method automatically extracts and annotates dates from the images to provide a chronological order of the equipment loss as well as a graphical overview.

The process shows promising results and manages to compile a large set of data, both the information provided by Oryx and the extracted dates from its imagery. Further, the automated process proves to be many times faster than its manual counterpart, showing a linear relationship between the number of images analysed and manhours saved. However, due to the limited development time – the process still has room for improvement and should be considered semi-automatic, rather than automatic. Nevertheless, thanks to the open-source design, the process can be continuously updated and modified to work with other databases, images, or the extraction of other strings of text from imagery.

With the rise of competent artificial image generation models, the study also raises the question if this kind of imagery will be a reliable source in the future when studying equipment losses, or if artificial intelligence will be used as a tool of propaganda and psychological operations in wars to come.

Keywords: Russian Equipment loss, Russo-Ukrainian War, Method development, Machine learning, Tesseract OCR, Oryxspioenkop, OSINT



Försvarshögskolan

Rapport – Självständigt Arbete

Kurs: Påbyggnadskurs Militärteknik, Självständigt arbete VT2023	
Kurskod: 1OF011	Poäng: 15 HP
Handledare: Kommendörkapten Thomas Svensson	Datum: 2023-04-11
Examinator: Docent Hans Liwång	Antal ord: 14093

Automatisk sammanställning och sammanfattning av dokumenterade ryska materielförluster i Ukraina – Metodutveckling

Sammanfattning

Sedan Rysslands provocerade invasion av Ukraina den 24e februari 2022 – har stora delar av de Förenta nationerna engagerat sig i århundradets mest signifikanta krig. Kriget har karaktäriserats av ryska krigsbrott, olagliga annekteringar, terror, propaganda samt en total avsaknad av respekt för folkrätt. I kontrast, har kriget även karaktäriserats av Ukrainas ovillkorliga motståndskraft, ett enat Europa och en ny dimension av underrättelseinhämtning från sociala medier.

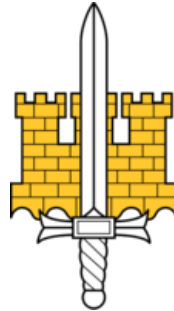
Genom internet, sociala medier, tillgängligheten av mobiltelefoner och Ukrainas militära och civila ansträngning att dokumentera rysk materiel – vart den befinner sig, vilken status den har samt vilken kvantitet den finns i, har öppen underrättelseinhämtning blomstrat på både professionell och amatörnivå. Dock, på grund av den kvantitet som denna data genereras i, kräver en helhetssammanställning en oerhörd insats.

Därav avser detta arbete ge en grund för sammanställning av materielförluster genom att tillhandahålla en automatiserad process för att extrahera data från en befintlig databas. Detta har exemplifierats genom att nyttja bildkollektioner från Oryxspioenkop, en grupp bestående av militäranalytiker som fokuserar på sammanställning av grafiskt material. Utöver detta så kompletterar processen befintliga data genom att inkludera datumet då materielen dokumenterats. Därigenom ges även en kronologisk ordning för förlusterna.

Processen visar lovande resultat och lyckas att effektivt och träffsäkert sammanställa stora mängder data. Vidare lyckas processen att överträffa sin manuella motsvarighet och visar på ett linjärt samband mellan antalet analyserade bilder och besparade mantimmar. Dock, på grund av den korta utvecklingstiden har processen fortfarande en del utvecklingsmöjlighet och förblir semiautomatisk, snarare än automatisk. Å andra sidan, eftersom processen bygger på öppen källkod, finns fortsatt möjlighet att uppdatera och modifiera processen för att passa annat källmaterial.

Slutligen, i och med den kontinuerliga utvecklingen av artificiell intelligens och artificiellt genererade bilder, lyfter studien frågan om denna typ av data kommer vara en trovärdig källa i framtida analyser av materielförluster, eller om det kommer att förvandlas till verktyg för propaganda och påverkansoperationer i ett framtida krig.

Nyckelord: Ryssland, Materielförlust, Rysk-ukrainska kriget, Metodutveckling, Maskininlärning, Tesseract OCR, Oryxspioenkop, OSINT



Special thanks to the National Defence University of Finland for hosting and supporting me during my research.

Table of contents

TABLE OF CONTENTS	3	<i>Optical Character Recognition methodology</i>	<i>15</i>
1 INTRODUCTION	5	<i>Strengths and Weaknesses.....</i>	<i>15</i>
NOMENCLATURE.....	5	<i>Assisting the machine.....</i>	<i>16</i>
Computer terminology	5	<i>Directing and controlling the result.....</i>	<i>17</i>
Research terminology.....	5	<i>Tesseract commands.....</i>	<i>17</i>
1.1 BACKGROUND.....	6	3.5 PREREQUISITES	18
1.2 PROBLEM.....	7	4 APPROACH	19
Aim	7	4.1 THE GENERAL WORK- AND DATAFLOW	19
Delimitation.....	7	4.2 ORYX SOURCE DATA	20
Research Questions	8	Image characteristic	20
1.3 PREVIOUS STUDIES.....	8	Data characteristics.....	21
2 OPEN-SOURCE DATABASES	9	Oryxspioenkop – Acquisition and Recording	21
2.1 DATABASES	9	Sorting Algorithm – Cleaning.....	22
Oryxspioenkop.....	9	Data Input – Extraction and Annotation	23
Sources for future studies.....	10	Downloading Images – Acquisition and Recording ..	24
UAwardata	10	4.3 PROCESSING METHODOLOGY	25
3 LOCATE, INTERPRET, AND UNDERSTAND THE DATA	11	Method 1 and 2 – Cleaning and Annotation	25
3.1 FRAMEWORK – BIG DATA.....	11	Method 3 – Cleaning	27
Volume, Variable and Velocity	11	Tesseract OCR – Cleaning, Annotation, and Integration	27
Big data processes.....	12	Manual Transcription – Cleaning, Extraction, Annotation, and Integration.....	27
3.2 QUALITY CONTROL.....	13	Sorting Algorithm – Cleaning and Sorting	27
DMAIC quality control	13	4.4 CALCULATION AND REPRESENTATION	29
3.3 PROCESS DEVELOPMENT AND EVALUATION	14	Data Input – Extraction and Annotation	29
Development	14	Calculation – Integration.....	29
Evaluation.....	14	Feedback – Analysis and Integration.....	29
Measuring Variables.....	14	Representation – Representation and Modelling.....	31
3.4 TESSERACT – OPTICAL CHARACTER RECOGNITION (OCR).....	15		

<i>Summary - Interpretation</i>	31	7.1 RESEARCH QUESTIONS	43
5 RESULTS AND ANALYSIS	32	7.2 FUTURE IMPLEMENTATIONS.....	43
5.1 METHOD EFFICIENCY.....	32	REFERENCES	44
<i>Sorting algorithm – Efficiency and accuracy</i>	32	ARTICLE (JOURNAL)	44
<i>Image Download</i>	33	ARTICLE (PERIODICAL)	44
<i>Pre-Processing</i>	33	PERSONAL INTERVIEW	45
<i>Manual Transcription</i>	34	REPORT.....	45
<i>Only using Tesseract</i>	34	WEB SITES	45
<i>Method 1</i>	34	APPENDIX 1 – REQUIREMENTS AND RECOMMENDATIONS	
<i>Method 2</i>	35		46
<i>Method 3</i>	35	SYSTEM REQUIREMENTS	46
<i>Method comparison</i>	35	<i>System</i>	46
<i>Pre-processing engine comparison</i>	36	SOFTWARE REQUIREMENTS	46
<i>Break-Even point</i>	37	<i>Acquisitions and recording</i>	46
5.2 RECOMMENDED METHOD	38	<i>Extraction, cleaning, and annotation</i>	46
5.3 USING THE METHOD ON SIMILAR DATABASES	39	<i>Integration, aggregation, and representation</i>	47
6 DISCUSSION AND FUTURE STUDIES	40	APPENDIX 2 – SOURCE CODE	48
6.1 STRENGTHS AND WEAKNESSES	40	APPENDIX 3 – DATA SAMPLE, ONE YEAR OF WAR	49
6.2 RESEARCH SUGGESTIONS.....	41	APPENDIX 4 – FUTURE STUDIES, DATA CONNECTIONS AND	
<i>Process improvements</i>	41	FLOWCHART	50
<i>Military technology</i>	41		
<i>Military analysis</i>	42		
7 CONCLUSIONS	43		

1 Introduction

Nomenclature

Computer terminology

Data, in the scope of this thesis, refers to the collection of digital values (bytes). Hence, data is unprocessed and requires context to convey information. In a similar manner to information, an individual piece of data is incoherent and seemingly useless. To illustrate with an example – the hexadecimal sequence [4f], [53], [49], [4e], and [54], are five pieces of data. Given a process and a context, such as Unicode Transform Format (UTF8), this set of data forms the word ‘OSINT’.

Information, on the other hand, refers to computer-processed, and structured data presented in a context. Information can be useful, but it can also be useless. In the previous illustration, the word ‘OSINT’ forms a piece of information. Given a couple more datasets, the sentence ‘OSINT is the acronym for Open-Source Intelligence’ is formed. The sentence still lacks context but can be new information depending on previous knowledge.

When combining the right pieces of information, a new context takes form, from which *intelligence* can be extracted. To further illustrate using the previous example – the information that ‘OSINT is the acronym for Open-Source Intelligence’ combined with another sequence of data that forms the sentence ‘OSINT analysis is further reviewed in this thesis’ would allow the reader to decide on whether to continue reading; or to stop.

Metadata is a collection of information within a piece of data that provides additional context. For instance, in a photo, metadata could inform what time a photo was taken, who took it, the manufacturer of the camera, the location it was taken, and camera settings such as shutter speed, focal length, etc. This data can be reviewed when

analysing the image to help understand the photos’ context.

A *Script* is a set of instructions given to the computer in a specific ‘scripting’-language that is carried out by a program, rather than directly by the computer’s processor.

An *Algorithm* is a set of instructions that a computer follows to achieve a goal, such as solving a mathematical problem or sorting large sets of data. An algorithm is comparable to a recipe in many ways – giving structure, instructions, and the chronological order required to perform a task. When used in this thesis, ‘algorithm’ will imply a script with the intent of sorting and cleaning data.

Scraping, when referred to in this thesis, will imply the act of storing data from a website in a non-intended way.

Research terminology

Approach, which can be found in section four, refers to the image-processing development.

Process, when mentioned, refers to a set of actions that need to be completed to achieve a desired result.

Method, in its turn, means the specific method or sub-method that can be used within a process to achieve the wanted result. This is further illustrated in sections four and five.

Tool implies a developed software, script or framework that can process, store, record and/or clean data required for a specific method.

1.1 Background

On the 24th of February 2022, Russia started Europe's largest war since the Second World War. The origin of this event started eight years prior, 27th of February, with the annexation of Crimea. Which since then has exponentially escalated in the period 2014 – 2022. The escalation started with pro-Russians shooting down a military transport plane in June 2014, killing 49 Ukrainian service personnel, and 298 lives were later lost in July 2014, when a Malaysia Airlines flight was shot down in eastern Ukraine.

Then, in 2015 Ukraine was plagued by several rocket attacks killing 30 and injuring many more in Mariupol. Consequently, these events caused Ukraine to gradually move- and form closer bonds with Europe and NATO during 2016 – 2020, resulting in Ukraine's new National Security Strategy 2020 – striving to become an official NATO country.¹

In early April 2021, Russia started a substantial military exercise in the neighbouring regions of Ukraine. 14th of April 2021 Ukraine's defence minister states that over 100,000 Russian troops were bordering Ukraine.² Later that year, in November 2021, President Zelenskyy confirms that large masses of Russian soldiers are still close to the Ukrainian border.

The following month Russia makes demands that denied NATO military access to eastern Europe as well as Ukraine to be accepted as a NATO member state. Later, in early February 2022, Russia

launches a large joint military exercise between Russia and Belarus – close to the Belarusian/Ukrainian border.³

Altogether these events resulted in President Putin recognizing the independence of the self-proclaimed *Luhansk People's Republic* and *Donetsk People's Republic*, on the 21st of February 2022, and commanded the Russian military into these areas for what he calls “peacekeeping duties”. Three days later, 24th of February 2022, Russian state television announces that a “special military operation” will be carried out in Ukraine, which marks the beginning of the Russian invasion of Ukraine.⁴

At the time of writing, this war has elapsed over a year and claimed hundreds of thousands of lives, millions of refugees, and billions of dollars.⁵

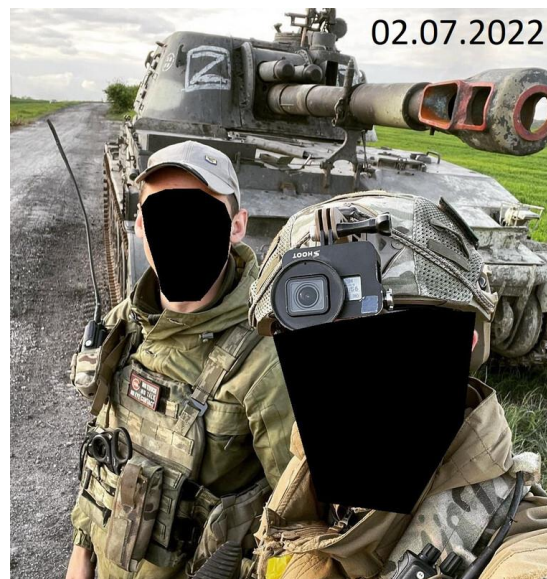


Figure 1, Soldiers posing in front of a Russian 152mm 2S3(M) Akatsiya, categorized by Oryx.

Retrieved from: <https://postimg.cc/HVKJcnnB>

During the war, the Ukrainian civilian population and its armed forces have proven themselves to be capable intelligence gatherers for the rest of the world's interested amateur and professional military analysts. By using social media such as Telegram, Twitter, TikTok, YouTube and many more, the Ukrainians have gathered thousands of images of battle-scarred Russian and Ukrainian equipment. For security reasons, many of these images have had their metadata containing information such as date taken, geolocation, camera model, etc, wiped off. However, a vast majority of these images have an embedded date

¹ Walker, 2023, pp. 15–16.

² Walker, 2023, pp. 15–16.

³ Walker, 2023, p. 27.

⁴ Walker, 2023, pp. 27, 30.

⁵ BBC, Ukraine war: US estimates 200,000 military casualties on all sides, 2022.

within the image (Figure 1). These images then contribute to the wars' big data.

The concept of 'big data' is the acquisition, processing, representation, analysis, and lastly interpretation of large amounts of digital data to support decision-making.

As a result of the growth of social media, the 'Internet of Things', the improvement of digital storage capacity, and smart devices being accessible by the general public – generated data is growing exponentially. Every social media post, YouTube video, cloud-stored security camera footage, fridge sensor, and so forth, leave its unique digital footprint and is stored on servers worldwide. Hence, the possibility to use and interpret this data creates a unique opportunity to gain intelligence which can be acted upon.⁶

1.2 Problem

Since neither side of the war wants to release the official numbers of their casualties and equipment loss for strategic and operational security reasons – the media throws itself over any numeric value released by experts or intelligence agencies. These numbers are further skewed by Russian and Ukrainian propaganda, with a tendency to portray the opposing side's losses in their favour.

At the same time, there exists plenty of well-established sources that gather data through Open-Source Intelligence. Due to the continuation of the war and no indication of its end, there are and will be plenty of research opportunities in gathering and analysing equipment loss.

However, since analysts have varying methods, niches, and use different data structures – these sources include a variety of information. Consequently, the information is continuously generated and updated, resulting in the manual labour needed to compile and understand the

information becoming more and more time-consuming.

Aim

The aim of this study is, hence, to explore an automatic data collection process, specialized in photographic material and data from one or more sources, that facilitates Open-Source Intelligence analysis. The process is exemplified with a database containing information from the current Russo-Ukrainian war but can be used, depending on implementation, to compile different sources.

Furthermore, the study aims to contribute to future analysis of Russian equipment losses – when a larger portion of the equipment is documented, collected, sorted, and accessible.

Delimitation

This study is aimed at exploring data collection and analysis, rather than military intelligence analysis. The result of the proposed data collection method is exemplified in Appendix 2 and 3, but the aim of the study is not to analyse Russian equipment loss.

To further delimit the method, the source material used is exclusively images (PNG or JPEG) that are directly downloadable. The included images can nonetheless originate from screenshots of a video, social media post, or other graphical media.

Due to the short time conducting this research, no further time was spent on developing a scraping method for sites such as Twitter or Facebook, which would be highly resourceful sites to use. Recommendations for future studies are also included in the final sections.

All source code and assets can be found in Appendix 2 – Source Code, the code is commented to explain what each line of code does. In section four, the basic principles of the method are described in further detail. However, to keep the

⁶ Gandomi & Haider, 2015, p. 138.

report concise, section four does not go into detail explaining what each step of code does.

Research Questions

How is Oryxspioenkop image collection a viable source to use for compiling equipment losses, and what information is, directly and indirectly, included on the website?

How would it be possible to automatically compile the data presented by Oryxspioenkop – utilizing both the directly included information and the embedded text information within the images?

How accurate and effective is the process, and at what data volume is the break-even point for automatization versus manual labour?

