Spis treści

1. Wstęp

Tematem naszej pracy jest stworzenie aplikacji, która umożliwi zamianę tekstu znajdującego się w dowolnym wybranym zdjęciu na mowę.

Projekt ten zostanie stworzony z wykorzystaniem języka Python oraz dostępnych dla niego bibliotek, oraz narzędzi.

Aplikacja będzie posiadać prosty graficzny interfejs użytkownika, który pozwoli na wybór obrazka, dostosowanie jego parametrów konwersji wraz z podglądem tych zmian, a co więcej umożliwi odtworzyć przy pomocy przycisku tekst zawarty wewnątrz wybranego obrazka, pozwoli również dostosować szybkość czytania oraz głośność, a także zapisać czytany tekst do pliku audio.

2. Przegląd aktualnych rozwiązań i literatury

Istnieje wiele aktualnych rozwiązań dla problemu czytania znaków z plików graficznych. Niektóre z nich są płatne lub mają pewne ograniczenia w darmowej wersji, inne natomiast są całkowicie bezpłatne, a co więcej część z nich jest na licencji open source, czyli pozwalają na zagłębienie się w kod programu i jego edycje pod własne potrzeby.

Wszystkie z tego typu aplikacji wykorzystują technologie o nazwie OCR (ang. Optical Character Recognition). Jest to technologia, która umożliwia optyczne rozpoznawanie znaków na plikach graficznych, drukowanych, a także zapisanych ręcznie [2]. OCR za pomocą kodów ASCII zmienia reprezentacje graficzne znaków na ich cyfrowe odpowiedniki. W ten sposób powstaje tekst, który może następnie zostać odczytany lub poddany dalszej edycji. W celu dokładniejszego działania programu stosuje się specjalne kroje pisma, a także dedykowane formularze i szablony. Oprogramowanie OCR wykorzystuje różne metody segmentacji obrazu, na przykład progowanie, aby wyodrębnić poszczególne znaki z obrazu, które następnie są najczęściej osobno klasyfikowane jako poszczególne litery. Zwykle w tym procesie wykorzystywane są sieci neuronowe. Zazwyczaj, by wyeliminować pomyłki, program sprawdza całość rozpoznanego tekstu lub poszczególne wyrazy pod kątem poprawności ortograficznej i gramatycznej danego języka [3].

Przykładową aplikacją tego typu jest Speechify, umożliwia ona użytkownikom robienie zdjęć tekstu, a następnie odsłuchiwanie go na głos. Jest to jedna z wielu funkcjonalności tego programu. Program ten jest udostępniony na licencji testowej. Pierwsza wersja aplikacji

została stworzona przez studenta college'u na Brown University w Stanach Zjednoczonych. Narzędzie miało pomóc mu nadążyć z czytaniem na zajęciach, ponieważ cierpiał na dysleksje. Obecnie aplikacja obsługuje ponad 20 języków, posiada kilka różnych głosów lektorów do wyboru, a także umożliwia dostosowanie prędkości czytania, nagrywania tekstu, wstrzymywania i rozpoczynania od zapisanego miejsca, a także obsługuje wiele rozszerzeń dla plików [1].

Inną tego typu aplikacją jest Balabolka. W przeciwieństwie do poprzedniego programu ten wykorzystuje jako lektora, syntezator mowy zainstalowany na komputerze użytkownika. Aplikacja ta pozwala także zmianę czcionki oraz jej koloru i umożliwia kierowanie procesem czytania z zasobnika systemowego lub przy wykorzystaniu skrótów klawiaturowych. Autorzy zaimplementowali również funkcję sprawdzania pisowni. Tak jak Speechify, pozwala także zapisywać audio, a odczyt jest realizowany dla wielu formatów plików. W przeciwieństwie do poprzedniego narzędzia jest jednak całkowicie bezpłatny [4].

Jedną z komercyjnych aplikacji jest Amazon Textract. Amazon Texttract został przedstawiony przez Amazon Web Services pod koniec listopada na konferencji AWS re: Invent Conference. Ta technologia jest oparta na uczeniu maszynowym i ma na celu wyeliminowanie ręcznej ingerencji w proces wydobywania danych z zeskanowanych dokumentów, nawet jeśli dane są dostępne w postaci tabel i formularzy. Amazon opisuje ją w ten sposób: „Amazon Texttract to usługa uczenia maszynowego (ML), która automatycznie wyodrębnia tekst, pismo odręczne i dane z zeskanowanych dokumentów. Wykracza poza proste optyczne rozpoznawanie znaków (OCR), aby identyfikować, rozumieć i wydobywać dane z formularzy i tabel". Usługa oferuje łatwy w obsłudze interfejs API, który ułatwia użytkownikowi końcowemu uzyskanie pożądanych rezultatów [5].

