

**WYDZIAŁ
MATEMATYKI
I FIZYKI STOSOWANEJ
POLITECHNIKI RZESZOWSKIEJ**

Wojciech Rolewski

**Wymiar pudełkowy i narzędzie do jego
wyznaczania**

Projekt inżynierski

**Opiekun pracy:
dr hab. inż. Dominik Strzałka, prof. PRz**

Rzeszów, 2024

1. Wstęp	3
1.1. Wprowadzenie	4
2. Geneza powstania wymiaru pudełkowego	5
2.1. Historia i rozwój koncepcji	5
2.2. Praktyczne zastosowanie wymiaru pudełkowego	6
3. Teoretyczny opis wymiaru pudełkowego	11
3.1. Definicja i charakterystyka koncepcji	11
3.2. Opis matematyczny metody	12
3.3. Wyznaczanie wymiaru pudełkowego obiektu	13
4. Tworzenie narzędzie do wyznaczania wymiaru pudełkowego	14
4.1. Wybór środowiska programistycznego	14
4.2. Analiza wymagań	15
4.3. Implementacja funkcjonalności	16
4.3.1. Wykorzystywane biblioteki	16
4.3.2. Funkcje obsługi interfejsu	17
4.3.3. Funkcje przetwarzania obrazu	18
4.3.4. Funkcje analizy danych	21
5. Zastosowania programu	23
5.1. Prezentacja działania	23
5.2. Analiza obrazów	26
5.3. Wnioski	30
Literatura	30

1. Wstęp

W świetle dynamicznego postępu naukowego i technologicznego, fraktale stanowią fascynujące pole badawcze, które przyciąga uwagę inżynierów ze względu na swoją wyjątkową strukturę i złożoność. Jako obiekty matematyczne o niezwykłej nieregularności i samopodobieństwie na różnych skalach, nie tylko zdobywają uznanie ze względu na swoją abstrakcyjną naturę, ale również odgrywają kluczową rolę w zrozumieniu struktur występujących w przyrodzie (Rys. 1). Ich zdolność do opisywania chaotycznych i złożonych wzorców sprawia, że stają się istotnym narzędziem w analizie, modelowaniu i rozwiązywaniu problemów w różnych dziedzinach inżynierii. Poprzez analizę fraktali, dążymy do lepszego zrozumienia ich roli w procesie projektowania, optymalizacji oraz tworzenia innowacyjnych rozwiązań.

Badania nad geometrią fraktalną wnoszą nowe spojrzenie na struktury, które wykraczają poza konwencjonalne pojęcie geometryczne. Jest to obszar, który gwałtownie rozwija się w kontekście projektowania inżynieryjnego, ponieważ umożliwia bardziej precyzyjne modelowanie naturalnych oraz sztucznych struktur.



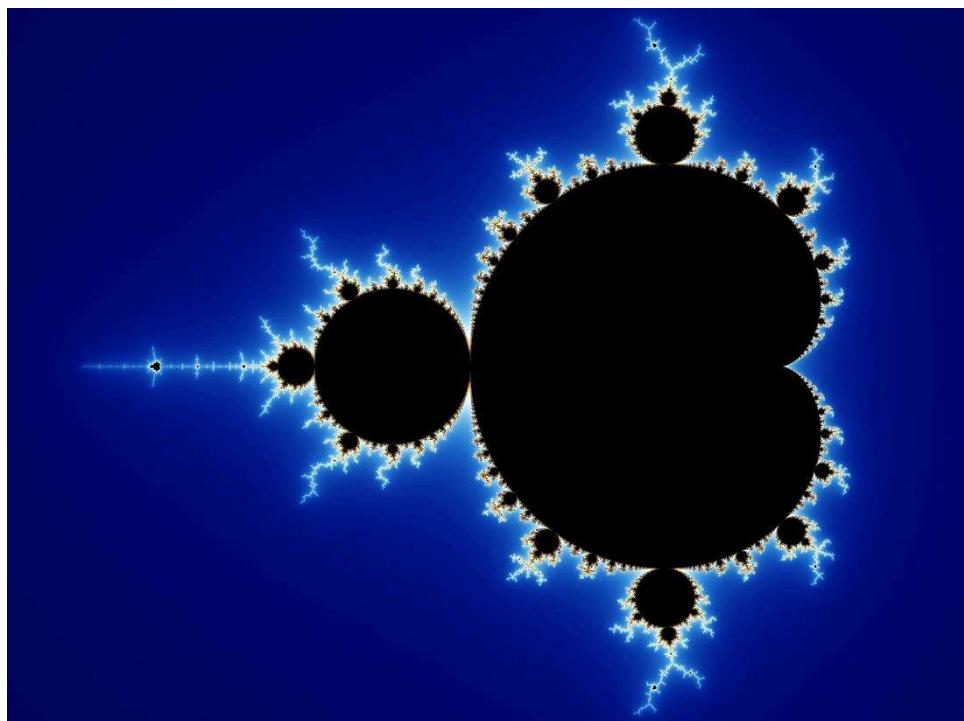
Rys. 1. Fraktalne oblicza natury.

Źródło: [9].

W ramach przedstawianego projektu autor skupi się na zgłębianiu geometrii fraktalnej, analizując jej fundamentalne aspekty matematyczne poprzez wykorzystanie wymiaru Minkowskiego-Bouliganda.

1.1. Wprowadzenie

Francuski matematyk Benoît Mandelbrot stworzył koncepcję fraktali w latach 70. XX wieku. W artykule „How Long is the Coast of Britain?” podjął temat statystycznego samopodobieństwa i wymiaru fraktalnego dając do zrozumienia, że większość obiektów naturalnych jest tak złożona i nieregularna, że nie da się ich dobrze opisać w prostych, dobrze znanych kategoriach, a te złożone obiekty wykazują silne samopodobieństwo. Z jego punktu widzenia linia brzegowa jest bardzo reprezentatywnym przykładem fraktala, który wykazuje dokładne lub statystyczne samopodobieństwo; ponadto wskazał, że jest to podstawową cechą fraktali. Oznacza to, że obiekty fraktalne można podzielić na wiele części, z których każda może być kopią całości w małej skali lub przynajmniej są do siebie podobne pod względem statystycznym, co dobrze obrazuje zbiór Mandelbrota (Rys. 2). Dlatego coraz więcej badaczy ma tendencję do przyjmowania wymiaru fraktalnego jako wskaźnika opisującego stopień samopodobieństwa i złożoności w geometrii.



Rys. 2. Zbiór Mandelbrota.

Źródło: [10].

Niezwykła użyteczność ujęcia fraktalnego i w konsekwencji zastosowanie go w analizie zjawisk będących przedmiotem badań najodleglejszych dyscyplin sprawiła, że pojęciem wymiaru fraktalnego zaczęto określać najrozmaitsze wymiary, często definiowane w całkowicie odmienny sposób i przyjmujące różne wartości dla tego samego obiektu.

2. Geneza powstania wymiaru pudełkowego

2.1. Historia i rozwój koncepcji

Genezy wymiaru pudełkowego można doszukiwać się w pracach naukowych takich matematyków jak Hermann Minkowski i Georges Bouligand. Obaj badacze wnieśli znaczący wkład w rozwój i zrozumienie geometrii fraktalnej, dającej podstawy tytuowej koncepcji.

Hermann Minkowski (Rys. 3), niemiecki matematyk, wprowadził pojęcie wymiaru Minkowskiego na początku XX wieku jako sposób pomiaru wymiarowości zbiorów i krzywych. Jego praca położyła podwaliny pod zrozumienie skomplikowanych właściwości geometrycznych fraktali i ich wymiarów. Pomyśły Minkowskiego przyczyniły się do eksploracji nieregularnych kształtów i struktur.



Rys. 3. Hermann Minkowski (1864 – 1909).

Źródło [11].

Georges Bouligand, francuski matematyk, rozwinął idee Minkowskiego i rozwinął koncepcję wymiaru pudełkowego pod koniec lat dwudziestych XX wieku. Praca Bouliganda polegała na podzieleniu obiektu fraktalnego na mniejsze pudełka o równej wielkości i określeniu liczby pudeł potrzebnych do pokrycia obiektu. Badając, jak zmieniała się ta miara wraz ze zmniejszaniem się rozmiarów pudełek, Bouligand dostarczył ilościowej miary fraktalnej złożoności i wymiarowości obiektu.

Wspólne prace wyżej wspomnianych naukowców położyły podwaliny pod wymiar Minkowskiego-Bouliganda, który stał się podstawowym narzędziem w badaniu i charakteryzowaniu fraktali i złożonych struktur geometrycznych. Wymiar ten znalazł zastosowanie w różnych dziedzinach, w tym w fizyce [4], biologii [2], analizie obrazów i grafice

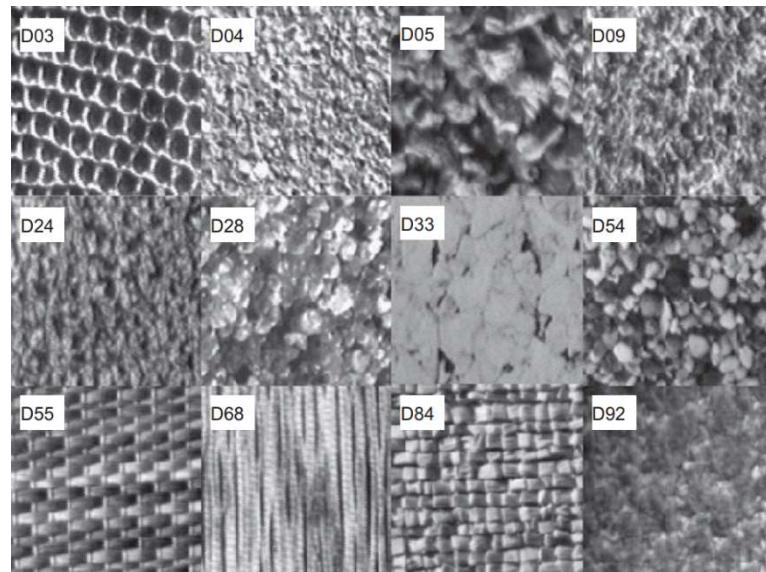
komputerowej [1], umożliwiając naukowcom ilościowe określanie i analizowanie skomplikowanych wzorców obiektów fraktałnych.

