POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Automatyka i Robotyka
SPECJALNOŚĆ: Systemy informatyczne w automatyce

# PROJEKT INŻYNIERSKI

Zastosowanie transformat falkowych
w zadaniach skalowania seam carving

Seam carving scalling based
on wavelet transform

AUTOR:
Karol Ankutowicz

PROWADZĄCY PRACĘ:
dr hab. inż. Przemysław Śliwiński,
I-6

OCENA PRACY:

WROCŁAW 2015

# Spis treści

# 1. Wstęp

## *1.1. Geneza projektu*

Współczesna technologia zarówno pod względem sprzętowym jak i programowym idzie zdecydowanie ku ciągłemu rozwojowi. Współczesne komputery jak i urządzenia mobilne są coraz mocniejsze a tworzone oprogramowanie umożliwia wykorzystanie tych możliwości. Wśród wielu współczesnych aplikacji poczesne miejsce zajmują te, które pozwalają nam na obróbkę grafiki – od wykonywania zdjęć poszczególnymi urządzeniami, przez ich przesyłanie aż do zaawansowanego przetwarzania. Niemal każdy korzysta z tego w różny sposób i w różnych celach.

Problemem z jakim często się spotykamy mówiąc o oprogramowaniu *przenośność* – problem ten oznacza konieczność produkowana innej wersji danego programu na dane urządzenie. Możemy się też na to napotkać przy okazji reprezentacji obrazów. Każdy sprzęt na którym możemy wyświetlić grafikę ma swoją określoną specyfikację tj. rozdzielczość urządzenia, proporcje obrazu itd. W konsekwencji może się zdarzyć, że potrzebujemy np. zmienić rozmiary zdjęcia tak, że zajdzie potrzeba zmiany jego proporcji. Niewiadomą wówczas pozostaje jakość zdjęcia, jakie otrzymamy. Wobec tego rozpoczęto prace nad takimi algorytmami, które pozwalają na obróbkę zdjęć biorąc pod uwagę jego zawartość. Wówczas do zadań takiego algorytmu należałoby np. zmiana tylko jednego z wymiarów przy jak najmniejszych zniekształceniach obiektów na obrazie. Takie wybiórcze skalowanie wydaje się być trudne, jednak w końcu prace w tym zakresie spowodowały stworzenie algorytmu, który tę funkcję realizuje. Algorytm ten nazywa się *seam carving* i jednym ze sposobów jego implementacji jest poświęcona ta praca.

Algorytm seam carving zaproponowany przez Shaia Avidana i Ariela Shamira jest nowatorski i prezentuje podejście do kwestii skalowania zdjęć inne niż szerzej znane metody. Metoda ta zdecydowanie przewyższa standardowe skalowanie, które wnosi zniekształcenia zawsze gdy wymiary obrazu skalujemy o inną wartość. Przewyższa również kadrowanie, przez które możemy utracić część istotnych informacji. Algorytm ten skupia się na zawartości obrazu a głównie nad sąsiedztwem danego piksela. W ten sposób określa, które według niego obszary na obrazie są bardziej istotne a które mniej. Dzięki temu końcowy efekt może zaskoczyć niejednego wymagającego użytkownika. Sposób działania algorytmu skłania do jasnego i przejrzystego wyznaczenia celu projektu.

### *1.2. Cel projektu*

Celem projektu jest zaimplementowanie programu realizującego algorytm skalowania *seam carving* w oparciu o funkcję falkową. Do osiągnięcia celu potrzebne jest zapoznanie się z teorią zastosowaniu falek jak i działaniem samego algorytmu *seam carving*. Kolejne rozdziały pracy pozwalają na przejście od teorii do praktyki w kwestii implementacji algorytmu oraz jego testowania.

W rozdziale pt. *Sygnały* zostały opisane podstawowe zagadnienia związane z definiowaniem sygnałów oraz ich odbioru. Rozdział ten zdecydowania stawia na przybliżanie właściwości obrazów, które traktujemy jako sygnały dwuwymiarowe. Pod samym słowem *obraz* aktualnie kryje się całe mnóstwo definicji i właściwości, że nie sposób się zająć wszystkimi. Również i w tej pracy została omówiona i zanalizowana zaledwie niewielka cząstka całości wiedzy o obrazach jaką w tej chwili posiada ludzkość.

Kolejny rozdział pt. *Falki* przybliża nam definicję, właściwości oraz zastosowanie tych funkcji. Samo słowo *falka* może nam wskazywać, że jest to zdrobnienia od słowa fala. Całkiem słusznie, ponieważ falki są funkcjami, które możemy przedstawić jako mały wycinek fali. Ogólnie ta dziedzina nauki jest młoda i dynamicznie się rozwija znajdując zastosowanie w przetwarzaniu obrazów czy dźwięków. My przyjrzymy się bardziej szczegółowo najstarszej spośród nich, mianowicie falce Haara. Sama definicja funkcji ma co prawda już ponad 100 lat lecz nadal jest bardzo użyteczna mimo zdefiniowania w późniejszym czasie wielu innych tego typy funkcji.

W rozdziale pt. *Skalowanie obrazów* przyjrzymy się dokładniej operacji zmiany rozmiarów obrazu. Okazuje się, że realizację tego przekształcenia można przedstawić na kilka sposobów. Po krótkim wprowadzeniu przez standardowe skalowanie skupimy się nad esencją tej pracy – algorytmem *seam carving*. Jest to nowoczesny algorytm skalowania zdjęć pozwalający na zmianę rozmiarów zdjęcia z uwzględnieniem jego zawartości. Celem stosowania tego algorytmu jest ukierunkowanie na zmiany obszarów zdjęcia o mniejszym znaczeniu. Zatem najistotniejsze obiekty znajdujące się na obrazie mają być jak najbardziej chronione przed obróbką.

Rozdział pt. *Implementacja własnego algorytmu* jest syntezą trzech wcześniejszych rozdziałów czyli: *Sygnałów*, *Falek* oraz *Skalowania obrazów*. Wiedza oraz wzory zebrane z tych rozdziałów oraz własna wizja na ich odpowiednie ułożenie w całość, pozwalają na zapisanie kodu źródłowego, który zrealizuje algorytm seam carving. Dobór odpowiedniego środowiska pracy pozwoli nam na usprawnienie działań i z pewnością sprawi, że implementacja algorytmu będzie szybsza i bardziej skuteczna.

W następnym rozdziale pt. *Doświadczenia* zajmiemy się badaniami właściwości naszego własnego algorytmu i porównamy z wybraną, już istniejącą implementacją. Jak do tej pory niewiele programów graficznych pozwala nam na zastosowanie tego typu skalowania. Po wybraniu jednego spośród powszechnie dostępnych programów będziemy mogli porównać działanie algorytmu z tym własnego autorstwa.

