|  |
| --- |
| MTU Kerry |
| The use of object detection to produce an audio description |
| Incorporating depth analyses in a 2D image |

|  |
| --- |
| Ivan Segade Carou T00219357  4-26-2024  **Submitted in part-fulfillment of the requirements for the award of**  **Bachelor of Science (Honours) in Computing with Software Development**  **Munster Technological University** |

A picture containing text

Description automatically generated

Table of Contents

[Abstract 4](#_Toc164976546)

[Short abstract 4](#_Toc164976547)

[Introduction 4](#_Toc164976548)

[Overview 5](#_Toc164976549)

[Scope 5](#_Toc164976550)

[Chapter 1. Project review 6](#_Toc164976551)

[Screen Readers 6](#_Toc164976552)

[History of Screen Readers 6](#_Toc164976553)

[Usage of Screen Readers 8](#_Toc164976554)

[Current screen readers tools 9](#_Toc164976555)

[Windows 9](#_Toc164976556)

[Java Access Bridge 9](#_Toc164976557)

[JAWS (Job Access With Speech) 9](#_Toc164976558)

[NVDA (Non-Visual Desktop Access) 10](#_Toc164976559)

[Linux 11](#_Toc164976560)

[ORCA 11](#_Toc164976561)

[Android 12](#_Toc164976562)

[Talk Back 12](#_Toc164976563)

[IOS 12](#_Toc164976564)

[Voice Over 12](#_Toc164976565)

[Web accessibility issues 14](#_Toc164976566)

[Screen Readers and programming 14](#_Toc164976567)

[AI technology 16](#_Toc164976568)

[Web tools 16](#_Toc164976569)

[Chat GPT and Screen Readers 17](#_Toc164976570)

[Image Recognition 18](#_Toc164976571)

[Introduction 18](#_Toc164976572)

[Evolution 19](#_Toc164976573)

[Machine learning algorithms 20](#_Toc164976574)

[Single Shot Detector (SSD) 20](#_Toc164976575)

[You Only Look Once (YOLO) 21](#_Toc164976576)

[Evaluation 22](#_Toc164976577)

[Desktop 22](#_Toc164976578)

[Touch devices 23](#_Toc164976579)

[Methodology 23](#_Toc164976580)

[Chapter 2. Sprint 1 25](#_Toc164976581)

[Find a valid Text to Speech engine. 25](#_Toc164976582)

[Read the FreeTTS documentation and create a class. 25](#_Toc164976583)

[Create the interface. 26](#_Toc164976584)

[Integrate the FreeTTS library into the interface 32](#_Toc164976585)

[Add the focus, action and key listeners 33](#_Toc164976586)

[Chapter 3. Sprint 2 34](#_Toc164976587)

[Download, install and read the Deep Java Library 34](#_Toc164976588)

[How to use the object detection technique with this library 34](#_Toc164976589)

[Find and install a library for working with JSON objects 36](#_Toc164976590)

[Create the necessary classes for working with the code 37](#_Toc164976591)

[Complete the code to generate the description of a picture 42](#_Toc164976592)

[Chapter 4. Sprint 3 47](#_Toc164976593)

[Adding Threads 47](#_Toc164976594)

[Adding a global key listener 49](#_Toc164976595)

[Chapter 5. Sprint 4 57](#_Toc164976596)

[How to create a custom dataset 57](#_Toc164976597)

[Adding overlapping to the description 62](#_Toc164976598)

[Comparing results 64](#_Toc164976599)

[Chapter 6. Sprint 5 68](#_Toc164976600)

[Adding depth by overlapping algorithm 68](#_Toc164976601)

[Integrating the interface in to the DJL 68](#_Toc164976602)

[Adding depth by overlapping algorithm 70](#_Toc164976603)

[Adding depth by central point algorithm 73](#_Toc164976604)

[Adding depth by bottom algorithm 75](#_Toc164976605)

[Implementing the algorithms 78](#_Toc164976606)

[Chapter 7. Sprint 6 84](#_Toc164976607)

[Adding a new interface class 84](#_Toc164976608)

[Testing the program 85](#_Toc164976609)

[Results 87](#_Toc164976610)

[Chapter 8. Future improvements 89](#_Toc164976611)

[Chapter 9. Conclusions 90](#_Toc164976612)

[References 91](#_Toc164976613)

[Appendix A 97](#_Toc164976614)

[Appendix B 98](#_Toc164976615)

# Abstract

This investigation tries to have an understanding of the functionality of screen readers and emulate them. Although screen readers have similar methodologies, often there are actions that are performed differently. Therefore, a comprehension about their internal performance was an interest challenge. Moreover, a second objective is the development of a program that would provide a description of the objects within a picture by using artificial intelligence and more specifically, object detection technology.

Most screen readers do not provide any description of the content of images. There are external applications that supply information about pictures, but mostly these descriptions are vague and just provide a slight mention of the objects.

The program language chosen was Java. Although Python is one of the most programming languages used in the artificial intelligence world, however this language does not suit visual impaired programmers, since indentation is part of its syntax and this is a visual aspect.

To use artificial intelligence with Java, the Deep Java Library was selected. This library provides the necessary code to work with artificial intelligence along with object detection. To generate the voice the FreeTTS (Free Text to Speech) library was used.

The result of this research has been to produce a user interface with its own voice, therefore no additional screen reader is needed. Users can enter the path to a picture and the program provides an audio description of the picture.

## Short abstract

The development of an application that descry es the objects contained within a picture, as existing aids for the visual impaired do not have ability.

# Introduction

## Overview

Text to speech software is an essential tool for the visual impaired people, but image recognition/description is a problem for this technology. Although pictures are recognised the description of what the image is of, is not provided. This project aims to develop a program that will describe the content of a picture and the relative position of the objects. The project also aims to provide an accessible interface to avoid the use of any additional screen reader for the use of this program.

## Scope

The development of a program for the description of an image and the translation of the resulting text to speech. The thesis’s main aims are:

1. Create the interface that the program will use to enter the picture path along with its own voice
2. Install the Deep Java Library and get familiar with it.
3. Develop the code for the object detection.
4. Integrate the interface and the code.
5. Expansion of the details described.
6. Developing algorithms to determine the relative position of objects to each within a picture.
7. Comparison of the performance of the developed algorithms.

# Chapter 1. Project review

## Screen Readers

A screen reader is software that recognizes and reads screen items and translates them into audio version. It is also compatible with Braille devices providing a physical output. Screen readers use speech to describe on screen actions by users.

With these applications blind and visually impaired people can use IT devices on a par with sighted users. This technology is developing a sense of normality for blind and visually impaired users.

Approximately 14% of the Internet utilise screen readers (Gunchenko, 2023). Among these users, 76% are blind, 18.5% have other visual issues. The rest of users have other kinds of disability that enables screen readers to help them use the Internet. Therefore, screen readers are not only for people with visual problems, but more users also rely on these tools (Bureau of Internet Accessibility, 2023). In the recent years, screen readers have become more important, with new uses appearing beyond the accessibility world. For example, screen readers are being used by drivers, for autonomous cars or in the medical world in order to detect diseases prematurely.

## History of Screen Readers

Despite there being some synthesis technology previously used with the operative system DOS, blind users have not managed to have an equal use of computers. It was not until the first screen reader appeared 8 years after DOS was released that blind users were able to take advantage of computer technology.

DOS is ASCII code based. This fact facilitated the ability of assigning a particular sound to a specific character. By knowing the key typed, the computer sent the information to a hardware device which produced synthesized speech for recognition by user(s).

When Graphical User Interfaces “GUI” appeared, new barriers were presented to blind users. As screen readers were designed to read just ASCII characters, they were not prepared for reading icons or graphical representations.

The problem to create a screen reader for “Windows” was that this operating system does not make public its interface control object \*explain what this is\*. Therefore, screen reader developers have to design special actions for each program. This fact meant that the screen reader was not able to read new applications. Therefore, blind users have been reluctant to switch to GUI operative systems.

After pressure from the blind community Microsoft created a working group called “Microsoft Active Accessibility” (MSAA) in 1997. Microsoft also published its user interface control objects, which facilitated the work of screen reader developers to create programs that can read the interface. (American fundation for the blind, 2023)

**Note**: When was this group created and when was the interface made public.

Apple also worked to make its products more accessible. The company had an internal working group call “Worldwide Disability Solutions Group” (WDSG). Temporarily, Apple was the best of all accessible software among all the choices. However, after a year of losses, Steve Jobs (Biography.Com Editors, 2023) decided to dismantle the department in 1998. Despite this, Apple continued to try to develop accessible products, but without the group, which decreased the quality of software accessibility.

In 2005 with APPLE OS X 10.4, Tiger, Apple released its new screen reader called “Voice Over” without additional charges. This version did not work as well as other third-party screen readers. However, from 2009, when Apple released the IPhone 3GS, “Voice over” was much improved by adding the first screen reader that used gestures on touch screens. After this, Apple added this functionality on OS X 10.6, Snow Leopard, to be able to use “Voice Over” on Mac computers. Nowadays, Apple implements “Voice Over” on all its products, and it has become the brand leader among blind touch devices users. (Blind, 2023)

Originally, ORCA was designed by Sun Microsystems by Marc Mulcahy in 2004. The “accessibility Program Office” was in charge of developing the idea, with contributions from many community members. The first version released in 2006. In 2010 Oracle bought Sun Microsystems, and the company stopped developing ORCA. Since then, volunteers and the company Igalia (Igalia, 2023) support ORCA. (Wikipedia, 2023)

Talk Back is the screen reader for Android (Android Accessibility, 2023). Its first release was in 2011 (Wikipedia, 2023) and it is part of the Android suit Accessibility Since 2017 with Android 9.

## Usage of Screen Readers

Screen readers use the “Text to Speech” technology. They are also improving and use the “OCR” (Optical Character Recognition). However, Screen readers are trying to implement AI to enhance their functionality. Text-To-Speech is an assistive technology that converts on screen text to voice. Java provides a free text-to-speech engine called freeTTS (source forge, 2023). A detailed description is given in Audio Technologies (Audio Technologies, 2023). For this reason, this library was selected for the implementation phase.

According to Hagargund et al. (Hagargund, Sharsha, Mitadru, & Eram, 2017) OCR is a technology that is capable of extracting text from scanned documents, pictures and images-only PDF and convert them into digital text format (Hagargund, Sharsha, Mitadru, & Eram, 2017). The main OCR limitation is that just text within the image is recognised and not a description of what the picture is of. This technology is not able to recognise any other kind of objects.

AI goes further than OCR by recognising objects on pictures. This technology is called “image recognition”. Some models look for signs, labels or handwritten words. Other models are trained to recognise independent objects on the images. The most advanced models are able to provide multi sentence descriptions rather than a single word descriptor.

The models evaluate the result according to two concepts: accuracy and relevance of the information. The model determines the output by using the “confidence” terminology. If the result is not accurate or relevant or even misleading, the description is discarded.

Confidence is a standard calculation that evaluates, in percentage, the probability of how correct the algorithm is detecting the objects in the image (Chumachenko, Gabbouj, & Iosifidis, 2022).

AI can even evaluate web pages accessibility to see if they fulfil the Web Content Accessibility Guidelines (WCAG) created by W3C (W3C, 2023). This technology can assess if the contrast or the text size are appropriate. Also, AI can check the images within the page and provide alternative text description.

## Current screen readers tools

## Windows

### Java Access Bridge

Java Access Bridge (Oracle, 2023) is a technology that enables Java based applications to work with assistive technologies on Microsoft Windows. This library is part of the Java Accessibility Utilities (Oracle, 2023) package that provides all the assistive technology for Java applications. For the library to work, the screen reader needs to know what the Java application is doing. For this purpose, the Java Access Bridge library connects to the JVM (Java Virtual Machine), then the Java Accessibility Utilities package, which reads the Java applications actions and sends them to the screen reader via Java Access bridge.

### JAWS (Job Access With Speech)

JAWS is a screen reader designed for Windows with the first version released in 1995 (Assistiv Labs, 2023). Due to its early release, JAWS is the most experienced and used screen reader on Windows (53.7%) (Eren, 2023). This is a commercial product and there are some package versions - Home, Student or Professional. Also, the licence may be bought by year or permanent. For example, the home edition goes from $95/year up to $1100/permanent (Freedom Scientific, 2023)

In order to be able to read Java programs, additional software is required to be installed. This driver is called “Java Access Bridge”. JAWS does not have this feature integrated, so users must install it if they want to be able to read Java programs.

The official JAWS web page lists some of the main features: (Freedom Scientific, 2023) as:

JAWS provides options for the customisation of the voice speech. This screen reader supports several languages, and in some of them with more than one voice option even for male or female. Moreover, other settings can be modified, as voice speed, vocaliser, eloquence. Users can even specify, when they type, if JAWS calls out just the key pressed, just the word typed or both. Also, users can choose if JAWS calls out the punctuations or just make the corresponding stops.

There are visual impaired users that work with Braille devices. JAWS can assist these users, since it holds drivers to be able to read the inputs from those devices, and transform the text into Braille format and pass it to the device. This screen reader also supports some audio book formats such as DAISY (DAISY, 2023).

JAWS provides its own optical character recognition to read text from images and PDF with no accessible format. JAWS can integrate a software created by Freedom scientific called PEARL. This application is used with a camera and can read pages i.e. from books or newspapers, and it converts into voice instantly (VisionAid, 2023)

JAWS is compatible with Microsoft products i.e. MS Office, and with other products, such as Firefox or Chrome. Also JAWS is compatible with the rest of Freedom scientific products, OpenBook, ZoomText, and Fusion.

### NVDA (Non-Visual Desktop Access)

This is another screen reader for Windows that was released in 2007 (Wikipedia, 2023). This software has a similar user interface to JAWS with the same key combinations for moving through the operative system and applications. This is a free product, only supported financially via donations (NVAccess, 2023). This is the second most used screen reader on Windows (38.7%) (Eren, 2023)

The NVDAA installation contains the “Java Access Bridge” driver integrated. So, users do not have to install this feature., since NVDA install it automatically.

According to the official NVDA web page, the main features of NVDA are:

NVDA supports a large number of languages, along with several options for some of them, including male and female voices. NVDA can also adds third party voices. This screen reader can change settings such as, speed, punctuation announcements or tones. NVDA also can configure when users type if it calls out the key typed, the word or both options.

NVDA can work with Braille devices. This screen reader can translate the text displayed on the screen into Braille format and send it to a Braille device. This screen reader can also do the reverse operation. Users that use Braille keyboards, can translate the Braille format into text format. NVDA has also the possibility of being used on touch devices.

Users can use the mouse, with NVDA describing what is under the pointer and says the position of the pointer on the screen. This removes the limitation of visual impaired users of working only with the keyboard.

NVDA adds the option of setting up a portable version. Users can install NVDA on a USB, and then when users connect that USB to a machine, the application runs and can be used on that machine without the need of a previous installation.

NVDA is compatible with the most common applications on Windows, such as, Microsoft Office, Google Chrome, Firefox, and many others that users work with on this operative system. NVDA is also compatible with 32 and 64 bits Windows versions. NVDA can be activated on the login screen in order to assist users from the most early machine phase.

## Linux

### ORCA

ORCA is a Linux screen reader, currently it is the native (preinstalled) screen reader for Linux operative system. It was designed by Sun Microsystems in Python and works specially with “Gnome” and “KDE” desktop environments. Nowadays ORCA is installed by default on the Linux distributions.

(GNOME Wiki, 2023)

According to the linuxlinks.com web page, the main features of ORCA are: (LinuxLinks, 2023)

ORCA enables users to change the synthesizers options i.e. verbosity, speed or pronunciation. This screen reader also allows users to choose the key type announcements: characters, words or both. ORCA also provides the option of magnifying the area where the focus on the screen is.

ORCA supports Braille devices. Users can interactive with Linux machines through Braille devices. Users can enter Braille format text and convert it into text format. Also, users can convert text format into text format in order to be able to read it on Braille format. Users can customise ORCA for using desktops or laptops keyboards.

## Android

### Talk Back

Talk Back (Android Accessibility, 2023) is the Android gestures-based screen reader. It is integrated in Android by default and is available on all Android devices. Samsung has its own customized Talk Back called Voice Assistant (Samsung, 2023). To activate or deactivate it, just press the volume buttons at the same time for 3 seconds. Also, by saying “Activate talk back” or “deactivate Talk Back” to the google assistant. According to the official web, the main features are: (Accessibility Help, 2023)

Talk Back allows users to interact with the touch device through voice commands. This feature removes the need to look for the options across the device menus for the most common actions. Talk Back also has a feature called Auditory feedback. This feature gives an explanation to the user with the possible options that the user has with the focused item.

Talk Back has its own text-to-speech engine. This is a common feature in all screen readers. Also, this screen reader has the speech-to-text feature. This feature converts the words that the user says into text format. This option removes the need of typing on the virtual keyboard. This screen reader has options for low vision users providing the option of changing the contrast colour and increasing the text size.

Talk Back is compatible with Braille devices. This screen reader can convert the text on the screen into Braille format and send it to a connected Braille device. This screen reader can also make the opposite operation of reading text in Braille format from a connected device and displays it on text format on the device screen.

Talk Back also provides options for users with hearing impairments. This screen reader provides subtitles or closed captions for multimedia files or videos. In order to assistive these users with these type of files.

## IOS

### Voice Over

Voice Over is the IOS gestures-based screen reader. It is integrated in IOS by default and is available on iPhone, iPad and Mac. To activate or deactivate it, just triple click on the home button or the side button. Also, by saying “Activate voice over” or “deactivate Voice Over” to Siri.

According to the official web page, the main features are: (Apple, 2023)

Voice Over provides a wide range of options to customise the voice used. Settings as, speed, pronunciation, verbosity or punctuations. This screen reader contains several voices for each languages, including male and female voices.

Voice Over provides shortcuts through gestures for accessing specific settings or even applications. Users must register the gesture and the option on the device that they want to access, then users can utilise the registered gesture in any screen.

Voice Over contains its own text-to-speech engine for converting the text on the screen into voice. This screen reader can also convert words dictated by users into text format on the screen. With this option, users do not have the need of typing the characters on the virtual keyboard.

Voice Over can be used with Braille devices. Users can connect a Braille device and read in Braille the text on the screen. Also, they can transfer the Braille text typed on the Braille device and send it to the device screen on text format.

Voice Over provides information about the item where the focus is by saying the possible options that users may do with the chosen item. This screen reader also calls out the content that there is on the place where users have their finger on the screen.

Voice Over has option for low vision users, by providing the magnification of the text or the area where the focus is. This screen reader also has the option of high contrast or dark mode, even for videos. Voice over can also stop or reduce video animations.

## Web accessibility issues

HTML is designed to be an accessible language. However, there are plenty of features that software companies develop to create web pages. Some of these new features are not accessible for screen readers (i.e., flash components). The issue is that many online entities do not consider web site accessibility testing as mandatory. Therefore, sometimes the major problem for having an accessible web site is the way of this was programmed.

**Note**: if stating facts and numbers you must have a reference to back them up.

Some most common Web accessibility issues: (B, 2023; Chatterjee, 2023)

1. Inappropriate Alternative Text on Images

2. Poor Contrast Text

3. Heading Hierarchy

4. Too Many Navigation Links

5. Poorly Structured Form labels

6. Using Non-descriptive Links

7. Lack of Keyboard Accessibility and Navigability

8. Embedding Inaccessible Documents

9. Forgetting about Mobile-First

10. Add Subtitles and Captions to Videos

## Screen Readers and programming

The JSR-381 (Java Specification Request) (Java Community Process, 2023) on the JavaVisRec package was developed to provide a solution for vision recognition in Java. This specification provides the necessary API that simplifies and standardises how Java deals with recognises, annotates and detects images. The different implementations rely on several platforms, such as DeepNets and MXNet. (InfoWorld, 2023)

Deep Java Library is an open source library (Deep Java Library, 2023) to train, develop and run deep learning models. This library can work with the Single Shot Detector (Liu, et al., 2023)

Apache MXNet is an open source deep learning framework and cross-platform that supports a wide range of programming languages such as Python, Perl or R. The framework allows to users to train, define and work with deep neural networks (Apache, 2023) . MXNet can work with the Single Shot Detector to train, evaluate and test this image recognition library.

OpenCV is a cross-platform library design for developing visual applications. Its main features are image processes, video capture and analyse such as face and object detection (Tutorials Point, 2023).