Genezę wymiaru Minkowskiego-Bouliganda można przypisać dążeniu do zrozumienia i ilościowego określenia złożoności nieregularnych kształtów i struktur.

2.2. Praktyczne zastosowanie wymiaru pudełkowego

Geometria fraktalna zapewnia model matematyczny dla wielu złożonych obiektów występujących w przyrodzie, takich jak linie brzegowe, góry i chmury. Obiekty te są zbyt złożone, aby posiadać charakterystyczne rozmiary i być opisane tradycyjną geometrią euklidesową. Samopodobieństwo jest podstawową właściwością fraktali i można je określić ilościowo za pomocą wymiaru fraktałnego. Istnieje wiele definicji wymiarów fraktałnych mających uzasadnienie w różnych sytuacjach. Metody ich szacowania możemy dzielić ze względu na wykorzystywane w tym celu sposoby obliczania, takie jak te oparte na geometrii (wymiar Hausdorfa czy Minkowskiego), analizie sygnałów (Analiza Fouriera) czy statystyce (wymiar entropijny). W pracy skupiamy się na jednej z metod geometrycznych, jaką jest wymiar liczenia pudełek. Jest to jedna z najbardziej popularnych metod, a powodem tego jest jego prostota i automatyczność. Jest on praktycznym narzędziem stosowanym na wielu płaszczyznach w celu analizy złożonych i nieregularnych struktur. Jego zastosowanie możemy spotkać w takich dziedzinach jak:

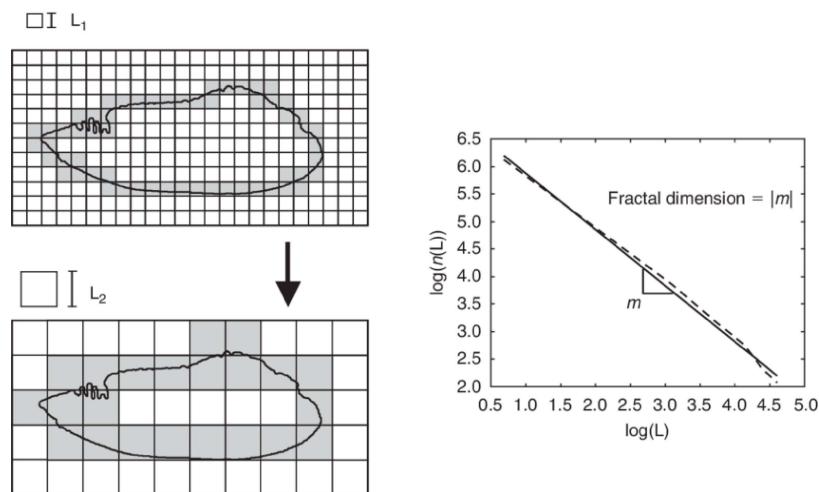
- 1) Analiza obrazu: W przetwarzaniu obrazu jest wykorzystywany do określania tekstury i złożoności kształtu obrazów. Pomaga w identyfikowaniu wzorów, tekstur i nieregularności w obrazach, które znajdują zastosowanie w dziedzinie rozpoznawaniu wzorców, klasyfikacji obrazów i wykrywaniu obiektów (Rys. 4). Istnieje wiele modyfikacji i usprawnień metody liczenia pudełek, które powstały w celu poprawienia dokładności szacunków. Aby poprawić dokładność obliczeń należy użyć jak najmniejszej liczby pudełek pokrywając przy tym całkowicie odpowiednią powierzchnię [1].



Rys. 4. Tekstury Brodatza.

Źródło:[1]

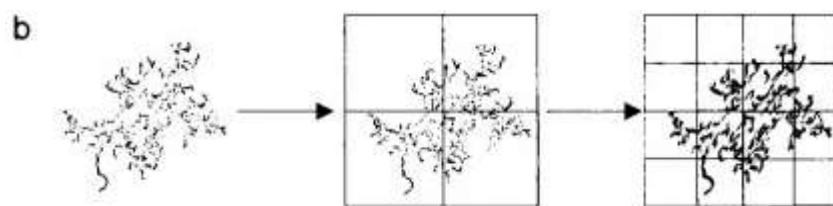
- 2) Analiza terenu: Do analizy i modelowania powierzchni terenu, takich jak krajobrazy i linie brzegowe (Rys. 5). Zapewnia wgląd w chropowatość, złożoność i fraktałny charakter tych powierzchni, co ma wpływ na badania środowiskowe, geologię i systemy informacji geograficznej.



Rys. 5. Ilustracja metody obliczania „pułapek” w celu oszacowania wymiaru fraktalnego konturu otolitu.

Źródło:[12]

- 3) Badania medyczne: Analiza fraktalna, w tym wymiar pudełkowy, jest wykorzystywana w różnych obszarach badań medycznych. Można go zastosować do badania złożoności struktur biologicznych (Rys. 6), takich jak rozgałęzienia naczyń krwionośnych, dróg oddechowych płuc i sieci neuronowych. Analiza ta pomaga w zrozumieniu organizacji i rozwoju tych systemów i może mieć wpływ na diagnostykę i leczenie chorób.

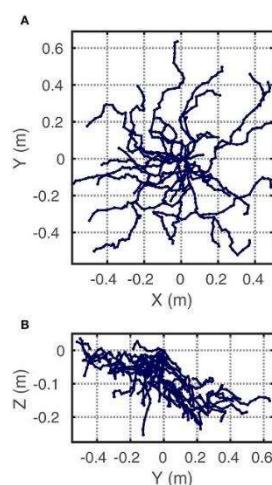


Rys. 6. Analiza komórki mikrogleju.

Źródło: [2]

Na powierzchnię badanego obszaru nakładane są kolejno siatki złożone z kwadratów o coraz mniejszej długości boku, w każdej siatce zliczana jest liczba kwadratów zawierających część badanej struktury. Tę liczbę przedstawia się na wykresie względem długości boku kwadratu w skali log-log. Wymiar fraktalny powiązany jest z nachyleniem linii regresji otrzymanej krzywej [2].

- 4) Badania botaniczne: Wymiar fraktalny, szacowany na podstawie liczenia pudełek, to miara używana do charakteryzowania złożoności anatomicznej rośliny lub charakterystyki wypełnienia przestrzeni do różnych celów (Rys. 7). Założenie statystycznego samopodobieństwa stanowi podstawę zasadności procedury.

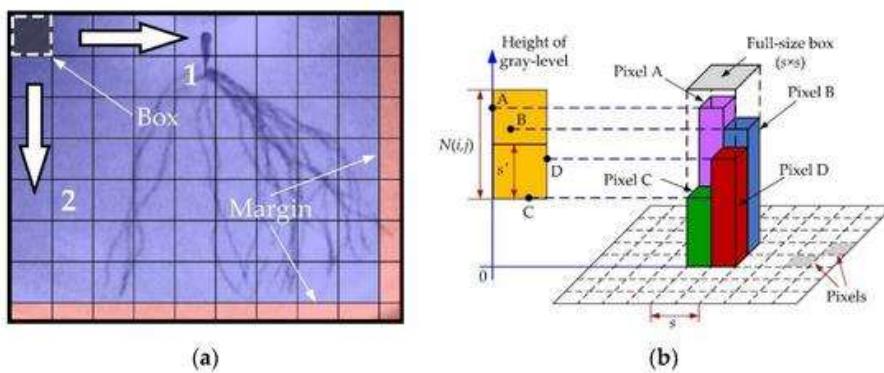


Rys. 7. Przykład reprezentacji systemu korzeniowego używanego w metodzie liczenia pudełek. Źródło: [3].

Procedura liczenia pudełek jest również obarczona błędem wynikającym z dowolnego rozmieszczenia siatki, znanym jako błąd kwantyzacji (QE), który jest ścisłe dodatni i zmienia się w

funkcji skali, co sprawia, że jest problematyczny w procedurze szacowania nachylenia. Celem badania było scharakteryzowanie wpływu błędu kwantyzacji, zapewnienie skutecznej metody zmniejszania tego błędu oraz ocena założenia statystycznego samopodobieństwa zbiorów danych źródłowych stosowanych w ostatnich badaniach. Grube systemy korzeniowe 36 krzewów scyfryzowano i poddano liczeniu pudełek. Zastosowano algorytm wyszukiwania wzorców w celu zminimalizowania błędu kwantyzacji poprzez optymalizację rozmieszczenia siatki [3].

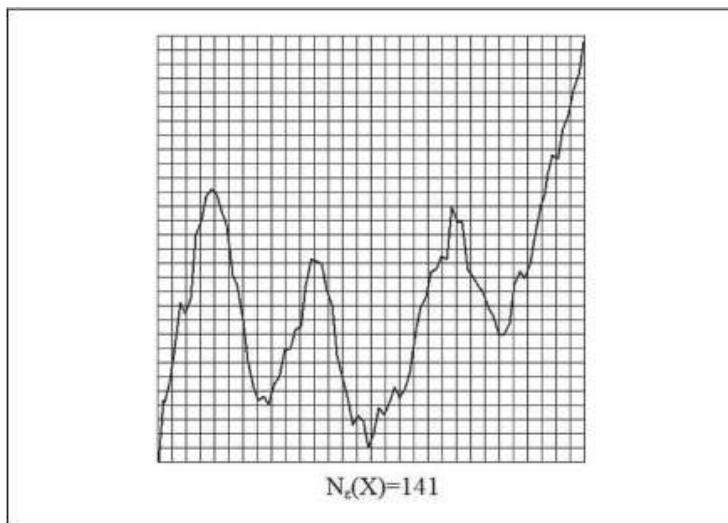
- 5) Nauka o materiałach: analiza fraktalna, w tym wymiar pudełkowy, znajduje zastosowanie w analizie materiałów do charakteryzowania ich mikrostruktury. Pomaga w zrozumieniu właściwości, takich jak porowatość, chropowatość i pole powierzchni, które są krytyczne w obszarach takich jak projektowanie materiałów.