W ten sposób przechodząc do rozdziału pt. *Podsumowania* będziemy mogli ocenić w jakim stopniu udało nam się zrealizować cel projektu oraz opisać całościowo efekty działania własnego algorytmu z wybranym, już istniejącym. Wskazanie wad i zalet własnego rozwiązania pozwoli nam w przyszłości na lepszą organizację pracy, rozszerzy poziom wiedzy oraz doświadczenia i pozwoli na wskazanie kierunku poprawy tak aby w przyszłości działać jeszcze efektywniej.

## 2. Sygnały

### *2.1. Definicja*

W najogólniejszym pojęciu [1] *sygnał* rozumie się jako bodziec niosący informację i zdolny działać na nasze zmysły lub narządy odbiorcze. Dzięki temu, zdajemy sobie sprawę, że wbrew pozorom pojęcie sygnał nie powinno nam się wyłącznie kojarzyć z najnowszą technologią. Generowanie sygnału może nastąpić z m. in. obiektów biologicznych, społecznych i technicznych, występujących w otaczającym nas świecie i zazwyczaj zawierają informacje o tych obiektach. Przykładem może być chociażby ludzka mowa, jako sygnał tworzony pod wpływem drgania strun głosowych. Dzięki możliwości nie tylko jego generowania ale również odbioru jego „treści", jesteśmy się w stanie porozumiewać.

Zajmijmy się jednak bardziej analitycznym podejściem do tematu. Zgodnie z definicją [2] sygnał jest to zmienność dowolnej wielkości fizycznej, którą można opisać za pomocą funkcji jednej $f(x)$ lub wielu zmiennych $f(x_1, x_2, x_3, \ldots)$. Przykładami zmiennych mogą być: temperatura, ciśnienie, prąd czy natężenie fali elektromagnetycznej. W praktyce, sygnały, które nas najbardziej interesują to sygnały będące funkcjami *czasu* $f(t)$ lub *położenia* w przestrzeni $f(x, y, z)$, np. $f(t) = \sin(2\pi ft)$, $f(t) = Ae'$, $f(x, y) = \exp(x^2+y^2)$. Zatem jako sygnał rozumiemy zmienność jakiejś wielkości fizycznej w funkcji wybranego argumentu np. ciśnienia lub napięcia elektrycznego w funkcji czasu czy chropowatość powierzchni w funkcji położenia.

Czasem okazuje się, że mimo iż odebraliśmy pewien sygnał, to tak naprawdę nie interesuje on nas jako całość ale tylko pewne jego elementy. Wówczas dokonujemy pewnej przemiany odebranego sygnału, przez co otrzymujemy jego *nową reprezentację*. Nowa reprezentacja sygnału uwypukla te cechy, które są dla nas najistotniejsze. Proces dokonywania tej przemiany nazywamy *przekształceniem* czy też *przetwarzaniem* sygnału.

### *2.2. Obraz jako sygnał dwuwymiarowy*

Jak już wiemy sygnały możemy reprezentować jako funkcję jednej lub wielu zmiennych. Spośród takiego wyboru ilości argumentów funkcji skupmy się na takiej, która ma dwie zmienne i oznaczymy ją jako $f(x, y)$. W momencie kiedy funkcja przyjmuje dwa argumenty to większość z nas odruchowo wyobraża sobie płaszczyznę – jeden argument to pozycja pozioma, drugi – pionowa. Dołóżmy do tego jeszcze pewną cechę dla każdej pary argumentów – jasność. W tym momencie otrzymujemy funkcję dwóch zmiennych $f(x, y)$

przedstawiającą jasność (w dowolnych jednostkach) w punktach o współrzędnych ($x$, $y$). Taką funkcję będziemy nazywać *obrazem*. Innymi słowami jest to dwuwymiarowa reprezentacja wycinka przestrzeni, która charakteryzuje się określonej rozkładem jaskrawości (barwy, przeźroczystości, gęstości przestrzennej).

Funkcja dwóch zmiennych może z łatwością stać się funkcją jednej zmiennej w momencie ustalenia jednego z argumentów [1]. Np. jeśli podstawimy $y=y_0$ to otrzymujemy funkcję jednej zmiennej $f(x, y_0)$, podobnie dzieje się przy podstawieniu $x=x_0$ – wówczas otrzymujemy również funkcję jednej zmiennej $f(x_0, y)$. W pierwszym przypadku otrzymujemy przekrój obrazu wzdłuż prostej równoległej do osi Y, w drugim przypadku natomiast przekrój wzdłuż prostej równoległej do osi X.

### *2.3. Właściwości obrazu*

Z pojęciem *obraz* wiąże się kilka cech ściśle z nim związanych jak i samych kilka jego różnych reprezentacji [1]:

– *jaskrawość* (*jasność*) *obrazu* – miara gęstości powierzchniowej natężenia światła, która decyduje o sile subiektywnego wrażenia jasności obrazu;

– *piksel* – najmniejszy i niepodzielny element obrazu cyfrowego, który może mieć przypisane własne, różniące się od sąsiednich elementów atrybuty. Nazwa pochodzi z *ang*. „picture element", czasem spotyka się też nawę *grel* (skrót od „graficzny element"[1]);

– *konwersja struktury obrazu* – sprzętowe lub programowe odwzorowanie ciągu danych opisujących obraz konturowy lub rastrowy;

– *rysunek ciągły* – odmiana obrazu konturowego, którego elementy składowe są opisane za pomocą współrzędnych względnych z wyjątkiem początkowego położenia elementu kreślącego obraz (strumienia elektronowego, głowicy kreślaka);

– *obraz światło-intensywnościowy* – obraz uzyskany w zakresie widmowym promieniowania widzialnego;

– *obraz wielo-widmowy* – zbiór nałożonych na siebie obrazów jednej i tej samej sceny, część z nich została utworzona w niewidzialnych ludzkim okiem zakresach widma promieniowania elektromagnetycznego;

– *obraz cyfrowy* – nieciągła postać obrazu (obraz zarejestrowany na nośniku komputerowym);

– *obraz rastrowy* – obraz powstały wskutek przekształcenia obrazu ciągłego na tablicę elementów zwanych pikselami. Każdemu pikselowi przypisano wektory cech typu poziomu jasności, barwa, nasycenie czy jaskrawość (również gęstość, przeźroczystość, przenikalność);

– *obraz konturowy* – inaczej rysunek liniowy – obraz złożony z linii i punktów, których dane mają ściśle określoną wewnętrzną strukturę;
– *obraz krawędziowy* – odmiana obrazu rastrowego, w którym każdy z pikseli jest przyporządkowany do jednej z dwóch grup – krawędziowy lub nie krawędziowy;
– *obraz plastrowy* (*warstwowy*) – obraz rastrowy przekroju porzecznego określonej sceny za pomocą (np. rentgenowskiego) tomografu komputerowego;
– *obraz trójwymiarowy (3W)* – stos (przyległych do siebie) obrazów rastrowych.

Oprócz obrazów świetlnych (czyli obrazów uzyskiwanych w zakresie promieniowania widzialnego), istnieje jeszcze klasa obrazów uzyskiwanych na innych zakresach promieniowania elektromagnetycznego a także obrazów tworzonych z wykorzystaniem innych zjawisk fizycznych, np. fal akustycznych.