In Python there are some libraries that can handle images. SCI is one of them. This library is for mathematical and scientific calculations (SCIPY, 2023). Essentially, pictures are pixels that represent mathematical values converted into colours. Therefore, this library has the features for working with images. For this purpose, the SCI library contains a specific module called mdimage (Multidimensional image) (SCIPY, 2023).

One of the framework’s that the Single Shot Detector uses to be developed is PyTorch (Imambi, Prakash, & Kanagachidambaresan, 2021). PyTorch is an open source machine library and is made in Python and supported by the Torch library (Tourch, 2023). This framework is used in deep learning by scientists that contains a large number of mathematical operations and a dynamic computation graphs. This means that developers do not have to wait to compile the whole code to check if the program works (Simplilearn, 2023).

Python is one of the most widely used programming languages for AI. One disadvantage of Python for visually impaired programmers is the fact that indentation is part of the syntax in Python. This feature adds complexity to coding for these programmers, as it is a visual cue requiring character by character examination of the code.

The Emgu library is an image recognition library for .NET compatible programming lang(Emgu, 2023). This library provides the necessary functions for working with OpenCV.

Google provides a cloud-based image recognition service. This service is called Google cloud Vision (Cloud Google, 2023). PHP has a client for working with this service (Googleapis, 2023) . This Google service is available for other programming languages and the service provides code examples for all of them. The Google cloud account provides access to this feature. The service has a free trail with 300 credits to be spent in 90 days. After that, the user will be charged for the use of the service. For this reason, this service was discarded for the implementation phase.

## AI technology

### Web tools

In the recent years, some web sites that process image recognition appeared. These tools provide an image description after processing the image.

One of these tools is the imagerecognize.com web site. After uploading an image picked from the Internet, the tool generated an output with a description. On this web site, the user can set the confidence parameter to fine tune the desire result. (imagerecognize, 2023)

A demonstration is given below.

A screenshot of a computer

Description automatically generated

Figure 1 Output

A screenshot of a computer

Description automatically generated

Figure 2 Settings

### Chat GPT and Screen Readers

After asking Chat GPT about how it can improve digital accessibility, the chat answered (OpenAI, 2023):

Ans. *AI models like ChatGPT can improve digital accessibility by enabling natural language interfaces for people with disabilities. For example, ChatGPT can provide text-to-speech conversion or text summarisation to make information more accessible for people with visual impairments. It can also help in generating alternative text descriptions for images to help people with visual impairments understand what’s on the screen. Additionally, AI can be used to automate accessibility testing and to provide more accessible content recommendations. These applications can make digital experiences more inclusive and accessible for people with disabilities. (How is AI tech like ChatGPT improving digital accessibility? - IA Labs, n.d.)*

In the last months OpenAI announced that ChatGPT will incorporate new features. One of them is the image recognition. This new capability will be available on the mobile ChatGPT version. The user will upload an image and the tool will return a description. (OpenAI, 2023). However, this new feature is just available for ChatGPT 4. Apart from this, ChatGPT is meant to be implemented on applications for image recognition. i.e. the “Be my eyes” application. This software was developed to help blind people to contact to sighted volunteers. Blind users made a video call and volunteers described the image to them.

With Chat GPT integrated on this application, blind users will send an image and Chat GPT will respond with a description of the image. This feature, that is not released yet, will give more independence to blind people. (Ní Chúláin, 2023)

ChatGPT has its own API (OpenAI, 2023). This service is just a link to the ChatGPT search textbox. This service enables the implementation of ChatGPT on a project through a request protocol. Users send a GET request and the API gives the response back. With this feature, users just can access the text ChatGPT version without the possibility of using the audio or the vision features. This service is charged $0.002 per 1000 tokens, equivalent to roughly 750 words. Chat Sonic has also its own API that connects with ChatGPT4 (writesonic, 2023). In order to use this API users must have a small group plan account. For this reason, using these APIs was discarded on the implementation phase.

## Image Recognition

## 

### Introduction

Guo et al. (Guo, et al., 2016) in their review article on deep learning discuss the application of this technology, on the image recognition tasks

Image recognition or image classification is a sub-category of computing vision technology that allows the detection and classification of objects on digital pictures, even videos. To carry out this process machine learning algorithms are used along with Artificial Intelligence to learn patterns and features. Once the model has learnt the patterns, it can detect and interpretate objects on pictures and videos with confidence.

In image recognition there are two versions, image classification and object detection. Image classification detects and predicts the objects that are within the image. In addition to that, object detection (Zhao, Zheng, Xu, & Wu, 2019) detects and predicts the objects in the image and also provides the location of the objects on the image and adds a box that surrounds the corresponding object.

Image recognition uses technologies such as, deep learning (AWS, 2023) or neural networks (Baheti, 2023), to process images, detect the objects within and categorise them. Digital images are made up of pixels which are represented by number values. Machines process this data using machine learning models, detects patterns on the images and ends up with the recognition of the objects appeared on the images.

Image recognition follows a few steps: Firstly the classification of the objects found on the image and assigns them to one of the predefined groups. Once the object is detected and localised on the image is rounded by a box to mark the shape. Image recognition also uses tagging, which detects all the objects found and label them. The segmentation recognises all the pixels that belong to the corresponding object.

By using AI, all the repetitive process such as, detecting and analysing images can be automated. AI uses image recognition to identify patterns, colours or textures. This feature is being used more frequently in several aspects in the real life such as (Arnold, 2023)

* Medicine. With AI medical scans can detect prematurely diseases with high accuracy.
* Security. With AI suspicious movements or objects can be detected
* Business. With AI companies can detect customer’s habits and check inventory.

### Evolution

The starting point of image recognition is considered with the book “Receptive fields of single neurons in the cat's striate cortex” (1959) by David Hubel and Torsten Wiesel. In this book the authors discover how vision neurons detect objects. First vision neurons look for the simple things such as borders and then they move on going more into detail. Image recognition still uses this logic nowadays.

In 1957 Russel Kirsch was able to convert images into numerical representation called pixels (Panos, 2023). With this format, machines were able to have an understandable representation of images. A photo of Russell’s son is considered one of the first digital images and became a symbol of this computing field.

Lawrence Roberts (Gregersen, 2022) is considered one of the pioneers of image recognition technology. Lawrence Roberts wrote a doctoral thesis in 1963 called “Machine perception of three-dimensional solids”. In this thesis Lawrence Roberts explained how to convert a 2D image into a 3D image and vice versa. The thesis was the base for later research into image recognition and 3D technology.

Seymour Aubrey Papert (Ellison, 2023) created in 1967 a research group called “Summer Vision Project” to study all the challenges that image recognition had at that time. The research did not succeed, however, a lot of experts consider this project as the starting point of AI based computing vision technology.

David Marr (Glennerster, 2007) Created in 1982 a research called “Vision: A computational investigation into the human representation and processing of visual information”. David explained that the vision system does not process images in one go, it processes images in a hierarchical way, starting from corners, borders and curves and then going into more complex objects.

A Japanese scientist called Kunihiko Fukushima created a self-organising artificial network called Neocognitron (Franklin Institute, 2023). This network consisted of convolutional rectangular layers with filters. The filters read the input values (the image pixels) and with the data the filters made calculations that were sent to the network to create the result. This network is considered the first convolutional network, a technology widely used in today’s image recognition applications.

From the late 1990s onwards, researchers started abandoning the hierarchical system and started to focus on the idea of recognising objects through their features regardless of factors, such as illumination, position and rotation.

Fei-Fei Li (Stanford Institute, 2023)created in 2007 an improved dataset called Imagenet. This dataset was designed to overcome the barriers that object detection came across i.e. underfitting, which led to unreliable data, because the model is not able to distinguish objects, leading to generalization of the results. Imagenet contains more than 3000000 well labelled images and can distinguish up to 5000 categories. This dataset was a big boost to image recognition technology.

## Machine learning algorithms

### Single Shot Detector (SSD)

SSD (Single Shot Detector) (Liu, et al., SSD: Single Shot MultiBox Detector, 2016). This model is trained on the Pascal VOC 2012 dataset and performs similar to YOLO, but it implements the anchor box technology (Gao, Du, Yang, & J., 2023). SSD uses neural networks (Guo, et al., 2016) as well and divides the image in several grid cells and they are responsible of detecting an object within. In the case of not recognising any object, that cell is considered part of the background. The objects found are rounded with a box to define the corresponding object. SSD provides: confidence of the result, size of the bounding box and the coordinates of the object on the image.

One drawback of SSD is that cannot detect properly small objects on images. Another disadvantage is that SSD gets slow when it uses huge models.

Single Shot Detector contains two components Backbone model and Head (arcgis, 2023). The Backbone model, is a pretrained image classification. The backbone model is the underlined neural network architectural (Yamashita, Nishio, Gian Do, & Togashi, 2018) that gets the input image a for further object detection processing. Different Backbone models are: VGG (Kaloni, Singh, & Tiwari, 2018), ResNet (Kaloni, Singh, & Tiwari, 2018) and MobileNet (Pujara, 2023). Head: After the backbone model extracts the different objects found, the head interpretates the result by adding a label defining the object along with the corresponding surrounded box, coordinates and the confidence of the result (arcgis, 2023).

SSD uses grid cells to divide the image in several parts to detect the objects that are within them. In the case of not detecting any object, the cell is considered as part of the background.

SSD uses anchor boxes for the detection and prediction. Once cell may have several anchor box, each one has different shape and size. The anchor box with the highest confidence of the result is assigned as the result of that specific cell. As objects have different shapes, sizes and even degrees, anchor boxes have several aspect ratio parameters that area implemented on the grid. For the different sizes objects may have, the anchor box has also he zoom parameter to adapt the anchor box to find objects with different size within the cell (arcgis, 2023).

SSD has receptive fields. This fields is the area in the image where the neuron is looking for a result. On SSD the receptive fields are the different layers that SSD uses to process images. It divides the image in different number of cells with different sizes, searching objects with different sizes in the image (arcgis, 2023).

### You Only Look Once (YOLO)

YOLO (You Only Look Once) (Jiang, Ergu, Liu, Cai, & Ma, 2022). This model is a real time object detection that uses Intersection over Union methodology (Rosebrock, 2023). YOLO divides the image in several grid cells, and each of this is responsible of detecting an object within it by using neural network. If an object is found, this is rounded by a box. YOLO uses neural networks architectures to generate high confidence on its results. Unlike previous models, YOLO iterates the image just once, which makes a very fast solution, ideal for live video streams. To carry out the process, YOLO divides the image in grid cells. YOLO provides: the confidence of the result, size of the bounded box,

YOLO implements the non-maximum suppression (NMS) (Gong, et al., 2021) technique. This technique is used after searching in the image for objects. The point is that sometimes the same objects is surrounded by several boxes, and as the boxes may have different sizes they my have different coordinates. NMS fixes this issue by removing the redundant boxes and leaving just one as the round object.

The most recent version of YOLO is the version 7. The main improvement as regards the previous version is that this one uses up to 9 anchor boxes. With this, YOLO can detect a wider range of objects with different shapes and sizes within the cell. Another new feature is the use of the Focal Loss technique (Lin, Goyal, Girshick, He, & Dollar, 2017). This technique is more accurate in order to find smaller objects. NMS improves the resolution up to 608X608 pixels. The last improvement is that YOLO V7 can process up to 155 frames per second. This makes YOLO the fastest object detection algorithm.

One drawback of YOLO is that has problems with objects that are close one to each other.

## Evaluation

### Desktop

JAWS is the most used screen reader (Eren, 2023) amongst Windows users in the visually impaired community. JAWS can read most of Windows components, which maybe associated with the longevity of its use in Windows, released 1995 (Assistiv Labs, 2023)

NVDA is the second most used screen reader in Windows (Eren, 2023). One advantage that this application has is the fact that Java Access Bridge is embedded whereas with JAWS this has to be installed by the user.

JAWS and NVDA are used through similar interfaces with keyboard and similar shortcuts. An additional feature in NVDA is the ability to use the mouse in a pointer read function, the text underneath the cursor been read. This feature would enable a visually impaired user the ability to utilise the mouse in some instances.

From personal experience it has been found that NVDA was able to read some programs better than JAWS. JAWS is not able to read the Cisco Packet Tracer command line, where NVDA was able. Then, the ideal situation would be to have both screen readers installed on a Windows machine, in order to maximise Windows usability.

ORCA still needs to improve on the graphical environment. It can read most of the items, but it is designed to use Linux through the command line. This screen reader needs to be developed more to get the graphical interface more accessible in the case the operative system wants to encourage blind users to use Linux.

### Touch devices

The two most popular screen readers on touch devices are Voice Over for iOS and Talk Back for Android. Voice Over is the most used screen reader with 72% and then Talk Back with 25.8% (Swan, 2023)

One reason that would explain the difference of usage between Voice Over and Talk Back might be that Talk Back is more recent than Voice Over. Therefore, Voice Over has more experience and knowledge on this field.

## Methodology

The project tries to provide an accurate description of a picture by using the Deep Java Library. With the object detection code. The information provided is the list of the detected objects, the coordinates and the size. With this information, apert from just say the objects and their position in the picture, the relative depth providing information on the stratification of objects in the pictures.

For this purpose, three algorithms were designed: the overlapping algorithm, the central point algorithm and the bottom and algorithm. The design of these algorithms is detailed in Sprint 5. These analyses methods were applied to a number of random pictures. The performance of each algorithm was tested by comparing the program result for each of the images with a the assistance of a sighted person as a allow accurate overview of the pictures.

Each picture was analysed using object detection to describe what it is in the picture and the algorithms to calculate the relative depth of the objects within the picture. Confirmation of the relative depth was done by a sighted person.

The process of selecting the pictures tried to find images where the number of objects was not very high because with too many objects the description becomes too large. As the model used is a generic model, images with street scenes were chosen. The pictures should contain objects with overlapping among them due to the overlapping is the base of the three algorithms used.

An initial target was to provide the option of using an built-in text to speech engine within the interface. Although the text to speech worked in the initial stage of the project, this feature became problematic upon the importation of the Deep Java Library. Another problematic area was the implementation of the global key listener. Although a global key listener worked, performance issues relating to repetitive key stroke announcements,(by the voice module) for a single key stroke.

# Chapter 2. Sprint 1

**Sprint 1. Creating the interface with a Text to Speech engine**

Start date: 15/01/2024

End date: 28/01/2024

|  |  |  |
| --- | --- | --- |
| **TASK** | **DETAILS** | **STATUS** |
| 1 | Find a valid Java Text to Speech engine | Complete |
| 2 | Read the documentation of the Text to speech engine and create a class that Generates audio | Complete |
| 3 | Create the interface that will be used by users | In Process |
| 4 | Integrate the Text to Speech engine into the interface | Complete |
| 5 | Add the action, focus and key listeners to the interface | Complete |
|  |  |  |

## Find a valid Text to Speech engine.

A Text to Speech engine was found on the Internet for Java applications. The engine is called FreeTTS (sourceforge, 2024). The FreeTTS was proofed by using a test program. That program instantiated a speech object and speaks out a text. For this purpose, the jar file of FreeTTS must be imported into the project.

## Read the FreeTTS documentation and create a class.

The official web page provides documentation and demos for working with this library. Therefore, a new Java project was created, using IntelliJ. This project is just to check how the FreeTTS library works. In this project a Java class that calls out the text entered was created. This Java class was copied from the demo section, to check if the library works properly in the IDE.

## Create the interface.

Then a new IntelliJ project was created. This project created the interface that the users will use. The interface consists of a textbox where users will enter a path, where the image is saved. Also a button, that will load the image and pass it to the object detector part, which will process the object detection. The interface uses the MVC design pattern to separate the three sections; Model, View and Controller. With this the application is more loosely coupled and clearer.

Explanation of the included code details: internal comments on the code are included, denoted by the use of “//” and “/\*\* \*//”, which precede the individual comments. These comments outline the function of the code which follows the comment.

\*\*\*\*\*\*\*\*\* code of the controller class. The controller *class that manages the instantiation of the interface and the voice of the program*

*/\*\*  
 \* class that manages the instantiation of the interface and the voice of the program  
 \* This is the controller layer of the application  
 \* Author: Ivan Segade Carou  
 \*/*public class CentralController {  
 // declaration the MyVoice class used  
 MyVoice myVoice;  
  
 */\*\*  
 \* Constructor of the controller  
 \*/* CentralController() {  
 // call to the method that instantiates the view class  
 display();  
 } // end constructor  
  
 public void display() {  
 //It instantiates the MyVoice class  
 MyVoice myVoice = MyVoice.*getInstance*();  
  
 // It calls out the presentation message of the program  
 myVoice.speak("Welcome to the central application");  
  
 // It instantiates the interface  
 CentralView centralView = new CentralView(this);  
 centralView.display();  
 } // end display  
  
  
 */\*\*  
 \* public method that calls the unique MyVoice class  
 \* and calls the speak method to call out the text passed as paramter  
 \*  
 \* @param textToSpeech  
 \*/* public void speak(String textToSpeech) {  
 myVoice = MyVoice.*getInstance*();  
 myVoice.speak(textToSpeech);  
 } // end of speak  
  
} // end of class

\*\*\*\*\*\*\*\*\* end of the code of the controller class

\*\*\*\*\*\*\*\*\* code of the MyVoice class. The MyVoice c*lass creates the voice that the program uses. It contains the method that converts the text passed as a parameter to speech*

*/\*\*Class that creates the voice that the program uses  
 \*It contains the method that converts the text passed as a parameter to speech  
 \* This is the model layer of the application  
 \* Author: Ivan Segade Carou  
 \*/*

*// import the libraries for the text to speech functionality*import com.sun.speech.freetts.Voice;  
import com.sun.speech.freetts.VoiceManager;  
  
  
  
  
public class MyVoice {  
 // variable that stores the unique instantiation of the class allowed in the program  
 private static MyVoice *uniqueInstance*;  
  
 // variables needed to create the voice, provided by the FreeTTS  
 VoiceManager voiceManager;  
 Voice voice;  
  
 */\*\*  
 \* private constructor to avoid create several instantiations of the class  
 \*/* private MyVoice() {  
// setting up the voice  
// selecting the voice used  
 String voiceName = "kevin16";  
 voiceManager = VoiceManager.*getInstance*();  
 voice = voiceManager.getVoice(voiceName);  
 voice.allocate();  
 } // end of constructor  
  
  
 */\*\*  
 \* public method that returns the unique instantiation of the class  
 \*  
 \* @return  
 \*/* public static MyVoice getInstance() {  
 // checks that the there is not a previous instantiation to create it  
 if (*uniqueInstance* == null)  
 *uniqueInstance* = new MyVoice();  
  
 return *uniqueInstance*;  
 } // end get instance  
  
  
 */\*\*  
 \* public method that calls out the text passed as a parameter  
 \*  
 \* @param textToSpeech  
 \*/* public void speak(String textToSpeech) {  
 voice.speak(textToSpeech);  
 } // end speak  
  
  
 */\*\*  
 \* public method that deallocate the resources assigned to the voice if needed  
 \*/* public void deallocate() {  
 voice.deallocate();  
 }  
  
} // end of class

\*\*\*\*\*\*\*\*\* end of the code of MyVoice class

\*\*\*\*\*\*\*\*\* the code of view class. The view *class that manages the interface along with its GUI components and listeners*

*/\*\*  
 \* class that manages the interface along with its GUI components and listeners  
 \* This is the view layer of the application  
 \* Author: Ivan Segade Carou  
 \*/*

*// import the libraries to add the GUI components and the listeners*import javax.swing.\*;  
 import java.awt.\*;  
import java.awt.event.\*;  
  
  
public class CentralView extends JFrame implements ActionListener, FocusListener, KeyListener {  
 // declare the controller and the GUI components  
 CentralController controller;  
 JFrame centralFrame;  
  
 JPanel centerPanel;  
  
 JButton speakButton;  
 JTextField textToSpeechField;  
  
 KeyListener keyListener;  
  
  
 */\*\*  
 \* constructor of the class that receives the controller as a parameter to access its methods  
 \*  
 \* @param controller  
 \*/* CentralView(CentralController controller) {  
 // the local controller is declared with the parameter  
 this.controller = controller;  
  
 // set the frame configuration  
 setFrame();  
  
 // add the listeners to the GUI components  
 addListeners();  
  
// add the GUI components to the frame  
 addFrame();  
  
 }// end constructor  
  
 */\*\*  
 \* method that is called to display the interface  
 \*/* public void display() {  
 // with the true value the frame and window can be displayed  
 centerPanel.setVisible(true);  
 centralFrame.setVisible(true);  
 }  
  
 */\*\*  
 \*The following methods are declared as private to avoid external access  
 \*/  
  
