POLITECHNIKA ŚLĄSKA W GLIWICACH
WYDZIAŁ INŻYNIERII BIOMEDYCZNEJ

Sprawozdanie

Biocybernetyka
Tytuł projektu: Symulator autonomicznego odkurzacza

Autor Pracy: Kinga Nowak, Justyna Herok, IiAM2

Zabrze, 23 stycznia 2020

1. Wstęp teoretyczny

Rozwój biocybernetyki oraz ogólniej - cybernetyki niesie za sobą wiele możliwości, zmian, ale też co za tym idzie – ułatwiania sobie pewnych zadań. Człowiek XXI wieku to człowiek, który chce mieć zawsze prostsze rozwiązanie. Dlatego rozwój wyżej wymienionych nauk, dodając do tego informatykę, automatykę i robotykę jest niezwykle ważny. Nasz projekt opiera się na podstawowej czynności porządkowej – odkurzaniu.
Z definicji odkurzacz to urządzenie elektryczne do odkurzania, wsysające powietrze wraz z kurzem. [1] Elektryczny odkurzacz został wynaleziony w 1901 roku i bez niego nikt sobie nie wyobraża życia. Nikt się nie spodziewał, że prawie wiek później firma iRobot zaprojektuje i doprowadzi do działania odkurzacz, który odkurza samodzielnie. Oprócz technicznych aspektów jest jeden najważniejszy – zaprogramowanie odkurzacza tak, aby dobrze samodzielnie pracował i odkurzał jak największą powierzchnię. Obecnie odkurzacze Roomba, czyli takie, które odkurzają same mają wiele funkcji i sposobów na dokładne sprzątanie. Są to czujniki, które wykrywają przeszkody, zanieczyszczenia. Może to być realizowane za pomocą różnych funkcji i algorytmów, które mają zapamiętany obraz i przykładowo gdy jest przeszkoda to odkurzacz wie jak się zachować, ponieważ został zaprogramowany na wypadek różnych przeszkód.

2. Założenia projektu

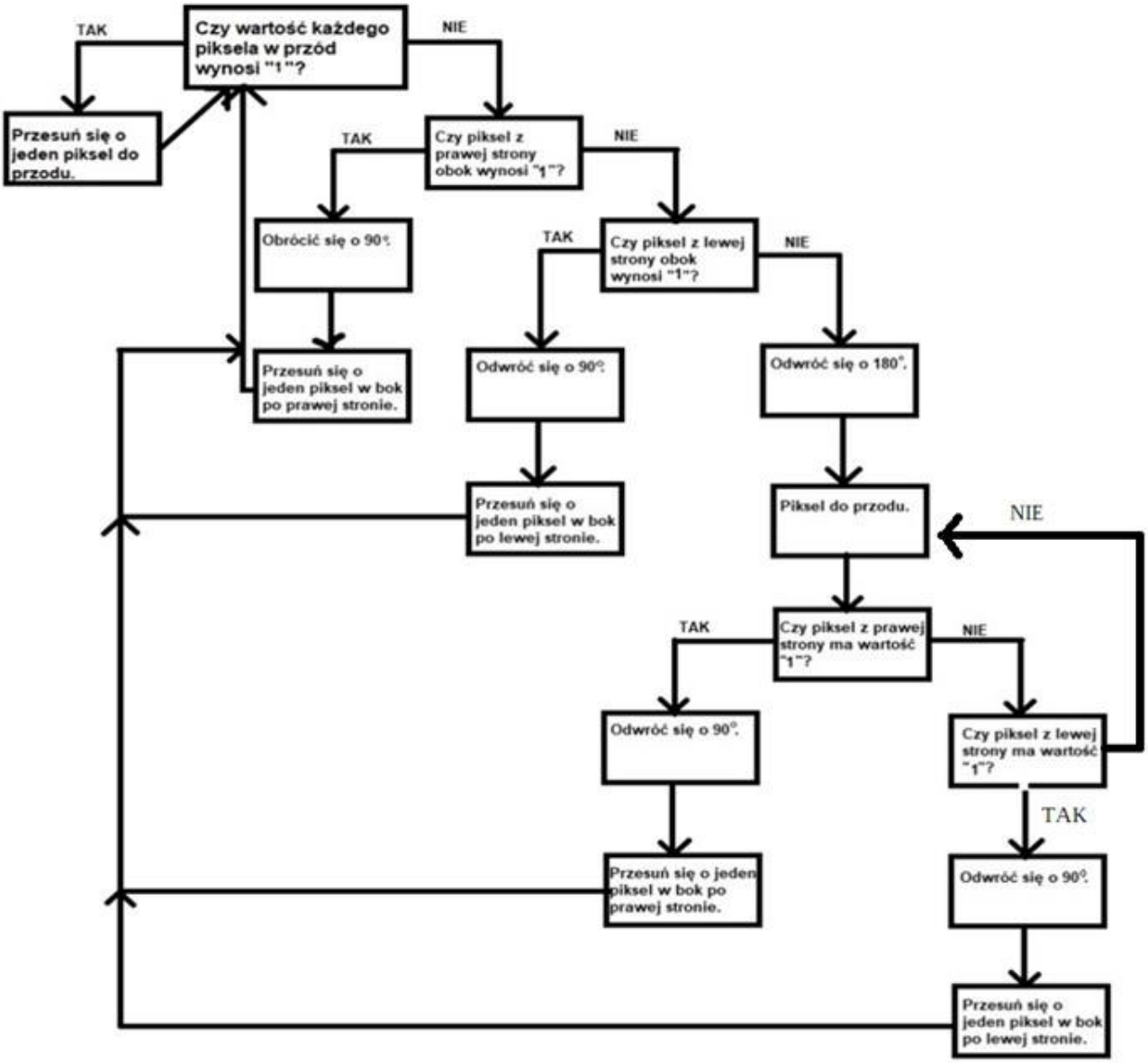
Celem projektu było opracowanie takiego działania, algorytmu, który pozwoli odkurzaczowi zapełnić (poodkurzać) całą, lub co najmniej większość powierzchni. Założono skuteczność odkurzania na poziom 90%, gdy osiągnie ten próg, odkurzacz może ukończyć działanie, lub iść dalej jeśli ma taką możliwość. Odkurzacz może być w polach, w których już był. Jeśli napotka przeszkodę, powinien iść w inną stronę. Podczas testowania programu można ocenić skuteczność na poziom 80-95%, co jest dobrym wynikiem jak na autorski algorytm.