1.3 Previous studies

The subject of using a machine to find text in images is an area that has been well-studied since the early 1990s. Image processing and optical character recognition (OCR) are two fields that are closely related. Numerous studies have been conducted to improve the accuracy and efficiency of these algorithms.

Three fields that have received special attention are document digitalization, transcription of handwriting, and number plate detection.

The former is frequently used to efficiently transcribe newspapers, historical documents, and other paper-based texts to a digitalized world. When dealing with images that contain a variety of colours and noise, it has been shown that using a binarization technique (further explained in section three) is an effective way to separate text from unwanted artefacts in images, consequently enhancing the results yielded by an OCR engine.⁷

Transcription of handwriting can be used to analyse individual styles, separate similar characters, and automate processes such as mail sorting, transcription of handwritten text and bank checks.⁸ In this field, even though it is not intended for it, the Tesseract engine (further explained in section three) has proved its potency as an all-around optical character recognizer.⁹

The latter field, number plate detection, is an important asset to law enforcement, toll collection, and traffic management. By using a machine rather than manual labour, the engine can extract the information within a number plate and match it against the authority's databases. In this area, the Tesseract engine has also shown its effectiveness when combined with binarization and other pre-processing methods – reaching a 92% precision rate distinguishing text within a detail-rich image.¹⁰

⁷ Ntogas & Ventzas, 2008, p. 41.

⁸ Zhang, et al., 2017.

⁹ Mursari & Wibowo, 2021, p. 177.

¹⁰ Poorani, et al., 2022, p. 136.

2 Open-Source Databases

Narrowing the study to one database is a liability for multiple reasons. First and foremost, if the database is not continuously updated, the method is quickly rendered obsolete. Then, if the database contains incorrect information, the method produces inaccurate information. Further, if the database is not considered trustworthy, the method is unreliable to use. Finally, the information within the database needs to be accessible. Therefore, it is vital to know how often a database is updated, if the data is reliable, and if it is accessible.

To evaluate different open sources for intelligence gathering, Lieutenant Colonel Joakim Paasikivi – one of Sweden's Russo-Ukrainian war experts, participates in assessing and discussing the sources used for this method.

With a military background starting in 1979 in Finland, Lieutenant Colonel Paasikivi has worked his way up from the reconnaissance troops to the highest military intelligence service in Sweden.

His merits also include working at the Swedish Armed Forces headquarters in the department of strategic management, 12 years at the Swedish Military Intelligence- and Security Service, where he was chief for the Balkan-, and the Mali desks (desk – temporary group that work with intelligence from a specific geographical area), as well as intelligence head teacher at the Baltic Defence College. At the time of writing, Paasikivi works at the Swedish Defence University's strategy department.¹¹

With accumulated professional expertise of 40 years, whereof 12 years at the highest military intelligence institution in Sweden – working with intelligence ranging from tactical to strategical,

Lieutenant Colonel Paasikivi has good insight into intelligence gathering and the Russo-Ukrainian war. With his expertise, Paasikivi helps by providing a critical view of the databases discussed in the following subsection.

2.1 Databases

To streamline and exemplify the method described in section four, the primary database used to collect the data is the Dutch equipment compilation and image collection website Oryxspioenkop.

Oryxspioenkop

Oryxspioenkop, or shortened, Oryx is an open-source and up-to-date website gathering data on Russian and Ukrainian equipment losses. The group does more than just collect data on the Russian invasion, but will when mentioned in this study, refer to their image collection.

The Oryx group is an independent analysis group consisting of a variety of different nationalities and has been cited by news sites such as BBC¹², CNN¹³, Forbes¹⁴ and The Guardian¹⁵ to mention a few. The group has been collecting and analysing conflicts around the world since 2013 but has, according to Google Trends¹⁶, received substantially more attraction since the Russian invasion of Ukraine.

In the context of this study, only the image collection of Russian equipment losses is being used. However, the method provided in section four can also be used, with a little tweaking, for other URL-image sources.

Furthermore, Paasikivi mentions that he has confidence in the material provided by Oryx and credits the site for professionally providing data in their niche area – image collection. He states that

¹¹ Paasikivi, 2023.

¹² BBC, Ukraine conflict: Why is Russia losing so many tanks?, 2022.

¹³ London, 2022.

¹⁴ Hambling, 2022.

¹⁵ Sabbagh, 2022.

¹⁶ Google Trends - Oryx, 2023.

he has been in contact with the sites' material since the recent conflict between Armenia and Azerbaijan.

However, in a discussion with the Lieutenant Colonel, it is also mentioned that there is a weakness in their line of work – that the continuous data stream needs to be collected, sorted, identified, and uploaded. A process that requires time and competence. He mentions that the contributors to the website are scattered across the globe, and some of them have never met. Hence, depending on how long the war persists and other conflicts that require attention, the group risks either burning out from the workload or needing to change their focus, resulting in the data becoming obsolete.

But on the other hand, Paasikivi finds their work to be well established and will, if the group can handle the workload, result in impactful data that can be used to understand the Russo-Ukrainian war in the future.

The content on Oryx is categorized into different vehicle types (for example Tanks, Radars) and subcategorized into vehicle models (for example T-72AV, PPRU-1 M), the unidentifiable vehicles are then categorized as well as possible (for example Unknown T-72, Unknown Radar).

The group states that the list only includes equipment of which there are photo- or video graphic evidence, and it does not include equipment such as small arms, missiles, mortars, loitering munitions, or civilian vehicles. Hence, the true amount of equipment loss is higher. Furthermore, if the origin of the vehicle cannot be established, it is not included on the list.

Lastly, the Oryx group claim that “All possible effort has gone into avoiding duplicate entries and discerning the status of equipment between captured or abandoned.”¹⁷

Sources for future studies

Since there is a plethora of sites and applications that give the user access to information concerning the Russian invasion, there is great potential in combining these sources to compile the information into open-source intelligence. For instance, Twitter, Telegram, Deepstate, Liveuamap, Reddit, UAwardata, and many more. Many of these sites focus on geo-data, such as troop, equipment, and vehicle sightings.

In the infancy of this study, the aim was to create a methodology that allows the user to combine different sources to cross-reference equipment loss to a certain unit. However, due to lack of time – the work of creating the correct cross-referencing database had to be left uncompleted. Nevertheless, the unfinished framework and proposed database will be included in the repository in Appendix 2 and exemplified in Appendix 4.

UAwardata

UAwardata is in similarity to Oryx an open-source website that collects data on the Russian invasion of Ukraine. UAwardata is created- and contributed to by Henry Schlottman, an independent open-source intelligence/operations analyst.

In contrast to Oryx, UAwardata is a database that collects information which is composed of operational/strategical intelligence concerning troop movements and events. The primary data it provides is geographical and is displayed on an interactive map with a time-ruler allowing the user to see the invasion from a top-down helicopter perspective. It also provides data on the location of different units, and to some extent, their equipment.

Depending on the type of information being gathered, and what intelligence is sought after, Paasikivi is sceptical about the usage of geographical map data for intelligence work. He adds that using such data can be helpful when

¹⁷ Mitzer & Janovsky, 2023.

trying to understand the bigger picture but is, in itself, not as valuable. He mentions that a fatal weakness is such data fails to provide context and details. He exemplifies this by saying that the decision-maker doesn't have to know, for example, that they are facing the 61st Separate Naval Infantry Brigade. However, what is of value for the Ukrainian commander is to know their opponents' battle worthiness, their equipment and what resistance they can offer.

But on the other hand, the data that the website provides is valuable for the recording and future analyses of the war. As well as that it can provide other pieces of data, that provide context for other pieces of information.

The open-source data that is provided by UAwardata gives dates, unit names coordinate in longitude and latitude, Schlattman's analysis, as well as to some extent the units' equipment and sources. If complemented with a database containing Russian units and their equipment, and then cross-referenced against the equipment loss documented by Oryxspioenkop, it would be possible to deduce, with certain plausibility, the equipment losses of different Russian units during the war.

In Appendix 4 possible connections between Oryx and UAwardata are exemplified, which is also mentioned as a future research suggestion in Section 7.

3 Locate, interpret, and understand the data

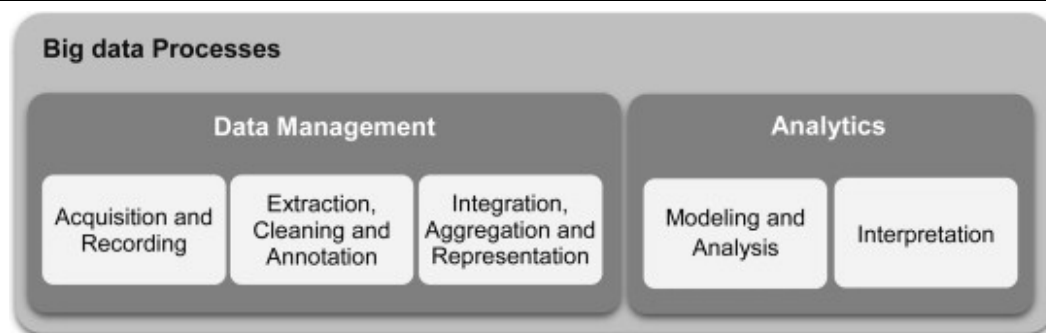


Figure 2, Model for Big data Processes, Gandomi and Haider,

3.1 Framework – Big data

Volume, Variable and Velocity

Big data is generally characterized by being produced at a high rate as well as the variability of its' size, origin, format, and structure. Depending on the dataset these factors can differ – therefore it is not possible to specify a certain range of data structures or amounts of bytes which are defined as big data.¹⁸

Volume

Depending on the system, data management methods, file structure, and compression – data can range from a few bytes to petabytes (10^{15} bytes) or more. Rather than focusing on byte size, big data is better understood by the magnitude of data.¹⁹ For instance, 20 million basic text documents, each consisting of one unique 8-character word (using capital A-Z and numbers 0-9), would create a dataset consisting of around 300 megabytes. While at the same time, a minute or two of high-quality mobile camera footage is the same size.

¹⁸ Gandomi & Haider, 2015, p. 138.

¹⁹ Gandomi & Haider, 2015, p. 138.

In the previous example, decoding and understanding the 8-character dataset would require a clever algorithm and a decent amount of computing power, while the video could easily be processed by one person in a matter of minutes.

Variety

Depending on how it is sorted and what data is interpretable for the machine, data can either be categorized as structured, semi-structured, or unstructured.

Structured data contributed, in the year 2010, to barely 5% of all existing data, and was primarily found in spreadsheets and structured databases. Its structured nature makes it easy to handle, interpret and analyse for both man and machine.²⁰

Unstructured data, on the other hand, is data that can be found in images, text, video, and audio.²¹ For a computer, processing such data is simple. However, for it to depict, interpret, and analyse the data is substantially trickier. Vice versa, the complex and detail-rich unstructured data is easy for the human brain to understand but is, on the other hand, demanding to process in large quantities. To illustrate, after watching a two-hour movie, a human can, within minutes, summarize the whole movie and describe plot twists, mood, script quality, and similarity to other movies they watched. The machine, on the other hand, can within minutes split the two hours of motion picture into 172,800 individual images and sort them in order of brightness and hue.

The third category – semi-structured data, falls in between structured and unstructured data, and its' boundaries are more loosely defined. Data such as text written in programming languages like Extensible Markup Language (XML) or Hyper Text Markup Language (HTML) is considered

semi-structured.²² Just like any language, for an untrained eye, both Markup Languages look like unstructured nonsense. However, hidden beneath is a structure of tags (Figure 3) that categorize the data within the tags.²³

```
<h3 style="text-align: left;">
    <span style="color: red;">
        Russia - 9441, of which:
        destroyed: 6007,
        damaged: 270,
        abandoned: 376,
        captured: 2788
    <br></span>
</h3>
```

Figure 3, Example of Hyper Text Markup Language. Bright colour indicates HTML code.

Velocity

The final dimension of Big Data is velocity – the speed at which it is generated, versus the speed it can be processed and acted upon.²⁴ Some sources produce a constant stream of data, such as security cameras. While others, such as social media, have distinct highs and lows.

Big data processes

What is the common denominator for all data is that if it is not acted on in time – it is useless. Therefore, big data does not serve a purpose if it is not recorded, extracted, integrated, analysed, and interpreted in time. Since time is the limiting factor, and big data is characterized by its magnitude, variable structure, and the speed it is generated – processing big data requires a simple, yet ingenious design. Gandini and Haider illustrate the basic principles of big data processing (Figure 2). If the process is fast and efficient the interpretation enables accurate decision-making in real-time.²⁵

²⁰ Gandomi & Haider, 2015, p. 138.

²¹ Ibid., p. 138.

²² Ibid., p. 138.

²³ Ibid., p. 138.

²⁴ Ibid., p. 138.

²⁵ Ibid., p. 141.

The process proposed by Gandini and Haider is intuitive for both man and machine. The first phase, *Data Management*, consists of accessing and acquiring the data, then extracting, cleaning, and sorting valuable information. Lastly integrating, compiling, and representing the data. Depending on the original data structure, this process can be more or less demanding. Under these circumstances the data transitions from a set of incoherent bytes – to well-structured and comprehensible information.

The organized data is then fed into the second phase, *Analytics*, where modelling and analytical techniques are used to extract intelligence from the data. Finally, information is interpreted to form intelligence to support decision-making.

3.2 Quality control

In an automated process, it is vital that the output keeps a high standard to reduce the need for post-processing. To improve a process and reduce waste, several methods exist that could be implemented. For instance – the *lean method*, six

sigma, a *plan-do-check-act-cycle*, or *DMAIC* (*Define, Measure, Analyse, Improve, Control*).²⁶

DMAIC quality control

The method described in section four is improved by using the iterative DMAIC process since it is found to suit the method-improvement process well (Figure 4).

The first step – *defining*, is the process of identifying the problem, and if there is potential and actual benefit in solving the problem. Then, *measuring* the process it seeks to improve, and collect quantitative data, to understand where the break-even point is. The measured data can then be *analysed* to understand the relationship between the process and the problem, as well as potential defects, quality problems and other waste that motivate problem-solving. After that, the defined and analysed problem in combination with the measured data can be used to *improve* the process. Lastly, to identify the further needs to improve the process, the output is *controlled*. If there is no more need for improvements, the cycle ends. If there is further room for improvement, the process begins anew and continues to do so until the desired process is achieved.

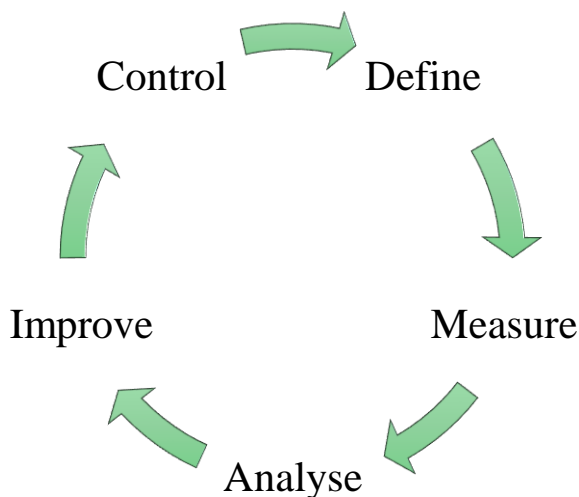


Figure 4, Illustration of DMAIC-cycle.

²⁶ Montgomery & Woodall, 2008, p. 335.

3.3 Process Development and Evaluation

Development

The general method development is described in further detail in section four and is written in chronological order. Due to lack of time, and the author's previous expertise using a certain set of tools, the development process is greatly influenced by the author's work methodology, preferred programming languages, and software. Section five and Appendix 1 showcase the used tools, as well as suggest alternative tools that could be used to solve the problem. Section four goes into further detail describing the different information types, the flow of information, subprocesses, as well as strengths and weaknesses of the process. Section six describes the strengths and weaknesses of the methodology in additional detail. During the time of development, the process went through different iterations to find a suitable and effective solution to the problem. The solution showcased in section four is *one* possible way to solve the problem. The expression – if your only tool is a hammer every problem will look like a nail, is applicable to understand the weakness of this methodology.

Evaluation

Section five goes into further detail explaining the results yielded by the different iterations of the process. To evaluate effectiveness, progress, and areas of improvement the DMAIC quality control cycle is used throughout sections four and five.

The first step is to *define* the problem. From a design perspective, the main goal of the process is

to extract and compile information as fast as possible. Hence, a critical factor to look at is the required processing time. Another factor is the accuracy of the process. These two factors constitute the effectiveness of the process. Since the process is being compared against its' manual counterpart, as long as it is more effective, it achieves the set goal. The manual counterpart is benchmarked in section five.

The second step is to assess and analyse the quantitative values of the process. This is achieved in section five where the result of the different iterations is *measured* and then *analysed* in order to understand possible improvements.

The third step consists of *improving* the process. During the development process, the improvement phase is continuous. After improving any iteration of the process, the yielded result is *controlled* in order to assess the effectivity of the new iteration. The results from the controlled process define the *measurement* in the next iteration, which in its turn will be *analysed*, and *improved* if needed.