  
 /\*\*  
 \* set the configuration of the frame  
 \*/* private void setFrame() {  
// create the frame along with the title  
 centralFrame = new JFrame("Central application");  
  
 // set the size of the window  
 centralFrame.setSize(500, 500);  
  
 // close the window with the X button of the window  
 centralFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 // type of frame  
 centralFrame.setLayout(new BorderLayout(2, 2));  
  
 // the frame is focused in on centralFrame.setFocusable(true);  
  
// declare the panel and add the button and the textbox  
 centerPanel = new JPanel(new GridLayout(1, 1));  
 textToSpeechField = new JTextField(30);  
 speakButton = new JButton("Speak");  
  
 } // end set frame  
  
  
 */\*\*  
 \* method that adds the action, focus and key listeners to the GUI components  
 \*/* private void addListeners() {  
 speakButton.addFocusListener(this);  
 speakButton.addActionListener(this);  
 textToSpeechField.addFocusListener(this);  
 textToSpeechField.addKeyListener(this);  
 addFrameListener();  
  
 speakButton.addKeyListener(keyListener);  
 } // end add listeners  
  
 */\*\*  
 \* method that adds the listeners to the frame  
 \*/* private void addFrameListener() {  
 keyListener = new KeyListener() {  
 // declare a key listener and allows to override the required methods  
 @Override  
 public void keyTyped(KeyEvent e) {  
  
 }  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
  
// get the code of the key pressed  
 int keyCode = e.getKeyCode();  
  
 // with the code, I get the text associated to the corresponding key  
 String keyPressed = KeyEvent.*getKeyText*(keyCode);  
  
 // call to the speak method in the controller to announce the text  
 controller.speak(keyPressed);  
  
 } // end key press  
  
 @Override  
 public void keyReleased(KeyEvent e) {  
  
 }  
 };  
  
 // add the declared key listener to the frame  
 centralFrame.addKeyListener(keyListener);  
 } // end add frame listener  
  
 */\*\*  
 \* method that add the GUI components to the frame  
 \*/* private void addFrame() {  
  
 centerPanel.add(textToSpeechField);  
 centerPanel.add(speakButton);  
  
 centralFrame.add(centerPanel, BorderLayout.*CENTER*);  
  
  
 } // end add frame  
  
  
 */\*\*  
 \* method that declares the action listener  
 \* when the user presses the button  
 \*  
 \* @param e the event to be processed  
 \*/* public void actionPerformed(ActionEvent e) {  
 // if the user pressed the button  
 if (e.getSource() == speakButton) {  
  
  
 // call to the method in the controller class  
 controller.speak(textToSpeechField.getText());  
 }  
 }// end action perform  
  
  
 */\*\*  
 \* method that declares the action when the GUI component has the focus  
 \*  
 \* @param e the event to be processed  
 \*/* public void focusGained(FocusEvent e) {  
 String textFocus = "";  
  
 // if the button has the focus  
 if (e.getSource() == speakButton)  
 //declare the text to speech  
 textFocus = "Speak, Button";  
  
 // if the textbox has the focus  
 else if (e.getSource() == textToSpeechField)  
 //declare the text to speech  
 textFocus = "Enter the text to speech, Text edit";  
  
  
 // call to the speak method in the controller to announce the text  
 controller.speak(textFocus);  
  
 } // end focus gained  
  
  
 */\*\*  
 \* method to be overide  
 \* they must be declared for inheriting the focus class  
 \* in this project they are not needed  
 \*  
 \* @param e the event to be processed  
 \*/* public void focusLost(FocusEvent e) {  
 } // end focus lost  
  
  
 */\*\*  
 \* methods from the key listener  
 \* only the press method is required in this project  
 \*  
 \* @param e the event to be processed  
 \*/* public void keyTyped(KeyEvent e) {  
 }  
  
 public void keyReleased(KeyEvent e) {  
 }  
  
 public void keyPressed(KeyEvent e) {  
// get the code of the key pressed  
 int keyCode = e.getKeyCode();  
  
 // with the code, I get the text associated to the corresponding key  
 String keyPressed = KeyEvent.*getKeyText*(keyCode);  
  
 controller.speak(keyPressed);  
  
 } // end key pressed  
  
  
} //end of class

\*\*\*\*\*\*\*\*\* end of the code of view class

## Integrate the FreeTTS library into the interface

The FreeTTS engine was imported into the project with the interface. A class in charge of the speech task was created, using code detailed previously. This class uses the Singleton pattern. The reason for this is that only one instantiation of the class is allowed, since having several speech objects is not required in this project. Only one voice can be used in the whole application.

## Add the focus, action and key listeners

Focus listeners were added to the textbox and the button. When these items are focused on, the listeners execute a function that announces the item where the focus is. For the textbox, specify what users must enter on it. Swing components do not provide accessibility options, such as alter text. Therefore, the message announced must be set manually.

The button was provided with an action listener. When the button is pressed, the program announces the text entered in the textbox. This is just provisional. The real purpose of the button is to load the image specified in the textbox into the object detection code in order to generate the output.

Also, a key listener was added to the frame. The reason for this is when the cursor is neither the textbox nor the button, the program announces the key pressed by the user. Then, an individual listener is added to the textbox and the button in order to announce the key pressed when the cursor is in either the textbox or in the button.

The interface is due to integrate the object detection section along with the corresponding code. Moreover, a global listener that will substitute the individual listener in each item of the interface is due to add.

# Chapter 3. Sprint 2

**Sprint 2. Creating the code for the object detection part with the Deep Java Library**

Start date: 29/01/2024

End date: 11/02/2024

|  |  |  |
| --- | --- | --- |
| TASK | DETAILS | STATUS |
| 1 | Download, install and read the documentation of the Deep Java Library | Completed |
| 2 | How to use the object detection technique with this library | Completed |
| 3 | Find and install a library for working with JSON objects | Completed |
| 4 | Create the necessary classes for working with the code | Completed |
| 5 | Complete the code to generate the description of a picture | Completed |

## Download, install and read the Deep Java Library

The library chosen for the object detection part was the Deep Java Library. This library has an official web page with documentation and some code examples (Amazon.com, Inc., 2024) . The web page also provides the link to the GitHub repository where users can download the entire library (amazon.com, 2024).

Once the library is downloaded, it is opened by using IntelliJ. The library can be opened by downloading the repository directly from the web page or by using the option in IntelliJ of opening a new project from a version control. There are also some web pages where users explain how to work with this library (Vasudevan, 2024) and YouTube videos that guide how to use the object detection code (Team, 2024).

## How to use the object detection technique with this library

The Deep Java Library provides a Java class that contains the necessary code for using the object detection technique. This class uses a picture on the library as an example and generates a result with the data. Once the class is found, it is run to check how the code works. Unexpectedly, the code gave an error. The problem is that the error message did not provide too much information about the origin of the problem.

\*\*\*\*\*\*\*\*\*\* Error message

> Task :examples:ObjectDetection.main() FAILED

Exception in thread "main" javax.imageio.IIOException: Can't read input file!

at java.desktop/javax.imageio.ImageIO.read(ImageIO.java:1308)

at ai.djl.modality.cv.BufferedImageFactory.fromFile(BufferedImageFactory.java:60)

at ai.djl.examples.inference.ObjectDetection.predict(ObjectDetection.java:55)

at ai.djl.examples.inference.ObjectDetection.main(ObjectDetection.java:49)

Execution failed for task ':examples:ObjectDetection.main()'.

> Process 'command 'C:\Program Files\Java\jdk-21\bin\java.exe'' finished with non-zero exit value 1

\*\*\*\*\*\*\*\*\*\* End of error message

Apparently, the Deep Java Library should work without any additional modification. After reading the code slowly, the problem was spotted. The path of the image contained the wrong path. Originally, it's “src/test/resources/”. The problem is that the working folder starts from examples rather than from src. Therefore, the right path is “examples/src/test/resources/”.

The Deep Java Library contains a class, called ObjectDetection.java, that uses an example picture and generates an output with the data that will be used later in the project. The information provided is:

* Class. The name of the object detected.
* Probability. How likely is that that the detection of that object is.
* Height. The height of the object in the image.
* Width. The width of the object in the image.
* X. the coordinate X of the object in the image.
* Y. the coordinate Y of the object in the image.

\*\*\*\*\*\*\*\*\*\* Output

{"class": "car", "probability": 0.99991, "bounds": {"x"=0.611, "y"=0.137, "width"=0.293, "height"=0.160}}

{"class": "bicycle", "probability": 0.95385, "bounds": {"x"=0.162, "y"=0.207, "width"=0.594, "height"=0.588}}

{"class": "dog", "probability": 0.93752, "bounds": {"x"=0.168, "y"=0.350, "width"=0.274, "height"=0.593}}

\*\*\*\*\*\*\*\*\*\* End of output

The code has a predict method that returns a DetectedObjects class (DJL, 2024). This class contains all the objects detected by the program. The DetectedObjects class has a method, called item, that returns the result. The specifications say that the method returns a type of object Classification. The problem is how to manipulate this object in order to retrieve all the data from the output to make the calculations. In the Deep Java Library there is not clear information about how to use this class as well as to extract the data.

## Find and install a library for working with JSON objects

After reading the output, it is clear that the output has a JSON object format. Therefore, a library for working with JSON objects must be imported in order to work with the data.

After a search, the first library that comes up is the JSON Simple library. An attempt was made to install the jar file for this library. However the compiler did not recognise the library. Then, the dependency was added to the build.gradle file. However, the compiler did not recognise it either. As a result, this library was discarded. Then, another library called com.google.json, was selected for working with JSON objects. The same problem arose as before . After a deep search, a YouTube video explains how to install this library in IntelliJ through the Maven repository (Fraser, 2024). After following the instructions given in the video, the JSON library was recognised by the compiler. Therefore, the data from the output can be manipulated with this library.

## Create the necessary classes for working with the code

Once having all the source code, as well as the tools for working with it, the classes to manipulate and process the data can be done.

First, a class to store the data for each object, call ObjectDetected, is created. This class stores the information retrieved from the predict method along with the description that contains the process of the information of the object in the image.

Explanation of the included code details: internal comments on the code are included, denoted by the use of “//” and “/\*\* \*//”, which precede the individual comments. These comments outline the function of the code which follows the comment.

\*\*\*\*\*\*\*\*\*\* Code of the ObjectDetected class

*/\*\**

*\* Class that stores the information for each object detected*

*\* Author: Ivan Segade Carou*

*\*/*

*// package which this class belongs to*

*package ai.djl.examples;*

*public class ObjectDetected {*

*// declaration of the attributes of the class*

*// the data the class stores from the detection*

*private String name;*

*private int height;*

*private int width;*

*private int positionX;*

*private int positionY;*

*private String description;*

*// constructor with no arguments*

*public ObjectDetected() {*

*}*

*// constructor with the data as a parameter*

*public ObjectDetected(String name, int height, int width, int*

*positionX, int positionY, String description) {*

*// saving the data from the parameters in the attributes*

*this.name = name;*

*this.height = height;*

*this.width = width;*

*this.positionX = positionX;*

*this.positionY = positionY;*

*this.description = description;*

*} // end constructor*

*// declaration of the getters and setters*

*public String getName() {*

*return name;*

*}*

*public void setName(String name) {*

*this.name = name;*

*}*

*public int getHeight() {*

*return height;*

*}*

*public void setHeight(int height) {*

*this.height = height;*

*}*

*public int getWidth() {*

*return width;*

*}*

*public void setWidth(int width) {*

*this.width = width;*

*}*

*public int getPositionX() {*

*return positionX;*

*}*

*public void setPositionX(int positionX) {*

*this.positionX = positionX;*

*}*

*public int getPositionY() {*

*return positionY;*

*}*

*public void setPositionY(int positionY) {*

*this.positionY = positionY;*

*}*

*public String getDescription() {*

*return description;*

*}*

*public void setDescription(String description) {*

*this.description = description;*

*}*

*// overriding the toString method to display the object data*

*@Override*

*public String toString() {*

*return "ObjectDetected{" +*

*"name='" + name + '\'' +*

*", height=" + height +*

*", width=" + width +*

*", positionX=" + positionX +*

*", positionY=" + positionY +*

*", description='" + description + '\'' +*

*'}';*

*}*

*} // end class*

\*\*\*\*\*\*\*\*\*\* end of the code of ObjectDetexcted class

Then, a class, called ObjectFunctionality, is created to process the values from the object attributes. This class process all the information and generates the data for the description attribute of the object with the data about the size and localization of the object in the image

\*\*\*\*\*\*\*\*\*\* Code of the ObjectFunctionality class

*/\*\*  
 \* Class that processes the values of the object and generates the description of the object  
 \* Author: Ivan Segade Carou  
 \*/*// package which the class belongs to  
package ai.djl.examples;  
  
public class ObjectFunctionality {  
 // declaration of the values to calculate the section of the image  
// the 15% of the value for objects with size less than that percentage  
 static final double *SMALLSIZE* = 0.15;  
 // the 40% after the 15% for objects that occupy that space in the image  
 static final double *MEDIUMSIZE* = 0.40;  
 // one third of the size of the image to localise the object in the image  
 static final double *ONETHIRD* = 0.33;  
 // two thirds of the size of the image to localise the object in the image  
 static final double *TWOTHIRDS* = 0.66;  
  
 */\*\*  
 \* method that calls the rest of the sub-methods that process the values  
 \* It receives the image values and the object to be processed  
 \* it returns the whole description already processed  
 \*  
 \* @param imageHeight  
 \* @param imageWidth  
 \* @param od  
 \* @return  
 \*/* public static String generateDescription(int imageHeight, int imageWidth, ObjectDetected od) {  
// declaration of the variables to work with the data  
 String result;  
 //variables that retrieve the data from the object  
 int objectHeight = od.getHeight();  
 int objectWidth = od.getWidth();  
 int objectX = od.getPositionX();  
 int objectY = od.getPositionY();  
// the result variable stores the value for each parameter  
 result = "\n" + *calculateHeightSize*(imageHeight, objectHeight);  
 result += "\n" + *calculateWidthSize*(imageWidth, objectWidth);  
 result += "\n" + *calculateHorizontalPosition*(imageWidth, objectX);  
 result += "\n" + *calculateVerticalPosition*(imageHeight, objectY);  
  
 return result;  
 } // end generate description  
  
  
 */\*\*  
 \* method that calculates the height of the object in the image  
 \* It receives the height of the image and the object  
 \* it returns the string with the description  
 \* it is private because non external classes can access it  
 \*  
 \* @param imageHeight  
 \* @param objectHeight  
 \* @return  
 \*/* private static String calculateHeightSize(int imageHeight, int objectHeight) {  
// the pixel value is less than this variable the object is small  
 int small = (int) (imageHeight \* *SMALLSIZE*);  
 //the pixel value for the calculation of the medium size range  
 int medium = (int) (imageHeight \* *MEDIUMSIZE*);  
 String result = "";  
  
// if statement that calculates small, medium and large size  
 if (objectHeight <= small)  
 result = "it has small height size";  
 else if (objectHeight <= medium)  
 result = "it has a medium height size";  
 else  
 result = "it has a large height size";  
  
 // it returns the description to the generate description method  
 return result;  
 } // end calculate height size  
  
 */\*\*  
 \* method that calculates the width of the object in the image  
 \* \* It receives the width of the image and the object  
 \* \* it returns the string with the description  
 \* \* it is private because non external classes can access it  
 \* \* @param imageWidth  
 \*  
 \* @param objectWidth  
 \* @return  
 \*/* private static String calculateWidthSize(int imageWidth, int objectWidth) {  
 // the pixel value is less than this variable the object is small  
 int small = (int) (imageWidth \* *SMALLSIZE*);  
 //the pixel value for the calculation of the medium size range  
 int medium = (int) (imageWidth \* *MEDIUMSIZE*);  
 //String result = "";  
  
 // if statement that calculates small, medium and large size  
 if (objectWidth <= small)  
 result = "it has a small width size";  
 else if (objectWidth <= medium)  
 result = "it has a medium width size";  
 else  
 result = "it has a large width size";  
  
 // it returns the description to the generate description method  
 return result;  
 } // end calculate width  
  
  
 */\*\*  
 \* method that calculates the vertical position of the object in the image  
 \* It receives the Y coordinate of the object and the height of the image  
 \* it returns the string with the description  
 \* it is private because non external classes can access it  
 \*  
 \* @param imageY  
 \* @param objectY  
 \* @return  
 \*/* private static String calculateVerticalPosition(int imageY, int objectY) {  
 // the pixel value is less than the variable the object is on the top   
 int top = (int) (imageY \* *ONETHIRD*);  
 // the pixel value for the calculation of the middle position   
 int bottom = (int) (imageY \* *TWOTHIRDS*);  
 String result = "";  
  
 // if statement that calculates the top, middle or bottom position   
 if (objectY < top)  
 result = "it is at the top";  
 else if (objectY <= bottom)  
 result = "it is in the middle";  
 else  
 result = "it is at the bottom";  
  
 // it returns the description to the generate description method   
 return result;  
 } // end calculate vertical position  
  
  
 */\*\*  
 \* method that calculates the horizontal position of the object in the image  
 \* it receives the X coordinates of the object and the width of the image  
 \* it returns the string with the description  
 \* it is private because non external classes can access it  
 \*  
 \* @param imageX  
 \* @param objectX  
 \* @return  
 \*/* private static String calculateHorizontalPosition(int imageX, int objectX) {  
// the pixel value is less than the variable the object is on the left   
 int left = (int) (imageX \* *ONETHIRD*);  
 //the pixel value for the calculation of the centre position   
 int right = (int) (imageX \* *TWOTHIRDS*);  
 String result = "";  
  
 // if statement that calculates the left, centre or right position   
 if (objectX < left)  
 result = "it is on the left";  
 else if (objectX <= right)  
 result = "it is in the centre";  
 else  
 result = "it is on the right";  
  
 // it returns the description to the generate description method   
 return result;  
 } // end calculate horizontal position  
  
  
} // end class

\*\*\*\*\*\*\*\*\*\* End of the code of the objectFunctionality class

## Complete the code to generate the description of a picture

Now that all the classes and libraries are created and installed, the core code for displaying the description of a picture can be made.

For this purpose, a class called ObjectDetectorDrive, was created This class centralises the whole code by using the rest of the classes created. It gets the image to be processed and performs all the required tasks to display the result. It calls the predict method that returns the data that will be used for the process. It converts the result into JSON objects and with them make the corresponding calculations. Finally this class outputs the description of the image.

The class called ObjectDetector contains the code provided by the Deep Java Library for processing the object detection. It has the predict method which contains all the code, parameters and settings that the Deep Java Library requires. The method runs the model Zoo which is used for processing the detection of the object within the image.

Explanation of the included code details: internal comments on the code are included, denoted by the use of “//” and “/\*\* \*//”, which precede the individual comments. These comments outline the function of the code which follows the comment.