3. Specyfikacja wewnętrzna

3.1 Opis algorytmu

Algorytm został opracowany autorsko, nie jest oparty na żadnym znanym algorytmie, lub nawet zaczerpniętym z takiego też algorytmu. Utworzony został przez przemyślenie pewnych kroków i iteracji. Polega on na tym, żeby uchwycić pewne elementy macierzy które są 1, czyli przestrzenią wolną do odkurzania. Natomiast przeszkody i ściany w programie to wartości 0. Wartości te biorą się z tego, że w programie Matlab, którym się posługujemy, użyto obrazków z funkcją rgb2gray, czyli w skali szarości i kolejno zmieniono tą macierz na typ logical, który w programie Matlab przyjmuje wartości 0 i 1 (biały i czarny).
Na początku zostaje stworzona macierz A. Kolejno zostaje stworzona macierz B, która jest kopią macierzy A, czyli mapy naszego pokoju. Na podstawie tych macierzy, algorytm sprawdza każdy piksel. W zależności od postawionych warunków "odkurzacz" wchodzi lub nie w ten piksel, czyli program rysuje dany krok "odkurzacza". Dzieje się to za pomocą elementów macierzy, gdzie krąży po wierszach i kolumnach i jeśli dalej dostrzeże pole białe to tam idzie, jeśli nie – może iść w inną stronę, gdy nie ma wyjścia musi się wrócić o kilka pól, lub wrócić się do początku. Kod, który wyraża algorytm składa się z wielu warunków dotyczących wierszy i kolumn, czy po prawej kolumna jest 0 czy 1, czy poniżej wiersz jest 0 czy 1. Sprawdza on też czy "odkurzacz" się nie zapętli i nie odkurza tego samego miejsca, czyli czy program nie rysuje w kółko tego samego schematu. Odpowiednio wtedy jest uformowane polecenie. Przykładowo: jeśli piksel z prawej strony wynosi 1, to odkurzacz ma tam iść, czyli obrócić się z pozycji "prosto" $\longrightarrow$ "w prawo" i zapełnić te pole. Użyte zostały również zmienne, które zapamiętują ostatnie położenie odkurzacza, aby odkurzacz nie ruszał ciągle z tego samego miejsca, ale z tego, w którym był za ostatnim razem. Dodatkowo użyto funkcji dzięki której możemy dowolnie ustawić miejsce rozpoczęcia pracy odkurzacza.

3.2 Schemat algorytmu



Rys. 3.1: Schemat algorytmu

3.3 Funkcje użyte w programie

Program działa w GUI – systemie, która ułatwia pracę i ją obrazuje za pomocą funkcji, które współdziałają ze sobą.

3.3.1 Funkcje typowe dla GUI:

hObject - identyfikator do kontrolki
Definicja użycia w matlabie:

```
function countClicks_OpeningFcn(hObject, eventdata,
handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
```

handles - zbiór uchwytów do wszystkich kontrolek – pusty do momentu wywołania funkcji w CreateFcn
Definicja użycia w matlabie:

```
function countClicks_OpeningFcn(hObject, eventdata,
 handles, varargin)
```

CallBack - funkcja jest wywoływana gdy na kontrolce została wykonana właściwa akcja
Definicja użycia w matlabie:

```
function countClicks_Callback(hObject, eventdata, handles)
```

Przykłady z programu:

```
function KingaNowak_JustynaHerok_OpeningFcn(hObject, eventdata,
handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function wybormapy_Callback(hObject, eventdata, handles)
```

3.3.2 Funkcje użyte w programie w celu jego prawidłowego działania

UIGETFILE – wyświetla standardowe okno otwierania plików, wybór obrazu
Przyjmuje obrazy w formacie: bmp, tif, tiff, jpg, jpeg, pcx, xwd, hdf, gif, png, cur, ico i zwraca okienko do wyboru obrazu.
Definicja użycia w matlabie:

```
[file,title] = uigetfile
gdzie 'file' to typ pliku a 'title' to napis, nagłówek
```

Przykład:

```
mapa=uigetfile('*.jpg','Wybierz obraz:');
```

IMREAD – wczytuje obraz
Definicja użycia w matlabie:

```
A = imread(filename)
gdzie 'A' to wczytany obraz, 'filename' to nazwa pliku
```

Przykład:

```
A=imread(mapa);
```

IMRESIZE – skalowanie obrazu mapy
Definicja użycia w matlabie:

```
B = imresize(A,scale)
gdzie 'B' to obraz wynikowy, 'A' obraz do przeskalowania,
'scale' to dana skala
```

Przykład:

```
A=imresize(A,0.15);
```

DOUBLE – zamiana obrazu na monochromatyczny lub RGB
Definicja użycia w matlabie:

```
Y = double(X)
gdzie 'Y' to obraz wynikowy, 'X' to obraz, który ma ulec zmianie
```