W sieci dostępnych jest również wiele darmowych rozwiązań online zawartych na stronach internetowych. Zazwyczaj narzędzia ta posiadają limit w postaci ilości przesyłanych zdjęć bądź stron dokumentów. Przykładową tego typu stroną jest https://texttospeech.io/online-image-to-text-reader lub https://products.aspose.app/ocr/image-to-speech. Wszystkie tego typu strony mają ograniczenia dotyczące, chociażby obsługiwanego języka tekstu, dostępnych lektorów oraz funkcji konfiguracji samego czytania.

Dla osób chcących wgłębić się w tematykę związaną z przetwarzaniem mowy oraz obrazu dostępnych jest wiele tytułów książkowych, które w sposób dokładny i zrozumiały dla

początkujących opisują sposób budowy tego typu narzędzia, a co więcej obejmują prace czołowych światowych ekspertów w tej dziedzinie. Przykładowymi tytułami tego książek są:

- Optical Character Recognition: An Illustrated Guide to the Frontier (The Springer International Series in Engineering and Computer Science, 502), Stephen V. Rice George Nagy, Thomas A. Nartker – książka ta oferuje spojrzenie na wydajność obecnych systemów OCR poprzez zilustrowanie i wyjaśnienie rzeczywistych błędów OCR. Przedstawia 280 ilustrowanych przykładów błędów rozpoznawania w taksonomii składającej się z defektów obrazowania, podobnych symboli, interpunkcji i typografii, a także omawia możliwe podejścia do poprawny dokładności tych systemów. W książce tej znalazło się także porównanie zdolności rozpoznawania znaków przez ludzi i komputery.
- Circuits, Signals, and Speech and Image Processing, Richard C. Dorf, Styczeń 13, 2006, Wydawnictwo CRC Press – książka ta zawiera wszystkie podstawowe informacje związane z obwodami elektrycznymi i komponentami, analizą obwodów, wykorzystaniem transformaty Laplace'a, a także przetwarzaniem sygnałów, mowy i obrazu za pomocą filtrów i algorytmów. Bada również pojawiające się obszary, takie jak synteza tekstu na mowę, przetwarzanie w czasie rzeczywistym i wbudowane przetwarzanie sygnałów. Każdy artykuł zawiera definicje terminów, odniesień i źródeł dalszych informacji.
- OCR with Tesseract, OpenCV, and Python, Adrian Rosebrock – jest kompletnym przewodnikiem szkoleniowym do opanowania optycznego rozpoznawania znaków. Pozwala na postawieniu swoich pierwszych kroków z OCR, a także zapoznaje nas silnikiem Tesseract i pokazuje jak go skonfigurować za pośrednictwem języka programowania Python oraz biblioteki OpenCV. W książce znajdują się także rozdziały poświęcone wykorzystaniu uczenia maszynowego do udoskonalania techniki rozpoznawania pisma, a także rozwiązywania łamigłówek typu Sudoku.

Opracowanie powyższych tytułów pozwoli nam w bardzo dokładny sposób opanować technikę rozpoznawania znaków z plików graficznych, a także stworzyć program, który następnie tak rozpoznane znaki przeczyta.

Ciekawostką z zakresu rozpoznawania znaków jest to, że rozpoznany przez użytkowników tekst z weryfikacji CAPTCHA, która zabezpiecza strony internetowe, wykorzystywany jest w oprogramowaniu OCR. ReCAPTCHA funkcjonuje od 2009 roku i w pierwszych 12 miesiącach

od wprowadzenia udało się zdigitalizować aż 17 tysięcy książek. Szacuje się, że aż 200 milionów kodów zabezpieczających CAPTCHA rozwiązywanych jest codziennie przez internautów na całym świecie. Proces ten zajmuje około 10 sekund, co daje sumarycznie niebagatelną ilość zrealizowanych 150 tysięcy roboczogodzin, każdego dnia [6].

3. Wykorzystane narzędzia

Nasz projekt został stworzony z wykorzystaniem języka Python wraz z bibliotekami:

- OpenCV – wieloplatformowa biblioteka funkcji wykorzystywanych do obróbki obrazu. Oparta jest na otwartym kodzie, zapoczątkowana została przez Intela.
- Tkinter – biblioteka Pythona umożliwiająca tworzenie interfejsu graficznego. Tkinter to darmowe oprogramowanie wydane na licencji Pythona.
- PIL – bezpłatna biblioteka typu open source dla języka Python, która dodaje obsługę otwierania, manipulowania i zapisywania wielu różnych formatów plików graficznych. Rozwój oryginalnego projektu, znanego jako PIL, został przerwany w 2011 roku. Następnie następca projektu o nazwie Pillow rozwidlił repozytorium PIL i dodał obsługę Pythona 3.x [8].
- Pyttsx3 – biblioteka konwersji tekstu na mowę w Pythonie. W przeciwieństwie do innych bibliotek działa w trybie offline wykorzystując systemowy syntezator mowy i jest kompatybilny z Pythonem 2, jak i 3.