\*\*\*\*\*\*\*\*\*\* Code of the ObjectDetectorDrive

*/\*\*  
 \* Class that executes the code that displays the description of an image  
 \* Author: Ivan Segade Carou  
 \*/*package ai.djl.examples;  
// import the necessary libraries for the class  
  
import java.util.List;  
import java.util.ArrayList;  
  
import com.google.gson.JsonElement;  
import com.google.gson.JsonObject;  
import com.google.gson.JsonParser;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
  
  
public class ObjectDetectorDrive {  
 */\*\*  
 \* method that is executed when the program is run  
 \*  
 \* @param args  
 \* @throws IOException  
 \* @throws ModelException  
 \* @throws TranslateException  
 \*/* public static void main(String[] args) throws IOException, ModelException, TranslateException {  
 // It declares the variable that stores the path to the image  
 Path imageFile = Paths.*get*("examples/src/test/resources/dog\_bike\_car.jpg");  
 // It creates an image class to be able to work with it  
 Image img = ImageFactory.*getInstance*().fromFile(imageFile);  
  
 // it declares a DetectedObjects variable  
 // the predict method returns the data that is used to generate the description  
 DetectedObjects detection = ObjectDetector.*predict*(img);  
  
 // it declares a variable that stores the data of each object  
 List<ObjectDetected> listOfObjects = new ArrayList<ObjectDetected>();  
  
 // the height and width of the image in pixels  
 int height = img.getHeight();  
 int width = img.getWidth();  
  
 // loop that iterates the list of detected objects  
 // and calls the createObject that converts the prediction into JSON objects  
 // it also calls the generateDescription method that converts the JSON object into string to stores in the objectDetected  
 // and it stores the data in theobjectDetected object  
 // it adds each object to the ArrayList  
 for (int x = 0; x < detection.getNumberOfObjects(); x++) {  
 String result = detection.item(x).toString();  
 ObjectDetected objectDetected = *createObject*(result);  
 String description = ObjectFunctionality.*generateDescription*(height, width, objectDetected);  
 objectDetected.setDescription(description);  
 listOfObjects.add(objectDetected);  
 }// end for  
 System.*out*.print("\nthe height is " + height + "\nthe width is " + width);  
  
 // loop that iterates the ArrayList and  
 // displays the description of each object  
 for (ObjectDetected od : listOfObjects)  
 System.*out*.print("\nName: " + od.getName() + "\nDescription: " + od.getDescription());  
  
 } // end main  
  
  
 */\*\*  
 \* method that receives the result of the prediction in String format  
 \* it converts String into a JSON object and extracts the values of each field  
 \* It returns an ObjectDetected with all the data inserted  
 \* It is private because non external classes can access it  
 \*  
 \* @param result  
 \* @return  
 \*/* private static ObjectDetected createObject(String result) {  
 // it declares a new empty ObjectDetected  
 ObjectDetected od = new ObjectDetected();  
  
 // it converts the String with the result into a JSONElement  
 JsonElement je = JsonParser.*parseString*(result);  
 // It converts the JSONElement into a JSONObject  
 JsonObject jo = je.getAsJsonObject();  
  
 // it retrieves the name value from the JSONObject and stores in the name variable in String format  
 String name = jo.get("class").toString().replace("\"", "");  
 od.setName(name);  
  
// It retrieves the bounds value where the data of the image is stored  
 // It stores the data in the bounds variable  
 String bound = jo.get("bounds").toString();  
  
 // it converts the String with the bounds into a JSONElement  
 JsonElement jee = JsonParser.*parseString*(bound);  
 // It converts the JSONElement into a JSONObject  
 JsonObject joo = jee.getAsJsonObject();  
  
 // It calls the fetchBounds method and passes the ObjectDetected object and the bounds value in String format  
 od = *fetchBounds*(od, jo.get("bounds").toString());  
  
// it returns the ObjectDetected with all the values stored  
 return od;  
 } // end create object  
  
  
 */\*\*  
 \* method that receives the ObjectDetected where the data is stored  
 \* and the bounds values in String format to convert it into JSON and extract the data  
 \* It returns the ObjectDetected object with all the data  
 \* It is private because non external classes can access it  
 \*  
 \* @param od  
 \* @param bounds  
 \* @return  
 \*/* private static ObjectDetected fetchBounds(ObjectDetected od, String bounds) {  
// it declares the variable that stores the corresponding value for storing in the corresponding field  
 int aux;  
 // it converts the String with the bounds into a JSONElement  
 JsonElement je = JsonParser.*parseString*(bounds);  
// It converts the JSONElement into a JSONObject  
 JsonObject jo = je.getAsJsonObject();  
// the fetchDecimals is called to convert the values in integer values  
 // it retrieves the X coordinate value  
 aux = *fetchDecimals*(jo.get("x").toString());  
 // it stores the value in the ObjectDetected  
 od.setPositionX(aux);  
  
 // it retrieves the Y coordinate value  
 aux = *fetchDecimals*(jo.get("y").toString().replace("\"", ""));  
 // it stores the value in the ObjectDetected  
 od.setPositionY(aux);  
  
// it retrieves the height coordinate value  
 aux = *fetchDecimals*(jo.get("height").toString().replace("\"", ""));  
// it stores the value in the ObjectDetected  
 od.setHeight(aux);  
  
 // it retrieves the width coordinate value  
 aux = *fetchDecimals*(jo.get("width").toString().replace("\"", ""));  
// it stores the value in the ObjectDetected  
 od.setWidth(aux);  
  
  
 // It returns the ObjectDetected with the values stored  
 return od;  
 } // fetch bounds  
  
 */\*\*  
 \* method that fetches the values from the bounds.  
 \* the values are defined as decimals values. Therefore they must be converted into integer values to be able to manipulate them  
 \* It receives the value as a decimal  
 \* It returns the value in integer format  
 \* It is private because non external classes can access it  
 \*  
 \* @param value  
 \* @return  
 \*/* private static int fetchDecimals(String value) {  
 return Integer.*parseInt*(value.substring(value.indexOf(".") + 1));  
 } // end fetch decimals  
  
  
} // end class

\*\*\*\*\*\*\*\*\*\* End of the code of the ObjectDetectorDrive

\*\*\*\*\*\*\*\*\*\* Code of the ObjectDetector

*/\*\*  
 \* Class that runs the model Zoo for the object detection  
 \* Author: Deep Java Library  
 \*/*package ai.djl.examples;  
  
// import the necesssary libraries for the class  
  
import ai.djl.Application;  
import ai.djl.ModelException;  
import ai.djl.engine.Engine;  
import ai.djl.examples.inference.ObjectDetection;  
import ai.djl.inference.Predictor;  
import ai.djl.modality.cv.Image;  
import ai.djl.modality.cv.ImageFactory;  
import ai.djl.modality.cv.output.DetectedObjects;  
import ai.djl.repository.zoo.Criteria;  
import ai.djl.repository.zoo.ZooModel;  
import ai.djl.training.util.ProgressBar;  
import ai.djl.translate.TranslateException;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
  
  
public class ObjectDetector {  
 */\*\*  
 \* method that performs the detection of objects in an image  
 \* by using the model Zoo  
 \* It configures the TensorFlow, engine and backbone used for the process  
 \* It receives the image to be processed  
 \* It returns a DetectedObject with the dedata of the detection  
 \*  
 \* @param img  
 \* @return  
 \* @throws IOException  
 \* @throws ModelException  
 \* @throws TranslateException  
 \*/* public static DetectedObjects predict(Image img) throws IOException, ModelException, TranslateException {  
  
 String backbone;  
 if ("TensorFlow".equals(Engine.*getDefaultEngineName*())) {  
 backbone = "mobilenet\_v2";  
 } else {  
 backbone = "resnet50";  
 }  
  
 Criteria<Image, DetectedObjects> criteria =  
 Criteria.*builder*()  
 .optApplication(Application.CV.*OBJECT\_DETECTION*)  
 .setTypes(Image.class, DetectedObjects.class)  
 .optFilter("backbone", backbone)  
 .optEngine(Engine.*getDefaultEngineName*())  
 .optProgress(new ProgressBar())  
 .build();  
  
 try (ZooModel<Image, DetectedObjects> model = criteria.loadModel()) {  
 try (Predictor<Image, DetectedObjects> predictor = model.newPredictor()) {  
 DetectedObjects detection = predictor.predict(img);  
  
 return detection;  
 }  
 }  
 } // end of predict  
  
  
} // end class

\*\*\*\*\*\*\*\*\*\* End of the code of ObjectDetector

Due to the dependencies of the Deep Java Library, the program must be connected to the Internet to be able to run the program.

# Chapter 4. Sprint 3

**Sprint 3 Adding Threads and global key listener**

Start date: 19/02/2024

End date: 03/03/2024

|  |  |  |
| --- | --- | --- |
| **TASK** | **DETAILS** | **STATUS** |
| 1 | Implementing threads to stop the voice in the interface | No completed |
| 2 | Implementing global key listener in the interface | No completed |
|  |  |  |

## Adding Threads

Some screen readers have the option of stopping the voice by pressing the control key. This is useful especially when the screen reader is announcing a long sentence and the user wants to interrupt the announcement. For this purpose, threads were chosen for emulating the same performance. This option would be useful particularly when the voice of the program is reading the text located in the textbox, since this may be too long and the user does not need to hear all the text.

The following code was added to the button action listener function in the CentralView class. As this class was already commented on in the previous sprint, only the new code was added.

Explanation of the included code details: internal comments on the code are included, denoted by the use of “//” and “/\*\* \*//”, which precede the individual comments. These comments outline the function of the code which follows the comment.

\*\*\*\*\*\*\*\*\*\* Code of the CentralView class

… rest of the code …

// declaration in the global declaration section

// the boolean variable for stopping the thread

private boolean stop = false;  
// the thread variable to work with

private Thread myThread;

… rest of the code …

// The new action performance function

// function is called when the user presses the button

public void actionPerformed(ActionEvent e) {  
 // if the user pressed the button  
 if (e.getSource() == speakButton) {  
// initialising the thread  
 // while loop that stops when the stop variable is set to true

// this statement is for stopping the thread myThread = new Thread(() -> {  
 while (!stop) {  
// try statement for running the thread catching the exceptions

try {

// call to the method speak in the controller class  
 controller.speak(textToSpeechField.getText());

// intermedium time between the execution of the thread  
 Thread.*sleep*(000);

// catch statement for avoiding crashing the program  
 } catch (InterruptedException ie) {  
 ie.printStackTrace();  
 } // end catch  
 } // end while  
 }); // end thread

// executing the thread

myThread.start();  
// avoiding more than one repetition

stop = true;  
  
 } // end if  
}// end action perform

… rest of the code …

// adding an if statement to the button key listener

// it checks when the user presses the control key and try to stop the voice

public void keyPressed(KeyEvent e) {  
  
// get the code of the key pressed  
 int keyCode = e.getKeyCode();  
  
 // with the code, I get the text associated to the corresponding key  
 String keyPressed = KeyEvent.*getKeyText*(keyCode);

// if statement that checks if the key pressed was the control key

// if if so, the stop variable is set to true to stop the thread  
 if (keyPressed.equals("Ctrl")) {  
 stop = true;  
   
 }  
  
 // call to the speak method in the controller to announce the text

// this call remains to announce the key pressed   
 controller.speak(keyPressed);  
  
 } // end key press

… rest of the code …

\*\*\*\*\*\*\*\*\*\*End of the new code of the CentralView class

Unfortunately, this implementation does not perform as desired. Once the speak method starts to run, the function does not stop until it finishes reading the text. The same code that handles the thread is moved to the CentralController class and the MyVoice class in their corresponding speak methods. The result of these modifications does not change the performance of the thread.

## Adding a global key listener

In this project a key listener was added to each GUI component. The ideal situation is that only one key listener handles this task. For this purpose, the idea is to add a global key listener to the frame to centralise the announcement of the key pressed. Java does not allow this implementation inherently. Some codes in the Internet say that they can perform that task. Therefore, the idea is to download these pieces of code, check them out and integrate them to the interface.

One solution is the GInputHook (dyorgio, 2024). The code was downloaded and opened with IntelliJ as a new project. Once the program is compiled, an error arises:

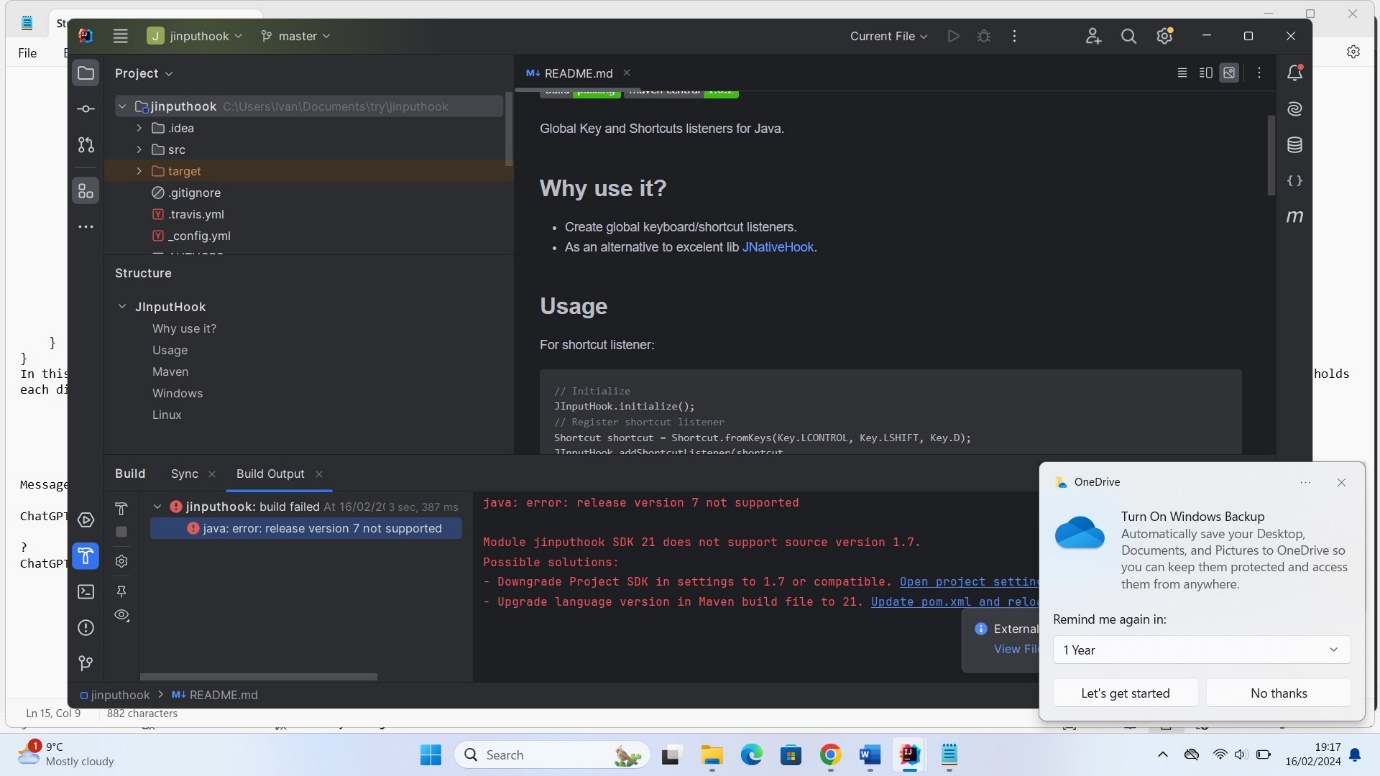
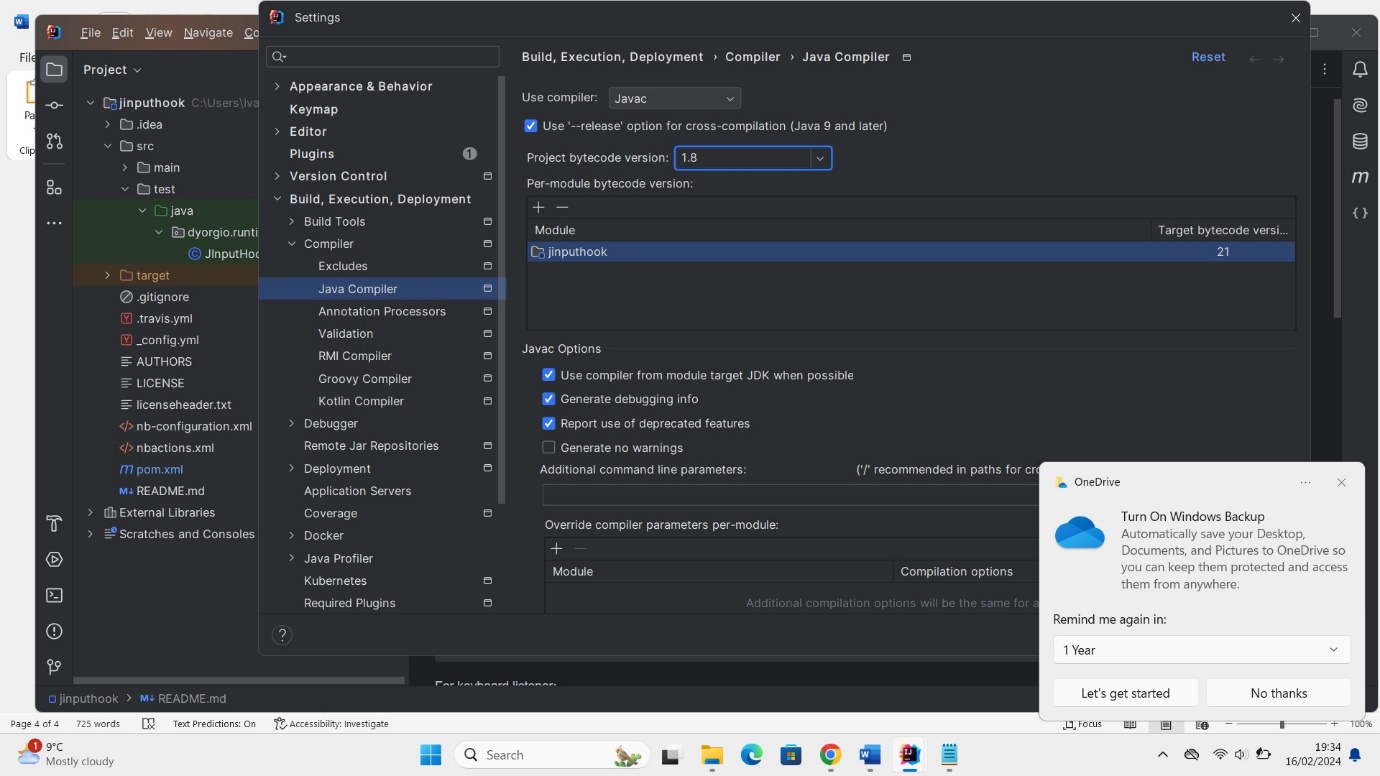
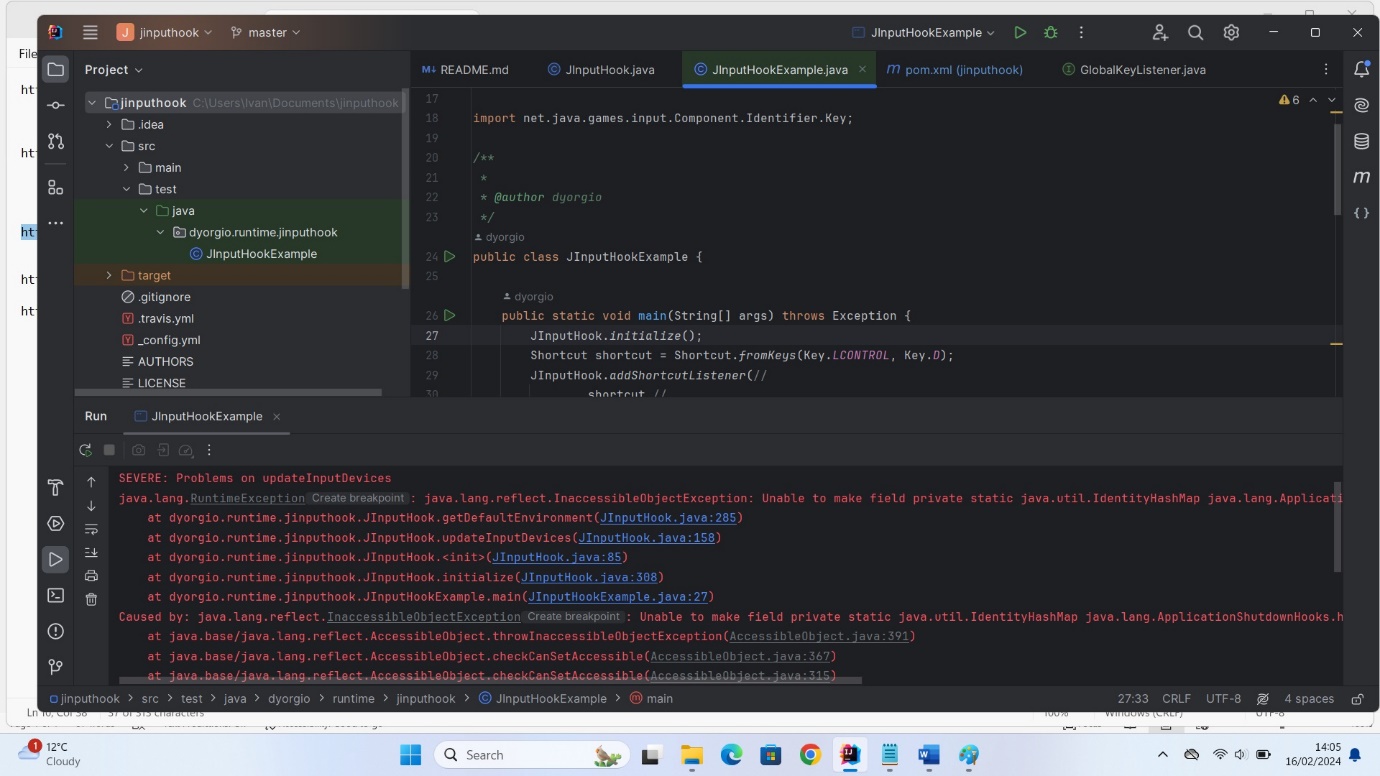


Figure 3 GInputHook release error

To fix this problem, the byte code value in the java compiler must be changed to 1.8. The option for changing this value is the settings menu. Once the new value is entered, the apply and Ok buttons must be pressed.