Rys. 8. Ulepszona metoda analizy obrazu do oceny właściwości izolatora. Źródło [4]:

Przykład analizy obrazu izolatora, którego kolejne rozgorzenia skutkowałyby zwęgleniem powierzchni i pogorszeniem izolacji. Do analizy tych obrazów (Rys. 8) zastosowano ulepszoną metodę różnicowego liczenia pudełek (IDBM) w oparciu o teorię fraktali. Badanie to może być korzystne w dążeniu do zrozumienia charakterystyki rozgorzenia podczas długotrwałej pracy. Zasugerowana metoda służy do szacowania właściwości propagacji wyładowań i charakteryzacji pogorszenia jakości na powierzchni izolatora [4].

- 6) Inwestycje finansowe: Dotychczas zaprezentowano wiele metod szacowania wymiaru fraktalnego mających zastosowanie w kwestiach finansowych. Nie wszystkie z nich mogą jednak nadawać się w przypadku specyficznego przedmiotu badania, jakim są szeregi czasowe cen bądź stóp zwrotu z instrumentów finansowych. Wymiar pudełkowy znajduje jednak zastosowanie także w dziedzinie inwestycji, a mianowicie jest jedną z metod szacowania wymiaru fraktalnego finansowych szeregów czasowych. Pokazuje jakie jest tempo zmian funkcji czy trajektorii procesu stochastycznego. Jest to jeden z argumentów przemawiający za wykorzystaniem w analizach właśnie wymiaru pudełkowego. Innym powodem dla którego warto po niego sięgnąć jest jego czytelna interpretacja ekonomiczna.



Rys. 9. Przykładowa siatka kwadratowa oraz $N_\varepsilon(X)$ dla danego Σ oraz zbioru X . Źródło: [8].

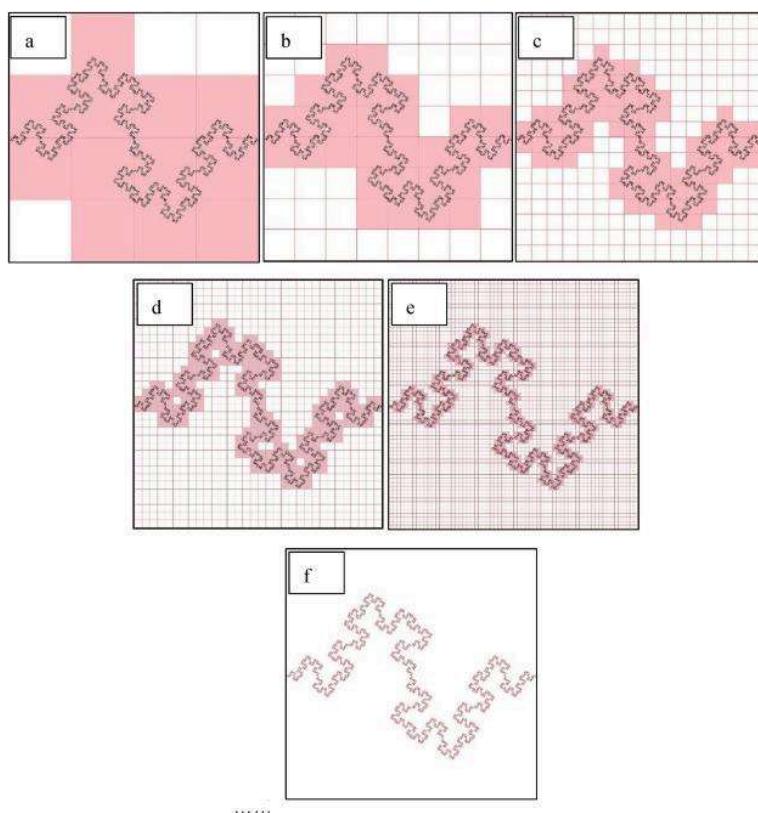
Wybór sposobu kalkulowania wymiaru pudełkowego jest uzależniony od stopnia prostoty danego podejścia dla analizowanego obiektu geometrycznego. Do wyznaczenia wartości wymiaru na Rys. 9 postępowano zgodnie z definicją wzoru (1). Możliwe jest uzyskanie odpowiedzi na pytanie jak zachowuje się badany obiekt w sytuacji malejących rozmiarów pokrywających go hipersześcianów. Mankamentem pojawiającym się przy stosowaniu wymiaru pudełkowego jest fakt, że nie dla wszystkich obiektów geometrycznych jest on dobrze określony, ponieważ nie zawsze istnieje granica wyznaczana we wzorze (1). W takiej sytuacji wyznaczamy wielkości zawsze istniejące, a mianowicie dolny i górny wymiar pudełkowy [8].

3. Teoretyczny opis wymiaru pudełkowego

W poniższych podrozdziałach autor skupi się na przedstawieniu definicji oraz omówieniu koncepcji wymiaru pudełkowego. Opisane zostaną także matematyczne aspekty metody oraz przedstawiony przykład obliczenia wymiaru na prostym przykładzie.

3.1. Definicja i charakterystyka koncepcji

Podstawową ideą wymiaru Minkowskiego-Bouliganda jest podział analizowanego obiektu na coraz mniejsze pudełka, nazywane też jednostkami pomiarowymi (Rys. 10), o stałej skali. Następnie zliczane jest, ile takich pudełek jest potrzebnych do pokrycia obiektu w zależności od rozmiaru pudełek. Wartość ta wynika z badania, jak ta liczba pudełek zmienia się wraz ze zmianą rozmiaru pudełek. Im bardziej nieregularny i fraktalny jest analizowany obiekt, tym większa jest różnica w liczbie pudełek dla różnych skal. Metoda ta umożliwia opisanie złożoności i skomplikowania obiektów o nieregularnym kształcie, takich jak wybrzeża czy skomplikowane linie brzegowe.



Rys. 10. Liczba N pudełek zmienia się wraz ze zmianą długości boku kwadratu.

Źródło: [5].

3.2. Opis matematyczny metody

W celu obliczenia wymiaru fraktalnego metodą pudełkową badany obraz należy umieścić na regularnej siatce składającej się z elementów o długości boków równej δ , a następnie policzyć, ile elementów siatki (pudełek) pokrywa obraz. Liczba, którą otrzymamy $N_\delta(F)$ będzie zależna od rozmiaru elementów siatki. W dalszych iteracjach należy stopniowo zmniejszać wartość F i określać odpowiednie wartości $N_\delta(F)$. Istota określenia wymiaru pudełkowego polega więc na obserwacji, jak zmienia się $N_\delta(F)$ przy zmianie F . Dla obrazów występujących w naturze liczba elementów występujących w kolejnych iteracjach nie jest stała, dlatego wymiar pudełkowy określa się jako wartość graniczną, gdzie długość pudełka zmierza do zera [6].

Bazując na definicji wymiaru Hausdorffa możemy otrzymać sformalizowaną definicję wymiaru pudełkowego.

Niech będzie dany dowolny ograniczony podzbiór F przestrzeni R^n .

Wymiar pudełkowy jest definiowany jako

$$D_b(F) = \frac{\log N_\delta(F)}{\log \frac{1}{\delta}} \quad (1)$$

przy założeniu, że ta granica istnieje, gdzie $N_\delta(F)$ jest jednym z następujących:

- najmniejsza liczba zamkniętych kul o promieniu δ pokrywających F ;
- najmniejsza liczba sześcianów o krawędzi δ pokrywających F ;
- liczba sześcianów powstały z siatek o wielkości oczek δ przecinających F ;
- najmniejsza liczba zbiorów o średnicy nie większej niż δ pokrywających F ;
- największa liczba rozłącznych kul o promieniu δ o środkach w F .

Wyrażenie (1) oznacza, że $N_\delta \propto \delta^{-s}$ dla małych δ , gdzie $s = D_b(F)$. Dokładniej można to zapisać jako:

$$N_\delta(F)\delta^s \rightarrow \begin{cases} \infty & s < D_b(F) \\ 0 & s > D_b(F) \end{cases}, \quad (2)$$

uwzględniając, że

$$N_\delta(F)\delta^s = \inf \left\{ \sum_i \delta^s \right\}. \quad (3)$$

Powyższą definicję można porównać z definicją wymiaru Hausdorffa. Wymiar pudełkowy jest zatem swego rodzaju uproszczeniem wymiaru Hausdorffa. Wymiary te jednak nie zawsze się pokrywają [7].

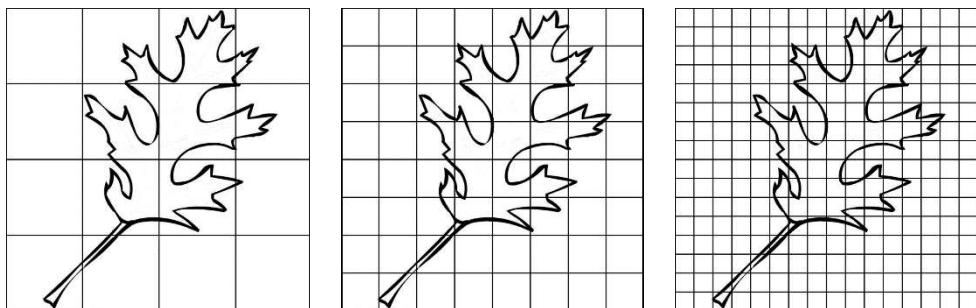
3.3. Wyznaczanie wymiaru pudełkowego obiektu

Przykładowym obiektem, którego wymiar pudełkowy zostanie wyznaczony będzie liść przedstawiony na Rys. 11. W tym celu autor postąpi zgodnie z krokami przedstawionymi w podrozdziale 3.2. Zakładając, że $N(\delta)$ jest liczbą pudełek o długości boku δ pokrywających obraz, wymiar pudełkowy takiego obrazu jest zdefiniowany zależnością, którą opisuje wzór (1), przy założeniu, że granica istnieje.