Measuring Variables

Due to the defined problem, and how the process is developed – it yields a set of variables that can be used to assess the effectivity of any given iteration. The used variables are limited to: the number of dates found, the time taken to compile the dates, and the accuracy – in other words 'how many dates were found, how long did it take, and how many could be found?'. Section four goes into further detail describing how, when, and where the different variables are extracted. Section five in its turn describes the results yielded by different iterations of the process.

3.4 Tesseract – Optical Character Recognition (OCR)

Tesseract is an open-source Optical Character Recognition (OCR) engine that utilizes the principles of machine learning to solve a specific set of tasks. Tesseract is generally recognized as one of the leading machine learning software capable of handling character recognition. The engine started as a PhD project in 1984 and was under development until 1995 when it was publicly revealed and outperformed many of the commercial engines. Later, in 2005, Tesseract was released for open source and its development is now contributed to, and sponsored by Google Inc.²⁷ Since its previous update, 4.0, Tesseract's character recognition engine is based on a Long Short-Term Memory (LSTM) neural network. To simplify – a machine learning methodology meant to imitate human learning, capable of creating connections and conclusions between data more accurately.²⁸

Optical Character Recognition methodology

Optical character recognition is required when trying to extract text embedded inside an image with a computer. Unlike a human looking at a computer screen, where a text document containing text is nearly identical to an image containing text, the machine cannot “see” the information stored in the image or text document.

The methodology required for a machine to optically recognize characters is a complex mathematical process. Tesseract achieves this with a set of algorithms that find lines, patterns, shapes, letters, words, and spacing in a text. It then compares these to similar characters it has encountered during training and estimates what character it has located.²⁹

Strengths and Weaknesses

Generally, tasks that are suitable for a machine to perform are categorized into *dirty*, *dull*, *dangerous*, or *difficult*. Transcribing large sets of data from one media (in this case – text embedded in a couple of thousand images), to another (Unicode text), is a tedious and dull task requiring many man-hours of work. Hence, the automatization of such a process with machine learning is an impactful contribution.

However, complex imagery containing a variety of colours and artefacts that resembles text – proves to be a tricky adversary for the machine (Figure 5).

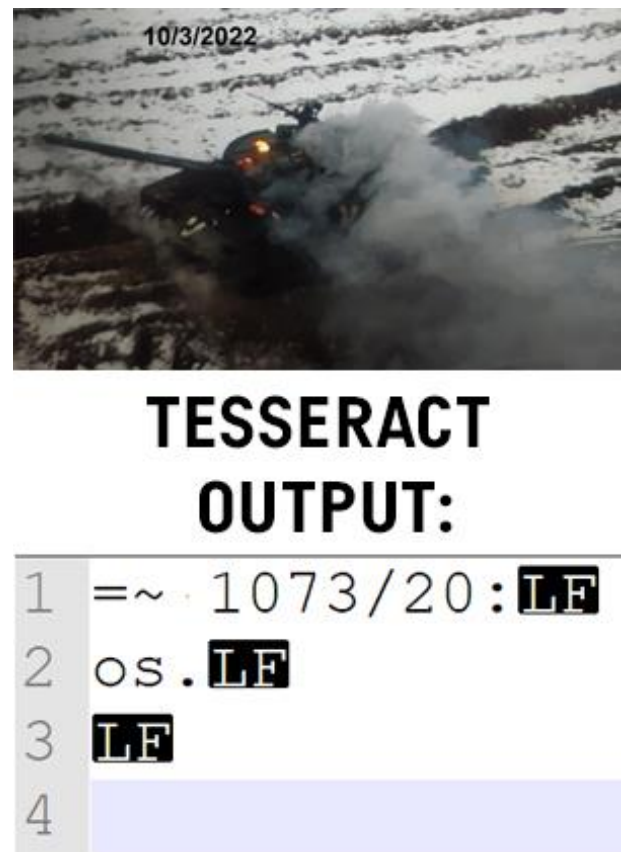


Figure 5, Upper – A colour image depicting a destroyed Russian T-80BV. Lower – what the OCR, using default settings, can ‘see’ within the picture. Vehicle identified and categorized by Oryx.

Retrieved from: <https://postimg.cc/1gF6Z9qf>

²⁷ Smith, 2007, p. 629.

²⁸ Tesseract User Manual, 2023.

²⁹ Smith, 2007, pp. 630–633.

To counter this, it is vital to direct and assist the machine's work.

Assisting the machine

Several methods can be implemented to help locate and identify characters – to improve the results given by Tesseract. For instance – rescaling, binarization, noise removal, dilation, rotation correction, adding borders, or correcting transparency (alpha channel).³⁰ The method described in section four relies on binarization.

Binarization

This is the process of transforming a range of values into a two-digit scale. For instance, a colour image turned into a black-and-white image.³¹ By using a threshold function, colours that are above a certain colour value turn into black, and all other nuances to white (Figure 6).

The threshold function illustrated in Figure 6 converts each pixel to either black or white, depending on the pixels' greyscale value. The normal greyscale used in editing software is based on 256 shades of grey, where 0 is represented by black and 255 by white. By using this mechanic certain features are enhanced while others are diminished, for instance, the date '10/3/2022' in Figure 6 is better visible when using threshold 50 than 150 or the original image, which helps the optical character recognizer.³²



Figure 6, Comparison between: Upper – full-colour image. Middle and Lower – What the image looks like when binarized with a threshold value of 150 or 50.

³⁰ Tesseract User Manual, 2023.

³¹ Binarization, 2023.

³² Binarization, 2023.

To illustrate, when running Tesseract on the same image as in Figure 5, but pre-processing it with a threshold value of 50, before using Tesseract, a fairly accurate result is yielded (Figure 7).

Directing and controlling the result

To further improve the output from Tesseract, it is beneficial to understand the basic settings that can be used to manipulate the optical character recognition algorithm.

The normal set of commands that can be given to Tesseract regulates what language it tries to recognize, output format, OCR engine mode, page

segmentation mode, or whitelisted characters.³³ Since Tesseract is operated with console commands, rather than a graphical user interface (GUI), this gives Tesseract an intimidating look for users who are unfamiliar with navigating a program using command-lines. However, this allows the user to customize the output to a high degree. The basic commands needed to launch Tesseract will be described below, for more detailed descriptions of other setting the reader is referred to the Tesseract manual.³⁴

Tesseract commands

The first basic commands to launch Tesseract OCR is illustrated in Figure 8. This command uses the default settings to input an image named 'Image_Name', formatted in jpg, and outputs the result to a text file named 'Output_File'. Different file formats are supported by Tesseract, to analyse a large set of images, the input file (yellow text in Figure 8) can be specified as a text file containing the names of all images to be analysed.

Page Segmentation Mode (PSM), illustrated in Figure 9, allows the user to force Tesseract into a specific reading pattern. There are 13 different modes, where the default settings use mode three, automatic page segmentation. Mode four assumes a single column of text with variable sizes. For more information about page segmentation, use the command:

```
Tesseract --help-psm
```

If the source images all are the same resolution, it is beneficial to use the dots per inch (dpi) command (Figure 10). By default, Tesseract adjusts this setting automatically, but when specifying the dpi – Tesseract has a better chance of reading the image.

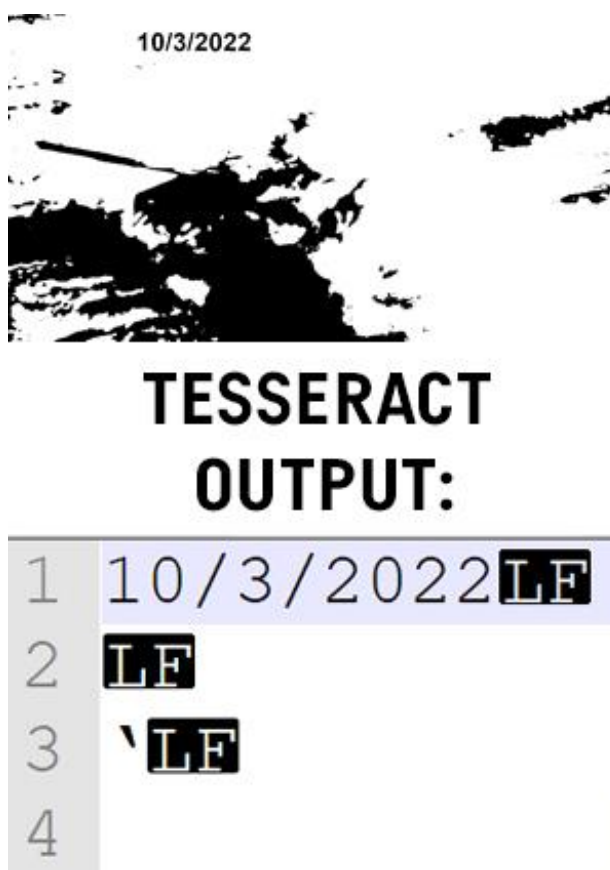


Figure 7, Upper – Threshold 50 pre-processed image. Lower – Tesseract output, with default settings. Character 'LF' indicate new lines of text.

³³ Tesseract User Manual, 2023.

³⁴ Tesseract User Manual, 2023.

```
C:\User>Tesseract Image_Name.jpg Output_File
```

Figure 8, Basic console command to launch Tesseract OCR [red text] on an image named Image_Name.jpg [yellow text] and output it into a file called Output_File [green text].

```
C:\User>Tesseract Image_Name.jpg Output_File  
--psm 4
```

Figure 9, Console command to launch Tesseract OCR using page segmentation mode four [turquoise text].

```
C:\User>Tesseract Image_Name.jpg Output_File  
--psm 4 --dpi 300
```

Figure 10, Console command to launch Tesseract specifying image resolution to 300 dots per inch [pink text].

```
C:\User>Tesseract Image_Name.jpg Output_File  
--psm 4 --dpi 300 -c tessedit_char_whitelist=0123456789./-\\
```

Figure 11, Console command to launch Tesseract whitelisting digits and characters – dot, slash, minus and backslash [white text].

```
C:\User>Tesseract Image_Name.jpg Output_File  
--psm 4 --dpi 300 -c tessedit_char_whitelist=0123456789./-\\ --oem 3
```

Figure 12, Console command to launch Tesseract with Optical Character Recognition Mode three [red text].

To further narrow down possible miss-readings from Tesseract, the user can whitelist characters, hence Tesseract will ignore any character not included on the whitelist. Figure 11 exemplifies this by whitelisting all digits ranging from 0-9, dots, slashes, minuses, and backslashes.

Finally, Figure 12 demonstrates the usage of the Optical character recognition Engine Mode (OEM). By default, Tesseract uses mode three, which is based on what is available. For more information about engine modes, use the command:

```
Tesseract --help-oem
```

3.5 Prerequisites

As the general theme of the method described in the following section is data acquisition, cleaning, sorting, and automatization – there is a specific set of computer knowledge, software, and hardware required to practice the method efficiently. Software and hardware requirements are described in Appendix 1. These requirements are dimensioned by the minimum requirements to run the tools.

Regarding the user's skill set – the more previous knowledge, the faster the method will yield results, but with a little curiosity and patience, there are no limitations.

4 Approach

The following section will explain the method development approach and principles the Microsoft Excel framework is built on.

4.1 The general work- and dataflow

The complete process that is explored in section four is illustrated in Figure 13. Later, when exemplifying specific sub-processes, the described process is highlighted (for example, see Figure 14). White background indicated that the process is tied to data management, while grey background is tied to the analysis of data.

The process starts at the top of the figure. Depending on the database, the second step will either be cleaning and organizing the source data or if the database natively supports downloading images the process continues directly to the processing phases (illustrated with a dashed line).

The point of the download phase is first and foremost to allow Tesseract to have data to work with – since the program does not natively analyse online resources. Secondly to secure the data, in case the source website goes offline or in other ways gets outdated, updated, or manipulated – allowing the user constant access to the same dataset.

After having downloaded the images, they can either be manually transcribed or fed into the Tesseract OCR Engine. Manual transcription implies that each image is looked at by a human, and then the date is manually written down on paper or computer.

The sorting algorithms will later be described in further detail. These algorithms have the purpose of structuring unstructured or semi-structured data, such as the output from Tesseract and the website HTML code.

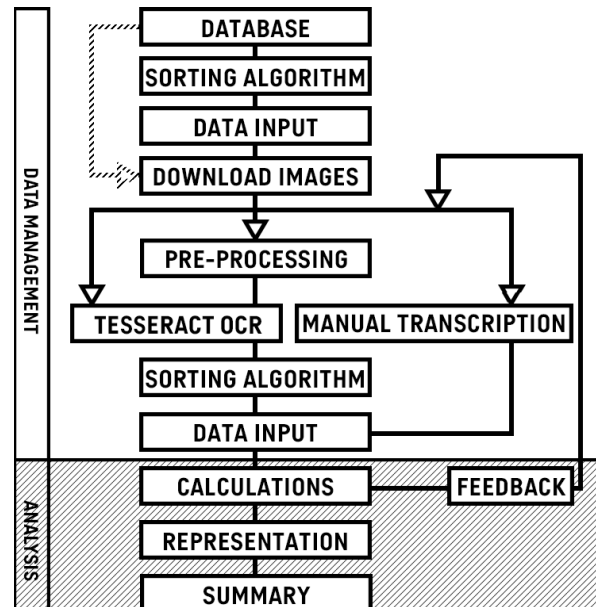


Figure 13, The complete data process explored in section 4. Arrows indicate multiple selections. General dataflow is top to bottom.

The two ‘data input’ steps indicate that the data needs to be stored temporarily or permanently, for the workflow to be efficient. In the method described in this section, all data input is done in Microsoft Excel. Allowing the software to sort, clean, store, and calculate.

When the data is cleaned, sorted, and stored, it forms information that can be calculated. After the calculating step, it is possible to interpret and analyse Tesseract’s accuracy. Hence allowing images to be fed back to the pre-processing phase, in case Tesseract failed to find a date within them. Since there is a variety between format, colour, background, size, and location of the dates within the images (Figure 15), pre-processing using one setting will not be able to distinguish the text in all images.

After having achieved a satisfactory result either using Tesseract, manual transcription, or combining the two, the data can be represented and summarized, enabling further analysis.

4.2 Oryx source data

Image characteristic

Looking at the images included in Oryx the general pattern is that most images have a date within them, and most of the dates are coloured black. In hindsight, roughly 97% of the images contain a date (see section five). The format of the images differs significantly. Some images are high quality (example 1220 x 800 pixels), while others are lower quality (example 320 x 187 pixels). Consequently, Tesseract must work with a wide range of resolutions, subsequently resulting in the dates within the image also varying.

Furthermore, the shape, size, position, and colour of the date vary. To illustrate, Figure 15 showcases three different variations of dates that can be found within the images.

The upper image depicts the date '13.11.2022' against a light blue sky. Directly under it is the product when using a threshold value of 37. The middle image contains the date '28/02/2022', using slashes '/' instead of dots '.' as a separator. The date is hard to detect – even for the human eye. Below it is the output when using a threshold value of 10, giving a clearer view of the date. The final image has the date 23.02.2023 embedded within it. The white date requires a threshold value of 225 to distinguish the text from the background.

In conclusion, these three threshold values capture three unique styles of dates within the image. However, if used incorrectly, for instance using a high threshold value on the two upper images or using a low value on the bottom image – will not yield any result (for example, see Figure 6).

Visually inspecting the dates embedded within the images, it is evident that a vast majority of them are either formatted using dots, slashes, backslashes, or minus signs as separators. Additionally, most of them are also formatted in the DD.MM.YYYY standard.

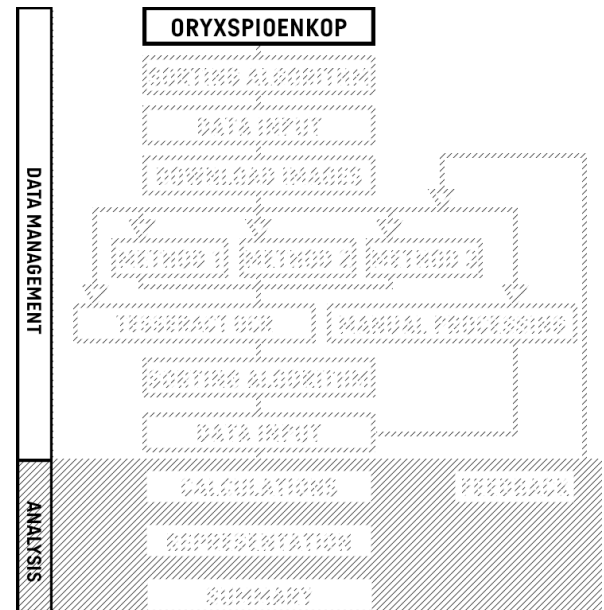


Figure 14, Method step 1 – Fetching data from Oryxspioenkop. The figure contains the entire data management and analysis process, the sub-process currently in focus is highlighted.



Figure 15, Three images with varying characteristics. Each image is split into two, original vs. binarized using a Threshold function.

Upper image - Date with sky as background. Black text. Dots as separator. Below it – Result with Threshold value of 37.

Middle image - Date with dense forest as background. Black text. Slashes as separator. Below it – Result with Threshold value of 10.

Bottom image - Date with muddy ground as background. White text. Dots as spacing. Below it – Result with Threshold value of 225.