Figure 4. Changing the byte code value

After this modification, the code is compiled again and a new error arises. This error consists of a error in the code.

Figure 5. Code error message

The problem is that to fix this error a deep reading of the whole code must be done and track all the related parts of code in the entire program. This is considered too large a task and too time consuming in the overall project. Therefore, this program was discarded.

A new program, called system-hook (kristian, 2024), that performs the same task was found. Therefore, the code was downloaded from the repository and opened with IntelliJ. Once the code is compiled, the same error message as the previous solution related with the release version arose. Then, the same procedure as before was followed, the byte code was changed to 1.8.

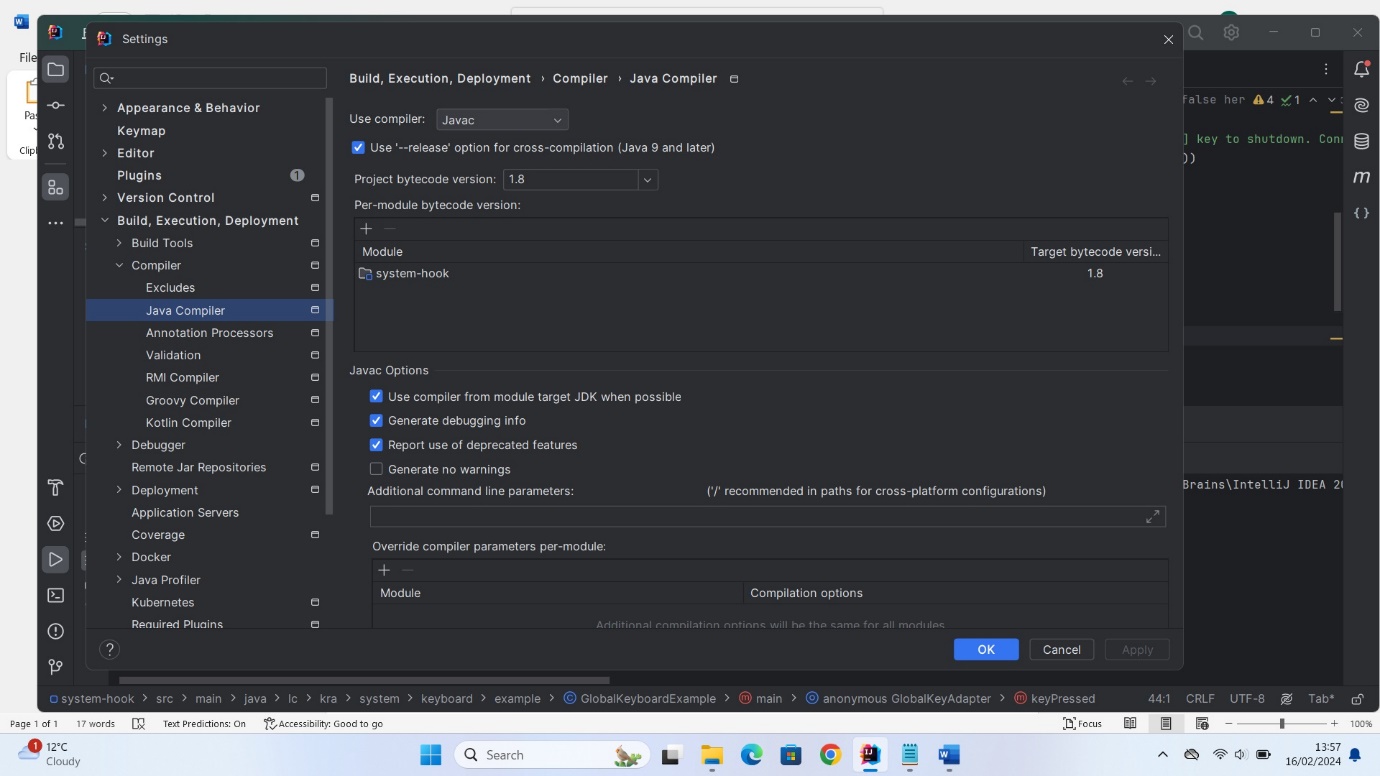


Figure 6. Changing the byte code value

After this modification, the program is compiled again and a new error comes up. A file called SystemHook-Windows-AMD64.DLL is missing from the code. The problem is that this file is not provided by the repository and is not found in the Internet. Therefore, the problem can not be fixed and as a result, this program was discarded from the overall project.

Then, a code that uses the KeyboardFocusManager class was found (Weatherhead, 2024) . This code uses an interface that implements the KeyEventDispatcher and contains a method called DispatchKeyEvent. This interface is used to be able to pass the class to the KeyboardFocusManager class that will call the method. Then a new class that implements the new interface, called KeyDispatcher, was created. The new class developed the method of the interface to handle the key events.

The KeyboardFocusManager class is used in the CentralView class. This is added to the JFrame listener created previously in the project. Then a new method, called setKeyPressed, is added to the CentralView class. This method contains the call to the speak function in the controller class. Then, this method is called from the KeyDispatcher class to announce the key name.

The logic of the process is that the JFrame listener will get all the key events before they access the GUI components in the frame. Afterwards, the JFrame returns the event to the corresponding GUI component. After implementing the code, the program is tested to check out the performance of the global key listener. To avoid confusions, the listeners added to the textbox and the button are removed from the file. Therefore, only the JFrame listener with the KeyboardFocusManager remains in the class as a key listener.

Once the program is run, The interface can announce the key pressed in any part of the frame. The problem is that depending on the key the program repeats the key name twice or three times. Obviously, this is not the result expected, since the voice must say the key name just once. For some reason, the KeyboardFocusManager performs the call to the setKeyPressed method twice or three times depending on the key. Finally, this code was discarded from the project, because the result does not meet the goal correctly.

Explanation of the included code details: internal comments on the code are included, denoted by the use of “//” and “/\*\* \*//”, which precede the individual comments. These comments outline the function of the code which follows the comment.

\*\*\*\*\*\*\*\*\*\* Code of the KeyEventDispatcher class

*/\*\*  
 \* Interface used with the KeyboardFocusManager class  
 \* Implements the method required by the KeyEventDispatcher interface  
 \* Author: Jon Weatherhead  
 \*/*// import the required libraries  
  
import java.awt.\*;  
import java.awt.event.KeyEvent;  
  
public interface KeyEventDispatcher extends KeyEventDispatcher {  
 boolean DispatchKeyEvent(KeyEvent e);  
  
 @Override  
 default boolean dispatchKeyEvent(KeyEvent e) {  
 return false;  
 }  
}

\*\*\*\*\*\*\*\*\*\* End of code of the KeyEventDispatcher class

\*\*\*\*\*\*\*\*\*\* Code of the KeyDispatcher class

*/\*\*  
 \* Class that implements the KeyEventDispatcher interface  
 \* It develops the dispatchKeyEvent method to get the key name and call the setKeyPressed function in the CentralView  
 \* Authors: Jon Weatherhead and Ivan Segade Carou  
 \*/*// import the required libraries  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.KeyEvent;  
  
class KeyDispatcher implements KeyEventDispatcher {  
 // It declares a CentralView variable to be used in the class  
 private CentralView view;  
  
 // It declares the constructor  
 public KeyDispatcher(CentralView view) {  
//The local CentralView variable gets the value of the CentralView object passed to the constructor  
 this.view = view;  
 } // end constructor  
  
 // It develops the method to get the key name and call the speak function  
 public boolean dispatchKeyEvent(KeyEvent e) {  
// get the code of the key pressed  
 int keyCode = e.getKeyCode();  
  
 // with the code, I get the text associated to the corresponding key  
 String keyPressed = KeyEvent.*getKeyText*(keyCode);  
  
 // if the event matches with the key pressed  
 if (e.getID() != KeyEvent.*KEY\_TYPED*)  
 // It calls the setKeyPressed function in the CentralView object  
 view.setKeyPressed(keyPressed);  
  
 // allow the event to be redispatched  
 return false;  
 } // end DispatchKeyEvent  
  
} // end class

\*\*\*\*\*\*\*\*\*\* End of code of the KeyDispatcher class

The CentralView class was modified and only the new code is shown.

\*\*\*\*\*\*\*\*\*\* Code of the CentralView class

… rest of the code …

// in the constructor

// It creates a new KeyboardFocusManager object

// It gets the current cursor

KeyboardFocusManager manager = KeyboardFocusManager.*getCurrentKeyboardFocusManager*();

// The KeyboardFocusManager object adds the event listener with a new KeyDispatcher object

// The KeyDispatcher class has the CentralView as a parameter

manager.addKeyEventDispatcher(new KeyDispatcher(this));

… rest of the code …

*/\*\*  
 \* method that is called by the KeyDispatcher class when a key is pressed  
 \*It calls the speak function in the CentralController class  
 \* It receives the key pressed name as a parameter   
 \* @param key  
 \*/*public void setKeyPressed(String key) {  
 controller.speak(key);  
} // end set key pressed

… rest of the code …

\*\*\*\*\*\*\*\*\*\* End of code of the CentralView class

Then, another code that performs a global key listener, called JavaNativeHook, was found (kwhat, 2024). Once it is downloaded and opened with IntelliJ the program was run. Initially, the code generates the same release version error as seen with other program. The same procedure was followed to fix the problem.

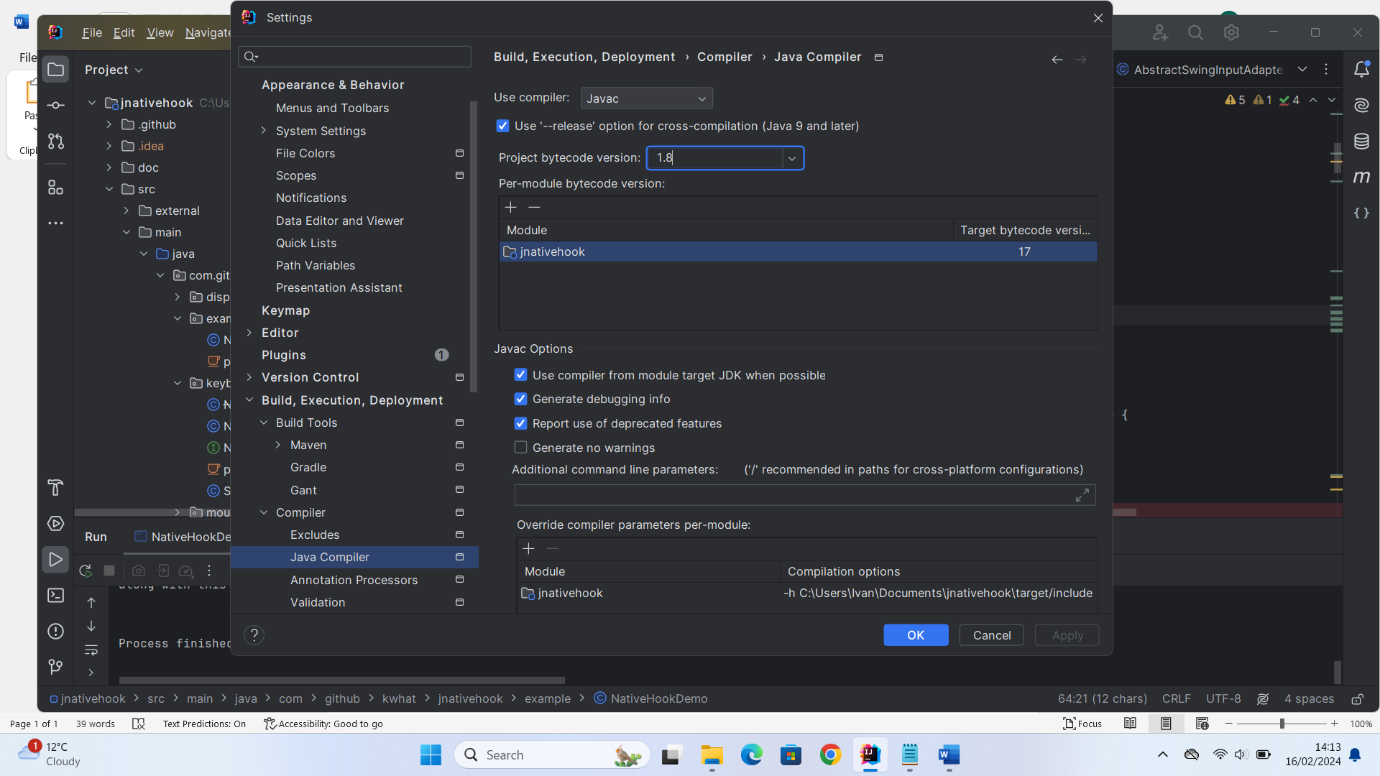


Figure 7. JNative GInputHook release error

Then, the file Jnativehook.dll must be downloaded and paste in the path \JNativeHook,

to get the program to work. This file identifies the devices used in the corresponding operative system. Then, the program was executed. A screen comes up and outputs the keys pressed.

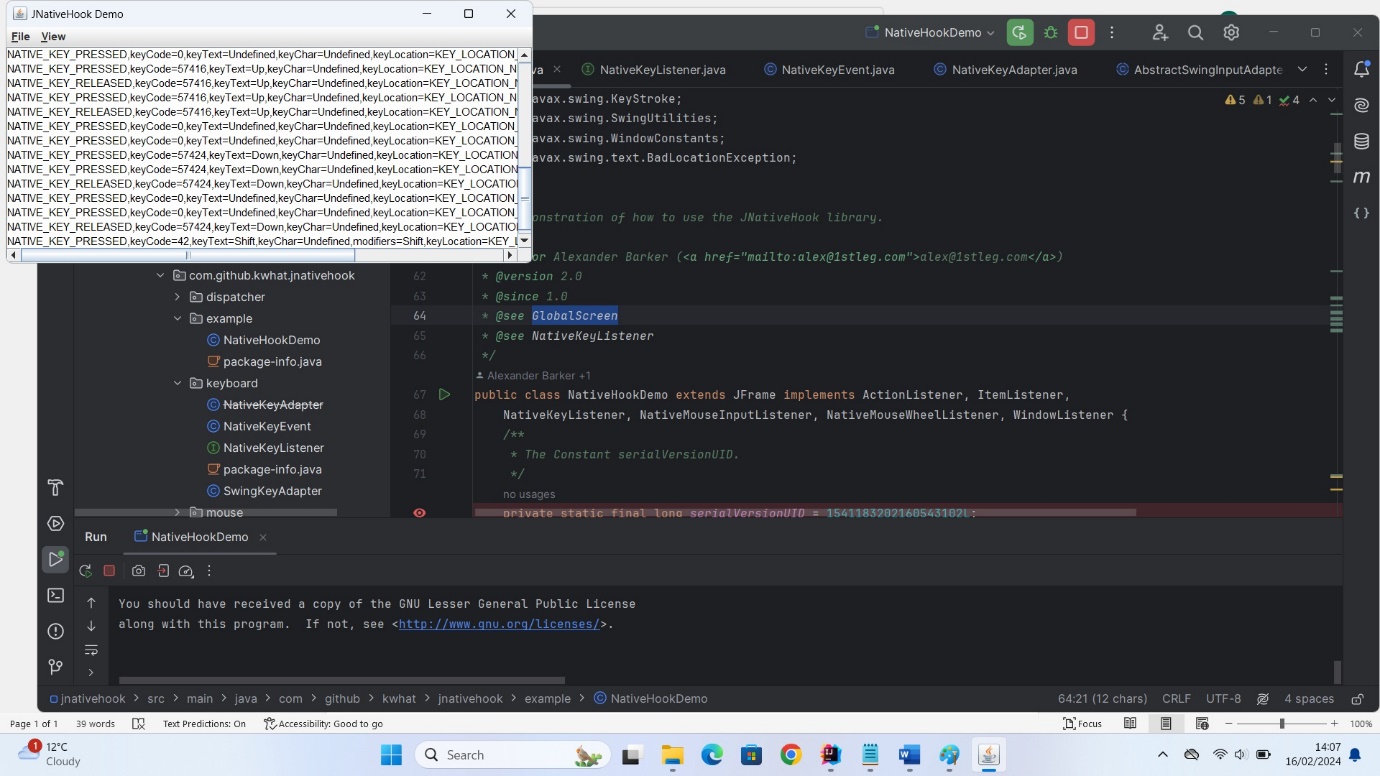


Figure 8. JNative code running

The problem with this code is the complexity of it. The task of integrating this code into the project is very tedious, due to the umber of files and dependencies involved in the program. This code also handles mouse events, but they are not needed for this project. However, it seems hard to get only the keyboard files and import them into the project. The sprint is finishing and the implementation of this code into the project would take too much time, and the global key listener is not a relevant feature of the project. For this reason, the code is discarded.

Although the global key listener is not implemented into the project, the fact that a global key listener can be implemented in Java is proved. Even though this task requires external code to get it to work.

# Chapter 5. Sprint 4

**Sprint 4: How to create a custom dataset, add the overlapping and compare the own approach**

Start date: 04/03/2024

End date: 17/03/2024

|  |  |  |
| --- | --- | --- |
| TASK | DETAILS | STATUS |
| 1 | Research how to create a custom dataset | Completed |
| 2 | Adding overlapping to the description | Completed |
| 3 | Comparing own approach with an alternative | Completed |

## How to create a custom dataset

There are two things that are necessary for creating a custom data set. The first one is a number of images that will be used during the process. Secondly the annotations of the images. A folder for the project must be created. Within this folder a two subfolders must be added, one for the images and the second for the annotations.

The images used must have several and different files for each object to be detected. These images must have the objects in different positions and perspectives for a better training and as a result, a better detection.

The annotations are the way of categorising or labelling an image in order to identify the object in the image. The label provides the features of the object that the machine learning model will use to detect it. There are two formats: XML and PASCAL VOC. The PASCAL VOC has the drawback that it cannot be used by non-object detection models.

A full example of a XML annotation is(IBM, 2024):

\*\*\*\*\*\*\*\*\*\* Example of XML annotation

<annotation>

<folder>JPEGImages</folder>

<filename>./JPEGImages/01\_01.jpg</filename>

<path>/JPEGImages/</path>

<source>

<database>Unknown</database>

</source>

<size>

<width>2000</width>

<height>2000</height>

<depth>3</depth>

</size>

<segmented>0</segmented>

<object>

<name>mitoses</name>

<pose>Unspecified</pose>

<truncated>0</truncated>

<difficult>0</difficult>

<bndbox>

<xmin>935</xmin>

<ymin>880</ymin>

<xmax>1015</xmax>

<ymax>960</ymax>

</bndbox>

</object>

</annotation>

\*\*\*\*\*\*\*\*\*\* End of example of XML annotation

The complexity of the label depends on the level of the project. A label can just be a simple description i.e. dog, or can be more specific i.e. dog breeds: boxer, Terrier or Bergamasco. The annotation also provides features such as the height and width of the corresponding object in the picture.

The task of labelling images can be tedious work, specially if the project contains a large amount of images. For this reason there are some tools that facilitate the task. With these applications users can select a group of images that belong to the same label class and label all of them at once. These tools generate the annotations in the correct format.

Examples of labelling applications are: LabelImg (PyPI, 2024) for Python. This is a basic tool that can be used for small or medium projects. For large projects the CVAT application (CVAT, 2024) or SuperAnnotate (SuperAnnotate, 2024)or SuperVisely (SuperVisely, 2024) are more suitable, providing more features, i.e. a team can work with the same project annotating the images for that project.

It is crucial that the annotation file name matches with the corresponding image file name. The model binds the annotation with the image through the file names. It is recommendable to split the data into training and validation parts. The first part is used to train the model and the second is to test that the model learnt properly. Typically, data is split 80% for training purpose and 20% for validation.

In the case of working with TensorFlow an additional text file is required. This text file is called Label Map. In this file, each class name is linked to an integer number that TensorFlow uses in the training and testing phases.

\*\*\*\*\*\*\*\*\*\* Example of label map file

item {

id: 1

name: "car"

item {

id: 2

name: "bike"

\*\*\*\*\*\*\*\*\*\* End of example of label map file

A pretrained model is required when developers when to create their own custom dataset. The reason of this is that pretrained models already have the architecture to interpretate and read images. Otherwise, developers would have to create that architecture by themselves and that is a very tedious task. Therefore, by using pretrained models in this process developers save a large amount of time and resources.