Przykład:

```
A=double(A)/255;
```

RGB2GRAY – zamiana obrazu na monochromatyczny
Definicja użycia w matlabie:

```
I = rgb2gray(RGB)
gdzie 'I' to obraz wynikowy, 'RGB' to obraz do zmiany
```

Przykład:

```
A=rgb2gray(A);
```

LOGICAL – zamiana elementów macierzy/obrazu na wartości logiczne – w tym przypadku na: 0 – czarny, 1 – biały
Definicja użycia w matlabie:

```
L = logical(A)
gdzie 'L' to elementy wynikowe, 'A' to elementy do zmiany
```

Przykład:

```
A=logical(A);
```

AXES – wybór odpowiedniej osi do rysowania
Definicja użycia w matlabie:

```
axes(Name, Value)
gdzie 'Name' to nazwa, 'Value' to wartości
```

Przykład:

```
axes(handles.os1);
```

INT16 – zamiana na system szesnastkowy
Definicja użycia w matlabie:

```
y = int16(X);
gdzie 'Y' to element wynikowy, 'X' to parametr
do zmiany na system szesnastkowy
```

Przykład:

```
w=int16(w);
k=int16(k);
```

NUM2STR - zamienia liczbę na ciąg znaków, w tym przypadku uzyskaną skuteczność odkurzania w procentach na napis
Definicja użycia w matlabie:

```
s = num2str(A, precision)
gdzie 's' to zamieniona liczba, 'a' to liczba zamieniana,
'precision' to precyzja z jaką ma być operacja wykonana
```

Przykład:

```
title(['Skuteczność odkurzania odkurzacza: ', num2str(zmienna)]);
```

IMSHOW – wyświetla obraz
Definicja użycia w matlabie:

```
imshow(I)
gdzie 'I' to obraz, który ma być wyświetlony
```

Przykład:

```
imshow(A)
```

XLIM, YLIM – rysowanie wykresu zależnego od dwóch wektorów na osiach 'x' i 'y'
Definicja użycia w matlabie:

```
___ = xlim(target, ___)
gdzie 'target' to odpowiednie parametry osi
```

Przykład:

```
ylim(handles.os2,[0,caloscp]);
xlim(handles.os2,[0,((j*i)-zajete)]);
```

SIZE – zwraca rozmiar macierzy
Definicja użycia w matlabie:

```
[sz1,...,szN] = size(___)
gdzie 'sz1', 'szN' to kolumny, wiersze, których rozmiar ma być zwrócony
```

Przykład:

```
[i,j]=size(A);
```

GINPUT – pozwala odczytać współrzędne punktów zaznaczonych na wykresie kursorem
Definicja użycia w matlabie:

```
[x,y] = ginput(n)
gdzie 'x' i 'y' to parametry punktu,
w który się kliknie,'n' to ilość kliknięć
```

Przykład:

```
[k,w]=ginput(1);
```

HOLD – pozwala narysować wykres w następnym okienku, bez usuwania poprzedniego
Definicja użycia w matlabie:

```
hold on
hold off
```

Przykład:

```
hold on
```

PLOT – rysuje wykres
Definicja użycia w matlabie:

```
plot(X,Y,LineSpec)
gdzie 'X', 'Y' to kolumny i wiersze, 'LineSpec' to typ linii
```

Przykład:

```
plot(k,w,'*g');
```

LINE – rysuje linię łamaną łączące dane punkty
Definicja użycia w matlabie:

```
line(x,y)
gdzie 'x', 'y' to wektory
```

Przykład:

```
line([zmienna_x zmienna_x],[zmienna_y zmienna_y-1], 'Color', 'red')
```

TITLE – dodaje tytuł do rysunku
Definicja użycia w matlabie:

```
title(txt)
gdzie 'txt' to tytuł, napis tekstowy
```

Przykład:

```
title('Trwa odkurzanie!');
title('Sprzątanie skończone!');
```

PAUSE – zatrzymuje funkcję
Definicja użycia w matlabie:

```
pause(n)
gdzie 'n' to czas pauzy
```

Przykład:

```
pause(0.1)
```

3.3.3 Pętle i ważne funkcje: FOR, WHILE, IF, END, ELSEIF

Potrzebne do powtarzania poleceń, stawiania warunków, kończenia warunków.
Definicja użycia w matlabie:

```
for index = values
   statements
end

while expression
    statements
end

if expression
    statements
elseif expression
    statements
else
    statements
end
```

Przykłady użycia:

FOR
Przykłady:

```
for x=1:j
    for y=1:i
        if A(y,x)==1
            caloscp=caloscp+1;
```

Definicja funkcji 'for':

```
for index = values
   statements
end
```

IF, ELSE, ELSEIF
Przykłady:

```
if B(zmienna_y,zmienna_x-1)==1
                    posprzatanep=posprzatanep+1;
                    end

elseif A(zmienna_y, zmienna_x+1)==1
```

Definicja funkcji 'if', 'eleif':

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

WHILE
Przykład:

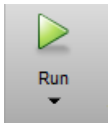
```
while A(zmienna_y, zmienna_x-1)==1
```

Definicja funkcji 'while':