W projekcie zostało także użyte narzędzie Tesseract. Jest to silnik optycznego rozpoznawania znaków dla różnych systemów operacyjnych. Jest to oprogramowanie darmowe, udostępnione na licencji Apache. Opracowane przez firmę Hewlett-Packard. W 2006 roku Tesseract został uznany za jeden z najdokładniejszych dostępnych silników OCR typu open source. Obecne dostępna jest wersja 5, która zostałą wydana w 2021 roku, po ponad dwóch latach testów i rozwoju. Obsługuje ponad 100 języków i 30 skryptów. Tak więc możliwe jest na przykład rozpoznawanie tekstu z mieszanką języków zachodnio- i środkowoeuropejskich przy użyciu modelu alfabetu łacińskiego, w którym jest napisany [7].

4. Implementacja

Pierwszym punktem implementacji aplikacje jest import potrzebnych bibliotek oraz instalacja narzędzia Tesseract, które jest wymagane do prawidłowego funkcjonowania aplikacji.

Została utworzona klasa „MyVariables”, która przechowuje zmienne dotyczące ścieżki do obrazka, obrazka, jego rozmiaru, a także tekstu odczytanego z niego. Klasa ta zawiera metody służące do ustawiania i pobierania wartości tych zmiennych [Rysunek 1].

Kolejną klasą, która została utworzona, jest „MyOptions”. Zawiera ona parametry używane do progowania obrazu, a także właściwości czytania takie jak np. głośność [Rysunek 2].

Okno główne aplikacji utworzone zostało z wykorzystaniem biblioteki Tkinter. Został zainicjowany obiekt klasy Tk(), która zawiera wszystkie metody konfiguracyjne okna. Następnie okno to zostało odpowiednio skonfigurowane poprzez ustawienie jego wymiarów, responsywności, koloru, czcionki itd. [Rysunek 3]. Wewnątrz okna została także utworzona etykieta, która przechowywać będzie w określonym miejscu obrazek.

Następnie znajdują się odpowiednie funkcje, które służą np. do zwracania konwertowanego obrazka, uruchomienia silnika tesseract w celu pobrania tekstu ze wskazanego obrazka, a także pojawiania okna do wyboru i zapisu pliku audio z wybranymi ustawieniami.

Kolejna funkcja służy do ustawienia wybranego obrazka w miejsce etykiety okna głównego aplikacji. Wewnątrz jej znajduje się warunek sprawdzający wymiary obrazka i jeśli jest taka potrzeba, to jest on skalowany [Rysunek 4].

Z ważniejszych funkcji, które zostały utworzone, jest także „Options”, które tworzy okno do konfiguracji parametrów obrazu i czytania. Funkcja ta wykorzystuje również bibliotekę Tkinter, a także utworzone przez nas funkcje do zmiany poszczególnych parametrów z wykorzystaniem np. slidera [Rysunek 5].

Aktualizacja wartości progowania oraz rozmycia realizowana jest przy pomocy utworzonej funkcji o nazwie „update_options”, która to po naciśnięciu przycisku Update jest uruchamiana [Rysunek 6]. Powoduje ona również aktualizacje podglądu obrazka, a także tekstu, który został rozpoznany.

Następnymi ważnymi funkcja są te, które służą do konwersji obrazu w celu pozbycia się

szumów, a także progowania liter. Dzięki temu aplikacja jest w stanie prawidłowo odczytać tekst nawet z niedoskonałych i zniekształconych obrazów. Konwersja realizowana jest z wykorzystaniem funkcji biblioteki OpenCV, która zawiera np. funkcje medianBlur do usuwania szumu [Rysunek 7].

Aplikacja zawiera również wiele innych funkcji, które wykorzystywane są np. do ustawienia parametrów czytania lub zapisania tekstu do przeczytania, lecz te omówione powyżej są jednymi z najważniejszych.

```
class MyVariables:
    src = ""
    img = None
    img_size = []
    saidText = ""

    Adrian Chmielowiec
    def __init__(self, src="", img=None):
        self._img = img
        self._src = src

    Adrian Chmielowiec
    def set_img(self, img):
        self._img = img

    Adrian Chmielowiec
    def get_img(self):
        return self._img

    Adrian Chmielowiec
    def get_src(self):
        return self._src

    Adrian Chmielowiec
    def set_src(self, src):
        self._src = src

    NeonDmn
    def set_size(self, size):
        self.img_size = size

    NeonDmn
    def get_size(self):
        return self.img_size
```

Rysunek 1 Klasa MyVariables

```
class MyOptions:
    volume = 1  # 0 - 1
    rate = 150
    blur_strength = 5
    treshold_blocksize = 11
    treshold_constant = 2
```

Rysunek 2 Klasa MyOptions

```
global panel
global textPanel
bOptionsOpen = False
core = MyVariables()
options = MyOptions()
# Okienko
root = Tk()

root.geometry("550x350")
root.resizable(width=True, height=True)
root.configure(bg='ghost white')
root.title("Text speaker")

Label(root, text="Image", font="arial 20 bold",
      bg='white smoke').pack(side=TOP, ipadx=5, ipady=5)

panel = Label(root, image="")
panel.pack(side=TOP, ipadx=5, ipady=5, expand=True)
textPanel = Label()
```