>  50 T-62M: (1, destroyed) (2, destroyed) (3, destroyed) (4, destroyed) (5, destroyed) (6, destroyed) (7, destroyed) (8, destroyed) (9, destroyed) (10, destroyed) (11, destroyed) (12, destroyed) (13, destroyed) (14, destroyed) (15, destroyed) (16, destroyed) (17, destroyed) (18, destroyed) (19, destroyed) (20, destroyed) (21, destroyed) (22, destroyed) (23, destroyed) (24, destroyed) (25, destroyed) (26, destroyed) (27, destroyed) (28, destroyed) (29, destroyed) (30, destroyed) (31, destroyed) (32, destroyed) (33, destroyed) (34, destroyed) (35, destroyed) (36, destroyed) (37, destroyed) (38, destroyed) (39, destroyed) (40, destroyed) (41, destroyed) (42, destroyed) (43, destroyed) (44, destroyed) (45, destroyed) (46, destroyed) (47, destroyed) (48, destroyed) (49, destroyed) (50, destroyed)

Figure 16, Screenshot from the Oryxspioenkop website depicting several hyperlinks leading to images of destroyed Russian T-62M tanks.

Data characteristics

Continuing by looking at the data within the website – the first problem that needs to be assessed is what structure it is stored in.

By taking a quick look at what hides directly beneath the surface of Oryxspioenkop, one can see that the front end of the website is composed of a collection of hyperlinks that allow the user to navigate to either an image or Twitter post by clicking at the status of a vehicle (Figure 16). This is achieved by linking the status text, for example ‘(2, destroyed)’ to a Uniform Resource Location (URL) link within the code (Figure 17). Looking at the code in Figure 17, it is possible to see certain patterns as well as identify useful information, such as – vehicle type ‘T-62M’, number of vehicles ‘50’, source ‘https://...’, and status ‘destroyed’.

Consequently, the data is semi-structured and, in contrast to structured data, requires certain processing before being useful. Looking at the data in Figure 17, the manual process to clean and sort these 20 lines of code is fast work. However, looking at all the data on Oryx, when structured similarly to that of Figure 17, adds up to over 15 thousand lines of code. Hence, sorting and cleaning the entire site requires a different approach.

Oryxspioenkop – Acquisition and Recording

In theory, the Oryx scraping method is to store all, or parts of the code that the web browser is given by the server hosting Oryxspioenkop. The code can easily be displayed by using the ‘inspect’ element

that most web browsers are natively equipped with. When the code is displayed (Figure 17), it is just a matter of copying and pasting the code into a text document to save it.

Technically, however, scraping images from Oryx can be made simpler than the previously described method. For instance, by using a pre-developed addon, software, website, or script that downloads all images directly from the website.

However, using such a method would undermine the work done by Oryx in categorizing and labelling the equipment. In other words, it would ignore information such as equipment type, number, and status that can only be found within the semi-structured code.

```
</li><li>&nbsp;50 T-62M:&nbsp;<a  
href="https://twitter.com/UAWeapons/status/154479  
7070883684353">(1, destroyed)</a>&nbsp;<a  
href="https://i.postimg.cc/X72RXDmM/1024-t62m-  
destr-11-01-23.jpg">(2, destroyed)</a>&nbsp;<a  
href="https://twitter.com/UAWeapons/status/157267  
2235726573568">(3, destroyed)</a>&nbsp;<a  
href="https://twitter.com/UAWeapons/status/158190  
8407967092738">(4, destroyed)</a>&nbsp;<a  
href="https://i.postimg.cc/Gt0Mw2m7/1010-t62m-  
destr.jpg">(5, destroyed)</a>&nbsp;<a  
href="https://twitter.com/UAWeapons/status/159047  
4674999558145">(6, destroyed)</a>&nbsp;<a  
href="https://i.postimg.cc/fy6S2qj0/1010-2x-t62m-
```

Figure 17, Parts of the corresponding HTML code to Figure 16. Fetched from Oryxspioenkop using the inspect element on the Web browser.

Sorting Algorithm – *Cleaning*

The sorting algorithm (Figure 18) is designed to clean HTML code in the specific format that Oryx organizes their websites. The source code can be found in Appendix 2 and is generalized to work even though parts of Oryx are written in different coding styles (presumably since there are multiple contributors).

The algorithm relies on the usage of regular expressions. There is a multitude of different regular expression languages, what they all have in common is that their primary task is pattern recognition and string replacement, hence they are a perfect tool for creating a cleaning algorithm.

Both sorting algorithms are based on the Perl Compatible Regular Expression (PCRE) library and are executed through the Python coding language. To further simplify and control the process, the sorting algorithm is launched by using the text editing software Notepad++ (NPP) as a framework. By using Notepad++, a graphic user interface is introduced into the cleaning and sorting process, allowing the user to see what text is cleaned to assess the efficiency and accuracy of the algorithm.

The algorithm's design philosophy is to be as careful as possible, first sorting and organizing white-listed phrases, then removing the most

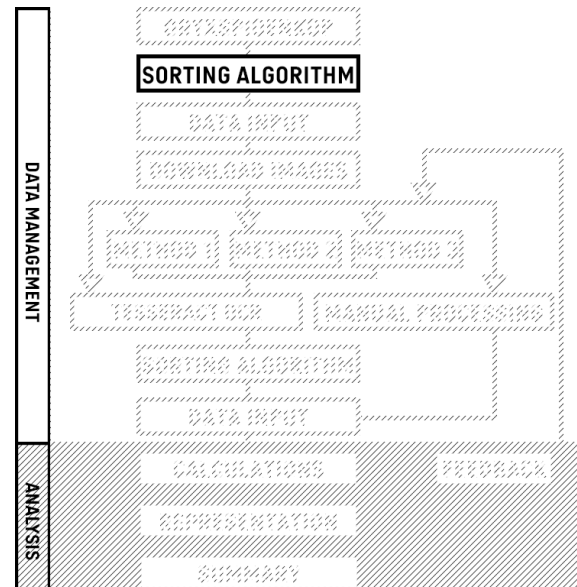


Figure 18, Method step 2 – Sorting and cleaning Oryx HTML code. The step is performed with Notepad ++ and a Python script.

evident clutter, and lastly working its way down – removing smaller and smaller clutter.

It exploits the semi-structured design of HTML code by knowing what tags are followed by useful information. For instance, looking at Figure 17, all URL links are preceded by the phrase ‘href=’’, the URL is then succeeded by a citation mark, a closed-angle-bracket and vehicle status ‘>(1, destroyed)’.

By taking advantage of this structure, the algorithm makes fast work of thousands of lines of code, in a matter of seconds. Outputting a sorted, cleaned,

```

50      T-62M:
        https://twitter.com/UAWeapons/status/1544797070883684353      (1, destroyed)
        https://i.postimg.cc/X72RXmM/1024-t62m-destr-11-01-23.jpg      (2, destroyed)
        https://twitter.com/UAWeapons/status/1572672235726573568      (3, destroyed)
        https://twitter.com/UAWeapons/status/1581908407967092738      (4, destroyed)
        https://i.postimg.cc/Gt0MW2m7/1010-t62m-destr.jpg      (5, destroyed)
        https://twitter.com/UAWeapons/status/1590474674999558145      (6, destroyed)
    
```

Figure 19, Output of the sorting algorithm. The information depicted is the sorted and cleaned code corresponding to Figure 17, formatted to work with Microsoft Excel.

and correctly formatted document, ready to be pasted into Microsoft Excel (Figure 19).

Data Input – *Extraction and Annotation*

When the data from Oryx is cleaned, sorted, and formatted as in Figure 19, it is just a matter of inputting the data (Figure 20) by copying and pasting it into Microsoft Excel. By extracting and annotating the data it is possible to control and monitor each image throughout the entire process. The Excel Framework can be found in the repository.

This is achieved by giving each URL a unique identification number (ID) (Figure 21). This identification number is tied to each URL and is further used as a more intuitive reference, rather than using the URL.

The identification number is used in a range of tasks throughout the process. Data that gets tied to an ID later in the process, can therefore also be tied to an image.

The structure and order of the identification numbers are not vital for the program. In the framework proposed in Appendix 2 – the ID is generated by adding row-number to 1,000,000, essentially allowing up to 8,999,999 images to be analysed (without altering the length of the identification number). The Excel framework found in Appendix 2 is programmed to automatically input and calculate all required formulas, the corresponding code for this function

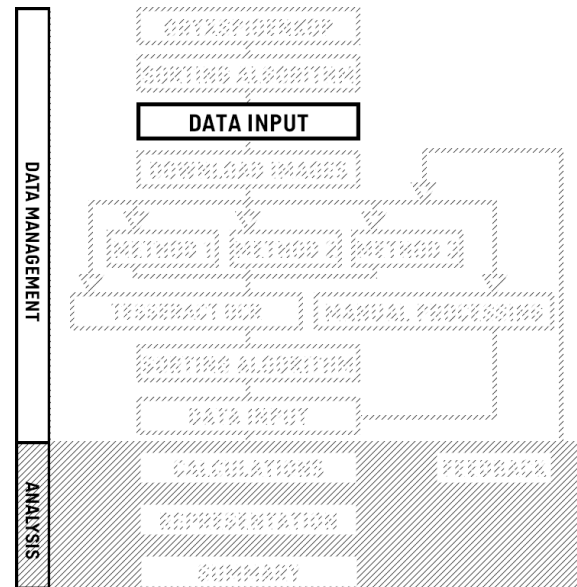


Figure 20, Method step 3 – Inputting the data into Microsoft Excel. The step is performed in Excel Sheet 'Database_Oryx_Automated'.

can be found in Appendix 2 repository named 'VBA code_Module 3.vbs'.

However, for pre-processing methods 1 and 2 (further explained in step 5), the file name being seven digits is vital for the framework to be able to deduce and differentiate between dates and control text.

Furthermore, constraining the identification number to seven digits and using the chronological order given by Oryx, simplifies organizing, downloading, and controlling the images.

AMOUNT	Unit	Vehicle	Links	Status	ID	Source
50		T-62M:	https://twitter.com/UAWeapons/status/1544797070883684353	(1, destroyed)	1000011	Twitter
			https://i.postimg.cc/X72RXDmM/1024-t62m-destr-11-01-23.jpg	(2, destroyed)	1000012	Image
			https://twitter.com/UAWeapons/status/1572672235726573568	(3, destroyed)	1000013	Twitter
			https://twitter.com/UAWeapons/status/1581908407967092738	(4, destroyed)	1000014	Twitter

Figure 21, A snippet of data from Figure 20 inputted into Excel. Columns from Left to Right: AMOUNT – The number of equipment counted by Oryx. UNIT – What kind of unit Oryx categorizes the vehicle/equipment as (not shown in figure). VEHICLE – The vehicle of equipment type categorized by Oryx. LINKS – Corresponding URL. STATUS – Status categorized by Oryx. ID – Each link's unique identification number (automatically inputted and calculated by excel). SOURCE – What kind of resource the URL refers to (automatically inputted and calculated by excel). Blue hyperlink indicates that the link has been clicked on.

Since batch downloading images is only possible on URL links that directly refer to an image (such as ID 1000012 in Figure 21) it is important to categorize which links can be used to download images (column seven in Figure 21). By doing this, each link gets tagged and can be sorted as either Twitter or an Image URL.

Downloading Images – Acquisition and Recording

As previously mentioned, to assure constant access to the images and give Tesseract favourable source material, it is vital to download all images that are included in Oryx (Figure 22).

In the Excel framework, this function is programmed to only require a few button clicks, allowing the user to specify the destination folder, and file name, as well as create a list of all images. The corresponding code for this function can be found in Appendix 2 repository named 'VBA code_Module 1.vbs'. After running the Excel download tool all required information is stored in the same folder (Figure 23).

However, using the downloading tool provided in the repository comes with one drawback – the code cannot distinguish between different image formats, such as '.PNG' or '.JPEG'. Consequently, after downloading an image formatted as a PNG the image will be saved with a JPEG extension, such an image might work at first glance with insensitive software such as Microsoft Paint or an image viewing software. But will, when batch-processed in software such as Photoshop or Gimp create problems. Therefore, there is a need to clean and integrate the images before continuing.

In total, during the period of 24th of February 2022 to 24th of February 2023, Oryx has documented 9392 vehicles and equipment. Whereof 6696 are downloadable images.

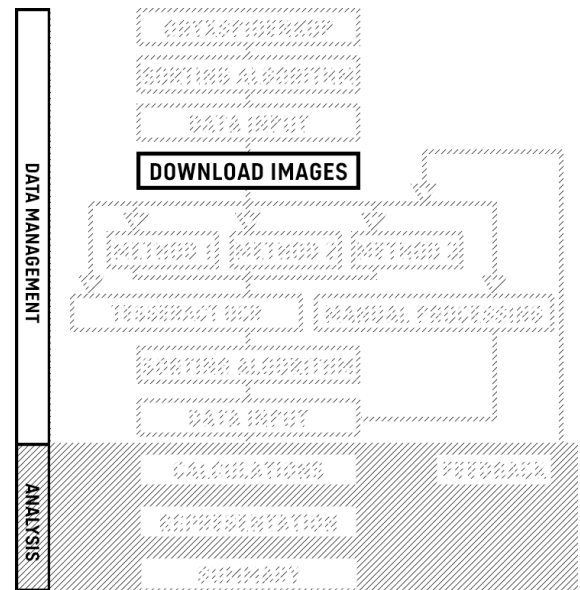


Figure 22, Method step 4 – Downloading the images. The step is done in Excel Sheet 'Image_Download'.



Figure 23, Example of downloaded images and generated text file on a Windows 10 based computer.

ImageMagick – Integration

Like Tesseract, ImageMagick is a free, open-source tool that is powered by console commands. By using ImageMagick – all images can be batch-converted into a correct format. Due to the program's insensitivity, images that are downloaded in a faulty format get converted to a

correct format, without noticeable quality loss, allowing integration of all images.

The following command can be used to operate ImageMagick using the native command prompt on Windows:

```
magick 1000012.jpg 1000012_new.jpg
```

The command 'magick' invokes the correct binary, launching the ImageMagick software, '1000012.jpg' – the input file to convert, and lastly '1000012_new.jpg' – the output file. Note that the input and the output files can have the same name, to eliminate the need to sort copied files.

4.3 Processing methodology

When addressing the question 'How would it be possible to automatically compile the data presented by Oryxspioenkop', it is wise to take into consideration that there are plenty of ways to alter a processing method. The following subsection describes three different pre-processing methods, only using Tesseract, and finally using manual labour (Figure 24). Methods one, two and three were developed in chronological order and are different pre-processing alternatives, each method having processed the same set of 6696 images.

By controlling the results yielded with different threshold values it is concluded that a threshold value of 11-20 has the highest chance of distinguishing dates within the images, then a value of 201-210, and lastly a value of 31-40. The results are further reviewed in Section 5.

Method 1 and 2 – Cleaning and Annotation

The first question that is addressed – is finding out if Tesseract is a suitable tool to use for extracting information from complex imagery. Since the tool is developed for scanning text documents, it would not be a reckless assumption to make that finding dates next to destroyed tanks is above its league. Even though it has proved its fidelity by looking at

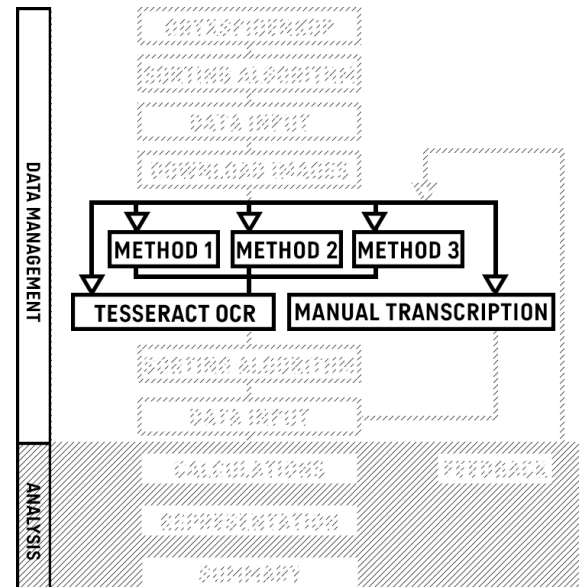


Figure 24, Method step 5 – Processing methodology. Methods 1 to 3 refer to pre-processing methods, these can either be using editing software or command-based functions. Tesseract OCR is done with the OCR engine. Manual transcription is performed by manually looking through each image individually,

```

77  FF14.02.2023
78  FF19.10.2022
79
80  FFFF19.10.2022
81
82  FF21.10.2022
83
84  FF15.11.2022
85
86  FFFF22.02.2023

```

Figure 25, A snippet of raw output from Tesseract, depicting eight analysed images – whereof six contains a date. Character 'FF' is the Form Feed character which separates analysed pages, represented by '\x0C' in American Standard Code for Information Interchange (ASCII).

number plates, the images provided by Oryx are characterized by more unpredictable text formats and backgrounds.

To answer this question, methods one and two analyse Tesseract's work methodology and accuracy. As will be further explained, the difference between the two methods is the order in which they process images.