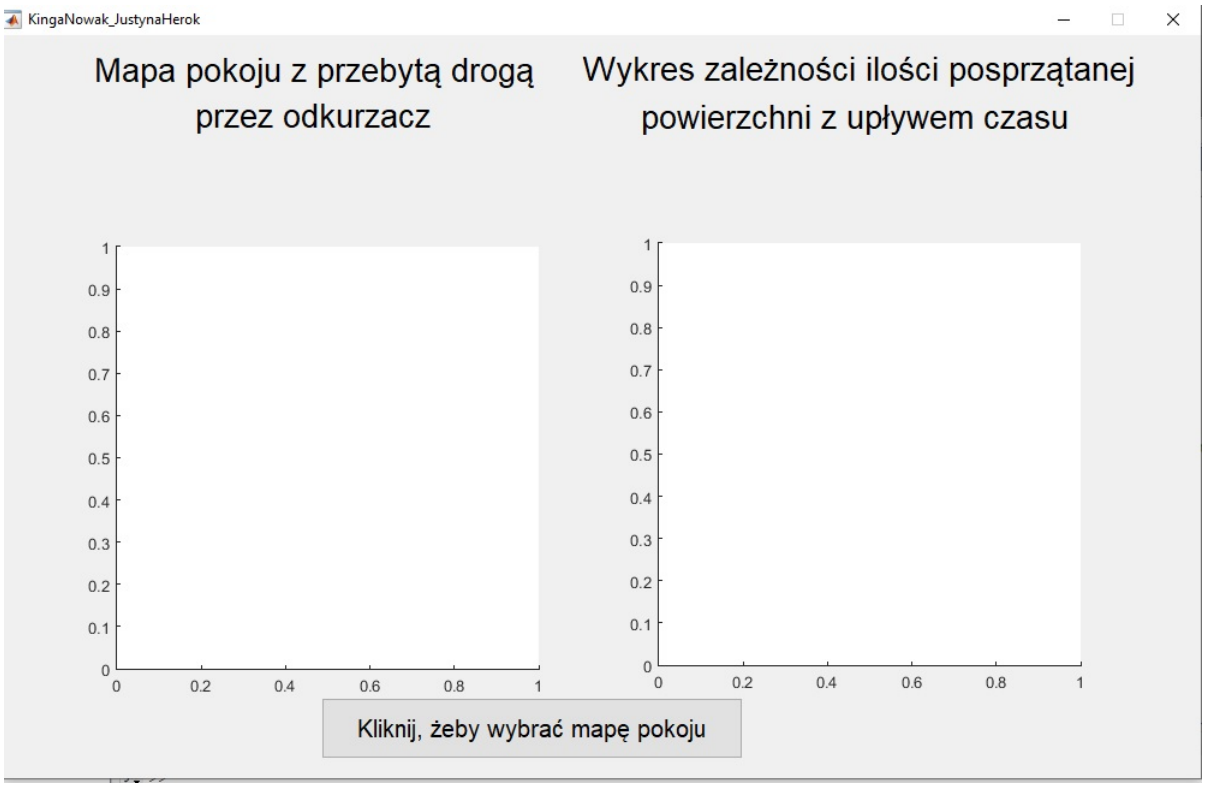
```
while expression
    statements
end
```

4. Specyfikacja zewnętrzna

Wchodząc w program należy kliknąć ikonkę 'Run'.



Wtedy otwiera się okno aplikacji GUI,
Następnie należy kliknąć na przycisk ⟶ 'Wybierz mapę pokoju'

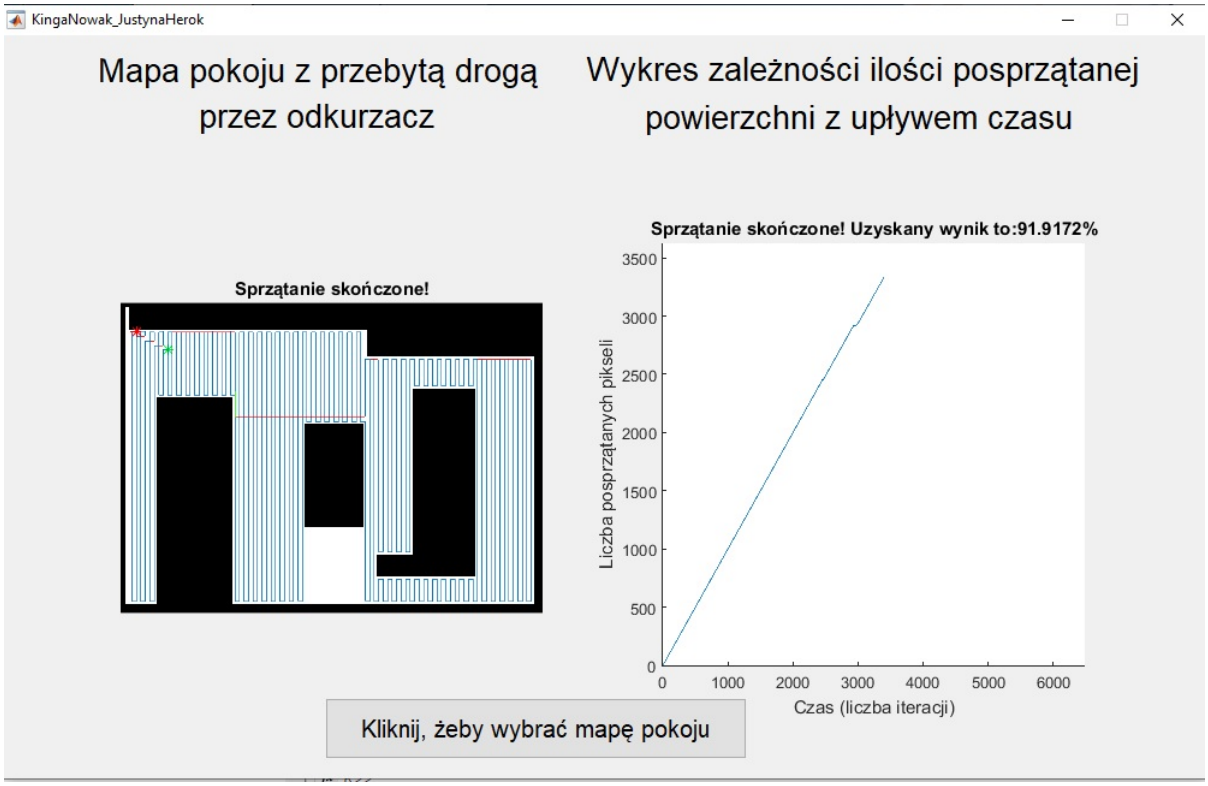


Rys. 4.1: Okno aplikacji GUI

Wtedy należy wybrać mapę pokoju i kliknąć w miejsce, w którym odkurzacz ma rozpocząć pracę.