A popular framework for working with machine learning and it also used in object detection is TensorFlow (TensorFlow, 2024) . This is an open source framework created by Google in the Python programming language.

TensorFlow provides tools for processing and load data. Also, it can use pre-trained models or created new custom models from the scratch. TensorFlow is multi-platform, it can be used whether in the cloud, in browser or in device. Moreover, TensorFlow can run models while these are in the production phase and it can maintain them. Therefore, developers can create complex neuron networks avoiding large amount of effort.

One of the main features of TensorFlow is the used of TFRecord format files (TensorFlow, 2024). TensorFlow converts the data and annotations in this format for a more efficient throughput of the process. These records are binary files, which reduce the size of data to be process. This is more notorious specially in large datasets. The inconvenient is that this type of format, unlike the other formats is non readable by humans, which is a problem in the case an error with these files arise.

Despite being TensorFlow designed to be used in Python, it can be used in other programming languages thanks to the APIs created. The most developed APIs are for C++, JavaScript and Java. There bindings for other programming languages such as C#, Ruby or Perl. However these ones are not so well developed nor supported.

Another open source framework for machine learning development is PyTorch (PyTorch, 2024). This framework was created in Python by Meta and uses the Torch library (Pypi, 2024). The Torch library is a scientific computing framework and was created originally for the LuaJIT project (Luajit, 2024). PyTorch is used for training and build neural networks and develop deep learning.

One of the main features of PyTorch and differentiates from TensorFlow is that PyTorch uses Dynamic Computation Graphs, when TensorFlow uses Static Graphs. This allows to build and modify neuron networks on-the-fly. This facilitates to work during the runtime (Yasar & Lewis, 2024) .

PyTorch just provides one API for other programming languages. This API is for C++ (PyTorch, 2024). This API can be used with Java by using the corresponding bindings. Another drawback of PyTorch is that it does not provide a large documentation, unlike TensorFlow.

There are some online tools that assist in the task of creating custom datasets, such as IBM (IBM, 2024). With the IBM Cloud and its Cluster Management Console, developers can create their own custom datasets through an automated process without any coding. Despite all, developers must follow the basic rules commented at the top of this document, such as having a folder for images, another for annotations and these ones must have XML format.

Google also provides a similar approach to IBM through the Google Cloud Console and the Vertex AI (Vertex AI , 2024). The steps to be followed are similar to the IBM Cluster Management Console. Both have the dataset option on their menus and the developers must upload their images and annotations. Then, developers must select the options that suit their project, such as the model used. Finally, the tool trains the dataset and creates the new custom dataset with a model as a result.

There are some concepts that developers must know when they work with machine learning and object detection.

Inference: it is the feature of making decisions after processing the image. It uses a trained model to detect the objects in the image and comes to the conclusions of the objects that will be showed as a result (Doyle, 2024).

Epoch: It is a entire iteration of the dataset in the training phase. The algorithm processes the dataset entirely, this is called a Epoch. Typically, Between 20 and 50 Epochs are used in the initial review (Simple learn, 2024) (Zhang, et al., 2021).

Neural network: It is a machine learning model that process data in a way similar to human neurons and make decisions based on that process. Neural network has layers of nodes, one input layer, one or more hidden layers and one output layer. Each node has its own threshold and weight. When the output of a node surpasses the corresponding threshold, the mode sends the data to the following layer. Therefore, the output of a node is the input of the node of the following layer.

On determining the input layer, weights are assigned to subsequent layers. These weights determine how important each variable is and are multiplied by the inputs. The product is summed to an activation function to determine the output which is compared to the threshold value (IBM, 2024).

Convolutional neural network (CNN): It is a variant of neural network that uses convolutional operations to process the images that are represented in a array of pixels. CNN differs from traditional neural network in its complexity of analysis of multiple layers, looking for patterns and relationships during the process. A hierarchy of layers is used to process the images. The complexity of the analysis increases with the depth performed, with the initial layers looking at basic image parameters and later layers are looking at more detailed parameters (IBM, 2024). An improvement of CNN is that can identify the main features without the need of human supervision. Some examples of convolutional neural network are: AlexNet, MobileNet, VGG-16 or Resnet.

Backbone: It is also called feature extractor network. The backbone computes the input images in order to provide a feature representation. This task is done in a high level when processing image. Then, this feature representation is used as a pattern in the following layers of the process of the image for a deeper computation. With his, all the layers can communicate among them and supply the result. In YOLO (You Only Look. Once) and SSD (Single Shot Detector) use convolutional neural networks as backbones (Li, et al., 2018)

## Adding overlapping to the description

A new feature to the description of the picture is added. The reason for this is to provide a deeper detailed information about the objects within the image. The program detects when objects are overlapped and calculates the percentage of the object area is overlapped by the other object area. For this purpose two more functions are added to the ObjectFunctionality class. The first one is called calculateOverlapArea. This function calculates the area of the common area between two objects. The second function is called calculatePercentageOfArea. This function calculates the percentage of the object area that is covered by the common area with the other object.

\*\*\*\*\*\*\*\*\*\* New code of the ObjectFunctionality class

… rest of the code …

*/\*\*  
 \* Method that calculates the common area between two objects using the pixel values  
 \* It receives two parameters. The two objects to check if there is overlapping between them  
 \*It returns the numeric value of the common area  
 \*  
 \* @param box1  
 \* @param box2  
 \* @return  
 \*/* public static int calculateOverlapArea(ObjectDetected box1, ObjectDetected box2) {  
 // Calculate the coordinates of the overlapping region  
 int overlapX = Math.*max*(box1.getPositionX(), box2.getPositionX());  
 int overlapY = Math.*max*(box1.getPositionY(), box2.getPositionY());  
  
 // Calculate the dimensions of the overlapping region  
 int overlapWidth = Math.*min*(box1.getPositionX() + box1.getWidth(), box2.getPositionX() + box2.getWidth()) - overlapX;  
 int overlapHeight = Math.*min*(box1.getPositionY() + box1.getHeight(), box2.getPositionY() + box2.getHeight()) - overlapY;  
  
 // Check for non-overlapping rectangles  
 if (overlapWidth <= 0 || overlapHeight <= 0) {  
 return 0; // No overlap  
 }  
  
 // Calculate overlap area  
 return overlapWidth \* overlapHeight;  
 } // end calculate overlap area  
  
  
 */\*\*  
 \* Method that calculate the percentage of the object area that is covered vy the common area with the other object  
 \* It receives the object area value and the common area value  
 \* It returns the numeric value of the object area that is overlapped  
 \* @param boxArea  
 \* @param overlapArea  
 \* @return  
 \*/* public static int calculatePercentageOfArea(int boxArea, int overlapArea) {  
 return (overlapArea \* 100) / boxArea;  
 } // end calculate percentage of area  
} // end class

\*\*\*\*\*\*\*\*\*\* End of new code of the ObjectFunctionality class

These two new methods are called from the ObjectDetectorDrive class. Two new for loops are added to the main function to iterate the ArrayList with all the detected objects and make the comparison among all of them.

The first for loop starts from the first item and finishes in the second last item. Then, the second one starts at the position of the first for loop plus one position up to the end of the ArrayList. The first iteration is the reference object and the second one iterates the objects that are compared to the first one.

The first for loop does not reach the end of the ArrayList because the last item already was compared between the rest of the items, Therefore, there is no need to reach that position. The second for loop starts at the position plus one because that is the current item and it does not make any sense to compare it to itself.

\*\*\*\*\*\*\*\*\*\* New code of the ObjectDetectorDrive class

… rest of the code …  
  
//Loop for the reference object  
for (int x = 0; x < listOfObjects.size() - 1; x++) {  
 // loop that compares the reference object to the rest of objects  
 for (int y = x + 1; y < listOfObjects.size(); y++) {  
 // It stores the value of the common area   
 int result = ObjectFunctionality.*calculateOverlapArea*(listOfObjects.get(x), listOfObjects.get(y));  
   
 // if the area is bigger than zero means there is common area   
 if (result > 0) {  
 // It displays the percentage of the reference object covered by the new one  
 System.*out*.println("The " + listOfObjects.get(x).getName() + " is overlapping by " +  
 listOfObjects.get(y).getName() + ObjectFunctionality.*calculatePercentageOfArea*((listOfObjects.get(x).getWidth() \* listOfObjects.get(x).getHeight()), result)  
 + "%");  
  
 // It displays the percentage of the new object covered by the reference object   
 System.*out*.println("The " + listOfObjects.get(y).getName() + " is overlapping by " +  
 listOfObjects.get(x).getName() + ObjectFunctionality.*calculatePercentageOfArea*((listOfObjects.get(y).getWidth() \* listOfObjects.get(y).getHeight()), result)  
 + "%");  
  
 } // end if  
 } // end for y  
  
} // end for x

} // end of main

… rest of the code …

\*\*\*\*\*\*\*\*\*\* End of the new code of the ObjectDetectorDrive class

## Comparing results

A question that arises with this investigation is: Are the results generated acceptable?. To solve this question, a comparison of the results generated with this investigation with an alternative solution is made. For this purpose, an alternative object detection approach is found (MediaPipe, 2024), which is written in Python in a Google Colab notebook. This solution uses the EfficientDet-Lite0 model trained with COCO dataset that comes from EfficientDet (Tan, Pang, & V. Le, 2024)and .

After running the code the result is some unexpected. The approach just detects one object, the bicycle (the biggest object in the picture and discarding the smaller objects). Even the data of the bounding boxes differs one from each other.

|  |  |  |
| --- | --- | --- |
|  | Own approach | Alternative approach |
| X coordinate | 162 | 283 |
| Y coordinate | 207 | 237 |
| Width | 594 | 498 |
| Height | 588 | 387 |

Table 1Comparing the codes own and alternative

\*\*\*\*\*\*\*\*\*\* Result of the alternative approach

detection\_result

DetectionResult(detections=[Detection(bounding\_box=BoundingBox(origin\_x=283, origin\_y=237, width=498, height=387), categories=[Category(index=None, score=0.73828125, display\_name=None, category\_name='bicycle')], keypoints=[])])

\*\*\*\*\*\*\*\*\*\* End of result of the alternative approach

\*\*\*\*\*\*\*\*\*\* Result of the own approach

[INFO ] - [

{"class": "car", "probability": 0.99991, "bounds": {"x"=0.611, "y"=0.137, "width"=0.293, "height"=0.160}}

{"class": "bicycle", "probability": 0.95385, "bounds": {"x"=0.162, "y"=0.207, "width"=0.594, "height"=0.588}}

{"class": "dog", "probability": 0.93752, "bounds": {"x"=0.168, "y"=0.350, "width"=0.274, "height"=0.593}}

\*\*\*\*\*\*\*\*\*\* End of result of the own approach

A screenshot of a computer

Description automatically generated

Figure 9. Result of the alternative approach

A screenshot of a computer

Description automatically generated

Figure 10. Result of the alternative approach

A screenshot of a computer

Description automatically generated

Figure 11. Result of the own approach

As a conclusion, the own approach, despite being created in Java, a not such a common programming language in the artificial intelligence world, can even provide better results than a Python solution, one of the most used programming languages in the artificial intelligence world.

# Chapter 6. Sprint 5

**Sprint 5: Integrate the interface into the DJL and adding the algorithm to determine the depth of the picture**

Start date: 18/03/2024

End date: 7/04/2024

|  |  |  |
| --- | --- | --- |
| TASK | DETAILS | STATUS |
| 1 | Integrate the interface into the DJL | Completed |
| 2 | Adding depth by overlapping algorithm | Completed |
| 3 | Adding depth by central point algorithm | Completed |
| 4 | Adding depth by bottom algorithm | Completed |
|  |  |  |

## Integrating the interface in to the DJL

In order to integrate the interface in to the Deep Java Library along with the rest of the code, a new package, called Ivan, is created and all the necessary classes are moved to this new package. With this, all the classes that were used for practising are separated from the code needed for the investigation.

This new package contains the follow classes: ObjectDetected, ObjectDetector, ObjectFunctionality, CentralController and CentralView. A new class, called IvanDrive, is created. This class is the driver class, the class that centralises the execution of the program. The code held in the ObjectDetectorDrive class is moved to the CentralController class.

Once the classes are implemented, the code is executed to check that the implementation is correct. After running the code an error is displayed related to the CentralView class. Apparently, the class must be serialised, a problem that did not happen when the interface was executed outside the Deep Java Library. To fix this problem, a line of code must be added to ignore the problem “@SuppressWarnings("serial")”

A screenshot of a computer

Description automatically generated

Figure 12. Error message.

Then the MyVoice class is implemented in to the package and consequently the FreeTTS library is imported. However, when importing this library The Deep Java Library does not recognise it. This is the same problem as when importing the JSON Simple library. Unfortunately, an alternative way to import this library was not found. Therefore, the program does not provide a voice output.

A screenshot of a computer program

Description automatically generated

Figure 13. Error message importing the FreeTTS

## Adding depth by overlapping algorithm

A question that arises when using the detected object technique is, if it is possible to identify the objects that are in front of other objects. The detected object code just provides the coordinates and the size of each object in the picture. With this information it is impossible to make an accurate prediction. However, data that is calculated in this investigation is the overlapping between objects. With this piece of data, some calculations can be made to try to identify the depth of the objects.

There is one circumstance where an accurate prediction can be made and the prediction is totally correct. If the overlapping area between two objects covers 100% of one object, it is clear that the inner object is in front of the other. If the situation was the other way around, the inner object would not be visible at all and the object detection code could not detect it.

With this logic, more calculations can be made. It is assumed that if the overlapping area covers between 90% and 99%, the likelihood that the object is in front of the other is high. The other way around, only 10% of the object would be visible and it is not very likely that the object detection code could detect it. Likewise, the same logic is used if the overlapping is between 76% and 89%, but the likelihood decreases. Then, if the percentage is 75% or less, it is possible that the object detection code can detect an object even if it is behind. Therefore, the code can not specify which object is front.

|  |  |
| --- | --- |
| Percentage | Definition |
| 100 | The likelyhood is total |
| 90 – 99 | It is more than likely |
| 76 – 89 | It is likely |
| < 75 | No prediction |

Table 2 Ranges for the likelihood

For this purpose, a new class, called OverlappingAlgorithm, is created. This new class now contains the methods that are related to the overlapping calculations: calculateOverlapArea and calculatePercentageOfArea. Also, the code that was hosted in the ObjectDetectorDrive is moved to this class in a getOverlap method. A new method, called, calculateDepth, is created. This method is similar to getOverlap, but rather than displaying just the percentages, uses the percentages to calculate the likelihood of the depth.

\*\*\*\*\*\*\*\*\*\* Code of the OverlappingAlgorithm class.

package ai.djl.examples.Ivan;  
  
*/\*\*  
 \* Class that holds all the methods related to the overlapping calculations  
 \*/*// importing the necessary libraires  
import java.util.List;  
  
public class OverlappingAlgorithm {  
  
 */\*\*  
 \* This method iterates the lis of objects and displays the percentage of the common overlapping area between two objects  
 \* It receives the list of objects as a List  
 \* It returns a String with the final result  
 \* It is public to allow the access outside the class   
 \* @param listOfObjects  
 \* @return  
 \*/* public static String getOverlap(List<ObjectDetected> listOfObjects){  
 String overlap = "";  
  
 //Loop for the reference object  
 for (int x = 0; x < listOfObjects.size() - 1; x++) {  
 // loop that compares the reference object to the rest of objects  
 for (int y = x + 1; y < listOfObjects.size(); y++) {  
 // It stores the value of the common area  
 int result = *calculateOverlapArea*(listOfObjects.get(x), listOfObjects.get(y));  
  
 // if the area is bigger than zero means there is common area  
 if (result > 0) {  
 // It displays the percentage of the reference object covered by the new one  
 overlap += "\nThe " + listOfObjects.get(x).getName() + " is overlapping by " +  
 listOfObjects.get(y).getName() + *calculatePercentageOfArea*((listOfObjects.get(x).getWidth() \* listOfObjects.get(x).getHeight()), result)  
 + "%";  
  
 // It displays the percentage of the new object covered by the reference object  
 overlap += "\nThe " + listOfObjects.get(y).getName() + " is overlapping by " +  
 listOfObjects.get(x).getName() + *calculatePercentageOfArea*((listOfObjects.get(y).getWidth() \* listOfObjects.get(y).getHeight()), result)  
 + "%";  
  
 } // end if  
 } // end for y  
  
 } // end for x  
  
 return overlap;  
 } // end get overlap

public static int calculateOverlapArea(ObjectDetected box1, ObjectDetected box2) {

// Same code as before

private static int calculatePercentageOfArea(int boxArea, int overlapArea)

// Same code as before

*/\*\*  
 \* Method that determines if one object may be in front of the other and the likelihood  
 \* It receives the names of the objects and the percentage of area  
 \* It returns a String with the result \* @param name1  
 \* It is private to avoid accessing outside the class  
 \*  
 \* @param name2  
 \* @param percentage  
 \* @return  
 \*/* private static String calculateLikelyhood(String name1, String name2, int percentage) {  
 if (percentage == 100)  
 return "\nThe " + name1 + " is in 100% in front of the " + name2;  
 else if (percentage >= 90)  
 return "\nit is more than likely that the " + name1 + " is in front of the " + name2;  
 else if (percentage > 75)  
 return "\nIt is likely that the " + name1 + " is in front of the " + name2;  
  
 return "";  
 } // end calculate likelyhood  
  
 */\*\*  
 \* Method that iterates the list of objects and calculates if one object may be in front of other object  
 \* It receives the list of objects as a List  
 \* It returns a String with the result  
 \* It is public to allow the access outside the class  
 \*  
 \* @param listOfObjects  
 \* @return  
 \*/* public static String calculateDepth(List<ObjectDetected> listOfObjects) {  
 String depth = "";  
  
 for (int x = 0; x < listOfObjects.size() - 1; x++) {  
 for (int y = x + 1; y < listOfObjects.size(); y++) {  
 // It stores the value of the common area  
 int result = *calculateOverlapArea*(listOfObjects.get(x), listOfObjects.get(y));  
  
 int percentage = *calculatePercentageOfArea*((listOfObjects.get(x).getWidth() \* listOfObjects.get(x).getHeight()), result);  
  
 depth = *calculateLikelyhood*(listOfObjects.get(x).getName(), listOfObjects.get(y).getName(), percentage);  
  
 percentage = *calculatePercentageOfArea*((listOfObjects.get(y).getWidth() \* listOfObjects.get(y).getHeight()), result);  
  
 depth = *calculateLikelyhood*(listOfObjects.get(y).getName(), listOfObjects.get(x).getName(), percentage);  
  
  
 }// end for y  
 } // end for x  
  
 if (depth.equals(""))  
 depth = "No information is available";  
  
 return depth;  
 } // end calculate depth  
  
} // end class

\*\*\*\*\*\*\*\*\*\* End of code of the OverlappingAlgorithm class.

## Adding depth by central point algorithm

Another calculation that may identify when an object is in front of another object is the central points locations of the bounding boxes. In this approach to determine which object is in front of the other it checks if the central point of one bounding box is inside of the other bounding box. Obviously, there may be situations where both central points can be inside of the other object. These cases are discarded and there is not a conclusion.

For this purpose, a new class, called CentralPointAlgorithm, is created. This class contains two method: calculateDepth and calculateDistance. The calculateDepth method returns the overall result and is called from the CentralController class.