### *2.4. Klasy obrazów*

Zajmijmy się klasyfikacją obrazów cyfrowych. Obrazy te dzielimy na 5 klas ze względu na postać reprezentacji danych w komputerze oraz sposób ich przetwarzania. Postać wizualna obrazu w tym wypadku jest sprawę drugorzędną [1]:
– klasa 1 – klasa obejmuje obrazy naturalne o pełnej gradacji kontrastów i odcieni. W komputerze są przedstawione jako dwuwymiarowe tablice pikseli, z których każdy z nich ma sprzężony wektor pewnych cech (m. in. barwa, intensywność, przenikalność). Człowiek nie jest w stanie rozróżnić poziomów jaskrawości, które różnią się mniej niż 0,5 %. Dzięki temu wystarcza 1 bajt do zakodowania obrazu barwnego;
– klasa 2 – klasa obejmuje obrazy dwuwartościowe (czarno-białe) lub kilkukolorowe, jak np. strona tekstu. W komputerze są reprezentowane przeważnie jako 1-bitowe tablice lub jako mapy, ze względu na to, że zawierają jednoznacznie określone obszary jednej barwy;
– klasa 3 – klasa obejmuje obrazy konturowe, np. wykresy czy kontury obszarów. Dane w komputerze najczęściej mają postać ciągu przyrostów ($\Delta x$, $\Delta y$) pomiędzy kolejnymi punktami;
– klasa 4 – klasa obejmuje obrazy punktowe. Obrazy tej klasy są złożone z punktów i wieloboków oddalonych od siebie w ten sposób, że nie mogą być reprezentowane przez kod łańcuchowy. Same punkty są reprezentowane przez tablicę ich współrzędnych. Punkty są połączone liniami prostymi, ewentualnie niekomplikowanymi krzywymi;
– klasa 5 – mapa głębi – struktury przestrzenne, zaliczane do struktur $2\ ^1/_2$ wymiarowych, ponieważ są macierzami przestrzennych punktów na powierzchni pewnego obiektu ale widzialnymi „perspektywicznie” z wybranego punktu w przestrzeni.

Podział ten traci na znaczeniu przez bardzo szybki postęp w tej dziedzinie. Dzięki temu coraz powszechniejsze jest stosowanie obrazów w jednolitej formie i pełnej reprezentacji (czyli jako obraz klasy 1).

Używając terminologii powyższego podziału obrazów na klasy, możemy natrafić na wiele różnych problemów związanych z przetwarzaniem informacji. Problemy te możemy określić jako przekształcenia pomiędzy klasami obrazów lub odwzorowania obrazów w ramach tej samej klasy. Wyróżniamy:

– Odwzorowania na wyższą klasę – dziedzina analizy obrazów;

– Odwzorowania na niższą klasę – dziedzina syntezy obrazów.

Obydwa odwzorowania oraz te w ramach każdej klas należą do dziedziny przetwarzania obrazów. Przykładem może być kompresja, która jest przekształceniem obrazów klasy 3 na obrazy klasy 2, z kolei wyostrzanie obrazu jest przekształceniem w ramach tej samej klasy.

## *2.5. Obróbka sygnałów dwuwymiarowych*

### 2.5.1. Przetwarzanie informacji wizualnej

W otaczającym nas świecie zjawiska fizyczne zazwyczaj mają charakter analogowy czyli ciągły. Jednak wobec wszechobecnej techniki komputerowej musimy nieco zmienić postać takich zjawisk. Nowa postać ma charakter cyfrowy – dla człowieka nienaturalny ale dla maszyny jak najbardziej pożądany.

Biorąc pod uwagę psychofizjologiczne właściwości percepcji wzrokowej człowieka najlepszą i pełną forma informacji jest obraz. Może być on również wzbogacony opisem symbolicznym. Wobec tego wymaga się aby komputer również był w stanie odbierać i wydawać wyniki w postaci obrazu.

Owe niedopasowanie było przyczyną powstania, już we wczesnym okresie rozwoju techniki nowej dyscypliny wiedzy – grafiki komputerowej. W ramach tej dziedziny prowadzone są prace nad wprowadzaniem do komputera informacji w postaci obrazu, jego przetwarzaniem oraz wyprowadzaniem. W szerszym znaczeniu grafika komputerowa obejmuje zespół środków sprzętowych i programowych mających na celu dopasowanie dwóch odmiennych elementów systemu:

- dyskretnego, z zasady działania i generowania wyników – komputera;
- analogowego, ze swojej natury – człowieka wraz z otaczającym go światem.

Dopasowanie polega na opracowaniu sposobów zamiany postaci cyfrowej na analogową oraz analogową na cyfrową.

Przetwarzanie informacji wizualnych i obrazowych przez komputer dzielimy na 3 części [3]:

- analizę obrazów;
- przetwarzanie obrazów;
- grafikę komputerową (syntezę obrazów).

### 2.5.2. Analiza obrazów

Termin *analiza obrazów* możemy odnieść do co najmniej trzech zakresów znaczeniowych. Podstawowe, wąskie znaczenie obejmuje metody tworzenia opisu oraz jego klasyfikowania w ramach określonych klas. Polega on na zamianie obrazu na abstrakcyjny opis który stanowią: zbiory liczb, symboli lub graf połączeń składowych pierwotnych tzw. pierwotniaków. Następnie taki obraz jest zakwalifikowany do jednej z wcześniej określonych klas. Przykładem może być rozpoznawanie znaków: znak wejściowy jest wydzielany z ciągu i jest zapamiętywana jego klasa. Jeśli chodzi o klasyfikowanie obrazów to mamy tu na myśli określenie właściwości, które mają wszystkie obiekty danej klasy. Przytoczony przykład klasyfikowania znaków może być rozważone jako przykład analizy, w którym na wyjściu jest jedynie nazwa obiektu.

W ramach analizy obrazów możemy wyróżnić kilka ważnych pojęć:

- percepcja – wydzielenie obiektu z całości i ustanowienie go jako osobnika;
- identyfikacja – ustalenie związku pomiędzy obiektem i jego opisem;
- rozpoznanie – zestawienie percepcji i identyfikacji czyli powiązanie wydzielania obiektu z tła i sprzężenie z nim opisu.

Analiza obrazów rozumiana w szerszym zakresie obejmuje również rekonstrukcje obiektów i scen trójwymiarowych na podstawie obrazów. Polega to na odtworzeniu budowy wewnętrznej obiektu lub jego kształtów w wyniku przeprowadzonej analizy obrazów danego obiektu i dodatkowych danych – więzi wynikających z zadania w ramach którego ta rekonstrukcja jest wykonywana.