Rysunek 3 Konfiguracja głównego okna aplikacji

```python
#Funkcja do zwracania przekonwertowanego obrazka
 NeonDmn
def opencv_to_tkinter(img):
    gray = cv2.split(img)
    img = cv2.merge(gray)
    im = Image.fromarray(img)
    return im

#Funkcja do uruchomienia silnika tesseract i pobrania oraz zwrócenia z wskazego obrazka tekstu
 Adrian Chmielowiec *
def ocr_core(img):
    # WARUNEKIEM DZIALANIA JEST POSIADANIE PROGRAMU TESSERACT. PONIZEJ TRZEBA PODAC SCIEZKE
    pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
    text = pytesseract.image_to_string(img)
    return text

#Funkcja do uruchomienia okna wyboru obrazka
 Adrian Chmielowiec +1
def open_file():
    # BIERZE SCIEZKE ABSOLUTNA I GDY ZNAJDUJA SIE SPACE TO POWODUJE BLEDY
    src = filedialog.askopenfilename(
        initialdir="Obrazy", title='Select a file')
    core.set_src(src)
    return src

#Funkcja do zapisywania pliku audio czytanego tekstu
 NeonDmn
def save_file():
    types = [('Audio file', '*.mp3')]
    file = filedialog.asksaveasfilename(
        filetypes=types, defaultextension=types)

    # Nic nie rób jak okno zamknięte przyciskiem "cancel"
    if (file is None):
        return

    # Aktualizuj ustawienia TTS
    set_property_speach()
    save_text()
```

Rysunek 4 Funkcje programu

```python
#Funkcja do ustawienia wybranego obrazka w miejsce etykiety okna glownego aplikacji
NeonDmn +1
def open_img():
    panel.config(image='')
    src = open_file()
    if (src == ''):
        return

    cv2img = cv2.imread(src)
    core.set_img(cv2img)

    img = Image.open(src)

    # Pobiera wymiary otwieranego obrazu
    imH, imW, Channels = cv2img.shape
    # Bool jest true jak szerokość obrazu jest większa niż jego wysokość
    bIsLandscape = (imW > imH)

    if (bIsLandscape):
        # Jeśli szerokość większa niż wysokość skaluj obrazek do szerokości i zachowaj proporcje przy wysokości
        imResizeRatio = 400 / imW
        imGotoSize = [400, round(imH * imResizeRatio)]
    else:
        # Skaluj do wysokości i zachowaj proporcje przy szerokości
        imResizeRatio = 200 / imH
        imGotoSize = [round(imW * imResizeRatio), 200]

    core.set_size(imGotoSize)
    update_preview_image()
    # Przekonwertuj z obrazka na tekst
    save_text()

    # Włącz przycisk do zapisywania
    butSave['state'] = NORMAL
```

Rysunek 5 Funkcja do ustawiania podglądu obrazka

```python
#Utworzenie okna do konfiguracji
 NeonDmn
def Options():
    # Otwiera okno opcji jak nie jest jeszcze otwarte
    global bOptionsOpen
    global textPanel
    if (bOptionsOpen):
        return

    bOptionsOpen = True
    # Utwórz okno opcji zależne od głównego okna
    optionsWindow = Toplevel(root)

    optionsWindow.geometry("550x450")
    optionsWindow.resizable(width=True, height=True)
    optionsWindow.configure(bg='ghost white')
    optionsWindow.title("Text speaker options")

    # Slider głośności
    setup_volume_slider(optionsWindow)

    # Parametry TTS
    Label(optionsWindow, text="Parameters", font="arial 20 bold",
          bg='white smoke').pack(side=TOP, ipadx=5, ipady=5)

    setup_rate_slider(optionsWindow)
    setup_image_parameter_options(optionsWindow)

    # delete
    textPanel = Label(optionsWindow, text="None", font="arial 11")
    textPanel.pack(side=BOTTOM)
    update_preview_text()

    # create_preview_image(optionsWindow)
    update_preview_image(True)

    optionsWindow.protocol(
        "WM_DELETE_WINDOW", lambda: option_window_destroy_sequence(optionsWindow))
```

Rysunek 6 Funkcja do tworzenia okna konfiguracji parametrów

```python
def update_options(e1, e2, e3):
    blur = int(e1.get())
    blocksize = int(e2.get())
    constant = int(e3.get())

    # blur i blocksize mają być nieparzyste i większe od 3
    if (blocksize < 3):
        blocksize = 3
    if (blur < 3):
        blur = 3
    if (blocksize % 2 == 0):
        blocksize -= 1
    if (blur % 2 == 0):
        blur -= 1

    if (int(e1.get()) == 0):
        blur = 0

    print(f"Updated: {blur}, {blocksize}, {constant}")

    # Wyczyść okna z wartościami
    e1.delete(0, 'end')
    e2.delete(0, 'end')
    e3.delete(0, 'end')

    # Wstaw prawidłowe wartości okien
    e1.insert(0, str(blur))
    e2.insert(0, str(blocksize))
    e3.insert(0, str(constant))

    # Ustaw nowe wartości w opcjach
    options.blur_strength = blur
    options.treshold_blocksize = blocksize
    options.treshold_constant = constant

    if (core.get_src() != ''):
        update_preview_image(True)
        # Wygeneruj TTS ponownie
        save_text()
        update_preview_text()
```