Rys. 4.2: Okno aplikacji podczas odkurzania

Kiedy będzie odkurzał widniał będzie napis nad obrazkiem z mapą pokoju 'Trwa odkurzanie!'



Rys. 4.3: Okno aplikacji po skończeniu odkurzania

Kiedy skończy działanie będzie napis 'Sprzątanie skończone'.
Będzie można również sprawdzić ilość poodkurzanych pikseli oraz procentowo – skuteczność sprzątania.

5. Komercyjne rozwiązania alternatywne

Obecnie w ofertę swojej firmy odkurzacz Roomba wprowadza wiele marek. Najpopularniejsza z pewnością jest firma iRobot. Cena waha się od 900zł do aż 6500zł. [2] Kolejna marka to Electrolux (2000zł - 2800zł). Wśród bardziej znanych marek znajdziemy również Samsung (1400zł – 3000zł). Mniej znane marki to: Tesla, Robojet, ILife, Ecovans, Zaco, Sensor i wiele innych. Ceny wahają się we wszystkich markach od 280zł do 6500zł. Każda marka proponuje coś nowego, może lepszego, o innym wyglądzie, odmiennym działaniu. Który lepszy, bardziej efektywny? Trzeba by było wypróbować każdy na różnych rodzajach powierzchni, na mniejszych czy większych obszarach. Skoro ceny są różne i różne są firmy produkujące takie odkurzacze to czym one się różnią? Przede wszystkim różnią się:
- ceną,
- wyglądem,
- czasem pracy – wytrzymałością bez ładowania (30-240 minut),
- czasem ładowania (100-250 minut),
- czujnikami – krawędzi, zbliżeniowymi, przeszkód, antykolizyjne, drgań,
- automatycznym powrotem do bazy (czy występuje czy nie),
- programatorem pracy,
- pojemnością zbiornika na kurz.

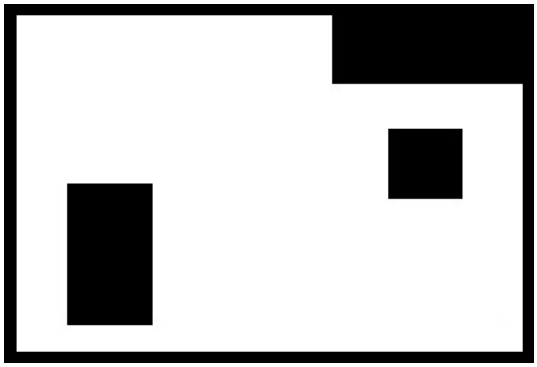
Na wiele aspektów więc można patrzeć kupując dany odkurzacz. Istnieją również takie, które oprócz tego, że czyszczą z kurzu, okruszków, dodatkowo myją powierzchnię. Jest to również dobre rozwiązanie dla paneli, kafelek.

6. Porównanie odkurzaczy

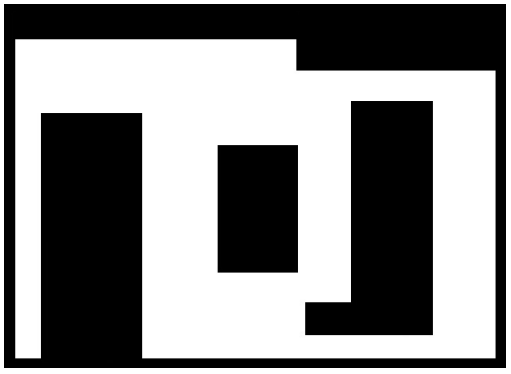
Stworzony algorytm, który pozwala interaktywnemu odkurzaczowi działać pozwala na skuteczność 92-95%. Skuteczność jest mierzona pikselami poodkurzanymi na wszystkie do poodkurzania i przekształceniem na procenty. Odkurzacze jakie można znaleźć w sklepach oprócz algorytmu mają dużą ilość czujników różnego rodzaju, które zastępują zapamiętywanie powierzchni jaką odkurzał, jak w przypadku algorytmu przez nas stworzonego. Czujniki wykrywają miejsca niepoposprzątane, gdzie widnieje kurz czy drobinki, które pozwoli wychwycić maszyna. Łączą się one z kamerą, razem są zaprogramowane tak, aby odróżnić elementy domu, wystroju od brudu, który ma posprzątać.
Nowością jest technologia Imprint Smart Mapping, w której odkurzacz tworzy wirtualne mapy pomieszczeń i przechowuje je w pamięci, dzięki temu podobnie jak w naszym programie – może zapamiętać miejsca w których już był, co znacznie poprawia efektywność działania.