Analiza obrazów rozumiana w najwyższym poziomie to ich analiza semantyczna – rozumienie obrazów i wnioskowanie o scenie zrekonstruowanej z obrazów. Badania nad zrozumieniem obrazów mają na celu rozszerzenie możliwości komputerów na takie jakie posiada ludzki system wzrokowy. Wówczas mogłyby one postrzegać otoczenie, następnie interpretować je i w końcu podejmować zaprogramowane akcje w ramach wykonywania

interpretacji. Bierze się to stąd, że postrzeganie obrazów to nie tylko luminacja i geometria sceny ale przede wszystkim intelektualne rozpoznanie i rozumienie otoczenia.

Rozpoznanie obrazów możemy przeprowadzić na kilka sposobów [3]:

- statystyczne – wykorzystanie rachunku prawdopodobieństwa i metod statystycznych do przydzielania osobników do określonych klas;
- strukturalne – postać obrazu jest reprezentowane przez pierwotniaki i zależności miedzy nimi w celu opisania i sklasyfikowania struktury wzorca;
- syntaktyczne – rodzaj rozpoznawania strukturalnego obrazów gdzie pierwotniaki i zależności między nimi są wyrażone w naturalnym lub sztucznym języku;
- piktograficzne – rozpoznawanie napisów obrazkowych;
- aspektowe dowolno-widokowe – metoda rozpoznawania trójwymiarowych obiektów na podstawie wybranych cech na obrazie uzyskanym z wybranego punktu widokowego.

### 2.5.3. Przetwarzanie obrazów

Termin *przetwarzanie obrazów* funkcjonuje w co najmniej dwóch zakresach znaczeniowych [3].

Przetwarzanie obrazów w wąskim sensie polega na ulepszaniu obrazu wejściowego dla potrzeb obserwatora. Przykładem może być zmiana rozmycia obrazu czy zmiana kontrastu co może spowodować wzmocnienie poszczególnych elementów obrazu. W ogólnym znaczeniu mamy tu do czynienia z różnego rodzaju operacjami wykonywanymi na obrazach: usuwaniu zakłóceń, kompresji, wyostrzaniu, polepszaniu kontrastu itp. Możemy to uzyskać za pomocą programu komputerowego, procesorów cyfrowych lub nawet sprzętu optycznego. Wszelkie operacje są wykonywane dla potrzeb oceny obrazu przez człowieka a parametry przetwarzania są formułowane przez operatora. System przetwarzający obrazy nie uwzględnia samej treści przedstawionej sceny.

W szerokim znaczeniu przetwarzanie obrazów może być rozumiane na dwa sposoby:

- przetwarzanie wstępne oraz zasadnicze – upraszczanie postaci a następnie jego klasyfikowanie;
- przetwarzanie niskopoziomowe oraz wysoko poziomowe – wydzielania cech obrazu w etapie początkowym i przetwarzania.

Podział ten bywa bardziej szczegółowy i może objąć 3 lub nawet 3 stopnie przetwarzania.

### 2.5.4. Grafika komputerowa

Termin grafika komputerowa podobnie jak przetwarzanie obrazów, funkcjonuje w dwóch zakresach znaczeniowych. W szerokim zakresie, jak już wiemy, jest to dopasowanie postaci cyfrowych i analogowych obrazu, wywodzące się z odpowiednio reprezentacji obrazu w komputerze oraz naturalnego postrzegania przez człowieka. W wąskim znaczeniu termin ten jest rozumiany jako generowanie obrazu na podstawie opisu matematycznego i reprezentowaniu jego na graficznym urządzeniu wyjściowym. W przypadku obiektów trójwymiarowych, ich reprezentowanie wymaga zadania punktu (punktów) widokowego i wykonaniu projekcji obiektów na płaszczyznę przed jego prezentacją. Grafika w tym znaczeniu obejmuje wiele różnych zastosowań. Cechą charakterystyczną jest to, że przeważnie programy graficzne są ukierunkowane na rozwiązywanie określonego, często wąskiego kręgu zagadnień.

W wąskim znaczeniu generowania obrazów, grafika jest zagadnieniem odwrotnym do analizy obrazów. Obydwie dziedziny jednak dotyczą tej samej fizycznej rzeczywistości. Dlatego podstawowe koncepcje i problemy są w obu dziedzinach takie same, mimo, że mają inny aspekt. Wspólną klasą problemów do analizy obrazów oraz grafiki jest: modelowanie, składowanie, kompresja oraz struktury danych obrazowych [3].

Termin grafika komputerowa uzupełniają poniższe terminy, będące częścią tej dziedziny:

- grafika interakcyjna – korzystanie z konsoli graficznej do komunikacji z komputerem w trybie konwersacyjnym;
- grafika bierna – używanie konsoli graficznej w trybie biernym;
- grafika konturowa – generowanie obrazów mających określoną strukturę wewnętrzną na podstawie rozkładów graficznych i współrzędnych, techniką konturowego rozwinięcia obrazu;
- grafika rastrowa (obrazowa) – generowanie obrazów na podstawie danych nie mających struktury wewnętrznej, techniką rastrowego rozwinięcia obrazu.

Danymi wejściowymi w grafice komputerowej są przeważnie opisy obrazów ale mogą to być też obrazy naturalne, uzyskane za pomocą urządzeń akwizycji obrazów, np. skanera lub kamery. Są one wykorzystywane do powstawania nowych obrazów lub obiektów, po dokonaniu na nich odpowiednich zmian i uzupełnień. Te możliwości i potrzeby przemawiają za jednolitym traktowaniem całości zagadnień dotyczących wprowadzania, analizowanie, przetwarzania, generowania i wyprowadzania informacji wizualnej, obrazowej i graficznej za pomocą komputera.

## 3. Falki

### *3.1. Definicja*

Zgodnie z definicją [3] falki (*małe fale*, *wavelets*) to rodziny funkcji $\Psi_{j,k}(t)$ zbioru liczb rzeczywistych w zbiór liczb rzeczywistych. Każda z nich jest wyprowadzona z funkcji-matki (funkcji macierzystej, podstawowej) $\Psi(t)$ za pomocą przesunięcia i skalowania:

$$\Psi_{j,k}(t) = \Psi(2^j t+k) \tag{3.1}$$

gdzie:

$\Psi(t)$ – funkcja matka,

$\Psi_{j,k}$ – falka (funkcja falkowa) o skali $j$ i przesunięciu $k$ w stosunku do funkcji-matki,

$j,k$ – liczby całkowite.

Funkcje te dążą do zera (lub wynoszą zero) poza pewnym przedziałem dla argumentu dążącego do nieskończoności. Ich suma ważona umożliwia przedstawienie z dowolną dokładnością funkcji okresowej. Tak więc funkcji Falkowe umożliwiają przedstawienie każdej funkcji ciągłej z określoną dokładnością poprzez sumę ważoną.

### *3.2. Przykładowe falki i ich zastosowanie*

Rozróżniamy kilkanaście różnych falek [4]: Haara, Daubechies, Symlets, Coiflets, Bioortogonalne, odwrotne bioortogonalne, Meyera, Gaussa, „Meksykański kapelusz”, Morleta, Shannona.