Tab. 1. Dane do obliczenia wymiaru pudełkowego.

δ	$N(\delta)$	$\log(\frac{1}{\delta})$	$\log(N(\delta))$	D_b
0,2500	11	0,6021	1,0414	1,4916
0,1250	32	0,9001	1,5051	
0,0625	87	1,2041	1,9395	

$$N(\delta) = 11, \delta = 0,2500 \quad N(\delta) = 32, \delta = 0,1250 \quad N(\delta) = 87, \delta = 0,0625$$



Rys. 11. Obrazowanie trzech kroków nakładania siatki pudełek na wyznaczany obiekt. Źródło: Opracowanie własne.

Praktycznie wymiar pudełkowy określa się w ten sposób, że rysujemy wykres $\log(N(\delta))$ w funkcji $\log(\frac{1}{\delta})$ i aproksymujemy go linią prostą. Nachylenie otrzymanej prostej jest wymiarem pudełkowym. Na Rys. 11 przedstawiono metodę obliczania wymiaru pudełkowego dla trzech wielkości elementów siatki. Dane wykorzystane do sporządzenia wykresu z Rys. 12 zamieszczono w Tab. 1. Wyznaczone nachylenie prostej będące zarazem szukanym wymiarem pudełkowym wyniosło 1,4916 co przedstawia Rys. 12.



Rys. 12. Ilustracja metody obliczania wymiaru pudełkowego dla danych z tab. 1. Źródło: Opracowanie własne.

4. Tworzenie narzędzie do wyznaczania wymiaru pudełkowego

W poniższych podrozdziałach autor skupił się na przedstawieniu wymagań oraz opisie implementacji funkcjonalności programu służącego do wyznaczania wymiaru pudełkowego. Uzasadnia wybór środowiska programistycznego i przedstawia wymagania, zarówno funkcjonalne jak i niefunkcjonalne jakie program powinien spełniać. Dokładnie opisuje także funkcjonalności programu, wykorzystywane biblioteki oraz utworzone klasy i funkcje.

4.1. Wybór środowiska programistycznego

Podejmując decyzję w kontekście wyboru środowiska programistycznego autor skupił się na poszukiwaniu wszechstronnego języka znajdującego zastosowanie w wielu dziedzinach. Mając na uwadze potrzebę przetwarzania obrazu, wybór padł na język Python oferujący szeroką gamę bibliotek dedykowanych przetwarzaniu obrazów. Inną kluczową zaletą wspomnianego języka jest czytelność tworzonego kodu oraz prostota składni. Nie można zapomnieć także o tym, że język ten cieszy się dużą popularnością wśród społeczności programistycznej, co przekłada się na dostęp do licznych zasobów, forów dyskusyjnych oraz dokumentacji online, co w razie problemów umożliwia sprawne znalezienie pomocy.

4.2. Analiza wymagań

Przed przystąpieniem do implementacji programu przetwarzającego obrazy należało dokładnie zdefiniować wymagania, które powinny być uwzględnione podczas tworzenia tego rodzaju oprogramowania. Jeżeli chodzi o wymagania niefunkcjonalne, program powinien:

- działać sprawnie nawet dla dużych obrazów, a czas przetwarzania nie powinien znacząco wzrastać wraz z rozmiarem obrazu;
- posiadać intuicyjny interfejs będący prosty w obsłudze pozwalający wybrać obraz, zrozumieć prezentowane wyniki oraz skorzystać z opcji zapisu zmodyfikowanego obrazu;
- być elastyczny i umożliwiać rozbudowę o dodatkowe funkcje.

Jeżeli natomiast mówimy o wymaganiach funkcjonalnych, należałyby skupić się na umożliwieniu dostępu do takich opcji jak:

- swobodny wybór obrazu w wielu formatach (PNG,JPG,JPEG,BMP) do przetworzenia w formie interaktywnego okna dialogowego;
- przetwarzanie wybranego obrazu, rysowanie siatki zgodnie z określonymi kryteriami i zliczanie zaznaczonych obszarów, a po przetworzeniu wyświetlenie zaznaczonych obszarów na obrazie;

- prezentowanie zmodyfikowanego obrazu z zaznaczonymi obszarami oraz wyświetlanie liczby zaznaczonych obszarów;
- prezentowanie wykresu, na którym punkty zaznaczane są automatycznie podczas zmiany rozmiaru pudełka;
- swobodne modyfikowanie parametrów obrazu takich jak rozmiar pudełek czy poziom kontrastu;
- dopasowywanie linii regresji do punktów wykresu;
- zapisywanie zmodyfikowanego obrazu z zaznaczonymi obszarami do wybranej przez użytkownika destynacji;
- panel interfejsu zawierający przyciski wykonujące wyżej wymienione funkcjonalności.

Analiza wymagań wykazała, że program powinien umożliwiać wybór, przetwarzanie, wyświetlanie wyników oraz zapis zmodyfikowanego obrazu. Dostosowanie do kryteriów akceptacji dla wymagań funkcjonalnych i niefunkcjonalnych zapewni efektywną i zadowalającą pracę z programem. Warto również zauważyc, że program powinien być łatwo rozwijalny, aby przyszłe dodatki czy modyfikacje były możliwe bez większych trudności.

4.3. Implementacja funkcjonalności

4.3.1. Wykorzystywane biblioteki

- **CV2** to biblioteka wykorzystywana do przetwarzania obrazów i analizy komputerowej, zawierająca wiele funkcji związanych z manipulacją obrazami. W programie użyta została w celu wczytania obrazu (`cv2.imread()`) oraz do rysowania siatki kwadratów na obrazie;
- **Tkinter** jest standardową biblioteką interfejsu użytkownika dla Pythona, umożliwiającą tworzenie prostych aplikacji desktopowych z interfejsem graficznym. W programie użyta została w celu stworzenia interfejsu użytkownika zawierającego etykiety, przyciski czy pola tekstowe;
- **Scale, IntVar, Button, Label, Entry** to elementy modułu Tkinter służące do tworzenia konkretnych elementów interfejsu użytkownika, takich jak zmienne całkowitoliczbowe (`IntVar()`), przyciski (`Button()`), etykiety (`Label()`) czy pola tekstowe (`Entry()`). W kodzie są elementami interfejsu i obsługi użytkownika;
- **Math** to moduł matematyczny w standardowej bibliotece Pythona zawierający funkcje matematyczne. W programie użyty został do obliczeń matematycznych takich jak funkcja logarytmiczna (`log()`);

- **Numpy** jest potężną biblioteką do obliczeń numerycznych w Pythonie zapewniającą wydajne operacje na tablicach i macierzach. W kodzie użyta została w celu utworzenia i manipulacji danymi tablicowymi;
- **Matplotlib** to biblioteka służąca do tworzenia wykresów i wizualizacji danych. W programie wykorzystywana do generowania wykresów w interfejsie użytkownika;
- **Matplotlib.backends.backend_tkagg.FigureCanvasTkAgg** umożliwia umieszczenie wykresu w oknie interfejsu za pomocą narzędzia FigureCanvasTkAgg integrującego wykresy Matplotlib z interfejsem Tkinter;
- **Sklearn** to biblioteka do uczenia maszynowego. Zawarty w niej moduł linear_model zawiera funkcję `LinearRegression()`, która w programie użyta została w celu dopasowania linii regresji do danych punktów na wykresie.

```
import cv2
from tkinter import Tk, filedialog, Scale, IntVar, Button, Label, Entry
from math import log
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from sklearn.linear_model import LinearRegression
```

Listing 1. Wykorzystywane biblioteki.

4.3.2. Funkcje obsługi interfejsu

- **Choose_image()** jest funkcją tworzącą okno dialogowe pozwalające na wybór pliku obrazu. Wywołuje funkcję `read_image()` a następnie `create_interface()`, gdy wybrany zostanie obraz.

```
def choose_image():
    root = Tk()
    root.withdraw()

    file_path = filedialog.askopenfilename(title="Choose an image file",
                                           filetypes=[("Image files", "*.png;*.jpg;*.jpeg;*.bmp")])

    if file_path:
        read_image(file_path)
        create_interface()
```

Listing 2. Funkcja `choose_image`.

- **Create_interface()** to funkcja tworząca interfejs użytkownika z wykorzystaniem modułu Tkinter. Dodaje etykiety, przyciski oraz pola tekstowe czy wykresy na ekran interfejsu. Przy starcie programu wywołuje funkcję `update_image()` przy starcie programu.

```
def create_interface():
```

```

    global image_height, image_width, image, reverse_sign,
pułapek_label, box_size_label, threshold_var, slope_entry, plot_frame

image_height = image_properties[1]
image_width = image_properties[2]
reverse_sign = False

root = Tk()
root.title("Parameter Adjustment")

    box_size_label = Scale(root, label="Box Size", from_=3, to=50,
orient="horizontal", length=200, command=lambda x: update_image(int(x),
int(threshold_var.get()), reverse_sign))
    box_size_label.set(7)
    box_size_label.pack()

    threshold_var = IntVar()
    threshold_label = Scale(root, label="Threshold", from_=0, to=255,
orient="horizontal", length=200, variable=threshold_var, command=lambda
x: update_image(int(box_size_label.get()), int(x), reverse_sign))
    threshold_label.set(100)
    threshold_label.pack()

    switch_button = Button(root, text="Switch Sign",
command=switch_sign)
    switch_button.pack()

pułapek_label = Label(root, text="Liczba pułapek: ")
pułapek_label.pack()

plot_frame = PlotFrame(root)

slope_label = Label(root, text="Wymiar pułkowy:")
slope_label.pack()

slope_entry = Entry(root)
slope_entry.pack()

draw_regression_button = Button(root, text="Draw Regression Line",
command=draw_regression_line)
draw_regression_button.pack()

clear_plot_button = Button(root, text="Clear Plot",
command=clear_plot)
clear_plot_button.pack()

save_button = Button(root, text="Save Modified Image",
command=save_modified_image)
save_button.pack()

    update_image(int(box_size_label.get()), int(threshold_var.get()),
reverse_sign)

root.mainloop()

```

Listing 3. Funkcja `create_interface`.