7. Mapy użyte w testach programu

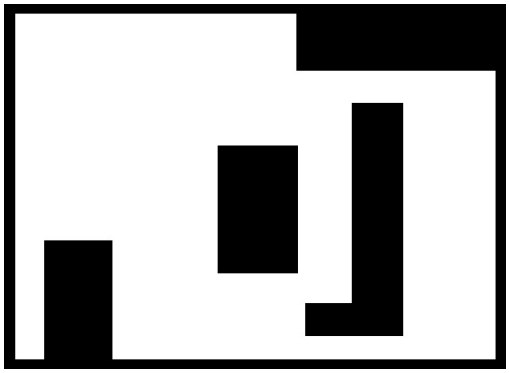
Poniżej są przedstawione mapy użyte w testach programu.
Mapa pierwsza jest zawarta w testach na rysunku 8.1 i 8.2, mapa druga - 8.3, mapa trzecia - 8.4, 8.5, 8.6.



Rys. 7.1: Mapa nr 1



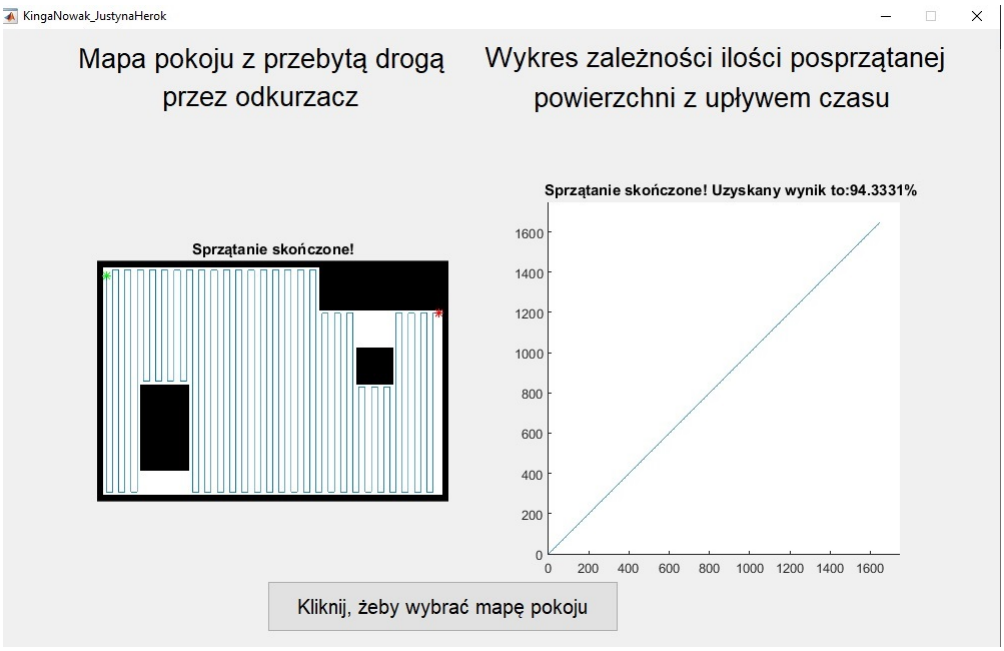