Falki są używane w analizie i przetwarzaniu sygnałów cyfrowych, w kompresji obrazów i dźwięków oraz w wielu innych dziedzinach.

Jednym z przykładowych zastosowań jest użycie falek w kompresji obrazów w standardzie JPEG 2000 [9], który jest uzupełnieniem bardzo popularnych technik kompresji JPEG. Algorytm JPEG 2000 polega na wykorzystaniu dyskretnej transformaty Falkowej DWT, która dzieli obraz na wysokie i niskie częstotliwości. Część odpowiadająca niskim częstotliwościom może być dzielona dalej ten sam sposób. Tak przygotowaną tablicę próbek dzielimy na bloki, które kwantuje się i koduje niezależnie od siebie. Stopień kompresji regulujemy poprzez wysyłanie tylko niektórych bloków oraz zmienną kwantyzację próbek.

W porównaniu do JPEG, standard JPEG 2000 charakteryzuje lepsza jakość obrazu przy tym samym stopniu kompresji. Możliwa jest również kompresja bezstratna, co czyni konkurencję dla innego formatu – PNG. Mamy też możliwość przeplotu danych – w miarę

przesyłania (np. przez sieć) próbek obrazu jego jakość stopniowo się poprawia. Tryb ten ma również standard JPEG ale w bardzo uproszczonej wersji. Format JPEG wyprzedza jednak JPEG 2000 pod względem złożoności obliczeniowej, co wyklucza jego całkowite zastąpienie nowszym standardem.

### *3.3. Falka Haara*

Falka Haara jest najstarszą i najprostszą z falek. Jest to para prostokątnych impulsów – ujemnego i dodatniego. Została zdefiniowana przez węgierskiego matematyka Alfréda Haara w 1910 r. [5]. Wówczas oczywiście nie nosiła takiej nazwy. Funkcję-matkę (falkę podstawową) Haara określa się wzorem:

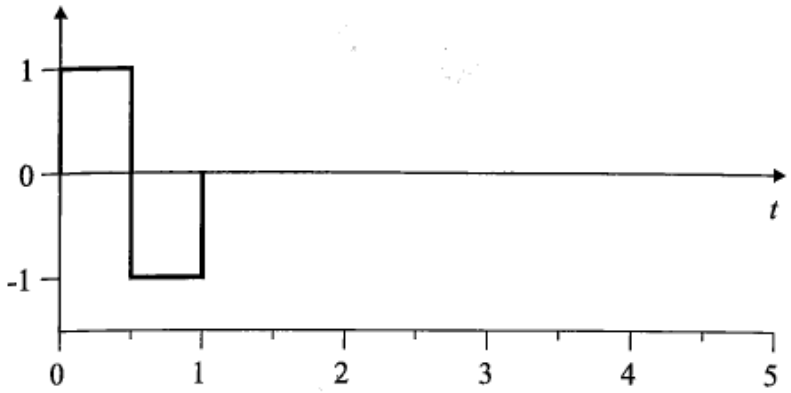
$$\Psi = \begin{cases} 1 & t \in \left[0, \frac{1}{2}\right) \\ -1 & t \in \left[\frac{1}{2}, 1\right] \\ 0 & \text{w pozostałych przypadkach} \end{cases} \tag{3.2}$$

Widzimy zatem, że dla przedziału $t \in [0,1]$ funkcja przyjmuje wartości różne od 0. Przedział ten jest nośnikiem tej falki.

Wzór (3.2) generuje zbiór falek Haara o elementach:

$$\Psi_{mn}(t) = 2^{-m/2}\Psi(2^{-m}t - n) \text{ dla } m, n = \ldots, -2, -1, 0, 1, 2, \ldots \tag{3.3}$$

Wykres funkcji-matki prezentuje się następująco:

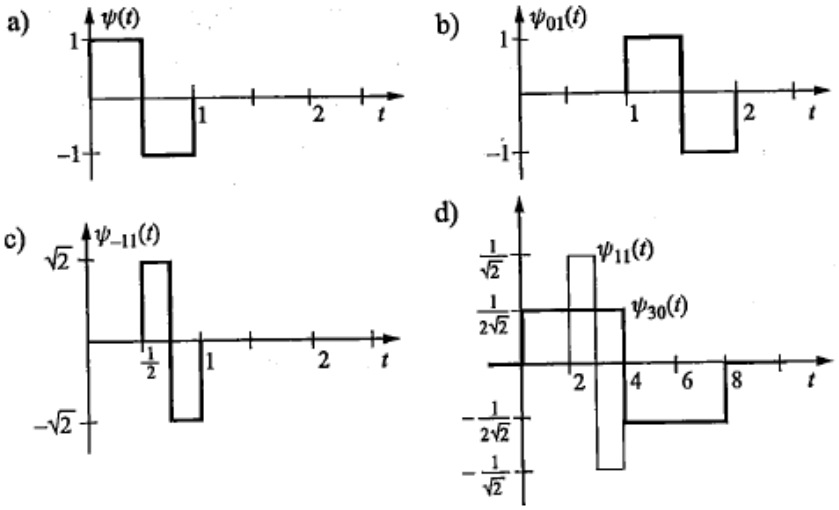



*Rys. 3.1. Wykres funkcji-matki falki Haara [1]*

Ponieważ nośnik funkcji $\Psi_{mn}(t)$ ma długość $2^m$, liczbę całkowitą $m$ nazywamy współczynnikiem skali zaś $n$ – współczynnikiem przesunięcia. Samo przesunięcie falki

podstawowej $\Psi$ zależy od współczynnika skali i wynosi $2^m n$. A zatem falki z tym samym współczynnikiem skali nie mają wspólnego nośnika, to znaczy ich przesunięcie jest całkowitą wielokrotnością długości nośnika. Jeśli dla falek o różnych skalach nośniki przecinają się, to wówczas falka o większej długości nośnika jest stała w przedziale równym nośnikowi falki o krótszym nośniku. Owe obserwacje prowadzą do ustalenia, że falki Haara są *funkcjami ortogonalnymi*. Współczynnik $2^{-m}/2$ w równaniu (3.3) jest współczynnikiem normalizującym i jest on odwrotnością pierwiastka kwadratowego z długości falki o współczynniku skali *m*. Współczynnik normalizujący $2^{-m}/2$ sprawia, że dla każdej falki całka jej kwadratu w przedziale nośnika (zwanej normą) jest równa 1 [5]. Kilka elementów zbioru falek Haara przedstawia Rysunki 3.2. a, b, c. Własność ortogonalności falek o różnych skalach ilustruje Rysunek 3.2. d, na którym nośnik falki „węższej" jest zawarty w przedziale, w którym falka „szersza" jest stała.