4.3.3. Funkcje przetwarzania obrazu

- **Read_image()** to funkcja, która wczytuje obraz z wybranego pliku przy użyciu funkcji `cv2.imread()`, a następnie zapisuje właściwości tego obrazu takie jak szerokość i wysokość w liście `image_properties[]`.

```
def read_image(file_path):
    global image, image_height, image_width
    image = cv2.imread(file_path)
    image_height, image_width, _ = image.shape
    image_properties.append(image)
    image_properties.append(image_height)
    image_properties.append(image_width)
```

Listing 4. Funkcja `read_image`.

- **Preprocess_image()** jest funkcją przetwarzającą obraz poprzez podział na pudełka o zadanym rozmiarze (`box_size`). Dla każdego obszaru oblicza średnią wartość koloru niebieskiego. W zależności od warunku (`reverse`) decyduje czy zaznaczyć obszar na obrazie. Zwraca zmodyfikowany obraz oraz liczbę zaznaczonych obszarów (`enclose_count`).

```
def preprocess_image(box_size, threshold, reverse):
    global image

    enclose_count = 0
    temp_image = image.copy()

    for h in range(int(image_height + 1 / box_size)):
        for w in range(int(image_width / box_size + 1)):
            cv2.rectangle(temp_image, (box_size * w, box_size * h),
            (box_size + box_size * w, 1 + box_size + box_size * h),
            color=(0, 0, 255), thickness=1)

            selection = temp_image[box_size * h:1 + box_size + box_size
            * h, box_size * w:box_size + box_size * w, 1]
            try:
                blue_mean = selection.mean()
            except RuntimeWarning:
                blue_mean = 0

            if (blue_mean >= threshold) if reverse else (blue_mean <
            threshold):
                cv2.rectangle(temp_image, (box_size * w, box_size * h),
                (box_size + box_size * w, 1 + box_size + box_size * h),
                color=(0, 0, 125), thickness=-1)
                enclose_count += 1

    return temp_image, enclose_count
```

Listing 5. Funkcja `preprocess_image`.

- **Update_image()** to funkcja aktualizująca obraz, który uległ przetworzeniu przez funkcję `preprocess_image()`. Aktualizuje etykiety interfejsu wyświetlając przy

tym liczbę zaznaczonych obszarów. Dodaje punkt do wykresu na podstawie rozmiaru pudełka (`box_size`) i liczby zaznaczonych obszarów.

```
def update_image(box_size, threshold, reverse):
    global pudelek_label, minkowski_label, plot_frame
    temp_image, enclose_count = preprocess_image(box_size, threshold,
reverse)

    pudelek_label.config(text=f"Liczba pudełek: {enclose_count}")

    if box_size > 0 and enclose_count > 0:
        plot_frame.add_point(log(1/box_size), log(enclose_count))

cv2.imshow("Gridded Image", temp_image)
```

Listing 6. Funkcja `update_image`.

- **Save_modified_image()** jest funkcją której celem jest zapisywanie zmodyfikowanego obrazu do pliku.

```
def save_modified_image():
    global image
    file_path = filedialog.asksaveasfilename(defaultextension=".png",
filetypes=[("PNG files", "*.png"), ("JPEG files", "*.jpg")])
    if file_path:
        cv2.imwrite(file_path, image)
```

Listing 7. Funkcja `save_modified_image`.

4.3.4. Funkcje analizy danych

- **Switch_sign()** to funkcja mająca na celu zmianę wartości flagi (`reverse_sign`). Wykorzystywana do zarządzania czy obliczane będą ciemne czy jasne elementy obrazu. Dodatkowo używając przycisku zmieniającego znak, powoduje czyszczenie wykresu oraz aktualizację obrazu na podstawie nowej wartości `reverse_sign`.

```
def switch_sign():
    global reverse_sign, plot_frame
    reverse_sign = not reverse_sign
    plot_frame.clear_plot()
    update_image(int(box_size_label.get()), int(threshold_var.get()),
reverse_sign)
```

Listing 8. Funkcja `switch_sign`.

- **Draw_regression_line()** jest funkcją rysującą linię regresji. Oblicza współczynnik nachylenia (`slope()`) i aktualizuje wykres.

```
def draw_regression_line():
    global plot_frame, slope_entry
    plot_frame.clear_regression_line()
    plot_frame.draw_regression_line()
```

```

if len(plot_frame.points_x) >= 2:
    X, Y = zip(*zip(plot_frame.points_x, plot_frame.points_y))
    X = np.array(X).reshape(-1, 1)
    Y = np.array(Y)

    model = LinearRegression()
    model.fit(X, Y)
    slope = model.coef_[0]
    slope_entry.delete(0, 'end')
    slope_entry.insert(0, f"{slope:.5f}")

```

Listing 9. Funkcja draw_regression_line.

- **PlotFrame** to klasa inicjalizująca wykres przy użyciu biblioteki matplotlib. Zawiera metody służące do dodawania punktu, czyszczenia wykresu, rysowania linii regresji oraz aktualizacji wykresu.

```

class PlotFrame:
    def __init__(self, root):
        self.figure, self.ax = plt.subplots()
        self.canvas = FigureCanvasTkAgg(self.figure, master=root)
        self.canvas_widget = self.canvas.get_tk_widget()
        self.canvas_widget.pack()
        self.points_x = []
        self.points_y = []
        self.regression_line = None

    def add_point(self, x, y):
        self.points_x.append(x)
        self.points_y.append(y)
        self.update_plot()

    def clear_plot(self):
        self.points_x = []
        self.points_y = []
        self.clear_regression_line()
        self.update_plot()

    def clear_regression_line(self):
        if self.regression_line:
            self.regression_line.remove()
            self.regression_line = None

    def draw_regression_line(self):
        if len(self.points_x) >= 2:
            X, Y = zip(*zip(self.points_x, self.points_y))
            X = np.array(X).reshape(-1, 1)
            Y = np.array(Y)

            model = LinearRegression()
            model.fit(X, Y)

            X_fit = np.linspace(min(X), max(X), 100).reshape(-1, 1)
            Y_fit = model.predict(X_fit)

```

```

        self.regression_line, = self.ax.plot(X_fit, Y_fit,
color='red', label='Regression Line')
        self.ax.legend()
        self.canvas.draw()

def update_plot(self):
    self.ax.clear()
    self.ax.plot(self.points_x, self.points_y, marker='o',
linestyle='None', color='b')
    self.ax.set(xlabel='log(1/δ)', ylabel='log(N_δ(F))',
                title='Fractal Dimension Estimation')
    self.ax.grid()
    self.canvas.draw()

```

Listing 10. Klasa PlotFrame.

- **Clear_plot()** jest prostą funkcją czyszczącą wykres poprzez wywołanie metody w instancji klasy **PlotFrame**.

```

def clear_plot():
    global plot_frame
    plot_frame.clear_plot()

```

Listing 11. Funkcja **clear_plot**.

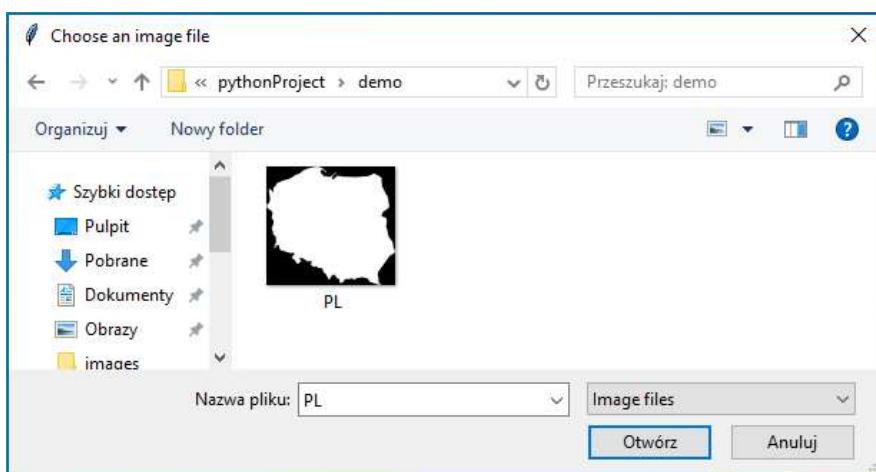
5. Zastosowania programu

W niniejszym rozdziale przedstawiona zostanie prezentacja działania programu mającego na celu wizualizację oraz wyznaczenie wymiaru pudełkowego dla wybranych obrazów. Przeprowadzone zostaną testy programu na obrazach o różnej rozdzielczości, kontraste kolorów oraz złożoności kształtów.