Since the output of Tesseract does not explicitly tell what image it has analysed, it is daunting to create an efficient sorting algorithm – since the process heavily relies on analysing the output, improving the regular expressions, and controlling the result (similar to the DMAIC-cycle). When batch analysing several images, Tesseract’s output works by segmenting each analysed image with a special character called ‘Form Feed’ (Figure 25).

To illustrate, when controlling the result in Figure 25, for instance, the date ‘15.11.2022’ on line 84, it would be hard to find what image has been analysed, since the output only gives the information within the image.

To correct this, it is possible to add control text into the image (Figure 26), hence either guaranteeing that Tesseract recognizes text within the image or proving that it is inaccurate when analysing images.

The common denominator for method one and two is that they resize the images, add control text, and uses a threshold function to binarize the image. However, the order in which they do so varies.

The reasoning behind resizing the images is to create a constant work environment for Tesseract, eliminating that differing resolutions may impact the result, as well as make the batch processing faster.

This is done by setting the largest dimension of the image to 900 pixels and rescaling the other dimension linearly. For example, an image with the dimensions 1200 x 900 pixels would be resized to 900 x 600 pixels.

Then, by adding control text to the image (Figure 26), the Tesseract output is guaranteed to include the document start, image name and end of the document (Figure 27). Consequently, aiding in controlling the output of Tesseract. The process of inserting the ID into the image is done automatically by using script



Figure 26, A binarized image depicting a captured T-62MV with added control text.

82	FF
83	1111111 1000072
84	15.11.2022
85	0000000 1000072

Figure 27, A snippet of Tesseract output when analysing the image depicted in Figure 26. Depicting the same result as line 84 in Figure 25.

‘Script_1_Photoshop_Filename_to_File.js’ from the repository. This action requires the usage of image editing software. In this case, Adobe Photoshop CC 2015 is used.

Method 1 achieves this by resizing, then binarizing, and lastly adding control text. While method two binarizes, then resizes, and lastly adds control text. The key difference is the order of binarization and resizing. Both methods then feed the image to the Tesseract OCR engine. The result provided by these methods (Exemplified in Figure 27) shows that Tesseract can, and with high accuracy, find dates within the images. In section 5, the difference between these methods is further reviewed.

Using the result from these methods, the sorting algorithm was able to be fine-tuned up to 99.035% accuracy (Section 5).

Method 3 – *Cleaning*

After assessing Tesseract's potency in finding dates, and developing an efficient and accurate sorting algorithm, method three aims to simplify and streamline the process. Avoiding the time-consuming tasks of resizing images and adding control text.

In other words, this method only requires binarization. Consequently, pre-processing with method three can either use image editing software such as Photoshop or Gimp, or command-based tools, such as Python or ImageMagick.

Tesseract OCR – *Cleaning, Annotation, and Integration*

Looking at Figure 24, the Tesseract engine can receive unprocessed or pre-processed images from any of the previously mentioned methods.

As mentioned in section three, the optical character recognition engine uses a set of algorithms to determine the content of an image. It is in this step that the dates within the images are extracted, cleaned, annotated, and integrated to form information that is comprehensible for the sorting algorithm.

By specifying what settings Tesseract should utilize, and what images it should analyse – this step does not require much effort. Since the process aims to extract dates from the images, Tesseract is given a whitelist designed to find digits and common date-separators:

```
whitelist=0123456789./-\\
```

Tesseract is described in more detail in Section 3; hence no further effort will be allocated to describe its principles in this section.

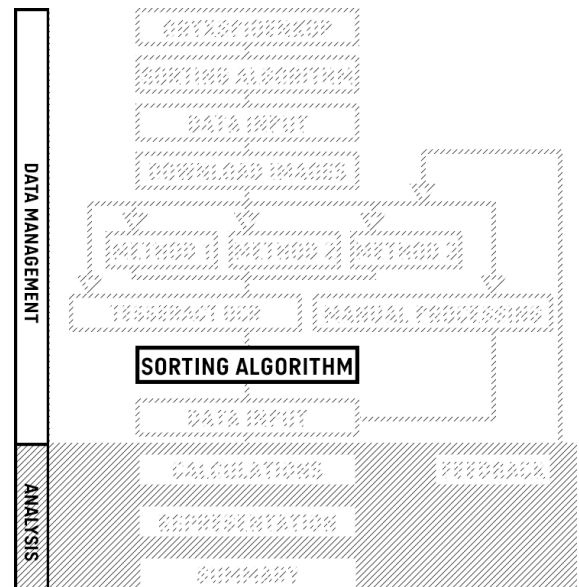


Figure 28, Method step 6 – Tesseract OCR output sorting algorithm. The step is performed with Notepad ++ and a Python script.

Manual Transcription – *Cleaning, Extraction, Annotation, and Integration*

In contrast to using Tesseract to batch analyse imagery, the transcription process requires each date to be inputted manually. Unlike using Tesseract, which relies on algorithms, manual transcription requires human time, effort, and attention. As is showcased in section five, the work needed to process a large number of images is time-consuming. Furthermore, it is either tedious and dull, or a costly task to perform.

The principle of manually transcribing images from Oryx is straightforward. First, choose the image to analyse, then search for any embedded dates, and lastly write down the date.

Hence, using this method is better suited for a small data set or when high levels of accuracy and attention to detail are required.

Sorting Algorithm – *Cleaning and Sorting*

The algorithm designed to sort Tesseract's output file works on the same principles as the HTML sorting algorithm previously addressed. The

algorithm is designed to process the raw output from Tesseract (Figure 28).

However, comparing the semi-structured data provided by Oryx to the first sorting algorithm, the Tesseract output is more unpredictable and, even though it contains certain structured elements such as Form Feed, is more prone to containing artefacts. These artefacts may include – the image containing unwanted numbers, interpreting background as characters, adding additional characters to dates, missing characters within dates, misinterpreting characters, or missing spacing within a date (Figure 29).

Consequently, the algorithm needs to handle all these issues, while at the same time, trying not to remove any vital information.

This is achieved in multiple steps, ranging from symmetrically structuring the document, to patching faulty dates. The complete algorithm with explaining comments is found in the repository. Therefore, this section will not go into detail explaining each step of the process. Nonetheless, the following paragraphs will briefly mention some of the more important steps.

First, by structuring and cleaning the document, the algorithm separates each distinguishable line, allowing it to treat every line as a possible date. Then by changing all non-digits to dots (for example backslash and minus) the dates become more homogenous. After this, the algorithm removes any line that contains less than 3 digits or multiple non-digits. The final steps include patching dates to the correct formats and removes any information it does not recognize as a date.

To illustrate, ‘1404/2022.’ on line 166 in Figure 29 becomes ‘14.04.2022’ on line 89, and ‘. 12.05.22.’ on line 179 becomes ‘12.05.2022’ on line 97, while ‘4559099 2’ on line 170 is not included on line 90.

152	FF09.04.2022	
153		
154	FF16.04.2022 16.0	
155		
156	FF26.04.2022	
157		
158	FF\	
159	FFFFFF12/04/2022	
160		
161	..	
162		
163	FF21/07/2022	
164	.	
165		
166	FF1404/2022 .	
167		
168	-	
169		
170	FF4559099 2	
171		
172	FF05/05/2022	
173		
174	FF01/04/2022.	
175		
176	\	
177	FF16/04/20224	
178	FFFFFF24.07.2022	
179	FF. 12.05.22.	
180	7 .	
181		
182	FF07.022023	
183	FF7 .	
184		
185	07.02.2023..	
186	FF10.03.2022.	
187		
188	FF10.03.2022	
189		
190	8	

81	FF	09.04.2022
82	FF	16.04.2022
83	FF	26.04.2022
84	FF	
85	FF	
86	FF	
87	FF	12.04.2022
88	FF	21.07.2022
89	FF	14.04.2022
90	FF	
91	FF	05.05.2022
92	FF	01.04.2022
93	FF	16.04.2022
94	FF	
95	FF	
96	FF	24.07.2022
97	FF	12.05.2022
98	FF	07.02.2023
99	FF	07.02.2023
100	FF	10.03.2022
101	FF	10.03.2022

Figure 29, Left – A snippet of unsorted Tesseract Output containing various artefacts. Right – The same snippet after running the sorting algorithm. Left figure contains 15 distinguishable dates in multiple formats, right figure contains the same dates, but is more structured.

4.4 Calculation and Representation

Data Input – *Extraction and Annotation*

Moving on to step 7 – the newly cleaned, sorted, and structured dates from the sorting algorithm, or manual transcription, are inputted into Microsoft Excel (Figure 30).

When manually transcribing, the process is continuous, inputting one date at a time to the correct cell. While inputting from Tesseract is achieved by automatically inserting all dates at the same time with a couple of pre-programmed button clicks into the Excel framework.

Calculation – *Integration*

Straightaway after importing, the dates can be calculated to evaluate the accuracy of the chosen method (Figure 31). This is represented in the Excel framework as ‘Total IDs’, ‘Found dates’, ‘Missing dates’, ‘Accuracy’ and ‘Incorrect dates’.

Total IDs are a count of how many images tied to an identification number have been analysed. While found dates equal the total number of inputted dates.

Vice versa, missing dates are calculated by subtracting the amount of found dates from the total amount of images. For example, if 7500 dates are found in a total of 10000 images, missing dates will equal 2500. Accuracy, on the other hand, is the quotient of ‘Found dates’ and ‘Total IDs’, represented in percentage. Using the previous example, accuracy will equal 75%.

Incorrect dates are calculated by looking at how many dates are within the examined period. For instance, if the period is February 2022 to February 2023, and 65 dates are classified as March 2023 – Incorrect dates will equal 65.

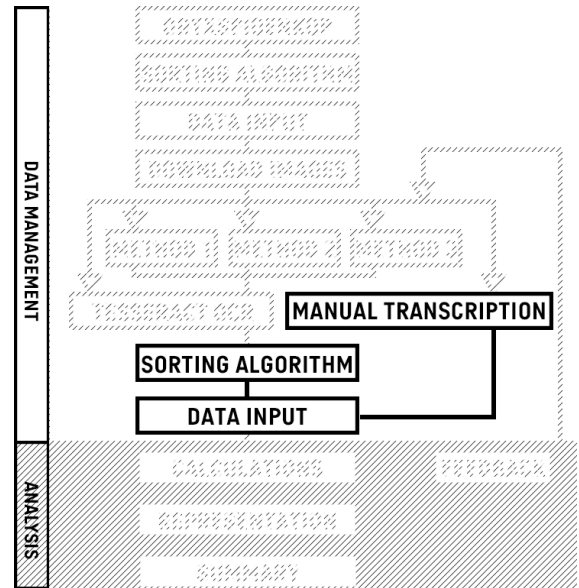


Figure 30, Method step 7 – Data input from either the sorting algorithm or from manual transcription. This step is done in Excel Sheet ‘Tesseract_Results’.

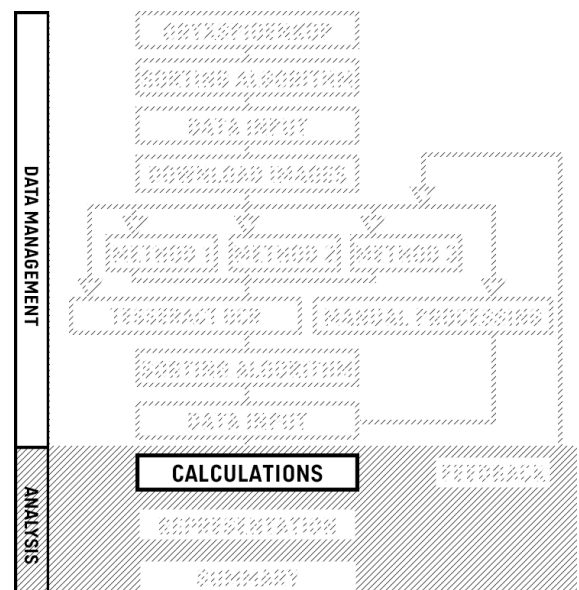


Figure 31, Method step 8 – Calculating the dates. This step is also done in Excel Sheet ‘Tesseract_Results’.

Feedback – *Analysis and Integration*

The reason behind using variables ‘Total IDs’, ‘Found dates’, ‘Missing dates’, ‘Accuracy’ and ‘Incorrect dates’ is to support the feedback process.

To simplify – the feedback process allows the user to re-process all images that Tesseract has not been able to find a date within. By analysing which

Identification numbers are tied to a date, and which are not, the framework re-runs the process (Figure 32). After the output is sorted and inputted, the new results are merged with the old.

Illustrating using the previous example, if 2500 images are re-processed using a different threshold value, and Tesseract finds 1000 new dates, there are now 8500 found dates, shrinking the missing dates from 2500 to 1500. Consequently, the accuracy is increased from 75% to 85%.

Then, after running the feedback process a final time, Tesseract finds 700 new dates, decreasing the missing dates to 800, and increasing the accuracy to 92%.

Only having 800 images left to process, the process can be repeated an infinite number of times, using different threshold values, or falling back to manual transcription, until a satisfactory result is achieved.

However, it should be noted that each iteration of the feedback tends to find a smaller percentage of dates. Consequently, it is recommended to use manual transcription after 3 batches, if the aim is to find all dates within the remaining images. At this stage, the problem tends to be the image quality, rather than the threshold value.

Finally, the dates are represented with their corresponding URL link and summarized in step 10 (Figure 33).

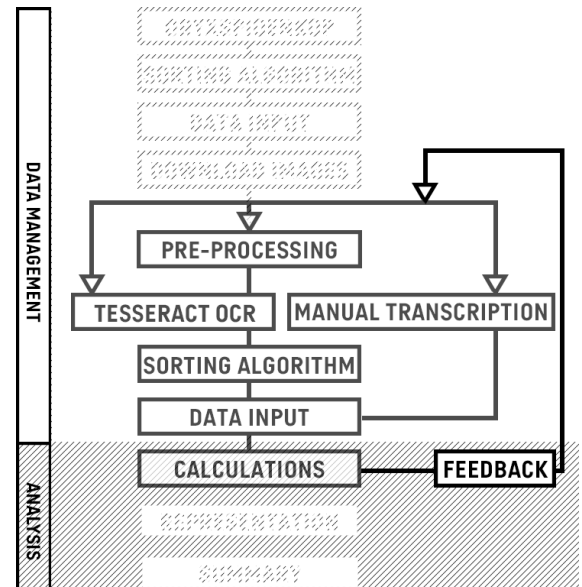


Figure 32, Method step 9 – The feedback process. This step is done in Excel Sheet 'Tesseract_Results', and 'Tesseract_Results_New_Batch'.

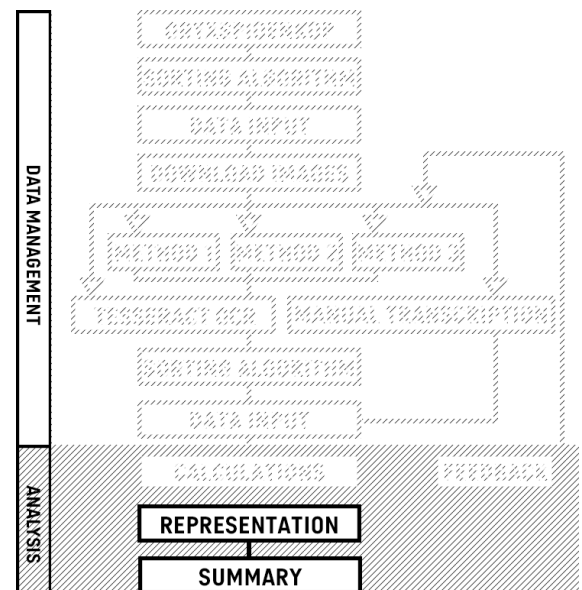


Figure 33, Method step 10 – Representation and Summary of the information. The results can be seen in Excel Sheet 'Database_Oryx_Automated'.

Links	Status	Date (Month)	Date (Tesseract)	ID	Source
https://i.postimg.cc/kX7BT8yc/65.jpg	(5, destroyed)	2022-Apr	09.04.2022	1000135	Image
https://i.postimg.cc/Dy6Y8N89/g7.jpg	(6, destroyed)	2022-Apr	16.04.2022	1000136	Image
https://i.postimg.cc/tg0SKYHf/2216.jpg	(7, destroyed)	2022-Apr	26.04.2022	1000137	Image
https://i.postimg.cc/MG3NKhKh/1003.jpg	(8, destroyed)	2022-May	01.05.2022	1000138	Image
https://i.postimg.cc/rwLTCjMW/554.jpg	(9, destroyed)	2022-Jun	02.06.2022	1000139	Image

Figure 34, A snippet from Excel Sheet 'Database_Oryx_Automated', depicting five URL links that have successfully been analysed. The date is represented as the full date in DD.MM.YYYY format 'Date (Tesseract)' and the shortened date in YYYY-mm format 'Date (Month)'.

Representation – *Representation and Modelling*

The representation step is done by taking the newly acquired and calculated dates and tying them to an URL link (Figure 34). This step is continuously performed after each calculation, allowing sampling of the result to check for anomalies.

For instance, opening the URL link tied to ID ‘1000138’ in Figure 34, it is evident that the date within does not correspond to Tesseract’s output (Figure 35). However, it is still identified as the correct month and year but demonstrates that the method has problems decoding certain text. But on the other hand, opening the four other links – it is evident that the method is still capable of accurately distinguishing embedded dates in varying quality to a high degree (Figure 35).