Rysunek 7 Funkcja do aktualizacji wartości progowania i rozmycia

```
#Funkcja do zwracania przekonwertowanego do skali szarosci obrazka
Adrian Chmielowiec
def get_grayscale(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#Funkcja do usuwania szumu z obrazu za pomoca medianBlur
NeonDmn +1
def remove_noise(image):
    if (options.blur_strength == 0):
        return image
    return cv2.medianBlur(image, options.blur_strength)

#Funkcja do progowania obrazu
NeonDmn +1
def thresholding(image):
    return cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                 cv2.THRESH_BINARY, options.treshold_blocksize, options.treshold_constant)
```

Rysunek 8 Funkcje do konwersji obrazu

5. Instrukcja obsługi i opis działania aplikacji

Utworzona aplikacji może zostać uruchomiona przez plik wykonywalny exe bądź także z pomocą wiersza poleceń i komendy python x, gdzie x jest ścieżką do lokalizacji skryptu głównego aplikacji. Istnieje także możliwość uruchomienia aplikacji przy pomocy środowiska programistycznego, jakim jest np. PyCharm czy też Visual Studio.

Uruchomienie aplikacji skutkuje wyświetleniem okna [Rysunek 9], które zawiera podgląd na obraz i funkcyjne przyciski. Okno można dowolnie przesuwać, a także manipulować jego rozmiarem przy jednoczesnym zachowaniu proporcji między poszczególnymi sekcjami.

Klikniecie przycisku „Open Image”, skutkuje pojawieniem się okna z systemu Windows w standardowej lokalizacji, zawierającej kilka testowych obrazów dla aplikacji [Rysunek 10]. Możliwe jest wybranie obrazu z dowolnej lokalizacji w komputerze. Aplikacja obsługuje większość formatów dla obrazów. Po wybraniu interesującego obrazka należy kliknąć następnie przycisk „Otwórz”. Okno systemowe zostanie zamknięte, a wewnątrz aplikacji znajdzie się podgląd wybranego obrazka [Rysunek 11].

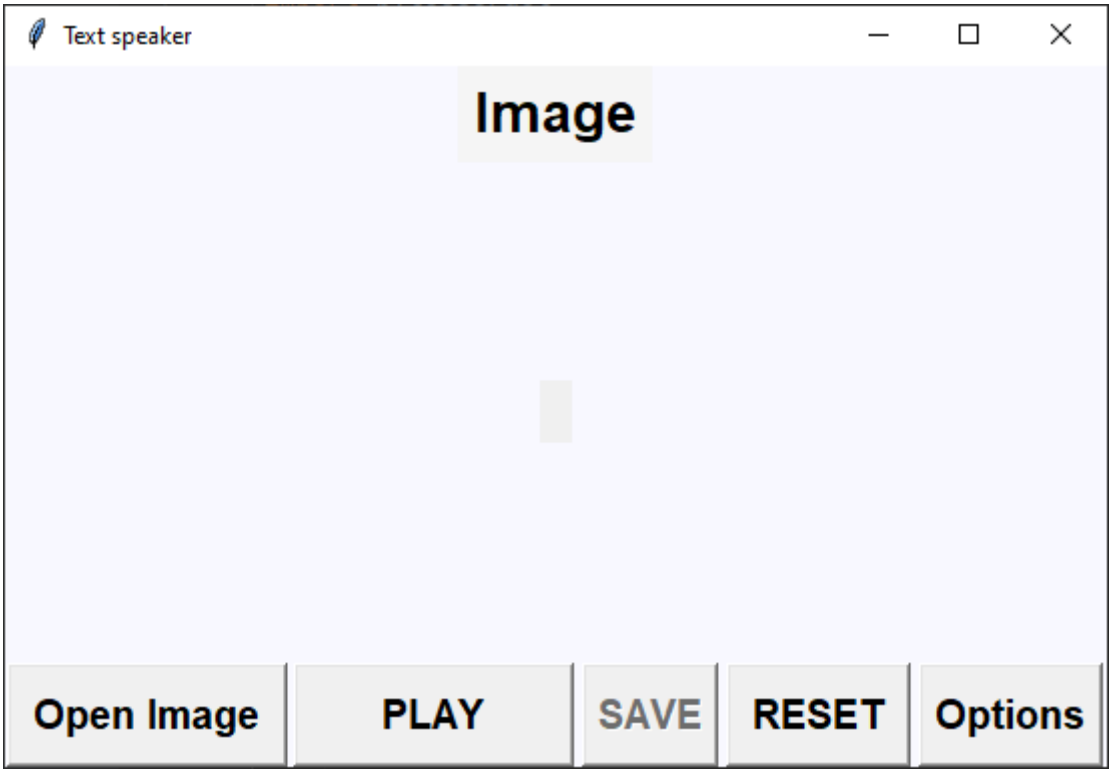
Po wczytaniu obrazka jest możliwość jego modyfikacji poprzez wartości progowania, a także rozmycia. Zmiana wartości tych ustawień umożliwia przycisk „Options”. Po jego naciśnięciu pojawia się nowe okno aplikacji, wewnątrz którego znajdują się wspomniane wyżej zmienne, a także parametry związane z głośnością czytania tekstu oraz szybkością [Rysunek 12]. Podgląd obrazka wewnątrz głównego okna aplikacji zostaje zaktualizowany po otwarciu okna, a także w momencie aktualizacji ustawień przyciskiem „Update”. Tak