Zaletą funkcji $\Psi mn$ jest ich dobra lokalizacja w czasie. Oznacza to, że długość ich nośnika równa $2^m$ jest skończona. W związku z tym dla $m \rightarrow -\infty$ uzyskujemy nieskończenie ostrą lokalizację w czasie, co pozwala na dowolną dokładność lokalizacji nieciągłości (w szczególności skoku) funkcji, której rozwinięcie określamy względem zbioru falek Haara. Duże i dodatnie *m*, odpowiadające falce o szerokim nośniku, jest kojarzone z ostrością lokalizacji względem częstotliwości oraz z odpowiednią stratą rozdzielczości względem czasu.




*Rys. 3.2. Przykładowe elementy zbioru falek Haara i ich ortogonalność [5]:*
*a) podstawowa falka Haara, b) falka o parametrach m = 0, n = 1;*
*c) falka o parametrach m = -1, n = 1, d) ilustracja ortogonalności falek Haara o różnych skalach*

Falkę Haara można wykorzystać również do analizy sygnału dwuwymiarowego. Wówczas obraz oryginalny jest dzielony na bloki 2x2 piksele i w efekcie otrzymujemy wydzielenie

poszczególnych cech obrazu na 4 części: cztero-punktową średnią, średni poziomy gradient, średni pionowy gradient, wartość diagonalna.

Do obliczeń korzystamy ze wzorów:

$$h_0 = \frac{1}{2}(a_{11} + a_{10} + a_{01} + a_{00}) \tag{3.4}$$

$$h_x = \frac{1}{2}(a_{11} + a_{10} - a_{01} - a_{00}) \tag{3.5}$$

$$h_y = \frac{1}{2}(a_{11} - a_{10} + a_{01} - a_{00}) \tag{3.6}$$

$$h_z = \frac{1}{2}(a_{11} + a_{10} - a_{01} + a_{00}) \tag{3.7}$$

gdzie:

$h_0$ – cztero-punktowa średnia;

$h_x$ – średni poziomy gradient;

$h_y$ – średni pionowy gradient;

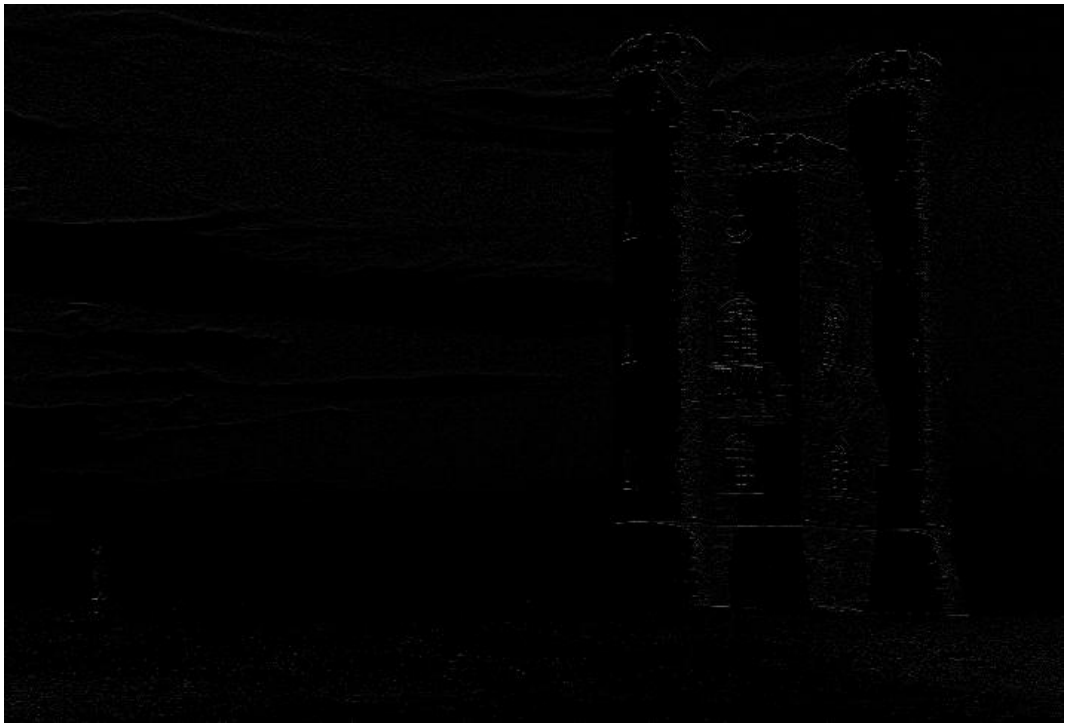
$h_z$ – wartość diagonalna

$a_{00}$, $a_{01}$, $a_{10}$, $a_{11}$ – współrzędne pikseli w kwadracie 2x2.

Przykładowy efekt rozdzielenia obrazu na tego typu bloki możemy zaobserwować na popularnym obrazie *Lena.png* (Rysunek 3.3.).

*Rys. 3.3. Wykorzystanie falki Haara do analizy obrazów (wydzielenie na 4 bloki) [10]*

To co nas będzie z tych informacji interesować to średni poziom gradientu poziomego lub pionowego, zależy od tego czy chcemy zmienić wysokość czy szerokość obrazu.



*Rys. 3.4. Średni gradient poziomy wybranego zdjęcia*

*Rys. 3.5. Średni gradient pionowy wybranego zdjęcia*

# 4. Skalowanie obrazów

## *4.1. Skalowanie oparte o interpolację*

Skalowanie obrazów jest jedną z najczęściej dokonywanych operacji na obrazach. W najprostszym rozumieniu polega na pomnożeniu współrzędnych każdego piksela obrazu przez odpowiedni współczynnik. Przyjmijmy, że dla osi *X* będzie to $S_x$ a dla osi *Y* będzie to $S_y$. Wówczas otrzymujemy wzory:

$$x' = x_0 S_x \tag{4.1}$$

$$y' = y_0 S_y \tag{4.2}$$

gdzie:

$(x_0, y_0)$ – współrzędne piksela przed skalowaniem;

$S_x$, $S_y$ – współczynniki skalowania względem osi odpowiednio *X* i *Y*;

$(x', y')$ – współrzędne piksela po skalowaniu.

Typowym zastosowaniem operacji skalowania jest *zooming* (ang. *zoom in*/*out* – *powiększać*/*pomniejszać*) czyli zbliżanie bądź oddalanie obiektów.

W ramach cyfrowego przybliżenia danego obrazu możemy zauważyć jednak niepożądane efekty zwane potocznie *pikselozą*. Objawia się to zwiększeniem rozmiarów pikseli przy dokładnie takim samym ich rozmieszczeniu względem siebie. Dobrze by było więc, gdyby przy powiększaniu rozdzielczość obrazu wzrastała. Mając do dyspozycji określoną „mapę" początkowych pikseli musimy uzupełnić brakujące obszary w nowo tworzonym obrazie. Utworzony piksel powinien być też możliwie najlepiej dopasowany do sąsiednich. Procesem takiego tworzenia nowych pikseli nazywamy *interpolacją*.