5.1. Prezentacja działania

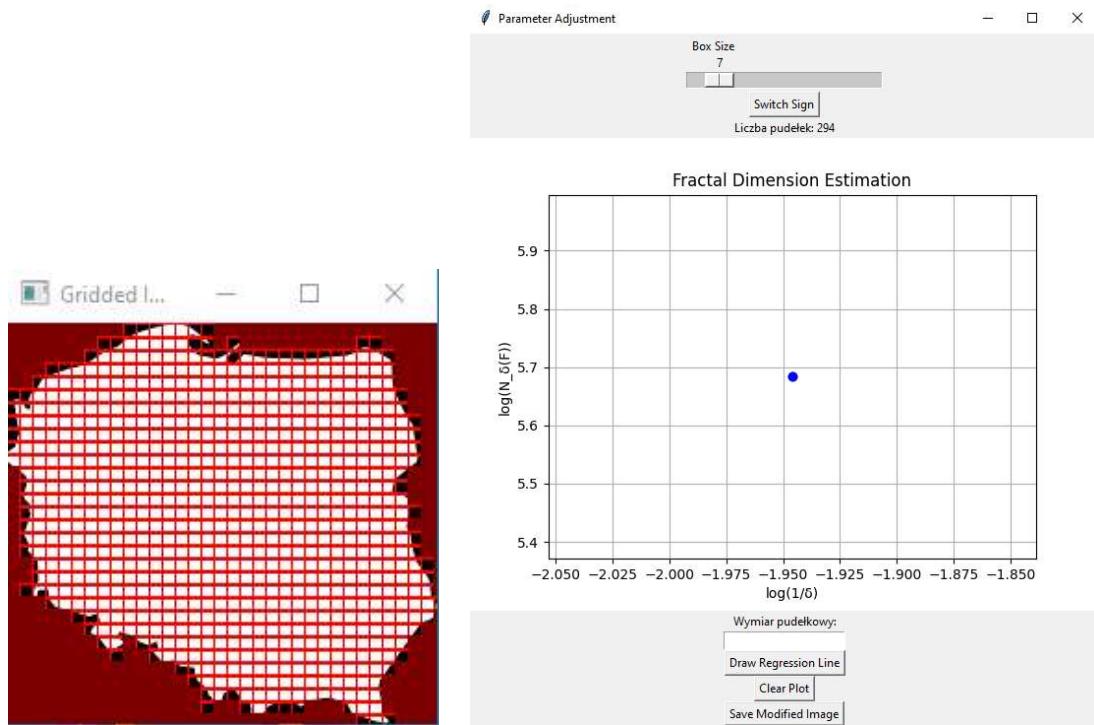
W rozdziale 4. opisane zostały poszczególne funkcjonalności kodu programu, natomiast poniżej przedstawione zostało praktyczne zastosowanie omówionych algorytmów. Do demonstracji wykorzystano umowny kontur mapy Polski.

Uruchamiając program następuje przekierowanie do okna umożliwiającego wybór obrazu. Umożliwia nam to wybór plików o rozszerzeniach PNG, JPG, JPEG oraz BMP (Rys. 13).

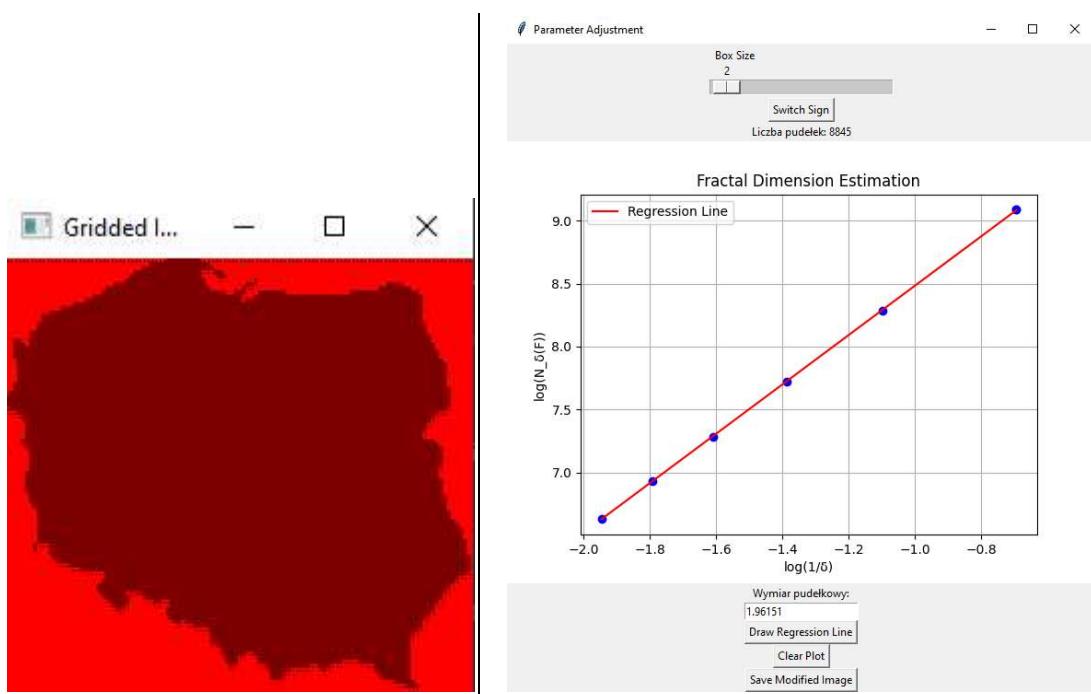


Rys. 13. Wybór obrazu.

Po wybraniu obrazu otwierane są okna wyświetlające obraz z nałożoną siatką kwadratów oraz okno interfejsu. W programie domyślnie rozmiar pudełek ustawiony jest na 7, a poziom kontrastu wyznaczany jest na podstawie metody Otsu służącej do automatycznego obliczania optymalnego progu na podstawie histogramu wartości pikseli w obrazie, oraz w taki sposób, aby zaznaczane i zliczane były obiekty o ciemnym kolorze (Rys. 14).



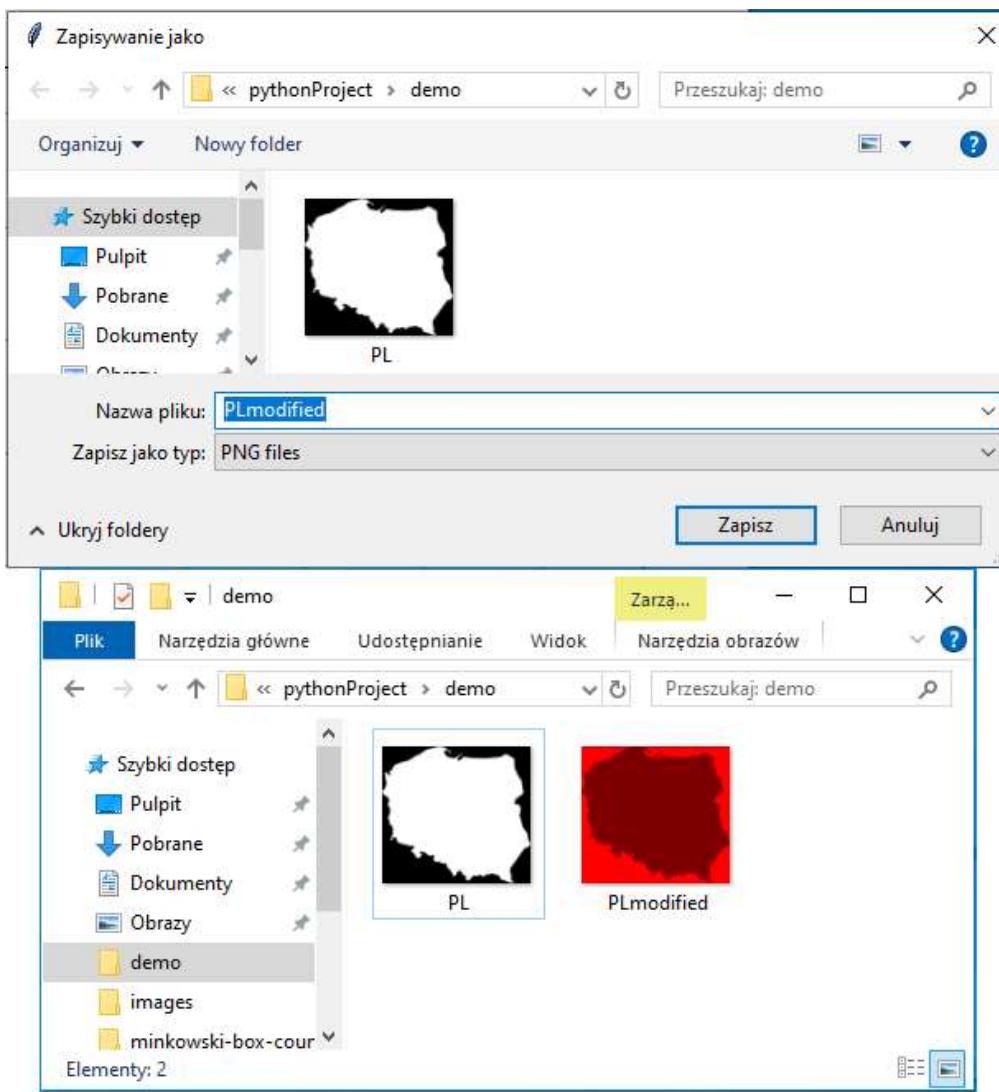
Rys. 14. Okna kolejno wyznaczanego obiektu oraz interfejsu.



Rys. 15. Okna interfejsu po zmianie parametrów.

Chcąc obliczyć wymiar pudełkowy mapy Polski, a nie jej otoczenia zmieniony został tryb zliczania tak, aby zaznaczał i obliczał elementy jaśniejsze. W tym celu należało użyć przycisku „Switch Sign”. Zmniejszony został także rozmiar pudełek z 7 na 2, co wygenerowało punkty na wykresie. Następnie poprzez użycie przycisku „Draw Regression Line” do punktów wykresu dopasowana została linia regresji. Jako iż wymiar pudełkowy określamy jako współczynnik kierunkowy dopasowanej na rysunku prostej, pod wykresem jest on także obliczany w momencie dopasowania linii regresji (Rys. 15).

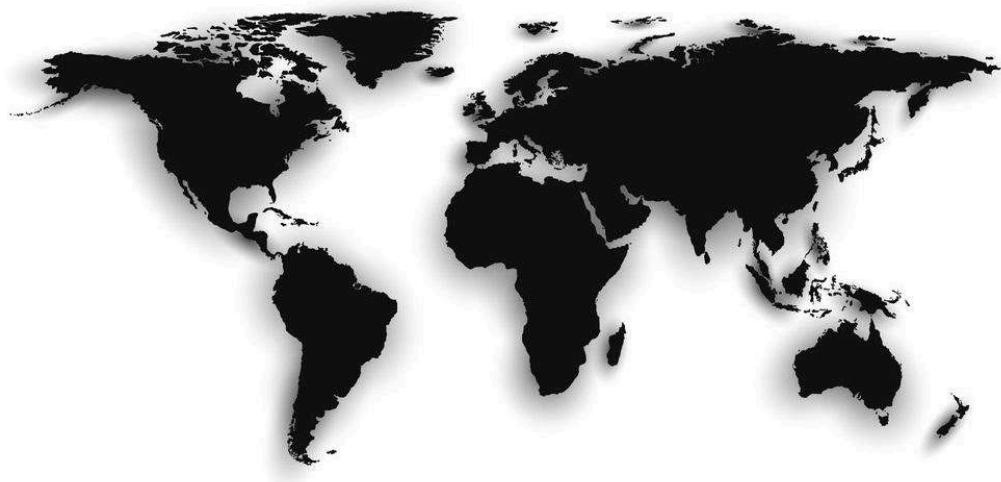
Na ekranie interfejsu jest także dostępny przycisk „Clear Plot” dający możliwość wyczyszczenia wykresu w dowolnym momencie, a także przycisk „Save Modified Image”, umożliwiający zapis zmodyfikowanego obrazu do wybranej destynacji (Rys. 16).



