ZAKŁAD SYSTEMÓW ZŁOŻONYCH
Wydział Elektrotechniki i Informatyki
ul. Wincentego Pola 2, 35-959 Rzeszów, tel. 17 865 1340
zsz.prz.edu.pl

WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Dokumentacja projektu

z przedmiotu

Python

Temat projektu : Opracowanie metody rysowania na kartce za pomocą drukarki 3D. Wejściem jest wygenerowane słowo przez AL w formie obrazu.

Imię i nazwisko	Piotr Dadel
Nr albumu	155687
Rok studiów	III EFZI

1. Wstęp

Zadaniem było utworzenie kodu który z zwykłego zdjęcia zawierającego podpis plik który umożliwił by napisanie go na kartce papieru przez odpowiednio przystosowaną drukarkę 3D.

Problem ten można było rozwiązać na kilka sposobów, ja wykorzystałem możliwość utworzenia z pliku JPG pliku SVG grafiki wektorowej a następnie przygotowanie na jego podstawie pliku Gcode wykorzystywanego przez drukarki 3D.

2. Wykonanie

W celu wykonania zadania użyłem kilku bibliotek takich jak :

OpenCV

svgwrite

numpy

svg_to_gcode

```
#Importowanie pakietów
import cv2
import svgwrite
import numpy as np
import svg_to_gcode
```

Pierwszym krokiem programu było pobranie obrazu za pomocą biblioteki openCV oraz przekształcanie go kolejnymi metodami aby uzyskać obraz składający się z wyraźnych krawędzi możliwych do zapisania w pliku wektorowym SVG.

```
##TWORZENIE PLIKU SVG

# Pobieranie zdjęcia do analizy
img = cv2.imread("img/fot4.jpg")
# Skalowanie zdjęcia do 70%
width = int(img.shape[1] * 70 / 100)
height = int(img.shape[0] * 70 / 100)
new_size = (width, height)
img = cv2.resize(img, new_size)
#wyświetlenie zdjęcia
cv2.imshow('Obraz', img)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Zmiana kolorów obrazu na odcienie szarości
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#zastosowanie fitlra w celu lepszego progowania obrazu
gray = cv2.medianBlur(gray, 5)

# Progowanie obrazu
thresh = cv2.adaptiveThreshold(gray, 200, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 11, 2)
thresh = cv2.bilateralFilter(thresh, 9, 60, 60)

# Szukanie konturów
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

W celu lepszego zaobserwowania jak budowane są kontury podczas wykonywania zadania dodatkowo dodałem linijki umożliwiające wyświetlenie obrazu z zaznaczonymi konturami w ilości wpisanej w programie, w przypadku niektórych podpisów kontury nie zawsze są ze sobą idealnie połączone co sprawia problemy w tworzeniu takiego pliku

```
# Sortowanie konturów wg długości (malejąco)
contours = sorted(contours, key=cv2.contourArea, reverse=True)
# Wybieranie x najdłuższych konturów
x = 1
contours = contours[:x]
# Rysowanie konturów
cv2.drawContours(img, contours, -1, (0, 255, 0), 2)
cv2.imshow('Contours', img)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Wybieramy najdłuższy kontur
contour = max(contours, key=cv2.contourArea)

# Konwersja konturu do postaci złożonej z punktów
curve = np.squeeze(contour)

# Utworzenie pliku SVG i dodanie krzywej do niego
dwg = svgwrite.Drawing('podpis.svg')
path = svgwrite.path.Path(d='M' + str(curve[0][0]) + ',' + str(curve[0][1]))
for point in curve[1:]:
    path.push('L', str(point[0]), ',', str(point[1]))
dwg.add(path)

# Zapisanie pliku SVG
dwg.save()
```

Kolejnym trudnym krokiem było utworzenie pliku Gcode który jest bardzo problematyczny ze względu na złożoność informacji jakie przechowuje:

-parametry drukarki (wielkość powierzchni roboczej itp.)

-parametry wydruku

-działania elementów drukarki jak wentylatory grzałki itp.

-szybkość druku lub pisania oraz każdy poszczególny ruch w celu uzyskania finalnego wydruku

Dużym problemem jest poznanie zawartość całego języka w których pisane są Gcody tak aby wszystkie parametry odpowiednio dostosować do naszej drukarki zazwyczaj kody generuje program do obsługi drukarki tzw. Slicer który na podstawie pliku stl generuje kroki ruchy i parametry wydruku wyklikane na interfejsie przez użytkownika.

```
##KONWERTOWANIE SVG DO GCODE

from svg_to_gcode.svg_parser import parse_file
from svg_to_gcode.compiler import Compiler, interfaces
from svg_to_gcode.formulas import linear_map

# Instantiate a compiler, specifying the custom interface and the speed at which the tool should move.
gcode_compiler = Compiler(interfaces.Gcode, movement_speed=100, cutting_speed=60, pass_depth=-20)

curves = parse_file("podpis.svg") # Parse an svg file into geometric curves

gcode_compiler.append_curves(curves)
gcode_compiler.compile_to_file("podpis1.gcode")
```

Program generuje plik Gcode ale jest on uszkodzony i nie są w stanie go otworzyć w celu podejrzenia programy typu Slicer.

3. Wnioski

Wykonanie tego zadania wymagało przeanalizowania i sprawdzenia wielu bibliotek języka python często nie działających w sposób oczekiwany, oraz nauki obsługi wykorzystanych bibliotek.

Największym problemem była konwersja między różnymi formatami plików i tym że dane biblioteki były tworzone w wielu przypadkach pod dane zastosowanie co utrudniało zastosowanie bibliotek do innych funkcji.

Kolejnym problemem utrudniającym tworzenie takiego programu przez brak informacji na ten temat w Internecie było to że darmowy program do obróbki grafiki **Inkscape** zawiera zestaw narzędzi rozwiązujący wszystkie problemy tego zagadnienia w przyjemnym interfejsie graficznym gdzie cały czas widzimy co dzieje się z naszym obrazem. Program Inkscape jest w stanie wygenerować plik SVG z pliku jpg a następnie za pomocą odpowiedniej wtyczki wygenerować plik Gcode z przestrzenną reprezentacją pliku SVG gotową do użycia przez drukarkę 3D.

Przez trudność i złożoność zagadnienia a także mój brak doświadczenia tworzony program ma wiele niedociągnięć i nadal wymaga włożenia wiele pracy aby wykonywał zakładane cele poprawnie.