Wyróżniamy kilka różnych metod interpolacyjnych [3]:

- najbliższy sąsiad – przypisanie wartości najbliższego piksela obrazu wejściowego względem obrazu skorygowanego;

- interpolacja dwuliniowa – wartość piksela jest obliczana na podstawie czterech najbliższych z zastosowaniem interpolacji dwuliniowej;

- interpolacja kosinusowa – oparta na funkcji $cos(x)$ między dwoma sąsiednimi pikselami;

- metoda sześciennej funkcji sklejanej (splan sześcienny) – nowa wartość piksela jest obliczana na podstawie szesnastu najbliższych mu pikseli z użyciem funkcji sklejanej trzeciego stopnia;

- metody Lanczosa – oparte na funkcji $sinc(x)$ $(=sin\pi x/\pi x)$ i uwzględniają sąsiadujące piksele w oknach o wielkościach 4x4, 6x6 lub 8x8. W przypadku sąsiedztwa 4x4 algorytm

daje zbliżone wyniki do interpolacji dwusześciennej. W pozostałych przypadkach wyniki są lepsze ale odbywa się to kosztem znacznie dłuższych obliczeń;

- rodzina metod na bazie funkcji *sinc*(*x*) w rozległym otoczeniu – metoda 64-elementowej funkcji *sincx*/*x*, w której wartość nowego piksela liczymy z aż 256 najbliższych pikseli z zastosowaniem aproksymacji funkcją *sincx*/*x*;

- inne metody oparte na funkcji *sinc*(*x*) z oknami – np. okna Welcha, Hanna-Hamminga, Backmana-Harrisa, Parzena.

## *4.2. Algorytm seam carving*

### 4.2.1. Działanie algorytmu

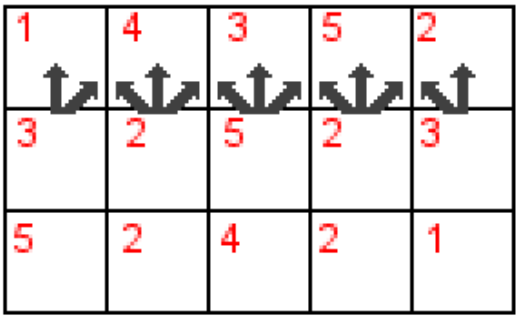
Seam carving (ang. seam – szew, carve – rzeźbić, krajać, wyryć; co można przetłumaczyć jako “wycinanie szwów”) jest nowoczesnym algorytmem skalowania zdjęć [6, 7, 8]. Został opracowany przez Shaia Avidana z Mitsubishi Electric Research Laboratories (MERL) oraz Ariela Shamira z Interdisciplinary Center i MERL. Algorytm opiera się na wyszukiwaniu *szwów* (ang. *seams*), czyli krzywych prowadzonych pionowo lub poziomo. Dzięki temu w miejscach gdzie te krzywe przebiegają istnieje możliwość usunięcia lub dodania pikseli. Proces wyszukiwania szwów jest ściśle związany z bezpośrednim sąsiedztwem danych pikseli.

W celu wyznaczenia szwu, dla każdego piksela przypisujemy wartość, którą możemy określić jako energię, kontrast czy wagę. Jest to wielkość charakteryzująca zmienność pikseli sąsiadujących w stosunku do analizowanego. Im większe różnice występują, tym większa jest energia danego piksela (schemat – Rysunek 4.1.). Naszym celem jest takie wytyczanie krzywych aby miały jak najmniejszą sumę energii. Aby to zobrazować przyjmijmy, że wyszukujemy szwów pionowo.

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 3 | 5 | 2 |
| 3 | 2 | 5 | 2 | 3 |
| 5 | 2 | 4 | 2 | 1 |

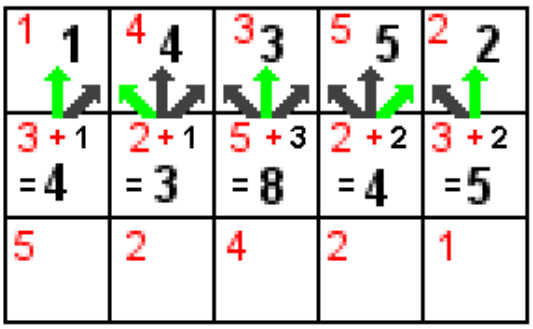
*Rys. 4.1. Przypisanie wag poszczególnym pikselom*

Następnym krokiem jest wyznaczenie dla każdego piksela z drugiego rzędu, sąsiadującego piksela z pierwszego rzędu o najmniejszej energii. Za sąsiedztwo rozumiemy stykanie się pikseli bokiem lub wierzchołkiem. Zatem skrajne piksele mają tylko dwie możliwości podczas gdy pozostałe – trzy (Rysunek 4.2.).



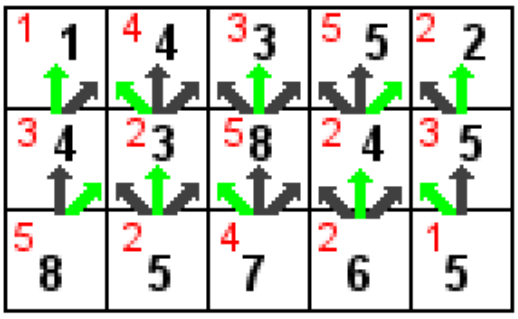

*Rys. 4.2. Wyznaczenie sąsiadujących pikseli z pierwszego rzędu dla drugiego rzędu*

Gdy już znajdziemy piksel z pierwszego rzędu o najmniejszej wadze spośród sąsiadujących z danym pikselem z drugiego rzędu, wówczas mamy wyznaczone pierwsze fragmenty szwów. Wagi obu pikseli w danym fragmencie dodajemy do siebie (Rysunek 4.3.).



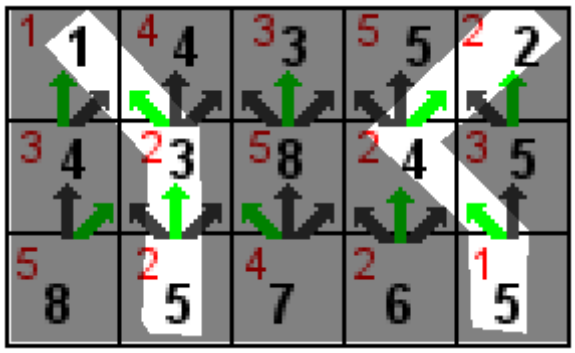

*Rys. 4.3. Sumowanie wag pikseli na wyznaczonych fragmentach szwów[11]*

W tym momencie oprócz wag poszczególnych pikseli mamy również wyznaczone sumy energii początkowych fragmentów krzywych. Od tego momentu dla każdego piksela z kolejnego rzędu nie przyporządkowujemy wagi sąsiadującego od góry piksela ale wagę całego fragmentu krzywej (Rysunek 4.4.).