Summary - *Interpretation*

The final step – Summary, aims to compile the extracted dates into a concise and interpretable preview. This preview can be modified depending on what information is desired. An example of how the data can be summarized is found in Appendix 3 – Data sample, One year of war. The complete summary can be found in Excel Sheet ‘Graphical Overview’.

Looking at the result in Figure 35, it is easy to assume that the method is accurate to 80%. However, since the provided data sample is less than 0.1% of the total dataset – this is likely a false estimate. A more accurate result is reviewed in the following section.

ID: 1000135 DATE (IN IMAGE): 09.04.2022 DATE (TESSERACT): 09.04.2022
ID: 1000136 DATE (IN IMAGE): 16.04.2022 DATE (TESSERACT): 16.04.2022
ID: 1000137 DATE (IN IMAGE): 26.04.2022 DATE (TESSERACT): 26.04.2022
ID: 1000138 DATE (IN IMAGE): 19/05/2022 Chernihiv Oblast DATE (TESSERACT): 01.05.2022
ID: 1000139 DATE (IN IMAGE): 02.06.2022 DATE (TESSERACT): 02.06.2022

Figure 35, Sample inspection of dates in Figure 34.
Black text – Same date in image as analysed by Tesseract.
Red text – Miss-match.

5 Results and Analysis

The following section discusses the results yielded from the approach described in section four. Methods 1, 2 and 3 refer to the same methods as the previous section.

5.1 Method Efficiency

Sorting algorithm – Efficiency and accuracy

When benchmarking the sorting algorithms, it is concluded that neither of the algorithms requires more than one minute to process all data, nor have any of them yielded unexpected results or crashed.

Looking closer at the result yielded by the HTML sorting algorithm, it manages to extract 9388 of the 9392 listed vehicles on Oryx, resulting in an accuracy of 99.96%. In total 6 rows contain an anomaly; all anomalies are tied to the ‘Status’ of the vehicle. Presumably, since there are a few inconsistencies in the Oryx source code.

Thanks to the semi-structured design of HTML the algorithm can extract information with high accuracy, resulting in less than 1 out of 2000 rows formatted in a faulty way. Further, no faulty URL links have been noted.

Assessing the accuracy of the Tesseract sorting algorithm is trickier since it is hard to distinguish if the algorithm fails to correct a date, or if Tesseract fails to find one. Further, as is exemplified in Figure 35, the algorithm can also accidentally make up new dates. It is possible to calculate the number of ‘impossible’ dates, assuming that the dates within the images are correct. For instance, if Tesseract and the sorting algorithm output a date that is ‘in the future’, for instance, ‘02.02.2024’, or before the war started it is safe to assume that the method has failed in extracting that date. Using this

Table 1, Automatic processing result.

Columns – Results from Tesseract, Method 1, Method 2, or Method 3. Rows – Each method is run through three batches. Using Threshold values 37, 220, then 10. Accuracy is calculated by taking ‘Total dates found’ divided by all images containing dates, in this case – 6504. Time taken is specified in hours and minutes, from the start of batch 1 until the results from batch 3 are finished. Dates per minute is the quotient of Total dates found and Time (min).

‘Impossible dates’ are dates that are outside the accepted range. Algorithm inaccuracy is the quotient of ‘Impossible dates’ and Total dates found.

	Only using Tesseract	Method 1	Method 2	Method 3
<i>Batch 1</i>	2692	4450	5284	4908
<i>Batch 2</i>	7	534	475	626
<i>Batch 3</i>	0	540	330	608
<i>Total dates found</i>	2699	5524	6089	6142
<i>Accuracy</i>	41.5%	84.93%	93.62%	94.43%
<i>Time (hh:mm)</i>	01:18	09:47	07:11	02:42
<i>Time (min)</i>	78	587	431	162
<i>Dates per minute</i>	34.6	9.4	14.1	37.9
<i>‘Impossible dates’</i>	66	52	49	95
<i>Algorithm inaccuracy</i>	2.44%	0.94%	0.80%	1.55%

assumption, Table 1, contains the statistic of each method’s algorithm inaccuracy.

As can be seen in Table 1, only using Tesseract yielded the poorest results in all categories except processing time. Which resulted in just about 24 dates per 1000 being corrupt.

Method 1, using pre-processing to resize, binarize and add control text resulted in less than 10 dates out of 1000 being faulty.

Moving to Method 2, pre-processing using binarizing, resizing and control text, the best algorithm accuracy can be found. Resulting in 8 dates out of 1000 falling outside the accepted range.

Method 3, only using a binarization, falls right between Method 2 and only using Tesseract –

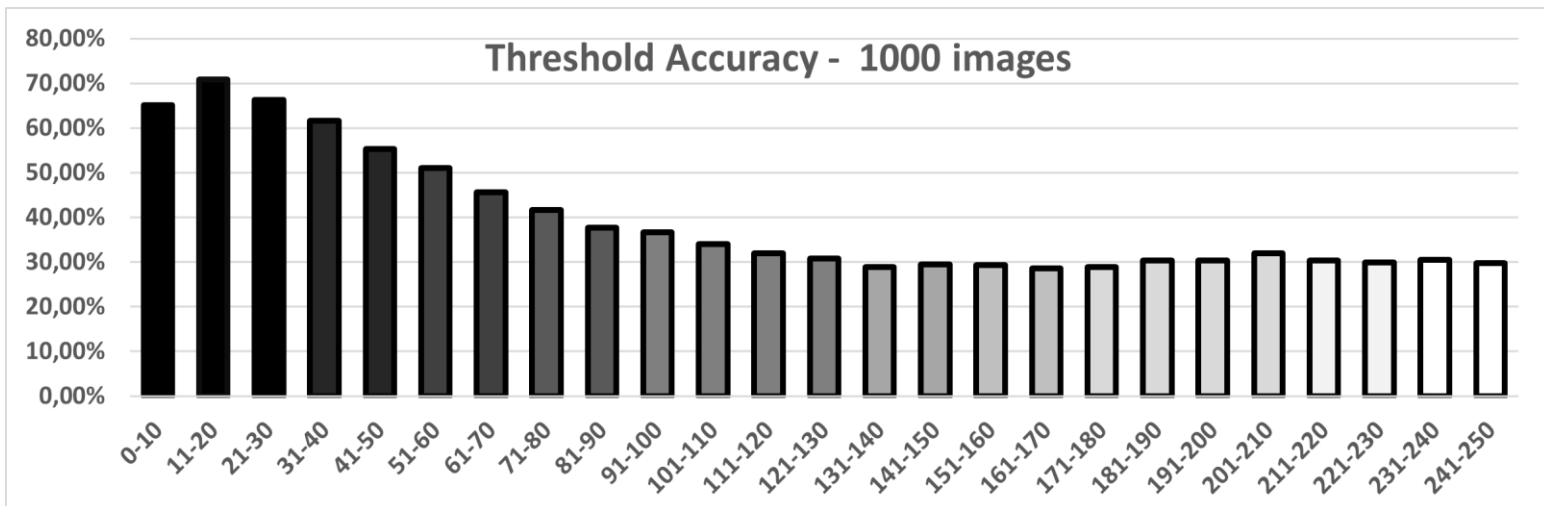


Figure 36, Threshold accuracy extracting dates from 1000 high-quality images. Thresholds range from 0-250. The values are grouped together in intervals of 10, represented by the mean accuracy. Stack colour indicates grayscale threshold – colours lighter than the threshold will be white, and darker or equal will be black in the output image. (Stack colours are only a visual indication and not the actual grayscale value)

resulting in around 16 dates out of 1000 being faulty.

It should be noted that the algorithm used for processing images without control text (Method 3 and only using Tesseract) slightly differs from the other algorithm since it does not have to sort any control text. This could explain why the two methods have a slightly heightened algorithm inaccuracy. The two algorithms are included in the repository.

Image Download

Out of the 6696 downloadable images from Oryx during the period of 24th of February 2022 to 24th of February 2023, 4678 images are formatted as ‘.jpg’ and 2018 images as ‘.png’.

After having run the download tool within the Excel framework, all 6696 images were successfully downloaded, however, 2018 of them are falsely formatted – resulting in roughly 30% of the images being corrupt.

Nevertheless, after having run the ‘ImageMagick’ image correction software, all 6696 images are usable.

Pre-Processing

Figure 36 demonstrates the accuracy of different threshold values ranging from 0 to 250 used on a set of 1000 random high-quality images from Oryx. The process is iterated 25 times, each iteration using the same data set.

As can be seen in the figure, when trying to extract dates from images using Tesseract and a threshold function, the most effective threshold value is between 11-20 which then progressively becomes less effective. Looking at the right end side of the Figure, values 201-210 seem to be the peak value when finding white text. Even though the graph looks to be the same from values 101-250, it is important to note that the different nuances work more or less effectively on specific images, as is illustrated in Section 3, Figure 6. Hence 30% efficiency on one side of the graph, will likely not find the same dates as 30% on the other side of the graph.

It should be noted that achieving these results is possible thanks to the developed process. The threshold values used during the process development are not based on this result. Rather they are based on trial and error. Hence, the results

described below can be further improved using the newly acquired ‘optimal’ threshold values.

The threshold value that, from the trial-and-error phase, gave the best general result is 37. Looking at Figure 37, Threshold 37 manages, on average, to find 72.89% of all dates, Threshold 220 in second place finding 39.69%, while Threshold 10 finds 30.81% of the remaining dates. The reason threshold 10 is underperforming in Figure 37 in contrast to the results yielded in Figure 36 – is that it is used on the images that threshold 37 failed to find dates in. To illustrate, if 1000 images were processed using threshold 37 it would, according to the figure, find 728 dates. The 272 remaining images would then be fed to Threshold 10 which would find 83 new dates. Finally, Threshold 220 would find 75 more dates from the remaining 189 images.

Manual Transcription

To assess what the automated method is competing against – the Manual transcription process had to be explored. To do this, 1000 images were manually transcribed in one sitting, with one 5-minute break after working for 60 minutes. The result of the manual transcription is presented in Table 2.

As can be seen, transcribing 1000 images took 129 minutes, averaging 7.4 dates per minute with an accuracy of finding dates in 95.4% of the images. Judging by the process being manual, and being controlled once more after transcription, the process is estimated to have 100% accuracy.

By assuming that the process can be continued without any loss in transcription speed, the entire set of 6696 images would take roughly 14 hours and 27 minutes (867 minutes) to transcribe.

Only using Tesseract

Subsequently, to assess the need for pre-processing the images, the results yielded by only using Tesseract are reviewed in Table 1. As can be seen,

Table 2, Results from manual transcription of 1000 images. ‘Time estimated’ is a calculation on the total number of man-hours needed to process all 6696 images, not including breaks.

Manual transcription	
<i>Images reviewed</i>	1000
<i>Dates found</i>	954
<i>Images containing date</i>	95.4%
<i>Time (hh:mm)</i>	02:09
<i>Time (min)</i>	129
<i>Dates per minute</i>	7.4
<i>Time estimated all images (hh:mm)</i>	14:27
<i>Time estimated all images (min)</i>	867

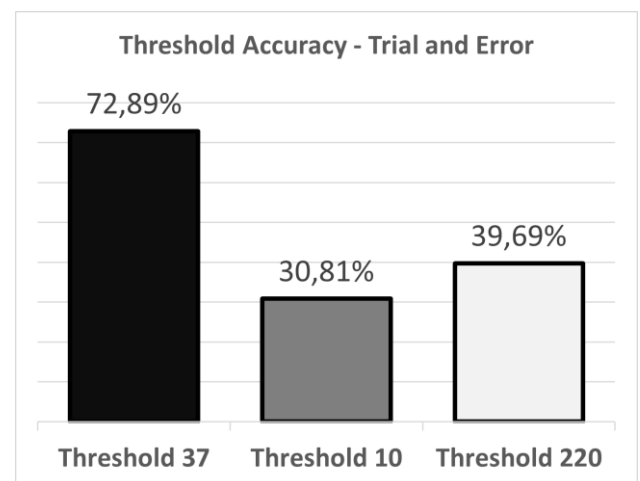


Figure 37, Three different Threshold Value's Accuracy in finding dates with Images. These values originate from the 'trial-and-error' phase. Percentages are taken from the mean value using Methods 1, 2 and 3.

the processing speed when only relying on Tesseract is many times faster than the manual counterpart, achieving over 30 extracted dates per minute. However, the process stagnates after the first batch, only marginally managing to find dates in the second and third batches. Resulting in just over 40% accuracy.

Method 1

As can be seen in Table 1, Method 1 performs the worst of the pre-processing methods, almost achieving 85% accuracy. Even though it is a decent portion, it is a slow process taking close to 10

hours. Since the computation time when inserting control text is drastically increased, the process is slow. On average, from start to end, the method finds 9.4 dates per minute. Compared to manual transcription's 7.4 dates per minute, this is not a very impressive result.

Since the method needs to insert control text, borders, and resize. It requires the usage of image editing software, or an ingenious script to work. The results in Table 1 are achieved using Photoshop.

Method 2

Then looking at the results of Method 2 in Table 1, it is evident that it outperforms the previous method in every aspect. Achieving an accuracy close to 94%, and significantly cutting the processing time, reaching 14.1 extracted dates per minute.

Similarly to Method 1, it also requires either editing software or a well-designed script. The results in Table 1 are achieved using Photoshop.

Method 3

Method 3 is the final iteration of the process. After having assessed the accuracy of Tesseract and creating an efficient sorting algorithm, this method aims to cut out unnecessary processing tasks.

By removing the steps of adding control text, the processing speed is increased by 169% compared to the previous method, achieving to extract 37.9 dates per minute, as well as increasing accuracy slightly over 94%, with the only drawback being an increase in 'impossible dates'.

In contrast to both previous methods, this method can, without hassle, be used with either a script or image editing software. The presented results are achieved using Photoshop. Consequently, in the following subsection, the difference between using image editing software and command-based software is further reviewed.

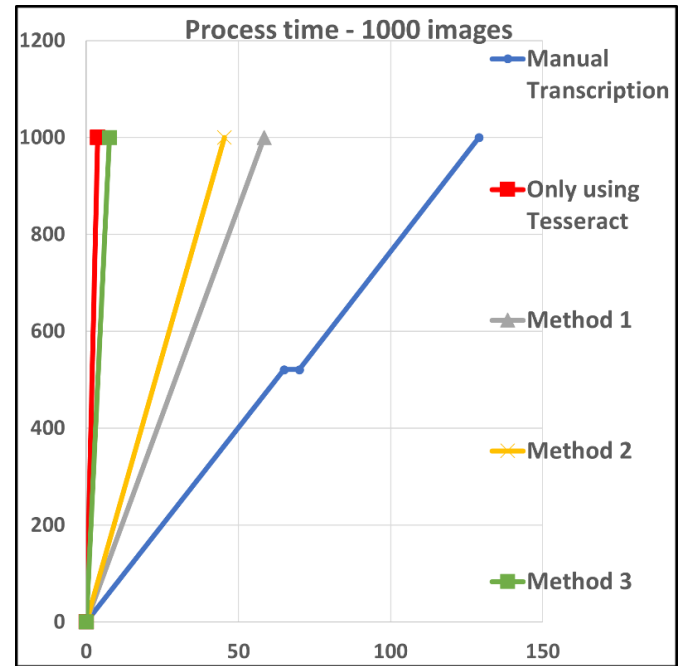


Figure 38, Graph depicting processing time required to process 1000 images using different methods. X-axis depicts time taken (in minute) to process Y number of images.

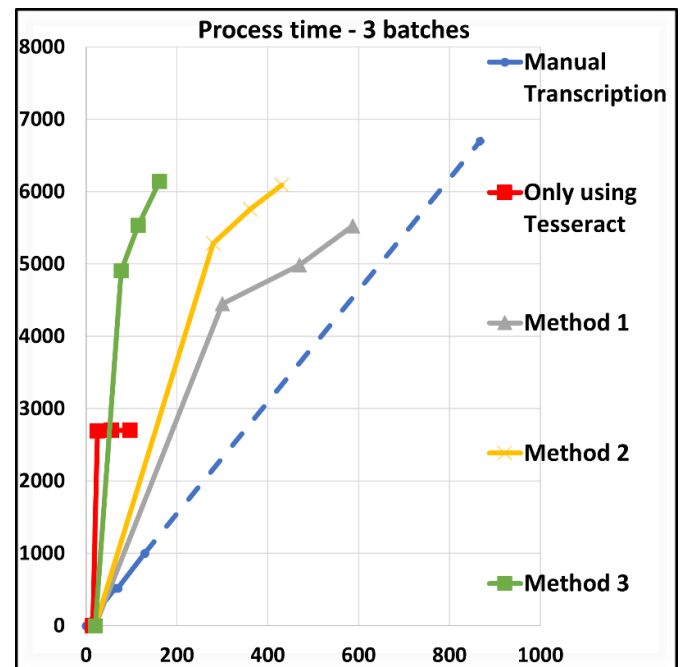


Figure 39, Graph depicting the number of dates found when processing 6696 images in three batches using different methods. X-axis depicts time taken (in minute) to extract Y number of dates.