przetworzony obraz zostaje następnie użyty przez system rozpoznawania znaków do utworzenia tekstu, który widoczny jest poniżej przycisku „Update” i reprezentuje on tekst, jaki zostanie odczytany na głos [Rysunek 13].

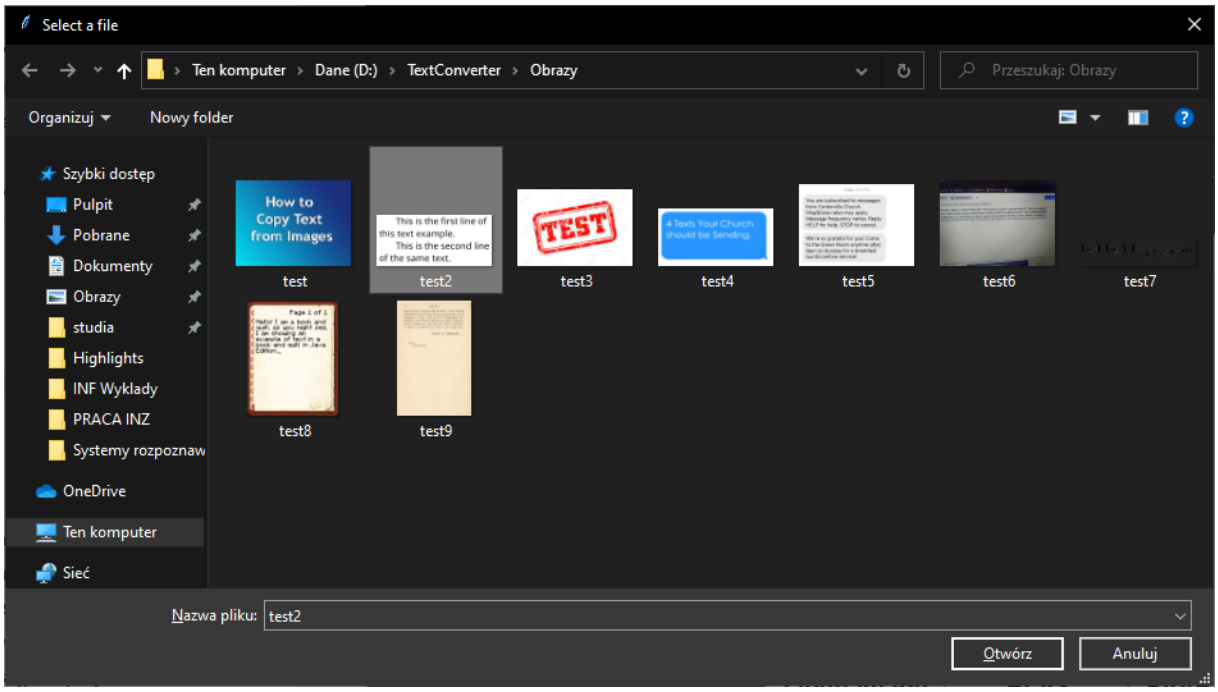
W celu odczytania tekstu z wybranego obrazka należy kliknąć przycisk „PLAY”. Po jego naciśnięciu lektor przeczyta tekst z przetworzonego obrazu. Jest możliwość zapisu dźwięku czytanego tekstu do pliku audio za pomocą przycisku „SAVE”, który po wciśnięciu uruchomi okno systemowe i pozwoli wybrać lokalizację zapisu pliku, który zostanie zapisany w formacie mp3.

Ostatnim z dostępnych przycisków jest „RESET”. Pozwala on na usunięcie wybranego obrazka.

Zamknięcie aplikacji odbywa się naciśnięciem „X” w prawym górnym rogu okna aplikacji.