Rys. 7.2: Mapa nr 2



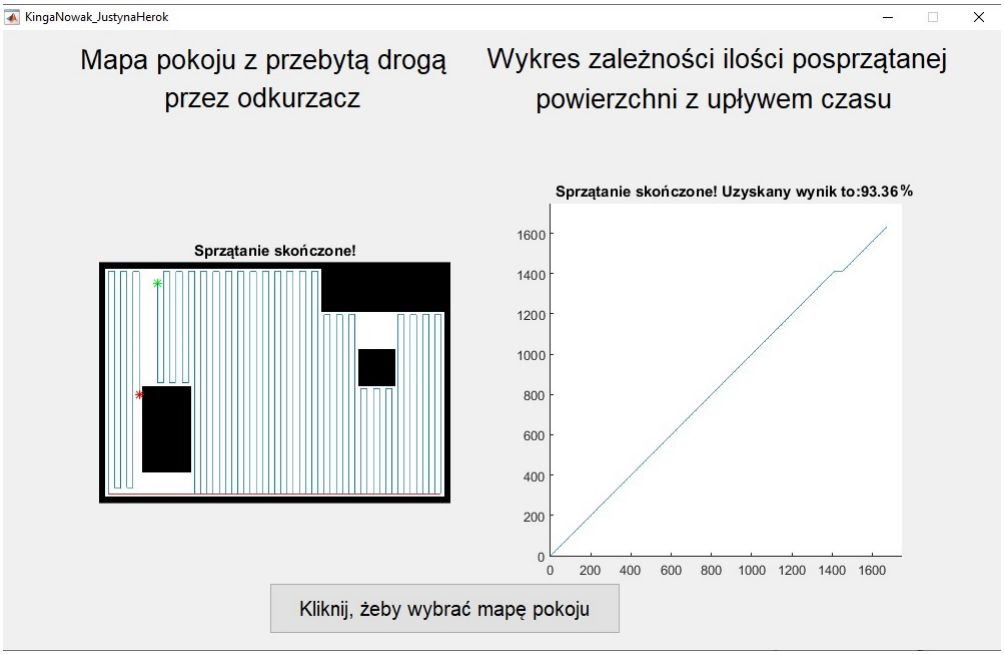
Rys. 7.3: Mapa nr 3

8. Wyniki testowania

Wyniki testów mapy nr 1

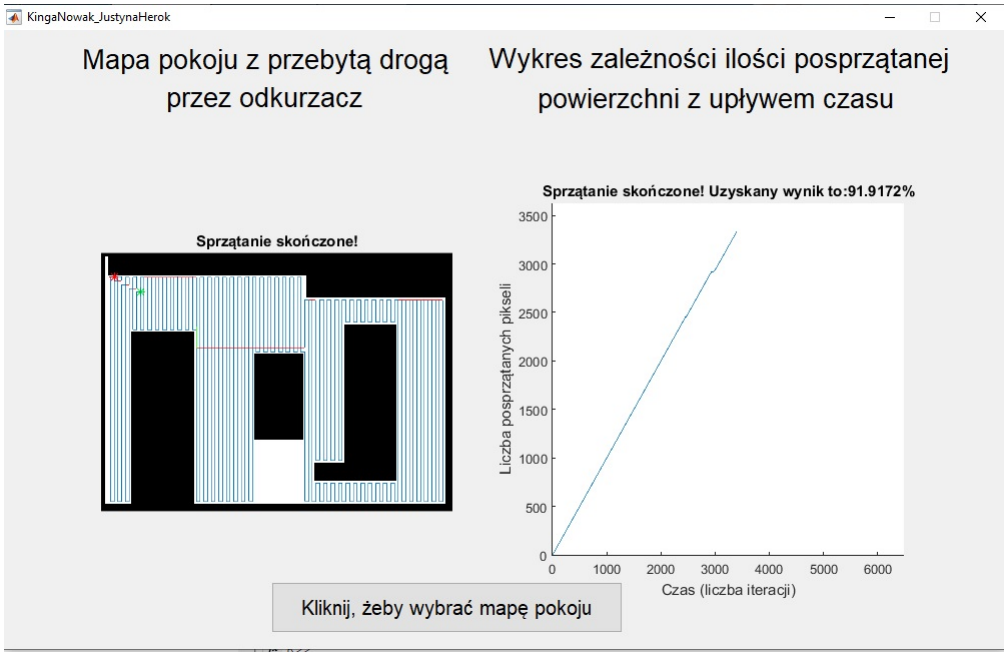


Rys. 8.1: Test mapy nr 1 - skuteczność 94,33%



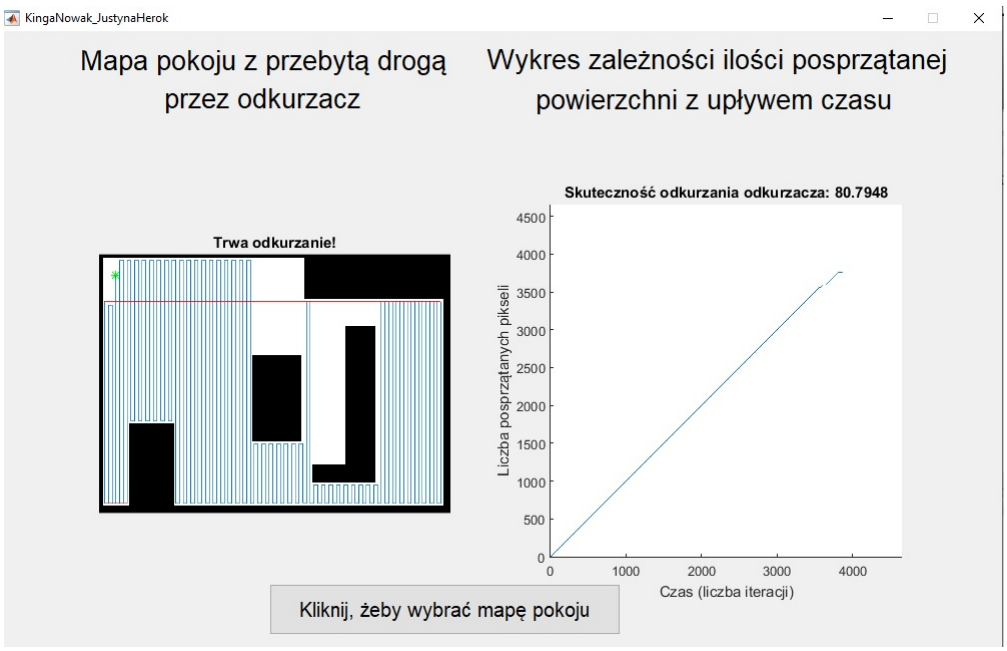
Rys. 8.2: Test mapy nr 1 - skuteczność 93,36%