\*\*\*\*\*\*\*\*\*\* Code of the CentralPointAlgorithm class.

package ai.djl.examples.Ivan;  
*/\*\*  
 \* Class that contains the methods to calculate if one object is in front of another object through the central points of the bounding boxes  
 \*/*// importing the necessary libraries  
  
import java.util.List;  
  
  
public class CentralPointAlgorithm {  
 */\*\*  
 \* Method that calculates the distance between the central point of one object and the bounding box of the other object  
 \* It receives an int array with the X and Y coordinates of the central point, and the ObjectDetected object that contains the data of the bounding box  
 \* It returns the distance between the central point and the bounding box.  
 \* If the central point is inside, the method returns 0  
 \* It is private to avoid accessing outside the class  
 \*  
 \* @param centralPoints  
 \* @param od  
 \* @return  
 \*/* private static double calculateDistance(int[] centralPoints, ObjectDetected od) {  
 // Declaring the variables for the calculation  
 int pointX = centralPoints[0];  
 int pointY = centralPoints[1];  
 int boxX = od.getPositionX();  
 int boxY = od.getPositionY();  
 int width = od.getWidth();  
 int height = od.getHeight();  
  
 // If the point is inside the bounding box, return 0  
 if (pointX > boxX && pointX < boxX + width && pointY > boxY && pointY < boxY + height)  
 return 0;  
  
  
 // Calculating the distance from the X and Y coordinates to the boudning box  
 int closestX = Math.*max*(boxX, Math.*min*(pointX, boxX + width));  
 int closestY = Math.*max*(boxY, Math.*min*(pointY, boxY + height));  
 double deltaX = pointX - closestX;  
 double deltaY = pointY - closestY;  
  
 // It returns the distance  
 return Math.*sqrt*(deltaX \* deltaX + deltaY \* deltaY);  
  
 }// end calculate distance  
  
 */\*\*  
 \* Method that iterates the list of the objects and compares one object with the rest of the objects individually  
 \* It receives the list of objects as a List  
 \* It returns the result of the iteration  
 \* It is public to allow the access outside the class  
 \*  
 \* @param listOfObjects  
 \* @return  
 \*/* public static String calculateDepth(List<ObjectDetected> listOfObjects) {  
 // Declaring the String that will contain the final result of the iteration  
 String result = "";  
// for loops that iterate the list of objects for the two objects to be compared  
 for (int x = 0; x < listOfObjects.size() - 1; x++) {  
 for (int y = x + 1; y < listOfObjects.size(); y++) {  
 // It calculates the distances for the two objects  
 int distance1 = (int) *calculateDistance*(listOfObjects.get(x).getCentralPoint(), listOfObjects.get(y));  
 int distance2 = (int) *calculateDistance*(listOfObjects.get(y).getCentralPoint(), listOfObjects.get(x));  
// if the distance of the first object is inside, but the second is not  
 if (distance1 == 0 && distance2 != 0)  
 result += "\nThe " + listOfObjects.get(x).getName() + " is in front of the " + listOfObjects.get(y).getName();  
  
 // if the distance of the second object is inside, but the first is not  
 if (distance2 == 0 && distance1 != 0)  
 result += "\nThe " + listOfObjects.get(y).getName() + " is in front of the " + listOfObjects.get(x).getName();  
  
 } // end for y  
 } //end for x  
  
 // If no detection is found a message is returned  
 if (result.equals(""))  
 result = "No information is available";  
  
 // It returns the result of the overall calculation  
 return result;  
 } // end calculate depth  
  
} // end class

\*\*\*\*\*\*\*\*\*\* End of code of the CentralPointAlgorithm class.

## Adding depth by bottom algorithm

The third way that is used in this investigation is to check when two objects have an overlapping area, the object that is located at the bottom, is considered to be in front.

For this purpose, a new class, called BottomAlgorithm, is created. This class contains three methods: calculateDepth, calculateBottom and calculateOverlapArea. As this calculation is based on the position of the objects according to the overlapping area, the method used previously in the investigation is used in this class as well. To ensure that there is a level of separation in the height of the two objects, the object at the bottom must be at least 10% beyond the bottom of the other object.

\*\*\*\*\*\*\*\*\*\* Code of the BottomAlgorithm class.

package ai.djl.examples.Ivan;  
  
*/\*\*  
 \*Class that holds all the methods related to the bottom algorithm to determine which object is at the bottom  
 \*/*// Importing the necessary libraries  
import java.util.List;  
  
public class BottomAlgorithm {  
  
 */\*\*  
 \* Method that iterates the list of objects and calculates if one object may be in front of other object  
 \* It receives the list of objects as a List  
 \* It returns a String with the result  
 \* It is public to allow the access outside the class  
 \*  
 \* @param listOfObjects  
 \* @return  
 \*/* public static String calculateDepth(List<ObjectDetected> listOfObjects) {  
 String depth = "";  
  
//Loop for the reference object  
 for (int x = 0; x < listOfObjects.size() - 1; x++) {  
 // loop that compares the reference object to the rest of objects  
 for (int y = x + 1; y < listOfObjects.size(); y++) {  
 int result = *calculateOverlapArea*(listOfObjects.get(x), listOfObjects.get(y));  
// if statement if there is overlapping  
 if (result > 0)  
 depth += *calculateBottom*(listOfObjects.get(x), listOfObjects.get(y));  
  
 } // end for y  
 } // end for y  
  
 // No resultswere generated  
 if (depth.equals(""))  
 depth = "No information is available";  
  
 return depth;  
 } // end of calculate depth by bottom  
  
  
 */\*\*  
 \* Method that calculates the common area between two objects using the pixel values  
 \* It receives two parameters. The two objects to check if there is overlapping between them  
 \* It returns the numeric value of the common area  
 \*It is private to avoid accessing outside the class  
 \* @param box1  
 \* @param box2  
 \* @return  
 \*/* private static int calculateOverlapArea(ObjectDetected box1, ObjectDetected box2) {  
 // Calculate the coordinates of the overlapping region  
 int overlapX = Math.*max*(box1.getPositionX(), box2.getPositionX());  
 int overlapY = Math.*max*(box1.getPositionY(), box2.getPositionY());  
  
 // Calculate the dimensions of the overlapping region  
 int overlapWidth = Math.*min*(box1.getPositionX() + box1.getWidth(), box2.getPositionX() + box2.getWidth()) - overlapX;  
 int overlapHeight = Math.*min*(box1.getPositionY() + box1.getHeight(), box2.getPositionY() + box2.getHeight()) - overlapY;  
  
 // Check for non-overlapping rectangles  
 if (overlapWidth <= 0 || overlapHeight <= 0) {  
 return 0; // No overlap  
 }  
  
 // Calculate overlap area  
 return overlapWidth \* overlapHeight;  
 } // end calculate overlap area  
  
 */\*\*  
 \* Method that calculates which object has the location at the bottom of the other object  
 \* the object at the bottom has to be at least from the 10% of the height of the other object  
 \* It receives the two object to compare their locations  
 \* It returns a String variable with the result of the calculation  
 \* It is private to avoid accessing outside the class  
 \*  
 \* @param od  
 \* @param od1  
 \* @return  
 \*/* private static String calculateBottom(ObjectDetected od, ObjectDetected od1) {  
//declaring the variable with the percentage of the height which is 10%  
 double LEVELBOTTOM = 0.1;  
  
 // if statements that compare the object locations and generate the result  
 if (od.getPositionY() > od1.getPositionY() + (od1.getHeight() \* LEVELBOTTOM))  
 return "\nThe " + od.getName() + " is in front of the " + od1.getName();  
  
 // if no object fulfil the condition it returns an empty String  
 if (od1.getPositionY() > od.getPositionY() + (od.getHeight() \* LEVELBOTTOM))  
 return "\nThe " + od1.getName() + " is in front of the " + od.getName();  
  
 return "";  
 } // end calculate depth  
  
  
} // end class

\*\*\*\*\*\*\*\*\*\* End of code of the BottomAlgorithm class.

## Implementing the algorithms

After creating the three algorithms, the next step is to modify the CentralView and the CentralController classes to adapt them to the new specifications. The CentralView class is modified by adding a new panel that contains 4 new buttons in the interface. Likewise, the CentralController class is modified by changing the “speak” method name for “load” that replaces the main method in the ObjectDetectorDrive class and this method calls all the calculateDepth methods that correspond to the required algorithm class. Then the method stores all the results in their corresponding variables. Also, a new method, called displayMessage, is added. This new method displays the information previously processed according to the selection of the user int the interface.

To avoid misspellings when selecting the option in the displayMessage method, a new enum class, called Choice, is created. This enum class contains the possible options to be executed and prevents entry of wrong options in the code.

\*\*\*\*\*\*\*\*\*\* Code of the Choice enum class.

package ai.djl.examples.Ivan;  
// emum with the different options available in the interface   
enum Choice {  
 *DESCRIPTION*,  
 *OVERLAPPING*,  
 *DEPTHOVERLAP*,  
 *DEPTHCENTRALPOINT*,  
 *DEPTHBOTTOM*} // end enum

\*\*\*\*\*\*\*\*\*\* End of code of the Choice enum class.

Moreover, the code that processes the location and size of each object in the picture is moved to a new method, called getDescription, in the CentralController class. With this, the class gains clarification and avoids the code smell of creating methods too large.

\*\*\*\*\*\*\*\*\*\* Code of the CentralController class.

package ai.djl.examples.Ivan;  
*/\*\*  
 \* Class that instantiates the interface and handles all the methods and classes that the program needs  
 \*/*

// The same imports as before

… rest of the code …

public class CentralController {  
// declaration of the CentralView object  
 CentralView centralView;  
 // declaration of the variables to work with the object detection process  
 List<ObjectDetected> listOfObjects = new ArrayList<ObjectDetected>();  
 String description = "No information available";  
 String overlapping = "No information available";  
 String depthOverlap = "No information available";  
 String depthCentralPoint = "No information available";  
 String depthBottom = "No information available";

… rest of the code …

*/\*\*  
 \* Method that calls all the required methods and classes to generate all the possible details that the program works with  
 \* It receives the String with the path to the picture to be processed  
 \* It is public to allow the access outside the class  
 \* It throws the required exceptions in the case an error related to these occurs  
 \*  
 \* @param imagePath  
 \* @throws ModelException  
 \* @throws TranslateException  
 \* @throws IOException  
 \*/*public void loadImage(String imagePath) throws ModelException, TranslateException, IOException {  
 // try statement that handles the possible exceptions  
 try {  
  
 // it declares the variable that contains the path to the picture  
 Path imageFile = Paths.*get*(imagePath.trim());  
  
 // It declares the Image object  
 Image img = ImageFactory.*getInstance*().fromFile(imageFile);  
 // It calls the method that generates the description  
 // It stores the information in the description variable  
 description = getDescription(img);  
  
  
 // It calls the method that generates the overlapping  
 // It stores the information in the overlapping variable  
 overlapping = OverlappingAlgorithm.*getOverlap*(listOfObjects);  
  
 // It calls the method that generates the depth using the OverlappingAlgorithm class  
 // It stores the information in the depthOverlap variable  
 depthOverlap = OverlappingAlgorithm.*calculateDepth*(listOfObjects);  
  
 // It calls the method that generates the depth using the CentralpointAlgorithm  
 // It stores the information in the depthCentralPoint variable  
 depthCentralPoint = CentralPointAlgorithm.*calculateDepth*(listOfObjects);  
 // It calls the method that generates the depth using the BottomAlgorithm  
 // It stores the information in the depthBottom variable  
 depthBottom = BottomAlgorithm.*calculateDepth*(listOfObjects);  
  
 // By default the description variable is shown  
 displayMessage(Choice.*DESCRIPTION*);  
  
 // the panel with the rest of the information buttons is displayed  
 centralView.buttonsPanel.setVisible(true);  
  
 } catch (IOException ioe) {  
 JOptionPane.*showMessageDialog*(null, ioe, "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
  
} // end load image

*/\*\*  
 \* Method that handles the selection of the user of the information picked in the interface  
 \* It receives a valid option of the Choice enum and displays the corresponding message  
 \* It is public to allow the access outside the class  
 \*  
 \* @param option  
 \*/*public void displayMessage(Choice option) {  
  
 switch (option) {  
 case *DESCRIPTION*:  
 JOptionPane.*showMessageDialog*(null, description, "Description", JOptionPane.*INFORMATION\_MESSAGE*);  
 break;  
  
 case *OVERLAPPING*:  
 JOptionPane.*showMessageDialog*(null, overlapping, "Overlapping", JOptionPane.*INFORMATION\_MESSAGE*);  
 break;  
  
 case *DEPTHOVERLAP*:  
 JOptionPane.*showMessageDialog*(null, depthOverlap, "Depth by overlapping", JOptionPane.*INFORMATION\_MESSAGE*);  
 break;  
  
 case *DEPTHCENTRALPOINT*:  
 JOptionPane.*showMessageDialog*(null, depthCentralPoint, "Depth by Central Point", JOptionPane.*INFORMATION\_MESSAGE*);  
 break;  
  
 case *DEPTHBOTTOM*:  
 JOptionPane.*showMessageDialog*(null, depthBottom, "Depth by bottom", JOptionPane.*INFORMATION\_MESSAGE*);  
  
 } // end switch  
  
} // end display message

*/\*\*  
 \* Method that calls the method that processes the picture and generate the detection  
 \* Also, it calls the corresponding methods to generate the description of the picture with the location and size of each object  
 \* It receives the image object to be process  
 \* It returns the String with the description  
 \* It is private to avoid accessing outside the class  
 \* It throws the required exceptions in the case an error related to these occurs  
 \*  
 \* @param img  
 \* @return  
 \* @throws IOException  
 \* @throws ModelException  
 \* @throws TranslateException  
 \*/*private String getDescription(Image img) throws IOException, ModelException, TranslateException {

// The same code as the main method in the ObjectDetectorDrive class

… rest of the code …

\*\*\*\*\*\*\*\*\*\* End of code of the CentralController class.

Then, in the interface a new panel with five buttons for each calculation made in the program is added. These buttons call the displayMessage function in the CentralController class. Each button passes the corresponding variable declared in the Choice enum class. Initially, the panel is hidden and is shown when the picture entered by the user is processed and the results are ready to be displayed. The buttons added are:

The “Description” button displays the message with the details of the location and size of each object. The “Overlapping” button displays the message with the details of overlapping among the objects. The “Depth by Overlapping” button displays the message with the details of which objects are in front of which objects using the OverlapAlgorithm class. The “Depth by Central Points” button displays the message with the details of which objects are in front of which objects using the CentralPointAlgorithm class. The “Depth by bottom” button displays the message with the details of which objects are in front of which objects using the BottomAlgorithm class.

\*\*\*\*\*\*\*\*\*\* Code of the CentralView class.

… rest of the code …

JPanel buttonsPanel = new JPanel(new FlowLayout());  
JButton descriptionButton;  
JButton overlappingButton;  
JButton depthOverlapButton;  
JButton depthCentralPointButton;  
JButton depthBottomButton;

… rest of the code …

private void setFrame() {

// new declaration of the new buttons

descriptionButton = new JButton("Description");  
overlappingButton = new JButton("Overlapping");  
depthOverlapButton = new JButton("Depth by overlapping");  
depthCentralPointButton = new JButton("Depth by Central Point");  
depthBottomButton = new JButton("Depth by bottom");

private void addFrame() {

// new GUI components to the interface

buttonsPanel.add(descriptionButton);  
buttonsPanel.add(overlappingButton);  
buttonsPanel.add(depthOverlapButton);  
buttonsPanel.add(depthCentralPointButton);  
buttonsPanel.add(depthBottomButton);  
  
centralFrame.add(buttonsPanel, BorderLayout.*SOUTH*);

private void addListeners() {

// Adding the action listeners to the new GUI components

descriptionButton.addActionListener(this);  
overlappingButton.addActionListener(this);  
depthOverlapButton.addActionListener(this);  
depthCentralPointButton.addActionListener(this);  
depthBottomButton.addActionListener(this);

public void actionPerformed(ActionEvent e) {  
 // if statement to know which button is pressed and call the corresponding method   
 if (e.getSource() == speakButton) {  
 // try statement that handles the possible exceptions   
 try {  
 controller.loadImage(textToSpeechField.getText());  
 } catch (IOException ioe) {  
 System.*out*.println(ioe);  
 } catch (ModelException me) {  
 System.*out*.println(me);  
 } catch (TranslateException te) {  
 System.*out*.println(te);  
 }  
 } // end if speak button  
 else if (e.getSource() == descriptionButton) {  
 controller.displayMessage(Choice.*DESCRIPTION*);  
  
 } // end if  
 else if (e.getSource() == overlappingButton) {  
 controller.displayMessage(Choice.*OVERLAPPING*);  
 } // end if  
 else if (e.getSource() == depthOverlapButton) {  
 controller.displayMessage(Choice.*DEPTHOVERLAP*);  
 } // end if  
 else if (e.getSource() == depthCentralPointButton) {  
 controller.displayMessage(Choice.*DEPTHCENTRALPOINT*);  
 } // end if  
 else if (e.getSource() == depthBottomButton) {  
 controller.displayMessage(Choice.*DEPTHBOTTOM*);  
 } // end if which button pressed  
}// end action perform

… rest of the code …

\*\*\*\*\*\*\*\*\*\* End of code of the CentralView class.

# Chapter 7. Sprint 6

**Sprint 6: Adding a new interface class and testing the program**

Start date: 8/04/2024

End date: 26/04/2024

|  |  |  |
| --- | --- | --- |
| TASK | DETAILS | STATUS |
| 1 | Adding a new interface class | Completed |
| 2 | Testing the program | Completed |
|  |  |  |

## Adding a new interface class

The three algorithm classes use the calculateDepth method to provide the corresponding result. In order to maintain consistency within the program a new interface class, called AlgorithmInterface, is added. This new interface is implemented for the three algorithm classes to ensure the required consistency in the code of the program. The interface contains the calculateDepth method only. As this method is static, the method must have some code that is overridden by the algorithm classes.

\*\*\*\*\*\*\*\*\*\* Code of the AlgorithmInterface class

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package ai.djl.examples.Ivan;  
  
import java.util.List;  
  
public interface AlgorithmInterface {  
 static String calculateDepth(List<ObjectDetected> listOfObjects) {  
 System.*out*.println("Method to be implemented in the algorithm classes");  
 return "";  
 }  
}

\*\*\*\*\*\*\*\*\*\* End of the code of the AlgorithmInterface class

## Testing the program

After processing 15 pictures the overall result is described in the table below (Table 4). It counts the number of times that the corresponding algorithm provides right results, false positives or blank results. Stock, royalty free images were obtained from pexels.com.

An example of the processing performed is outlined below. The program provides a description output with the nobject name along with its location and size. Moreover, the overlapping among objects is provided. Furthermore the use of the three algorithms provide the relative position of the objects. The individual results for each image are provided in Appendix B.

|  |  |
| --- | --- |
| Image | Description |
|  | Car Medium height and medium width size From the right and the top  Bicycle Large height and large width size From the left and the middle  Dog Large height and medium width size From the left and the middle |

Table 3 Picture with description

A screenshot of a computer

Description automatically generated

Figure 14. Description pop-up window

A screenshot of a computer error message

Description automatically generated

Figure 15. Overlapping pop-up window

A screenshot of a computer error

Description automatically generated

Figure 16. Depth by the overlapping algorithm pop-up window

A screenshot of a computer error

Description automatically generated

Figure 17. Depth by the central point algorithm pop-up window

A screenshot of a computer error

Description automatically generated

Figure 18. Depth by the bottom algorithm pop-up window

## Results

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Right  results | False  positives | Blank  results |
| Depth by overlapping | 9 | 0 | 8 |
| Depth by central point | 10 | 2 | 6 |
| Depth by bottom | 20 | 5 | 1 |

Table 4 Overall results

Based on the results, the performance of each algorithm can be analysed and compared. From the results in table XX , the algorithm with more results is the bottom algorithm, along with the least amount of blanks. However this algorithm is also the algorithm with more false positives.

The bottom algorithm despite the number of false positives, has the highest detection. This maybe due to the practise of photographers framing method, often positioning the context of the photograph in the foreground. A good example is when there is a big group of people to take a photo, people is ordered in rows. The first row is the lowest one and the following rows are getting higher gradually.

Therefore, the choice of the algorithm depends on the requirements of the user. If user wants to have high reliability of results with a low probability of false positives, regardless of blank results, the overlapping algorithm and the central point algorithm are the right options. Both provide very similar numbers in their final analyses. However, if the user wants to have the highest probability of getting results and low number of blank results, the bottom algorithm is the right option. However with this there is a percentage of possible false positives (20%) the user must bear in mind.

The overlapping algorithm and the central point algorithm provide very similar results. Although the central point algorithm has come probability of failure to recognise objects, as this algorithm may provide a few false positives when the overlapping algorithm does not. As well, the central point algorithm has more probability of offering results. Even though the method of calculation is different.

A possible reason for the good performance of the central point algorithm is that it considers that outer object as the predominant one. Therefore, an inner object must have the central point within the bounding box of the outer object, resulting in the conclusion that the inner object is in front of the outer object. An example would be when a person holds a baby. The central point of the baby is inside the area of the person.

In images where an objects bounding box is fully covered by a second objects bounding box, the inner object may not be detected. As overlapping is required for all three algorithms to operate, as sufficient level of overlap is required to allow use of the algorithms. In the situation where no overlapping occurs only individual object descriptions can be provided.