Method comparison

To summarize the different methods – Figures 38, 39 and 40 illustrate the result from Table 1.

The initial processing speed is illustrated in Figure 38. Showcasing that when looking at the small sample size, only using Tesseract is the fastest method.

However, looking at Figure 39, it is evident that what Tesseract wins in speed, it loses in accuracy. Looking at the final two batches from Method 1, it is also illustrated that it tends to become less effective than manual transcription.

Finally, comparing the accuracy of each method in Figure 40, it is evident that using a pre-processing method is vital when wanting to achieve a practical result. As well as using manual transcription is a good complementary method when wanting to find as many dates as possible.

Pre-processing engine comparison

To control the processing environment when using the three methods, the results presented in Table 1 and Figures 37-40, are achieved by using Photoshop as the pre-processing engine. Furthermore, the processing time is composed of all required actions, such as setting up software, changing windows, navigating folders, etcetera.

Removing these factors unveils a new processing result shown in Table 3. The table compares processing using either Adobe Photoshop or ImageMagick. As can be seen in the upper half of the figure, when only running one instance of the software Photoshop outperforms ImageMagick. Looking at the specifications of the host computer (Appendix 1), this is a natural result since Photoshop can calculate using both the graphical- and central processing units, while ImageMagick relies on only the central processing unit (CPU).

However, looking at the lower half of the Table – the stress test shows a different result. Since Photoshop relies on a graphical user interface, and natively can only run in one instance, tweaking the program's performance is very limited, only performing slightly faster. ImageMagick, on the other hand, can be run in multiple instances. When

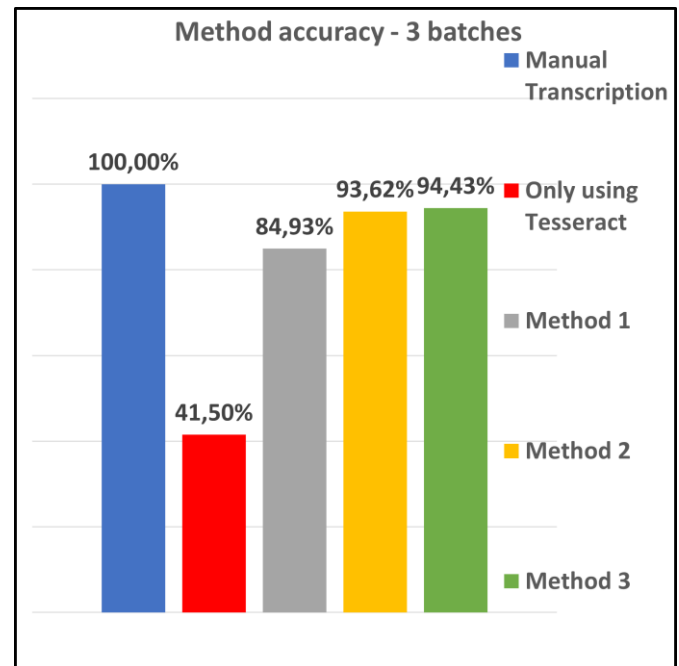


Figure 40, Stacks depicting the accuracy in extracting dates using different processing methods.

Table 3, Comparison between binarization pre-processing speed using Photoshop compared to ImageMagick. Upper half of the Table – Processing speed using the normal data set with one instance. Lower half of the Table – Stress testing the software with over 40,000 images.

	Photoshop	ImageMagick
Images	6696	6696
Processing time (min)	35	40
Images per minute	191	167.4

	Photoshop	ImageMagick
Images	6 x 6696	6 x 6696
Processing time (min)	206	77
Images per minute	195	521.7
Performance increase	2.09%	211.6%

processing six instances of 6696 images with ImageMagick, the output performance was increased by over 200%

The reason behind ImageMagick performing better on six instances is likely to be tied to the operating system throttling the software, not allowing it to use the full potential of the processor. However, by

running several instances, this throttling is surpassed.

Nevertheless, it should be noted that using newer software, different hardware, or operating system will produce unique results, hence it is important to experiment to find the best setup for each system.

Break-Even point

Assuming that the limiting factor comparing manual transcription with the automatic process is the time it takes to set up the required software and learn the process – the break-even point analysing 6696 images would be 705 minutes (manual transcription time minus Method 3 processing time). In other words, if the software can be set up, and the process learned in less than 11 hours and 45 minutes – it is more beneficial to automate the process.

However, this is when the sample size is just under 7000 images. When analysing more images, the setup time can be longer. To illustrate using the results from Table 1 and assuming a sample size of 15,000 images – the manual processing time would be roughly 2000 minutes while using the automated process described in Table 1 would require roughly 400 minutes. Consequently, the setup time only needs to be shorter than 26 hours and 40 minutes (1600 minutes) to be faster than manual transcription.

Looking from a different perspective, assuming that the setup time is 90 minutes – the break-even point would be 827 images (Equation 1). Using the same equation, Figure 41 showcases different break-even points (Y-axis) depending on setup time (X-axis). It should be noted however, that improving the automatic process speed lowers the number of images needed to reach the break-even point – as will be discussed in the following sub-section.

After running the entire process of installing the software, learning the method, setting up,

Equation 1, Values from Table 1. y representing the number of images needed for Manual transcription (Right) to equal Method 3 with 90 minutes of setup time (Left).

$$\begin{aligned}\frac{y}{37,9} + 90 &= \frac{y}{7,4} \\ 90 &= \frac{y}{7,4} - \frac{y}{37,9} \\ 90 &= \frac{30,5y}{280,46} \\ y &= 827,58 \dots\end{aligned}$$

Equation 2, General calculation for break-even point. y = number of images processed. x = setup time [minutes]. A = Automatic processing [Dates per minute]. M = Manual transcription speed [Images per minute].

$$\frac{y}{A} + x = \frac{y}{M}$$

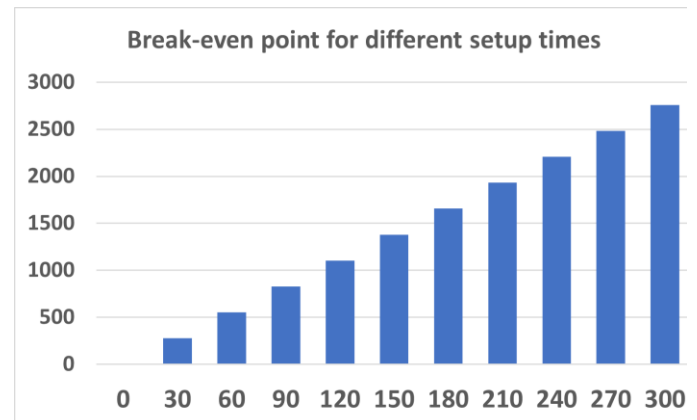


Figure 41, X-axis – setup time ranging from 0-300 minutes. Y-axis – Images needed to reach the break-even point. Calculations are performed using Equation 1.

downloading, etcetera, on a computer operated by a colleague without any previous knowledge of this study – it is concluded that 120 minutes is a realistic setup time. Starting the timer after giving the instructions and stopping when batch 3 is inputted into Excel. In other words, if processing more than 1000 images, it could be beneficial to use the automated process (not including manual transcription). Another interesting factor is that when the test was conducted (24th of March), the Oryx database included more than 7000 images. In other words, more than 10 images have been added per day to the database since the original process was developed.

5.2 Recommended method

Looking at the results, the recommended method to find black or white embedded text is using three batches. Batch one using threshold values 15, finding black text. Batch two uses value 215, which finds white text. Batch three using threshold 35 to find darker grey colours. Then, if accuracy is wanted, either experiment with other threshold values or manually transcribe the last percentages of images.

When choosing a pre-processing engine, the recommended software is ImageMagick, since it can be run in multiple instances, is easy to set up and already required to patch images earlier in the process.

Using the provided framework, scripts and algorithms in the repository are recommended to speed up and facilitate the process.

Test running the process one last time, using these factors and the same set of images, the following result is produced (Table 4). This shows that utilizing ImageMagick instead of Photoshop greatly increased the processing speed (from 37.9 dates per minute in Table 1 to 76.1), but at the same time slightly losing in accuracy (from 94.43% in Table 1 to 94.05%)

Table 4, The final iteration of the automatic process. Upper table – The results from the automatic processing. Middle table – The results from the manual transcription. Lower table – The total results using both automatic processing and manual transcription. Accuracy is the quotient of 6504 and the number total dates found.

Step 1 – Automatic processing	
<i>Batch 1, Threshold 15</i>	5235
<i>Batch 2, Threshold 215</i>	709
<i>Batch 3, Threshold 35</i>	173
<i>Total dates found</i>	6117
<i>Accuracy</i>	94.05%
<i>Time (hh:mm)</i>	01:21
<i>Time (min)</i>	81
<i>Dates per minute</i>	75.5
<i>'Impossible dates'</i>	79
<i>Algorithm inaccuracy</i>	1.28%
Step 2 – Manual transcription	
<i>Remaining images</i>	579
<i>Dates found</i>	387
<i>Images not containing a date</i>	192
<i>Time (hh:mm)</i>	00:59
<i>Time (min)</i>	59
<i>Dates per minute</i>	6.6
<i>Images per minute</i>	9.8
Result	
<i>Images</i>	6696
<i>Dates found</i>	6504
<i>Time (hh:mm)</i>	02:20
<i>Time (min)</i>	140
<i>Time (min) per 1000 images</i>	21
<i>Average dates per minute</i>	46.5
<i>Percentage of images containing dates</i>	97.1%

Further, the results showcase that on average, 971 images out of 1000 contain dates, with an average speed of 21 minutes per 1000 processed images.

Thanks to the increased processing speed using ImageMagick – combining automatic processing and manual transcription still manages to outperform the previous methods, averaging 46.5 dates per minute. Also showcasing that 97.1% of the images within the Oryx database contain dates (Table 4).

Using Equation 2 with the results yielded in Table 4, the break-even setup time for 6696 images using both automatic processing and manual transcription – would equal 12 hours and 40 minutes. This time includes installing the exclusive software, scripts and algorithms required for the automatic process, learning the process, and processing the images.

5.3 Using the method on similar databases

The method has proved to work well when used on other image collections from Oryxspioenkop. For instance, to look at the Ukrainian equipment documentation during the Russian invasion, or from other wars.

However, when used on another website, the process is less effective. The process requires that the website contains embedded URLs that can be used to download images. Nonetheless, with a bit of modification, the method can work well on other sites as well. If all images are already available, the later steps of the process can be used to analyse any embedded text within the images.

However, it should be noted that these results originate from a couple of improvised tests that have not been further analysed. Looking at the algorithms, these are also specified to handle dates in DD.MM.YYYY format, hence they need to be altered if different information is desired.

6 Discussion and future studies

6.1 Strengths and weaknesses

Looking at the two previous sections, the automated date-extraction process is faster than its manual counterpart, proving that it is capable of extracting dates from a large set of images with adequate precision. The process can within a few hours handle thousands of images. However, it still has problems detecting certain date formats and a small, but not unnoticeable, tendency to create ‘fake dates’ when Tesseract misreads text, and the algorithm fails to correct it. Consequently, manual transcription is a great complement to the automated process.

Further, the process still has a lot of room to be improved. Algorithm efficiency, accuracy and self-correcting behaviour can be improved further. The framework can be more user friendly and/or more automated.

A big drawback of the process is that, in its current state, it is a ‘one-time tool’ that does not allow the user to include more images from Oryx without having to re-run the whole process. To illustrate, if the user has data from February, and wants to incorporate the newest data from March, the whole process must be rerun using all images, and not only the newly added ones.

Comparing the images that are included on Oryx between the 24th of February 2023 and the 24th of March, when the test subject benchmarked the process, it is noted that more than 300 images has been added during the period. Showcasing the Oryx group’s continuous work and updates. However, by not including equipment documented on Twitter the method misses almost a third of the possible data that can be analysed from Oryx.

In other words, if the process is used recurrently and the user wants to add new data from month to month, the current state of the framework requires

rerunning everything to not include duplicated vehicles – since the status of a vehicle can change during that month (for example from captured to destroyed).

Since the framework requires human inputs to process images, there is plenty of room for sampling and it makes troubleshooting easier, without having to rework all the code. However, this step is only a strength since the process is not fully developed, a well-developed process should not require to be corrected. This could also be used to argue that the process in itself is semi-automatic, rather than automatic. But the extraction method, on the other hand, is automatic.

As previously mentioned, there is a lot of room for improvements. The code, framework and pre-processing method can be further improved – to reduce faulty information created by the algorithms, make the process more user-friendly and reduce the risk for faulty data-inputs.

Furthermore, the Tesseract OCR engine is not fully explored and could include more settings that give more accurate results. Then there are also additional pre-processing methods that could be incorporated to enhance the results.

Looking at the interoperability of the process, when used on other Oryx image collections, there is no need to alter the code. If used on any other collection, there might be need to correct certain rules in the algorithms or change the appearance of the framework. Additionally, the method is heavily reliant on accurate sorting algorithms and will not function efficiently without them.

But by using a repository to handle all source code – continuous updates and opportunity for others to contribute is made possible. Furthermore, this also allows the framework to be open-source and

controlled by others to not include malicious content.

By utilizing Microsoft Excel as the primary analytical tool, the method loses its aim to be completely free to use and open source, since the desktop version of Excel requires a valid licence and is not open source. On the other hand, this is motivated by Excel *excelling* at handling large amounts of data and allowing the user to write complex code using the Visual Basics programming language.

Raising the question if Oryx does not already have this information compiled; is a relevant critique of this specific method. However, looking at the transparency and professionalism of the Oryx group, it would be odd if they did not include this information if they possessed it. Furthermore, even if they did possess it, they could still refrain from sharing it.

Similarly, if the website is altered, the process could be rendered useless. But by looking at their previous image collections, it is unlikely that they will completely change their website standard. And even if they did, the process could be altered to suit the new standard.

6.2 Research suggestions

Process improvements

As previously mentioned, the developed process has room for improvement. The biggest contribution would be incorporating a scraping method for Twitter into the process allowing for more images to be analysed and more data. The process could also include other databases, or try to extract other information from the images, such as position. Further improving algorithm accuracy, Tesseract settings, or adding the possibility to update the Excel database without having to rerun the whole process.

Military technology

During the development of the process, looking from a military technological perspective, it became evident how easy it would be to fake the images used in Oryx. For instance, by using machine learning, images that depict destroyed equipment can be created in quantity. Figure 42 illustrates an example of the current potential of machine-created photorealistic imagery. For an untrained eye, or looking quickly at the image, the image looks to be real. However, looking closely – the artefacts of the generated image can be seen in better detail (Figure 43). Nonetheless, the process of generating the image took less than 2 minutes.

When more imagery of damaged and destroyed modern equipment is published for artificial intelligence to use as training data, it is plausible that AI will be able to, in near future, produce unique and photorealistic images depicting war-torn equipment. Either side of a war can then use these images to inflate the view of the ‘actual’



Figure 42, An artificially generated image depicting a destroyed tank using OpenAIs DAL-E 2 deep learning engine. The image is generated by using the line ‘a destroyed russian T-80 tank on a field with smoke’. Illustrating the power of AI tools.

losses. Since artificially generated images of humans are already reaching a level where humans cannot distinguish between real and fake ones,³⁵ it is not unlikely that engines focusing on equipment and landscape can reach the same levels.

Since Russia is famous for using troll factories, it is not implausible that such a strategy could evolve in a current, or a future, conflict. Raising the question if these images will be a reliable source in the future.

Military analysis

For military analysis and war studies, the process could be rerun in a later phase of the war when more data is collected. As is showcased in Appendix 3, this data could be used to illustrate the battle frequency of the war or other perspectives that are dependent on quantitative values. The different vehicular losses could also be analysed, for instance why the data shows that certain types of vehicles are overrepresented during different periods of the war.

As mentioned at the start of this report, the initial aim was to develop a method that combines different sources to analyse the equipment losses.



Figure 43, Zoom in on Figure 42. Showcasing the artefacts of the AI-generated image.

Due to lack of time, this aspect had to be abandoned and only using Oryx became the primary research goal. However, as illustrated in Appendix 4, it is interesting to research if combining Oryx image collection with another database can be used to rule out what unit the documented equipment belongs to.

³⁵ Nightingale & Farid, 2021.

7 Conclusions

7.1 Research Questions

How is Oryxspioenkop image collection a viable source to use for compiling equipment losses, and what information is, directly and indirectly, included on the website?

To summarize the study, Oryxspioenkop is a frequently updated, reliable, and accessible source that can be used to compile and review equipment losses during the Russian invasion of Ukraine.

The Oryx group identifies, sorts, and categorizes the equipment. However, to access the text information embedded within the imagery there is a need to use an extraction method, either manual or automatic.

How would it be possible to automatically compile the data presented by Oryxspioenkop – utilizing both the directly included information and the embedded text information within the images?

The proposed automatic method extracts dates from the images which are then combined with the existing information provided by Oryx. The combined information can then be used to depict the frequency, type, status, and amount of documented equipment during the war which is then graphically illustrated. This information is extracted by using a set of tools and then compiled by using Microsoft Excel as a framework.

How accurate and effective is the process, and at what data volume is the break-even point for automatization versus manual labour?