Wyniki testów mapy nr 2

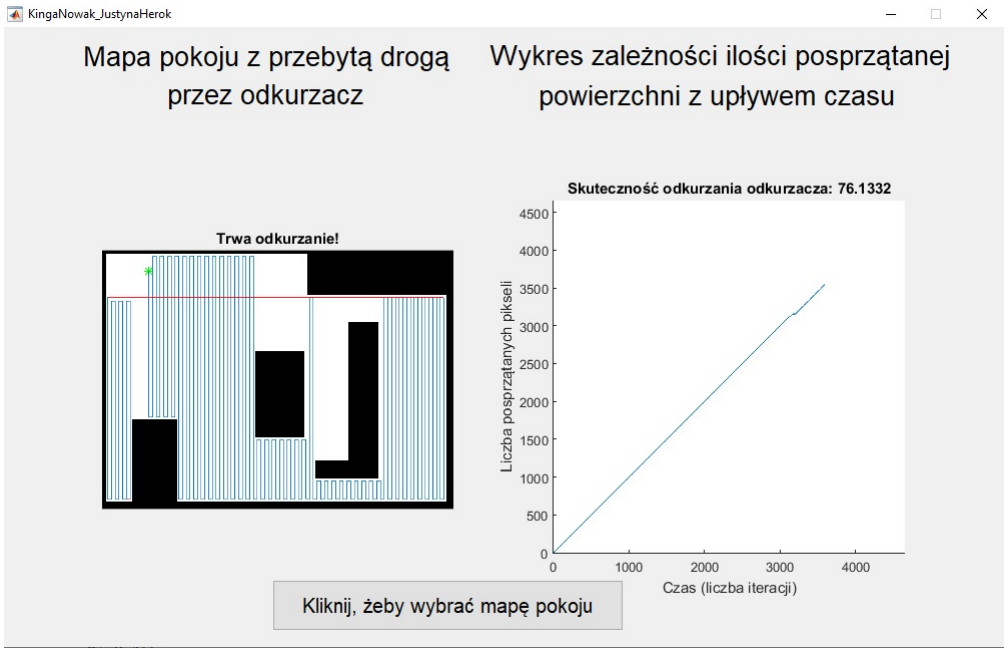


Rys. 8.3: Test mapy nr 2 - skuteczność 91,92%

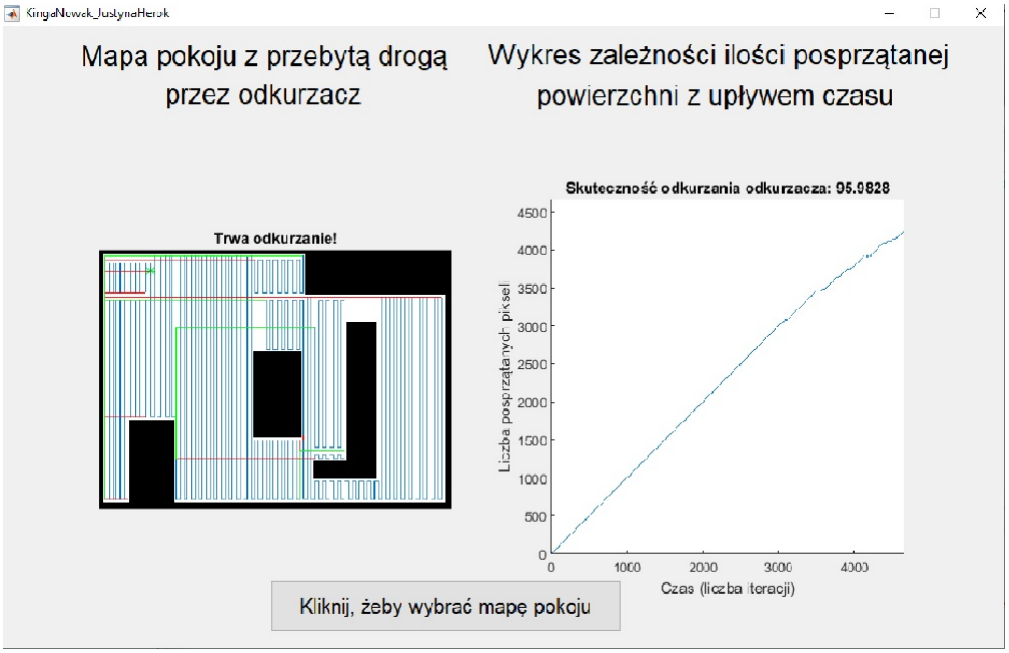
Wyniki testów mapy nr 3



Rys. 8.4: Test mapy nr 3 - skuteczność 80,79%



Rys. 8.5: Test mapy nr 3 - skuteczność 76,13%



Rys. 8.6: Test mapy nr 3 - skuteczność 95,98%

9. Podsumowanie

Wyniki działania programu są zadowalające, zgodnie z oczekiwaniami mogą osiągać nawet 95%.
Przy testowaniu mapy nr 1 uzyskane wyniki to 94,3% (Rys. 8.1) oraz 93,4% (Rys. 8.2), są to dobre wyniki, przestrzeń niepoodkurzana jest niewielka i jest ona w jednym lub w dwóch kawałkach, z czego w obu testach jest to te same miejsce, co świadczy o tym, że algorytm idąc po kolejnych iteracjach pomija przejście do konkretnych kolumn i wierszy, co związane jest z warunkami w algorytmie.
Przy testowaniu mapy nr 2 uzyskany wynik to 91,9% (Rys. 8.3). Obszar niepoodkurzany jest pod elementem, gdzie jest mało miejsca, wynika to z postawionych warunków i z tego, że program po osiągnięciu danej skuteczności nie musi już iść dalej, jeśli z każdej strony miejsce jest zapełnione.
Wyniki uzyskane w testach mapy nr 3 to: 80,8% (Rys. 8.4), 76,1% (Rys. 8.5) i 96% (Rys. 8.6). Przy testowaniu mapy nr 3 zauważono, że program się zawiesza, może to być spowodowane wysokim użyciem procesora, gdyż mapa posiada dużo przestrzeni do odkurzania. Podczas testowania pozostałych mapy nr 1 i nr 2 nie zauważono tego typu problemów w działaniu programu.
Wybór punktu startowego we wszystkich trzech mapach ma wpływ na skuteczność odkurzania, dlatego, że program wykonuje algorytm, ale w innej kolejności.

10. Problemy

Tworząc program można było się spotkać z wieloma problemami. Głównie dotyczyły one samego pomysłu na algorytm, później jego prawidłowego działania, aby odkurzacz nie zatrzymywał się w momencie, gdy dochodzi on do ściany (czarnego piksela). Reszta funkcji nie stanowiła większego problemu.
Jednym z problemów było stworzenie odpowiedniego wykresu, który będzie pokazywał ilość zapełnionej powierzchni. Po testowaniu programu pojawiły się problemy z działaniem i przejściem odkurzacza po elementach macierzy. Program zapętlał się i nie działał zgodnie z zamierzeniami, nie przechodził do pustych pól, ale chodził po zapełnionych i nie zatrzymywał się.
Wszystkie te problemy były jednak w dużym stopniu lub całkowicie do rozwiązania.

Bibliografia

[1] Słownik języka polskiego

[2] Sklep RTV EURO AGD