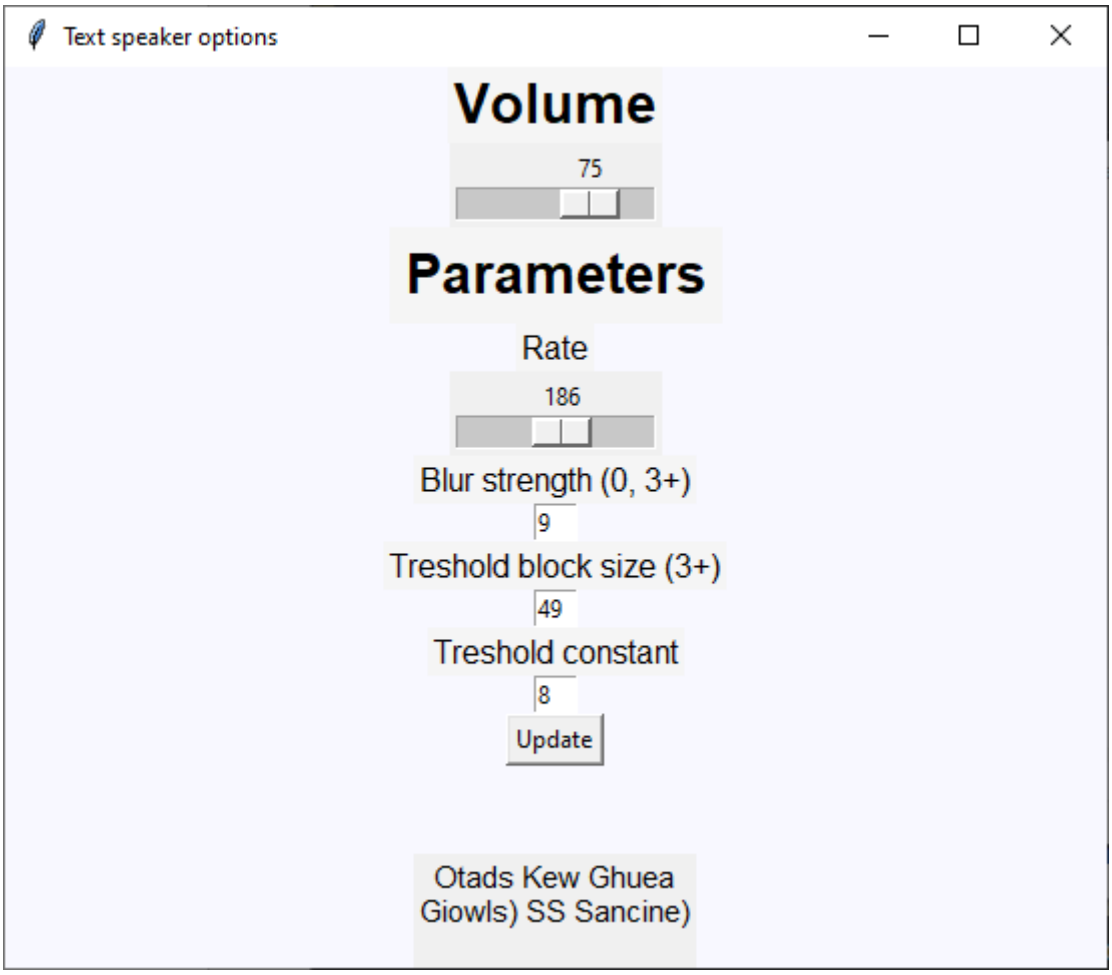


Rysunek 9 Okno główne aplikacji

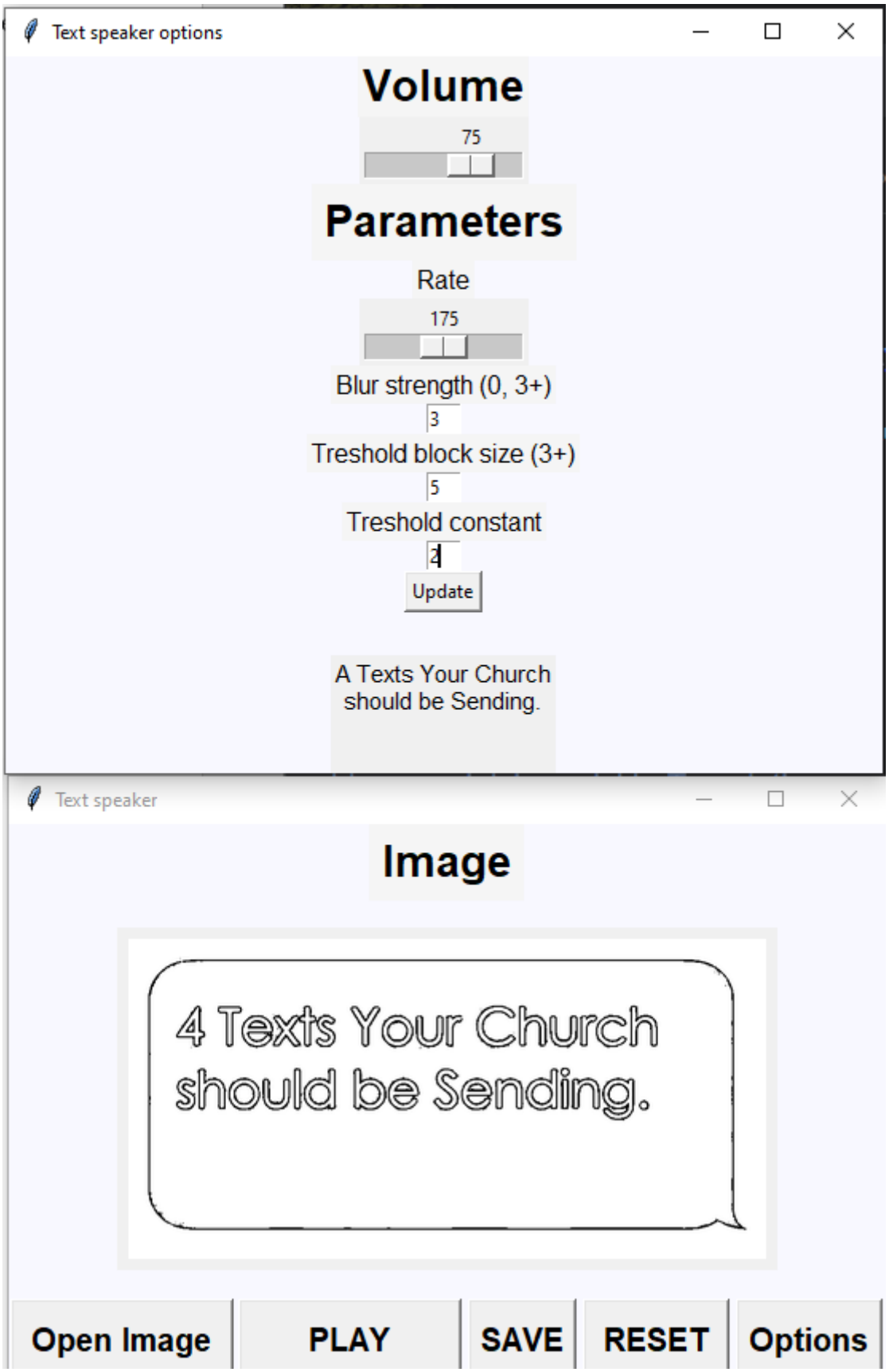


Rysunek 10 Okno wyboru obrazka

Rysunek 11 Podgląd wybranego obrazka wewnątrz aplikacji



Rysunek 12 Okno modyfikacji parametrów obrazka i czytania



Rysunek 13 Modyfikacja obrazka

6. Podsumowanie

Utworzona aplikacja spełnia wszystkie założenia projektowe. Powstało narzędzie pozwalające na odczytanie tekstu z dowolnego obrazka, który zostanie wgrany, a ponadto jest możliwość zapisu czytanego tekstu do pliku audio i odsłuchania go w dowolnym momencie na dowolnym urządzeniu. Aplikacja posiada graficzny interfejs, który jest prosty i pozwala szybko odnaleźć się w programie użytkownikom nawet o niskim stopniu zaawansowania. Okno do modyfikacji parametrów progowania obrazu pozwala na, dostosowanie tych wartości tak by program, jak najdokładniej odwzorował tekst znajdujący się wewnątrz obrazka. Ponadto jest możliwość manipulowania głośnością i szybkością czytania tekstu przez lektora. Aplikacja nie wymaga również połączenia internetowego, tak jak ma to miejsce w większości tego typu narzędzi.