Rys. 16. Zapis zmodyfikowanego obrazu do wybranej destynacji.

5.2. Analiza obrazów

Będąc zaznajomionym z działaniem programu oraz jego funkcjonalnościami można przystąpić do analizy obrazów o różnych rozmiarach, poziomach kontrastu oraz skomplikowania. Na Rys. 17 przedstawiono czarno-białą mapę świata. Obraz jest jednak zdecydowanie większej rozdzielczości, a także zawiera o wiele bardziej nieregularne kształty.

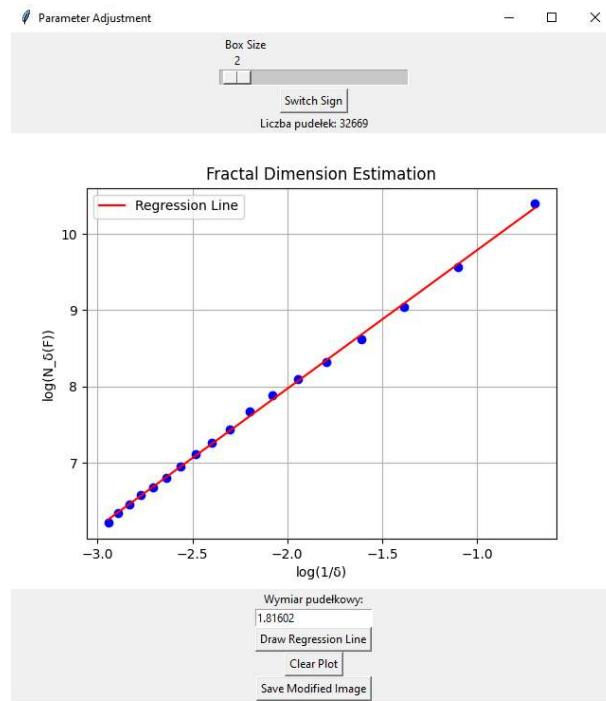


Rys. 17. Czarno-biała mapa świata.

Tak jak w przykładowym zastosowaniu następuje wczytanie pliku obrazu, tym samym wyświetlając okna interfejsu. Tym razem jednak będzie zwiększoną liczba punktów na wykresie poprzez wczytywanie siatki pudełek o rozmiarach kolejno od 2 do 20 w skali programu (Rys. 18 - Rys. 19).



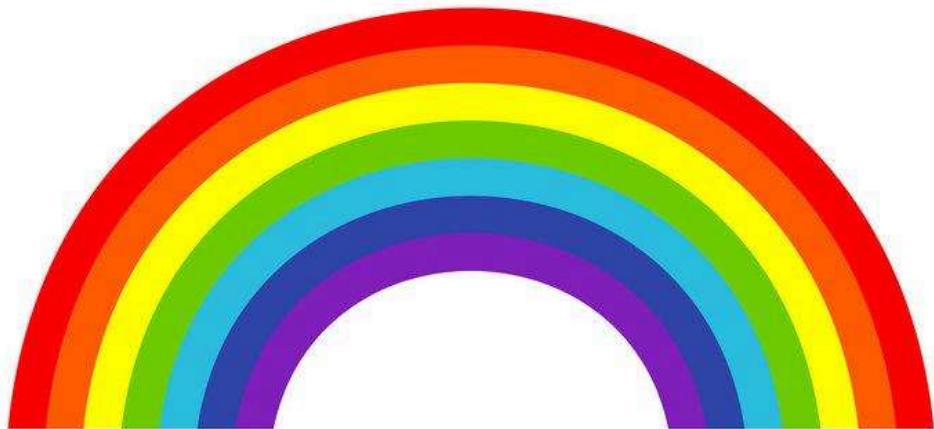
Rys. 18. Mapa świata z wyznaczonymi pudełkami dla kontynentów w skali 2.



Rys. 19. Interfejs do mapy świata z rysunku 18.

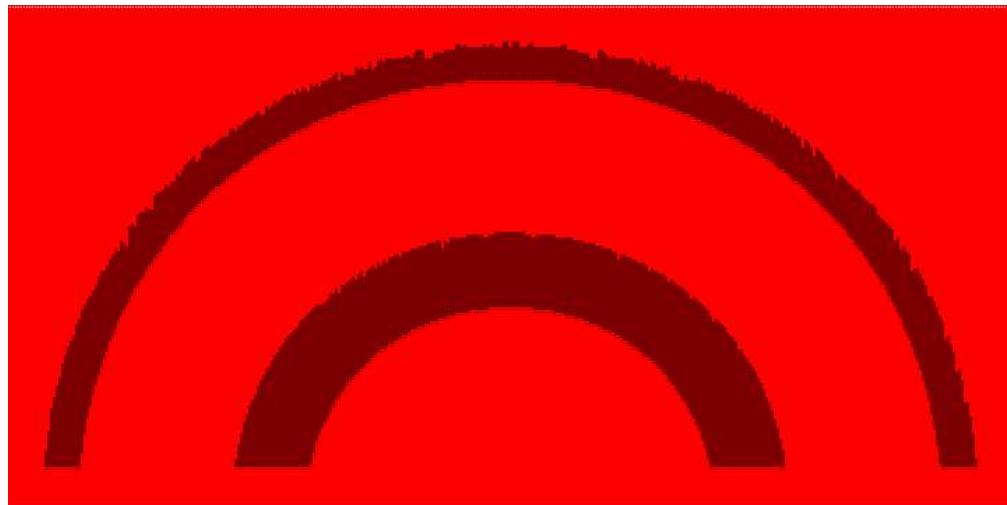
Jak prezentuje Rys. 18 siatka składa się z tak małych elementów, iż nie są one dostrzegalne gołym okiem. Porównując go jednak z Rys. 17 można stwierdzić że pokrycie kontynentów wykonane jest z dużą dokładnością, a punkty na podstawie których rysowana jest linia regresji rozmieszczone są bardzo regularnie. Wymiar pudełkowy wyniósł 1,81602, a liczba pudełek 32669.

Wiedząc już, że dla czarno-białych obrazów program działa bez zarzutów, warto sprawdzić, jaki jest próg dla warunku zaznaczania kwadratów siatki. Posłużyono się w tym przypadku klasycznym obrazem tęczy (Rys. 20).

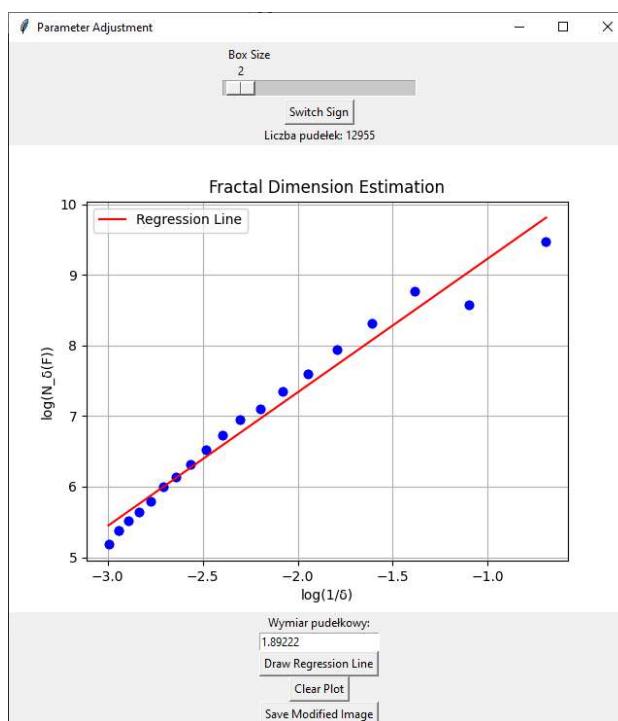


Rys. 20. Obraz tęczy do analizy kontrastu.

Jak można zauważyć na Rys. 21 program ustawiony na wykrywanie i zliczanie elementów ciemnych, jako zgodne z kryteriami z rysunku tęczy przyjmuje kolor fioletowy, ciemny odcień niebieskiego oraz kolor czerwony. Daje to jasny obraz w jakich granicach kolorów należy się poruszać, aby analiza obrazów przyniosła zamierzony efekt.



Rys. 21. Obraz tęczy z nałożoną siatką zliczającą ciemne elementy.



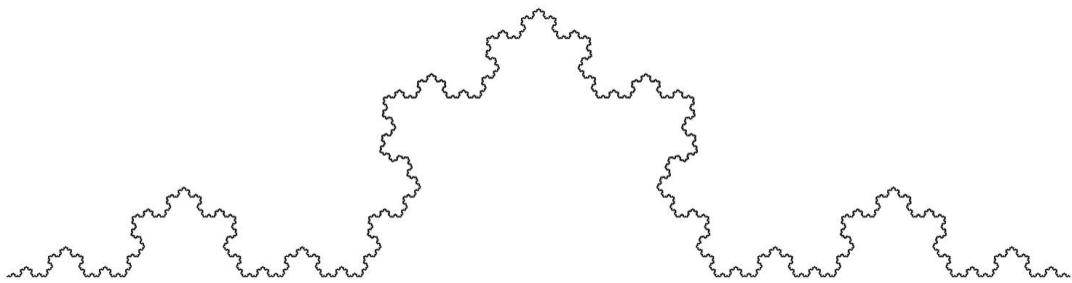
Rys. 22. Interfejs tęczy z rysunku 21.