# Chapter 8. Future improvements

One improvement would be the ability to differentiation between repeated objects in a image, example “Person 1” and “Person 2”. This may be done by adding a number when the object type is repeated

Another good improvement would be the segmentation of the detected objects for a following process. The reason for this is that when an object is detected, the technology does not usually detect if there is another inner object within. Therefore, to segment the detected object and process it as a new picture to try to find more objects within would provide more information.

To detect colours would be another improvement. Obviously, to detect all the colours that an object may have would be confusing. Therefore, the predominant colour should be detected.

The use of more models will improve accuracy across object types. This could also be developed to help identify subsets of identified objects, i.e. greyhound as a type of dog or car models. The combination of this with developing facial recognition processes may allow individuals be identified.

Perhaps another Java library may be used for a similar project. A library that can accept external resources easier. Another text to speech engine may be used. The FreeTTS works reasonable fine, but it is a bit slow when processing the text. Therefore, a lighter solution might be implemented.

# Chapter 9. Conclusions

The Java programming language can be used for artificial intelligence purposes. The Deep Java Library is a good tool that proves that fact. Although this library seems is not very friendly when using external resources. The import of the JSON and FreeTTS libraries are a good example: being tedious to import the JSON library and impossible to import the FreeTTS. Perhaps, it is the complexity of the internal design of the Deepa Java library that makes it difficult to utilise it with external resources.

A good description of pictures can be done with the existing tools. If with very few data, such as just the coordinates and size of the objects in a picture, some calculations can be made to determine if an object is in front of another object. Calculations so obvious as the ones using in this project: overlapping, central point and the bottom of the objects, can make some conclusions that are reasonable.

With a more developed technology, such as the 3D, more accurate description of images can be made. Nowadays, with devices with several lends, this process is not so complex as it was some years ago. The depth of the images can be detected much easier and will not be a guess from a 2D image.

There are other techniques to determine the depth of an image: getting the depth by map or stereo vision, the buffer algorithm that stores the depth of each pixel or the painter algorithm that depends on the order of the objects are from the viewer and does not work with overlapped objects These algorithms need extra information and are designed for specific situations. For the purpose of this project, generic algorithms must be used.

Ideally Java could develop its own artificial technology because the Deep Java Library still needs libraries such as TensorFlow, developed in Python. More native libraries might help a better evolution of Java in this field. Obviously, Python is a predominant language in the artificial intelligence world and this is the future of the IT world in the coming years. Therefore, Java may loose some IT market and as result, less companies will choose this programming language.

# References

amazon.com. (2024, 01 29). *deepjavalibrary/djl: An Engine-Agnostic Deep Learning Framework in Java*. Retrieved from github.com: https://github.com/deepjavalibrary/djl

Amazon.com, Inc. (2024, 01 29). *DJL - Deep Java Library*. Retrieved from djl.ai: https://djl.ai/

American fundation for the blind. (2023, 10 01). *Home | American Foundation for the Blind*. Retrieved from afb.org: https://www.afb.org/aw/1/4/16165

Android Accessibility. (2023, 10 2). *Get started on Android with TalkBack - Android Accessibility Help*. Retrieved from support.google.com: https://support.google.com/accessibility/android/answer/6283677?hl=en

Apache. (2023, 11 16). *Apache MXNet | A flexible and efficient library for deep learning.* Retrieved from mxnet.apache.org: https://mxnet.apache.org/versions/1.9.1/

arcgis. (2023, 11 16). *How single-shot detector (SSD) works? | ArcGIS API for Python*. Retrieved from developers.arcgis.com: ArcGIS Developers

Arnold, V. (2023, 11 17). *AI in Image Recognition: Benefits, Applications and Challenges.* Retrieved from neuroflash.com: https://neuroflash.com/blog/ai-in-image-recognition-benefits-applications-and-challenges/

Assistiv Labs. (2023, 10 16). *JAWS Screen Reader Overview*. Retrieved from assistivlabs.com: https://assistivlabs.com/assistive-tech/screen-readers/jaws

Audio Technologies. (2023, 10 01). *FreeTTS : a Java speech synthesizer | Internet with a Brain*. Retrieved from .web3.lu: https://www.web3.lu/freetts-a-java-speech-synthesizer/

AWS. (2023, 11 14). *What is Deep Learning? - Deep Learning Explained - AWS*. Retrieved from aws.amazon.com: https://aws.amazon.com/what-is/deep-learning/#:~:text=Deep%20learning%20is%20a%20method,produce%20accurate%20insights%20and%20predictions.

B, G. (2023, 10 15). *Top 5 Website Accessibility Issues Every Developer Must Avoid & Fix.* Retrieved from hurix.com: https://www.hurix.com/5-most-common-web-accessibility-issues-to-avoid-fix/#:~:text=Screen%20readers%2C%20which%20help%20specially,a%20common%20web%20accessibility%20issue.

Baheti, P. (2023, 11 14). *The Essential Guide to Neural Network Architectures.* Retrieved from v7labs.com: https://www.v7labs.com/blog/neural-network-architectures-guide

Biography.Com Editors. (2023, 10 12). *Steve Jobs: Biography, Apple Cofounder, Entrepreneur*. Retrieved from biography.com: https://www.biography.com/business-leaders/steve-jobs#main-content

Bureau of Internet Accessibility. (2023, 09 29). *How Screen Readers Are Used in Accessibility Testing.* Retrieved from boia.org/blog: https://www.boia.org/blog/how-screen-readers-are-used-in-accessibility-testing

Chatterjee, S. (2023, 10 15). *10 Most Common Web Accessibility Issues to Solve for | BrowserStack.* Retrieved from browserstack.com: https://www.browserstack.com/guide/common-web-accessibility-issues#:~:text=Cluttered%20navigation%20links%20present%20a,navigation%20if%20not%20coded%20well.

Chumachenko, K., Gabbouj, M., & Iosifidis, A. (2022). Chapter 11 - Object detection and tracking. In A. Iosifidis, & A. Tefas, *Deep Learning for Robot Perception and Cognition* (pp. 243-278). Academic Press.

Cloud Google. (2023, 11 06). *Vision AI | Google Cloud*. Retrieved from cloud.google.com: https://cloud.google.com/vision?hl=en

CVAT. (2024, 03 05). *CVAT*. Retrieved from cvat.ai: https://www.cvat.ai/

DAISY. (2023, 10 06). *DAISY Format - The DAISY Consortium*. Retrieved from daisy.org: https://daisy.org/activities/standards/daisy/

Deep Java Library. (2023, 11 03). *djl.ai*. Retrieved from djl.ai: https://djl.ai/

DJL. (2024, 02 01). *DetectedObjects (Deep Java Library 0.2.1 API specification)*. Retrieved from djl.ai: https://javadoc.io/static/ai.djl/api/0.2.1/ai/djl/modality/cv/DetectedObjects.html

Doyle, S. (2024, 03 06). *AI 101: Training vs. Inference.* Retrieved from backblaze.com: https://www.backblaze.com/blog/ai-101-training-vs-inference/#:~:text=Efficiency%3A%20Training%20is%20typically%20a,data%20is%20a%20key%20feature.

dyorgio. (2024, 02 25). *dyorgio/jinputhook: Global Key and Shortcuts listeners for Java.* Retrieved from github.com: https://github.com/dyorgio/jinputhook

Emgu. (2023, 11 1). *Emgu CV: OpenCV in .NET (C#, VB, C++ and more)*. Retrieved from emgu.com: https://www.emgu.com/wiki/index.php/Main\_Page

Eren, I. (2023, 10 21). *A Brief Introduction to JAWS, NVDA, and VoiceOver | CSS-Tricks - CSS-Tricks.* Retrieved from css-tricks.com: https://css-tricks.com/comparing-jaws-nvda-and-voiceover/

Fraser, B. (2024, 02 04). *GSON: Use in IntelliJ - YouTube.* Retrieved from youtube.com: https://www.youtube.com/watch?v=Qc9EfiepfWs

Freedom Scientific. (2023, 10 16). *JAWS® – Freedom Scientific*. Retrieved from freedomscientific.com: https://www.freedomscientific.com/products/software/jaws/

Gao, M., Du, Y., Yang, Y., & J., Z. (2023, 11 14). Adaptive anchor box mechanism to improve the accuracy in the object detection system. *Multimedia Tools and Applications*, 27383–27402. Retrieved from mathworks.com: Skip to content

Gong, M., Wang, D., Zhao, X., Guo, H., Luo, D., & Song, M. (2021). A review of non-maximum suppression algorithms for deep learning target detection. *Proc. SPIE 11763, Seventh Symposium on Novel Photoelectronic Detection Technology and Applications, 1176332 (12 March 2021).* Kunming.

Googleapis. (2023, 11 1). *googleapis/google-cloud-php-vision*. Retrieved from github.com: https://github.com/googleapis/google-cloud-php-vision

Gunchenko, N. (2023, 09 30). *Everything You Need To Know About Screen Readers | A.Checker.* Retrieved from accessibilitychecker.org: https://www.accessibilitychecker.org/blog/screen-readers/#:~:text=Let's%20look%20at%20another%20survey,reader%20to%20navigate%20the%20web.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing,*, 27 - 48.

Hagargund, A. G., S. V., M. B., & E. F. (2017, 10 12). Image to speech conversion for visuallyimpaired. *International journal of latest research in engineering and technology*, 09 - 15. Retrieved from ibm.com/blog: https://www.ibm.com/blog/optical-character-recognition/

IBM. (2024, 03 11). *Create an image dataset for object detection - IBM Documentation*. Retrieved from ibm.com: https://www.ibm.com/docs/en/scdli/1.2.1?topic=dataset-images-object-detection

IBM. (2024, 03 07). *What are Convolutional Neural Networks? | IBM*. Retrieved from ibm.com.

IBM. (2024, 03 07). *What is a Neural Network? | IBM*. Retrieved from ibm.com: https://www.ibm.com/topics/neural-networks

Igalia. (2023, 10 01). *Accessibility Tools and Technologies | Igalia - Open Source Consultancy and Development*. Retrieved from igalia.com: https://www.igalia.com/technology/accessibility

imagerecognize. (2023, 10 22). *Free Image Recognition Online*. Retrieved from imagerecognize.com: https://imagerecognize.com/#uploaded-img

Imambi, S., Prakash, K. B., & Kanagachidambaresan, G. (2021, 11 17). Programing with Tensorflow: solution for Edge computing application. In K. Praqash, & G. R. Kanagachidambaresan, *Programming with Tensorflow* (pp. 87 - 104). Cham: Springr. Retrieved from techtarget.com: https://www.techtarget.com/searchenterpriseai/definition/PyTorch

InfoWorld. (2023, 10 22). *Machine Learning in Java is Speeding Image Processing | InfoWorld*. Retrieved from infoworld.com: https://www.infoworld.com/article/3601711/machine-learning-in-java-is-speeding-image-processing.html

Java Community Process. (2023, 11 3). *The Java Community Process(SM) Program - JSRs: Java Specification Requests - detail JSR# 381*. Retrieved from jcp.org: https://www.jcp.org/en/jsr/detail?id=381

Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2022, 11 14). A Review of Yolo Algorithm Developments. *Procedia Computer Science*, 1066 - 1073. Retrieved from datacamp.com: https://www.datacamp.com/blog/yolo-object-detection-explained#main

Kaloni, S., Singh, G., & Tiwari, P. (2018). ResNet and VGG Backbone. *researchgate.net*, 1 - 5.

kristian. (2024, 02 27). *kristian/system-hook: Global Keyboard / Mouse Hook for Java applications.* Retrieved from github.com: https://github.com/kristian/system-hook

kwhat. (2024, 03 02). *kwhat/jnativehook: Global keyboard and mouse listeners for Java.* Retrieved from github.com: https://github.com/kwhat/jnativehook

Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., & Sun, J. (2018). DetNet: Design Backbone for Object Detection. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Ed.), *Computer Vision – ECCV 2018* (pp. 339-354). Munich: Springer Link.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollar, P. (2017). Focal Loss for Dense Object Detection. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (pp. 2980 - 2988). IEEE.

Liu, W., Anguelov, D., Erhan, D., Christian, S., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *Computer Vision – ECCV 2016* (pp. 21 - 37). Springer, Cham.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2023, 11 07). *[1512.02325] SSD: Single Shot MultiBox Detector.* Retrieved from arxiv.org: https://arxiv.org/abs/1512.02325

Luajit. (2024, 03 10). *The LuaJIT Project*. Retrieved from luajit.org: https://luajit.org/

MediaPipe. (2024, 03 15). *object\_detector.ipynb - Colaboratory.* Retrieved from colab.research.google.com/: https://colab.research.google.com/github/googlesamples/mediapipe/blob/main/examples/object\_detection/python/object\_detector.ipynb

NVAccess. (2023, 10 16). *NV Access*. Retrieved from nvaccess.org: https://www.nvaccess.org/

OpenAI. (2023, 09 30). *ChatGPT*. Retrieved from chat.openai.com: https://chat.openai.com/

OpenAI. (2023, 10 22). *ChatGPT can now see, hear, and speak*. Retrieved from openai.com: https://openai.com/blog/chatgpt-can-now-see-hear-and-speak

OpenAI. (2023, 11 5). *Introducing ChatGPT and Whisper APIs*. Retrieved from openai.com: https://openai.com/blog/introducing-chatgpt-and-whisper-apis

Oracle. (2023, 10 26). *Java SE Desktop Accessibility - Java Access Bridge For Windows OS*. Retrieved from oracle.com: https://www.oracle.com/java/technologies/javase/javase-tech-access-bridge.html

Oracle. (2023, 10 26). *Java SE Desktop Accessibility Downloads*. Retrieved from oracle.com: Skip to content

Pujara, A. (2023, 11 16). *Image Classification With MobileNet | Built In.* Retrieved from builtin.com: https://builtin.com/machine-learning/mobilenet#main-content

PyPI. (2024, 03 05). *labelImg · PyPI*. Retrieved from pypi.org: https://pypi.org/project/labelImg/

Pypi. (2024, 03 10). *torch · PyPI*. Retrieved from pypi.org: https://pypi.org/project/torch/

PyTorch. (2024, 03 09). *PyTorch*. Retrieved from pytorch.org: https://pytorch.org/

PyTorch. (2024, 03 10). *PyTorch C++ API — PyTorch main documentation*. Retrieved from pytorch.org: https://pytorch.org/cppdocs/

Rosebrock, A. (2023, 11 14). *Intersection over Union (IoU) for object detection - PyImageSearch.* Retrieved from pyimagesearch.com: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Samsung. (2023, 10 2). *galaxy-mobile-voice-assistant | Samsung US*. Retrieved from samsung.com.

SCIPY. (2023, 11 1). *Multidimensional image processing (scipy.ndimage) — SciPy v1.11.3 Manual*. Retrieved from docs.scipy.org: https://docs.scipy.org/doc/scipy/reference/ndimage.html#module-scipy.ndimage

SCIPY. (2023, 11 1). *SciPy*. Retrieved from scripy.org: https://scipy.org/

Simple learn. (2024, 03 06). *What is Epoch in Machine Learning? | Simplilearn*. Retrieved from simplilearn.com: https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-epoch-in-machine-learning#:~:text=Each%20time%20a%20dataset%20passes,training%20the%20machine%20learning%20model.

Simplilearn. (2023, 11 16). *What is PyTorch, and How Does It Work? | Simplilearn.* Retrieved from simplilearn.com: https://www.simplilearn.com/what-is-pytorch-article#what\_is\_pytorch\_and\_how\_does\_it\_work

source forge. (2023, 09 05). *FreeTTS 1.2.3 - A speech synthesizer written entirely in the Java(TM) programming language*. Retrieved from sourceforge.io: https://freetts.sourceforge.io/

sourceforge. (2024, 01 15). *FreeTTS 1.2.3 - A speech synthesizer written entirely in the Java(TM) programming language*. Retrieved from freetts.sourceforge.io: https://freetts.sourceforge.io/#who\_authors

SuperAnnotate. (2024, 03 06). *SuperAnnotate | Empowering Enterprises with Custom LLM/GenAI/CV Models*. Retrieved from superannotate.com: https://www.superannotate.com/

SuperVisely. (2024, 03 05). *Supervisely: unified OS for computer vision*. Retrieved from supervisely.com: https://supervisely.com/

Swan, H. (2023, 10 14). *Browsing with a mobile screen reader - TetraLogical.* Retrieved from tetralogical.com: https://tetralogical.com/blog/2021/10/05/browsing-with-a-mobile-screen-reader/#:~:text=The%20WebAIM%20screen%20reader%20user,and%20the%20rest%20at%202.2%25.

Tan, M., Pang, R., & V. Le, Q. (2024, 03 15). *[1911.09070] EfficientDet: Scalable and Efficient Object Detection.* Retrieved from arxiv.org: https://arxiv.org/abs/1911.09070

Team, @. (2024, 01 31). *DJL 101 3 ndarray - YouTube.* Retrieved from youtube.com: https://www.youtube.com/watch?v=XMEEsYXxNrg&list=PLC1JzXeHJitDmZBHupMGZE2zHzU8fEvfD&index=3

TensorFlow. (2024, 03 09). *TensorFlow*. Retrieved from tensorflow.org: https://www.tensorflow.org/

TensorFlow. (2024, 03 09). *TFRecord and tf.train.Example  |  TensorFlow Core*. Retrieved from tensorflow.org: https://www.tensorflow.org/tutorials/load\_data/tfrecord

Tourch. (2023, 11 16). *Torch | Scientific computing for LuaJIT.* Retrieved from tourch.ch: http://torch.ch/

Tutorials Point. (2023, 10 22). *OpenCV Tutorial*. Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/opencv/index.htm#:~:text=OpenCV%20is%20a%20cross%2Dplatform,use%20OpenCV%20in%20your%20applications.

Vasudevan, R. (2024, 01 31). *Introducing Deep Java Library(DJL) | by Rakesh Vasudevan | Towards Data Science.* Retrieved from towardsdatascience.com: https://towardsdatascience.com/introducing-deep-java-library-djl-9de98de8c6ca

Vertex AI . (2024, 03 11). *Create a dataset for training image object detection models  |  Vertex AI  |  Google Cloud*. Retrieved from cloud.google.com: https://cloud.google.com/vertex-ai/docs/image-data/object-detection/create-dataset

VisionAid. (2023, 10 21). *Pearl | VisionAid*. Retrieved from visionaid.co.uk: https://www.visionaid.co.uk/pearl#:~:text=PEARL%20brings%20blind%20and%20low,picture%20of%20your%20reading%20material.

W3C. (2023, 10 22). *WCAG 2 Overview | Web Accessibility Initiative (WAI) | W3C*. Retrieved from w3.org: https://www.w3.org/WAI/standards-guidelines/wcag/

Weatherhead, J. (2024, 02 29). *Java Global JFrame Key Listener – Planetjon.* Retrieved from planetjon.ca: https://planetjon.ca/java-global-jframe-key-listener-3089

Wikipedia. (2023, 09 30). *Google TalkBack - Wikipedia*. Retrieved from Wikipedia.org: https://en.wikipedia.org/wiki/Google\_TalkBack

Wikipedia. (2023, 10 16). *NonVisual Desktop Access - Wikipedia*. Retrieved from Wikipedia.org: https://en.wikipedia.org/wiki/NonVisual\_Desktop\_Access#:~:text=The%20project%20was%20started%20by%20Michael%20Curran%20in%202006.&text=NVDA%20is%20programmed%20in%20Python,General%20Public%20License%20version%202.

writesonic. (2023, 11 5). *Chatsonic API With ChatGPT Superpowers*. Retrieved from writesonic.com: https://writesonic.com/chatgpt-api-chatsonic

Yamashita, R., Nishio, M., Gian Do, R. K., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *insights imaging*, 611–629.

Yasar, K., & Lewis, S. (2024, 03 10). *What is PyTorch?* Retrieved from techtarget.com: https://www.techtarget.com/searchenterpriseai/definition/PyTorch#:~:text=PyTorch%20is%20an%20open%20source,platforms%20for%20deep%20learning%20research.

Zhang, J., J. Humaidi, A., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., . . . Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big. *Data 8*, 53.

Zhao, Z., Zheng, P., Xu, S., & Wu, X. (2019). Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 321 - 3232.

# Appendix A

The code for this project can be found in the following repository. The code needs the entire Deep Java Library. For this reason, the repository contains the whole library. To find the code of this project, users are required to go

examples>src>main>java>ai>djl>examples>Ivan

The link to the repository is:

<https://github.com/Segade/DJL>

# Appendix B