Narzędzie to ma możliwość rozwoju poprzez dodanie dodatkowych opcji oraz modułów takich jak np. wybór lektora i języka. Istnieje także możliwość usprawnienia samego systemu odczytywania znaków przez dołączenie do niego uczenia maszynowego.

Biorąc pod uwagę powyższe informacje, można stwierdzić, że utworzona aplikacja po przetestowaniu i usprawnieniu mogłaby zastąpić inne tego typu oprogramowanie dostępne na rynku.

7. Bibliografia

1. Speechify - https://speechify.com/blog/turn-image-to-speech-with-speechify/?landing_url=https%3A%2F%2Fspeechify.com%2Fpl%2Ftekst-do-mowy-online%2F (dostęp 28.01.2023)
2. OCR - https://www.ironmountain.com/pl/resources/data-sheets-and-brochures/o/ocr-what-is-it (dostęp 28.01.2023)
3. OCR zasada działania - https://pl.wikipedia.org/wiki/Optyczne_rozpoznawanie_znak%C3%B3w (dostęp 28.01.2023)
4. Balabolka - http://www.cross-plus-a.com/pl/balabolka.htm (dostęp 28.01.2023)
5. Amazon Texttract - https://webuters.medium.com/what-is-amazon-textract-and-why-is-it-a-game-changer-77ae08c7125b (dostęp 28.01.2023)
6. ReCAPTCHA - https://ocrwdokumentach.pl/jak-digitalizujemy-ksiazki-o-tym-nie-wiedzac/ (dostęp 28.01.2023)
7. Tesseract - https://en.wikipedia.org/wiki/Tesseract_(software) (dostęp 28.01.2023)
8. PIL - https://en.wikipedia.org/wiki/Python_Imaging_Library (dostęp 28.01.2023)