Jak widać na Rys. 22 punkty wykresu wraz ze zmianą wielkości pudełek nie są już tak regularnie ułożone w jednej linii, jak to było w przypadku poprzednich wykresów. Oznacza to, że liczba zaznaczanych pudełek siatki zmienia się znacznie dla niektórych rozmiarów kwadratów. Ma

to z pewnością związek z różnymi kolorami na obrazie, przez co są one różnie traktowane przez funkcję weryfikującą kryterium akceptacji.

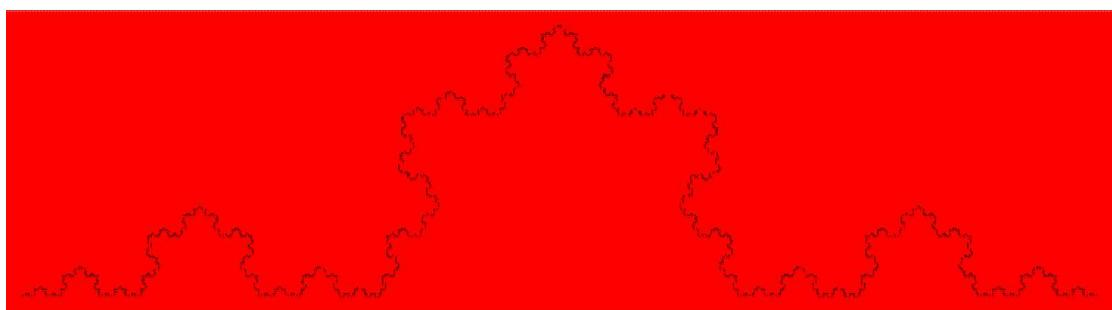
Mając świadomość tego, jak program radzi sobie z dużymi obiektami oraz wiedząc jak traktowane są poszczególne kolory przez funkcję zarządzającą kryterium kontrastu, sprawdzone zostanie jak program funkcjonuje, jeżeli chodzi o wyznaczanie cienkich obiektów. Aby to zweryfikować posłużyono się powszechnie znaną krzywą Kocha

(Rys. 23).



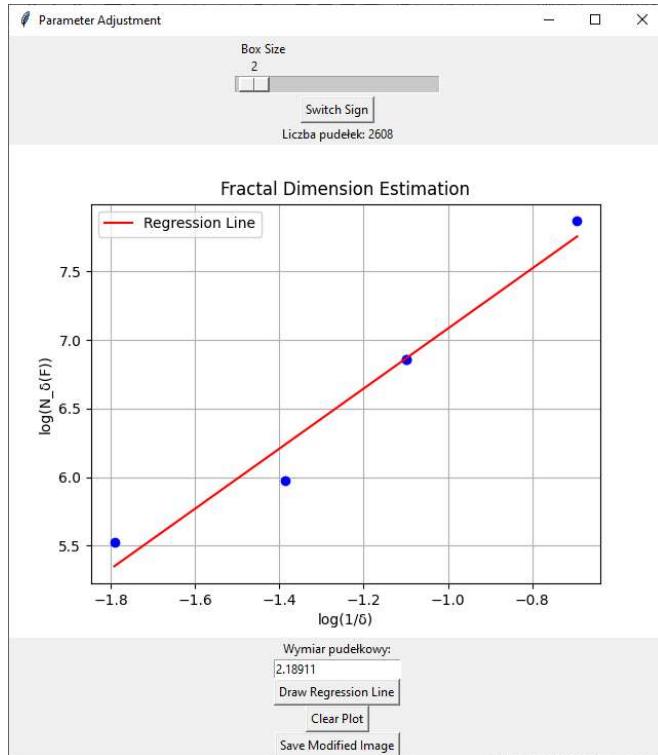
Rys. 23. Krzywa Kocha.

Wyznaczając wymiar pudełkowy takich obiektów jak krzywa Kocha, od początku operowano na mniejszym rozmiarze siatki, aby program wykrył linie wyznaczanego obiektu. Wydaje się, że wyznaczany obiekt został w większości pokryty tak jak należy, jednak w przybliżeniu widać, że ze względu na złożoność krzywej na obrazie wyjściowym występują pewne niedoskonałości, lecz kolejne zmniejszenie rozmiaru siatki nie jest już możliwe (Rys. 24).



Rys. 24. Krzywa Kocha z nałożoną siatką kwadratów.

Dodatkowo mniejsza liczba rozmiarów pudełek, na których operujemy spowodowała zmniejszoną liczbę punktów, do których dopasowana był linia regresji (Rys. 25). Wyznaczony wymiar pudełkowy różni się jednak znacznie od teoretycznego wymiaru krzywej Kocha. Jest to spowodowane tym, że na komputerze rozmiar pudełek możemy zmniejszać tylko do pewnego momentu, a wynik logarytmu jest podawany w zaokrągleniu.



Rys. 25. Interfejs programu dla krzywej Kocha z rysunku 5.12.

5.3. Wnioski

Program oferuje elastyczne narzędzie do badania różnorodnych obiektów. Jego interfejs jest intuicyjny, umożliwia wybór plików o różnych rozszerzeniach i pozwala na dostosowanie parametrów analizy. Domyślne ustawienia oraz funkcje automatycznego obliczania kontrastu sprawiają, że program jest łatwy w użyciu. Zmiana parametrów umożliwia bardziej precyzyjną analizę obiektów. Przykładowa analiza mapy świata, tęczy i krzywej Kocha pokazuje skuteczność programu w różnych warunkach. Może być używany do analizy zarówno prostych, jak i bardziej skomplikowanych obrazów, dostosowując się do różnych poziomów kontrastu i kształtów.

6. Podsumowanie

W pierwszej części pracy zwrócono uwagę na historię i rozwój koncepcji wymiaru pudełkowego, a także na praktyczne zastosowania tej metody. Pozwoliło to na zrozumienie ewolucji idei oraz wartości praktycznych wymiaru pudełkowego. Następnie przedstawiono teoretyczny opis tytularnej metody, obejmujący definicję, charakterystykę oraz opis matematyczny koncepcji. Podkreślone zostały kluczowe elementy teoretyczne, stanowiące podstawę do zrozumienia działania programu. Przechodząc do tworzenia narzędzia wyznaczania wymiaru pudełkowego dokładnie opisano ten proces, rozpoczynając od wyboru środowiska programistycznego, poprzez analizę wymagań, po implementację funkcjonalności. Uzasadniony został wybór środowiska programistycznego oraz kluczowe aspekty związane z projektowaniem funkcji przetwarzania obrazu oraz interfejsu użytkownika. Następnie prezentowane były praktyczne zastosowania programu, poprzez przeprowadzenie testów na różnego rodzaju plikach wejściowych. Wykazały one, że oprogramowanie może być używane do analizy różnorodnych obiektów, dostosowując się do różnych obiektów i kształtów. Praca nad projektem dała możliwość zgłębiania wiedzy na temat geometrii fraktalnej, a w szczególności wymiaru pudełkowego oraz jego praktycznych zastosowań. Jako osobisty wkład w wykonanie projektu autor uważa analizę oraz dobór odpowiedniej literatury związanej z omawianym tematem oraz wybór odpowiednich technologii i implementacje funkcjonalności programu dostosowanych do wymagań postawionych przez zagadnienie. Mimo licznych trudności związanych ze stworzeniem funkcji, która bez zarzutu będzie dokonywała analizy obrazów różnego poziomu skomplikowania, projekt został zakończony sukcesem.

Literatura

- [1] L. Jian, D. Qian, S. Caixin, „An improved box-counting method for image fractal dimension estimation”, *Pattern Recognition*, vol. 42, p. 2460-2469, November 2009.
- [2] D. Orłowski, Z. Sołtys, „Analiza fraktalna kształtu komórek”, *Postępy Biologii Komórki*, vol. 29, p. 423-433, 2002.
- [3] M. Bouda, J. s. Caplan, J. E. Saiers, „Box-Counting Dimension Revisited: Presenting an Efficient Method of Minimizing Quantization Error and an Assessment of the Self-Similarity of Structural Root Systems”, *Front Plant Sci.* 2016 Feb 18;7:149. doi: 10.3389/fpls.2016.00149. PMID: 26925073; PMCID: PMC4758026.
- [4] Xiaolong Li, Chen Cao, Xin Lin, „Improved Image Analysis Method to Evaluate Tracking Property under Successive Flashover Based on Fractal Theory”, *Energies* 2021.
- [5] W. Jiaxin, J. Xin, M. Shuo, T. Jinbo, „An effective method to compute the box-counting dimension based on the mathematical definition and intervals”, *Results in Engineering*, Vol. 6, 2020.
- [6] Z. Omietek, „Badanie samopodobieństwa obrazów metodą analizy fraktalnej” *Barometr Regionalny Nr 1(23)*, 2011.
- [7] P. Łabędź, „Algorytmy fraktalne w cyfrowej analizie obiektów przestrzennych”, Kraków 2016.
- [8] R. Buła, „Implikacje teorii rynku fraktalnego dla oceny ryzyka inwestycji finansowych”, p.27-28, 81-83, Warszawa 2019.
- [9] A. Zavalny, „Measuring the Length of Coastlines”, June 17, 2021.
- [10] W. Beyer, „Mandelbrot set”, dostęp 7.12.2023,
<https://prettymathpics.com/beyers-mandelbrot-zoom-sequence/>.
- [11] V. Pašić, „Fractal Dimension”, *Mathematics Seminar Project*, dostęp 8.12.2023,
<http://www.pmf.untz.ba/vedad/pdf/Fractal%20Dimension.pdf>.
- [12] J. Piera, V. Parisi-Baradad, E. Garcia-Ladona, A. Lombarte, L. Recasens, J. Cabestany, „Otolith shape feature extraction oriented to automatic classification with open distributed data”, *Marine and Freshwater Research* 56(5), January 2005.