Looking from the developer's perspective – the automatic process works for what it is designed, and it manages to significantly outperform its

manual counterpart. However, the process still has room for improvement, and is at the moment semi-automatic, rather than automatic, due to the rough design.

In a similar fashion, looking from a user's perspective – there is still room for improvement and a more intuitive design.

Looking at the break-even point, having benchmarked the process with different conditions, it is concluded that it is favourable to use the automatic process. However, there is still room for improvement to increase accuracy and efficiency.

The recommended method is to use the automated process to extract most of the information, then resolving to manual transcription if additional accuracy is needed. The method can, as is illustrated in section five, in just over two hours extract dates from roughly seven thousand images – and can be made even faster!

7.2 Future implementations

For a follow-up of this study, there is a multitude of perspectives that can be studied. To mention a few – improving the process, analysing the results yielded from the process, combining multiple databases to generate intelligence, or exploring the future and impact of artificially generated imagery.

Looking forward, to when more images have been gathered and the Russian invasion has come to an end, the hours devoted to this method development and thesis construction will hopefully be a good kick-off for anyone wanting to contribute to the research field or conduct further studies in similar areas.

References

Article (journal)

- Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35(2), 137-144. doi:10.1016
- Montgomery, D. C., & Woodall, W. H. (2008). An Overview of Six Sigma. *International Statistical Review / Revue Internationale de Statistique*, 76(3), 329-346. Retrieved from <http://www.jstor.org/stable/27919650>
- Mursari, L. R., & Wibowo, A. (2021). The Effectiveness of Image Preprocessing on Digital Handwritten Scripts Recognition with The Implementation of OCR Tesseract. *Computer Engineering and Applications*, 10(3), 177-186. Retrieved from <https://comengapp.unsri.ac.id/index.php/comengapp/article/view/386/237>
- Nightingale, S. J., & Farid, H. (2021). AI-synthesized faces are indistinguishable from real faces and more trustworthy. *Proceedings of the National Academy of Sciences* 119,(8), 1-3. Retrieved from <https://doi.org/10.1073/pnas.2120481119>
- Ntogas, N., & Veintzas, D. (2008, July). A binarization algorithm for historical manuscripts. *WSEAS International Conference on COMMUNICATIONS*, 12, 41-51. Retrieved from <http://www.teithessaly.gr/dbData/Dimosieyseis/03-com.pdf>
- Poorani, G., Krithick Krishnna, B. R., Raahul, T., & Praveen Kumar, P. (2022). Number Plate Detection Using YOLOV4 and Tesseract OCR. *Journal of Pharmaceutical Negative Results*, 13(3), 130-136. doi:10.47750
- Zhang, X. Y., Bengio, Y., & Liu, C. L. (2017). Online and offline handwritten Chinese character recognition: A comprehensive study and new benchmark. *Pattern Recognition*, 61, 348-360. Retrieved from <https://www.sciencedirect.com/science/article/abs/pii/S0031320316302187>

Article (periodical)

- BBC. (2022, April 11). Ukraine conflict: Why is Russia losing so many tanks? *BBC*. Retrieved February 27, 2023, from <https://www.bbc.com/news/world-61021388>
- BBC. (2022, November 10). Ukraine war: US estimates 200,000 military casualties on all sides. *BBC*. Retrieved February 24, 2023, from <https://www.bbc.com/news/world-europe-63580372>
- Hambling, D. (2022, May 25). Ukraine Is Knocking Increasing Numbers Of Russian Drones Out Of The Sky — With Help From Russian Corruption. *Forbes*. Retrieved February 28, 2023, from <https://www.forbes.com/sites/davidhambling/2022/05/25/ukraine-is-knocking-increasing-numbers-of-russian-drones-out-of-the-sky/?sh=4f28e91f635a>
- London, B. (2022, April 28). Russia's tanks in Ukraine have a 'jack-in-the-box' design flaw. And the West has known about it since the Gulf war. *CNN*. Retrieved February 28, 2023, from <https://edition.cnn.com/2022/04/27/europe/russia-tanks-blown-turrets-intl-hnk-ml/index.html>
- Sabbagh, D. (2022, April 06). As Ukraine war enters new phase, can western arms turn the tide? *The Guardian*. Retrieved February 28, 2023, from <https://www.theguardian.com/world/2022/apr/06/as-ukraine-war-enters-new-phase-can-western-arms-turn-the-tide>

Personal Interview

Paasikivi, J. (2023, February 20). Självständigt arbete – Ryska materielförluster, Källor OSINT. (C. Zaff, Interviewer)

Report

Smith, R. (2007). *An Overview of the Tesseract OCR Engine*. Google Inc. Retrieved February 29, 2023, from <https://github.com/tesseract-ocr/docs/blob/main/tesseractcdar2007.pdf>

Walker, N. (2023). *Conflict in Ukraine: A timeline (2014 - present)*. House of Commons Library, Uk Parliament. Retrieved from <https://researchbriefings.files.parliament.uk/documents/CBP-9476/CBP-9476.pdf>

Web sites

ACLED. (2023, February 24). *Ukraine Conflict Monitor*. Retrieved February 24, 2023, from ACLEDdata: <https://acleddata.com/ukraine-conflict-monitor/#1677782254184-ea664901-3576>

Binarization. (2023, March 04). Retrieved March 04, 2023, from DeepAI: <https://deepai.org/machine-learning-glossary-and-terms/binarization>

Google Trends - Oryx. (2023, February 24). Retrieved February 24, 2023, from Google Trends: <https://trends.google.com/trends/explore?date=all&q=oryx%20news,oryx%20ukraine,oryx%20equipment,oryx%20russia,oryxspioenkop>

Mitzer, S., & Janovsky, J. (2023, February 24). *Attack On Europe: Documenting Russian Equipment Losses During The 2022 Russian Invasion Of Ukraine*. Retrieved February 24, 2023, from Oryxspioenkop: <https://www.oryxspioenkop.com/2022/02/attack-on-europe-documenting-equipment.html>

Schlottman, H. (2022, November 11). *Russian advances in Ukraine*. Retrieved February 24, 2023, from Uawardata: <https://uawardata.com/>

Tesseract User Manual. (2023, March 02). Retrieved March 02, 2023, from Tesseract-OCR: <https://tesseract-ocr.github.io/tessdoc/Home.html>

Appendix 1 – Requirements and recommendations

System requirements

The system requirements should be seen as a recommendation of the minimum system specifications and software needed to effectively perform the automated process described in section four. These specifications are not a hard limit but are denominated from the highest system requirement (specified by the software developers) from all the required software.

System

	Minimum	Recommended	Done with
Internet-connection	Stable internet-connection	More than 5 MB/s download-speed	0,8-2 MB/s download-speed
OS	Windows, Linux, or Mac	Windows	Windows 10 Pro, 64-bit
RAM	8 GB	16 GB	16 GB DDR4, 1200 MHz
GPU	DirectX 12 Support, 1.5 GB GPU memory	GPU Capable of running Image rendering software	6 GB GPU memory GTX 1660 TI
CPU	Multicore, 64-bit, 2 GHz	Multicore, 64-bit, more than 2.7 GHz	Multicore, 64-bit, 3.2 GHz Ryzen 7 3750H
Memory	More than 15 GB available HDD	More than 15 GB available SSD	200 GB Available Kingston SSD SATA-2

Software requirements

Acquisitions and recording

	Minimum	Recommended	Done with
Web Browser	Can inspect HTML code	Google Chrome, Mozilla Firefox, DuckDuckGo, Opera	Google Chrome
Storing	Latest version of Microsoft Excel with VBA and Macro function enabled		

Extraction, cleaning, and annotation

	Minimum	Recommended	Done with
Text cleaning software	Support for Regular Expressions Search and replace functions	Notepad ++	Notepad ++ With PythonScript plugin
Extraction, Cleaning, and Annotation	Latest version of Microsoft Excel with VBA and Macro function enabled		

Image correction software	Insensitive image correction software capable of changing png to jpg	ImageMagick, Paint.net	ImageMagick
Image Processing	Adobe Photoshop, GIMP, ImageMagick		ImageMagick / Adobe Photoshop CC 2015

Integration, aggregation, and representation

	Minimum	Recommended	Done with
Text cleaning software	Support for Regular Expressions Search and replace functions	Notepad ++	Notepad ++ With PythonScript plugin
Image Processing	Adobe Photoshop, GIMP, ImageMagick		ImageMagick / Adobe Photoshop CC 2015
Optical Character Recognition Software	Tesseract OCR		
Integration, aggregation, and representation	Microsoft Excel with VBA and Macro function		

Appendix 2 – Source code

GitHub Repository

https://github.com/CarlZaff/Tesseract-Date-Extraction_Oryxspioenkop

Oryx Sorting Algorithm (Python)

https://github.com/CarlZaff/Tesseract-Date-Extraction_Oryxspioenkop/blob/main/Algorithm1_Oryxs_HTML.py

Tesseract Sorting Algorithm (Python)

https://github.com/CarlZaff/Tesseract-Date-Extraction_Oryxspioenkop/blob/main/Algorithm2_Tesseract.py

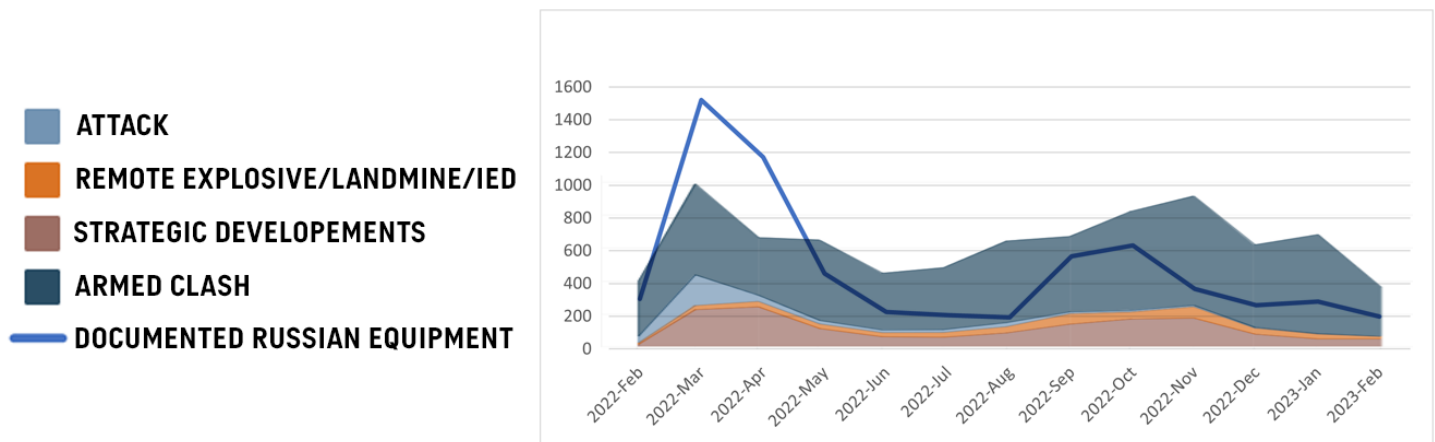
Excel Document Clean Framework (Macro Enabled)

https://github.com/CarlZaff/Tesseract-Date-Extraction_Oryxspioenkop/blob/main/Framework_Oryx_Automatic.xlsm

Photoshop Filename into File (csharp / JavaScript)

https://github.com/CarlZaff/Tesseract-Date-Extraction_Oryxspioenkop/blob/main/Script_1_Photoshop_Filename_to_File.js

Appendix 3 – Data sample, One year of war



Appendix Figure – X-axis: Months ranging from February 2022 to February 2023, Y-axis: Number of documented Russian equipment and political events in Ukraine.

Blue line – Documented Equipment, depicts the number of vehicles that have been documented during each month of the war. The data is retrieved from Oryxspioenkop using the method presented in this report. The colour-coded background depicts ‘Political Violence Events’, retrieved from the Armed Conflict Location & Event Data project (ACLED), documenting Attacks (light blue), Remote explosive, landmines, Improvised explosive devices (orange), Strategic developments (brown) and Armed clashes (dark blue).³⁶

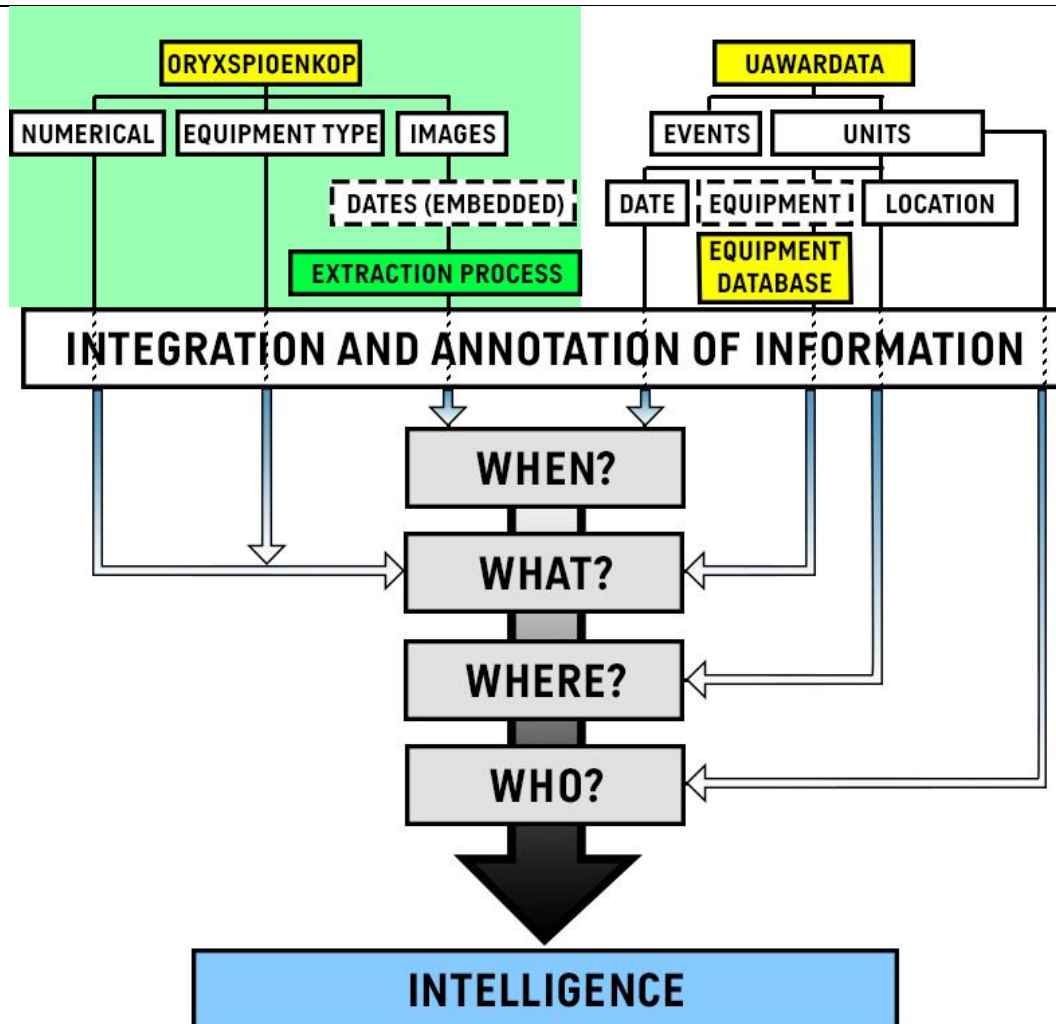
The Oryxspioenkop data sample in the Appendix Figure contains dates extracted from 6696 unique images. The total dataset from Oryxspioenkop consists of 9392 images and Twitter posts. Since only images are examined with this method – 71% of the dataset is included.

This data could, in theory, translate to the battle frequency which has occurred during the Russian Invasion of Ukraine, with a slight offset/displacement on the x-axis, since the images are documented *after* the vehicle has been destroyed, captured, damaged, or abandoned. This is exemplified by the graph depicting *Weekly Organized Political Violence Events* in Ukraine, fetched from The Armed Conflict Location & Event Data Project (ACLED).

However, it is important to note that even though there might appear to be a correlation, this does not guarantee that the two datasets are correlated. This is merely a reflection from the author since there has been no further analysis of the data.

³⁶ ACLED, 2023.

Appendix 4 – Future studies, Data connections and flowchart



Appendix Figure, Schematic suggestion of Data connections and flowchart to cross reference two public databases.

Legend: Yellow background – Database. White background – Information from Database. Grey background – Desired intelligence. Green background – Method discussed in this study. Blue background – New intelligence acquired from cross-referencing the databases. Dashed lines – Missing information that needs to be complemented.

Since both databases provide **WHEN** a unit or equipment has been sighted, and UAwardata can be complemented to include **WHAT** equipment is in each unit – it could be possible, with a certain probability, to cross-reference **WHERE** the equipment included in Oryxspioenkop is located, and **WHO** has lost it, as well as how many they have lost. This process could also be implemented by using another source than UAwardata, which includes units, the date they have been sighted, and their location. It should be noted however, that such a method has several flaws, for instance, a discrepancy between when a unit and equipment is sighted, or several units using the same equipment (as is the case in the Russian armed forces).