*Rys. 4.4. Obliczanie sum energii w kolejnych rzędach pikseli (liczby czarne) [11]*

Przechodząc kolejno przez wszystkie wiersze od góry do dołu otrzymujemy na końcu sumy wag wszystkich krzywych wyznaczonych na podstawie wag poszczególnych pikseli. Ilość szwów jest równa szerokości obrazu. W celu dalszej obróbki zdjęcia, czyli zmiany szerokości zdjęcia wyszukujemy krzywe o najmniejszej sumie energii (Rysunek 4.5.)




*Rys. 4.5. Wyszukiwanie krzywych o najmniejszej sumie energii [11]*

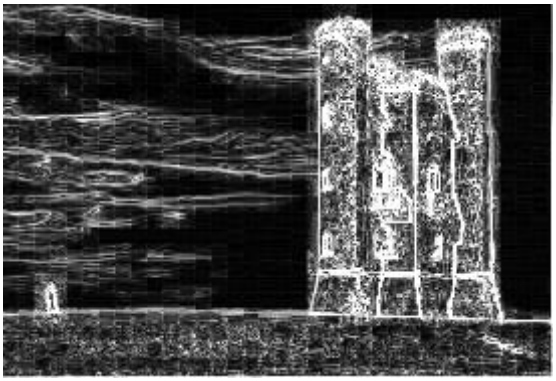
Dzięki wyznaczeniu szwów mamy możliwość zmiany rozmiarów obrazu. Jeżeli chcemy zwiększyć rozdzielczość to wówczas w miejscu szwów następuje utworzenie pikseli, jeżeli zmniejszyć to wówczas szwy są po prostu usuwane. Trzeba również zauważyć, że po każdej tego typu operacji piksele należące do szwu i z nim sąsiadujące mają inną energię Przez ten fakt po każdorazowej zmianie konieczne jest ponowne przeliczenie energii każdego z pikseli.

Wartość wag poszczególnych pikseli jest ściśle związana z bezpośrednim sąsiedztwem danego piksela. Im wyższy skok wartości sąsiednich pikseli tym wyższa wartość wagi analizowanego piksela a zatem mniejsze prawdopodobieństwa zawarcia go w krzywej. Małe różnice wartości energii możemy zaobserwować w momencie kiedy tło jest jednolite lub istnieją łagodne przejścia pomiędzy kolejnymi barwami. Z kolei duże zmiany wartości możemy zaobserwować na granicach dwóch obiektów np. kontur jakiegoś obiektu na jednolitym tle jest łatwy do wykrycia w ten sposób. Możemy wówczas zauważyć, że w tych miejscach jest dużo mniejsze prawdopodobieństwo wyznaczenia fragmentu szwu, wobec tego obiekt ten jest niejako „chroniony” przez ten algorytm. I to jest cecha charakterystyczna algorytmu seam carving – wszystkie istotne obiekty są maksymalnie chronione przed zniekształceniem, podczas gdy obróbce zostają poddane obszary o zbliżonych wartościach. Przyjrzyjmy się temu bliżej na poniższym przykładzie.

Jako obiekt przyjmijmy sobie zdjęcie z Rysunku 4.6. W prawej części jest na nim przedstawiony zamek, z lewej strony proporcjonalnie mniejszy człowiek, w tle na pierwszym planie mamy trawnik a na dalszym planie niebo z niewielkim zachmurzeniem. Naszym celem jest zmniejszenie szerokości zdjęcia zatem szwów będziemy szukać pionowo.

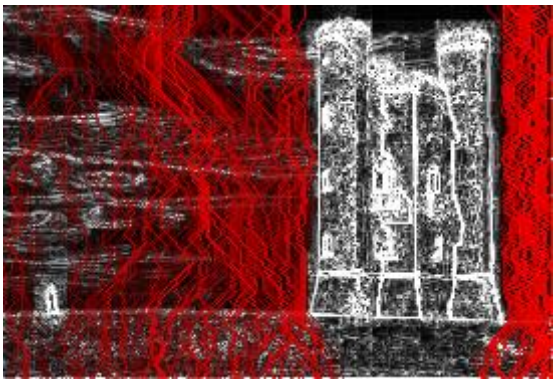
*Rys. 4.6. Zdjęcie przed obróbką [11]*

Początkowym etapem jest stworzenie czarno-białej *mapy energetycznej* zdjęcia (Rysunek 4.7.) – każdy piksel jest odpowiednio zobrazowany przez jego energię. Im kolor piksela jest jaśniejszy tym jego wartość energii jest większa. Zauważmy, że największe wagi (a zatem największa zmienność sąsiednich pikseli) występują na najistotniejszych elementach zdjęcia – człowieku oraz zamku.



*Rys. 4.7. Mapa energetyczna zdjęcia [11]*

Kolejnym etapem jest wyszukiwanie szwów na podstawie wag poszczególnych pikseli. Znalezione szwy oznaczone są na Rysunkach 4.8. i 4.9. kolorem czerwonym. Zwróćmy uwagę, że całkowicie pomięte są główne obiekty zdjęcia – człowiek oraz zamek co jest ściśle związane wysoką energią pikseli składających się na te obiekty.

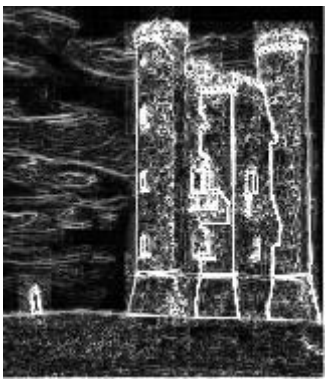


*Rys. 4.8. Szwy na mapie energetycznej [11]*

*Rys. 4.9. Szwy na oryginalnym obrazie [11]*

Następnie w celu zmniejszenia rozmiarów zdjęcia usuwamy szwy. Efekt końcowy możemy zobaczyć na Rysunku 4.10. Ponadto dla porównania ponownie została przeliczona energia wszystkich pikseli (Rysunek 4.11.). Można zauważyć, że w porównaniu do Rysunku 4.7. zostały znacznie zredukowane czarne obszary z lewej jak i z prawej strony zamku. Są to obszary pikseli o najmniejszych wagach i zostały one głównymi elementami szwów.

*Rys. 4.10. Efekty po skalowaniu seam carving [11]*



*Rys. 4.11. Mapa energetyczna zdjęcia po przeskalowaniu [11]*

Efekty jakie możemy uzyskać tym algorytmem są bardzo dobre w porównaniu z obróbką zdjęć innymi metodami. Dla porównania zdjęcie z Rysunku 4.12. poddaliśmy trzem różnym metodom zmiany rozmiarów zdjęcia: metodą skalowania opartego o interpolację (Rysunek. 4.13.), kadrowania (Rysunek 4.14.) oraz metodą seam carving (Rysunek 4.15.).

*Rys. 4.12. Zdjęcie oryginalne [11]*

*Rys. 4.13. Zdjęcie poddane skalowaniu opartego o interpolację [11]*

*Rys. 4.14. Zdjęcie poddane kadrowaniu [11]*

*Rys. 4.15. Zdjęcie poddane skalowaniu seam carving [11]*

Przy zastosowaniu tradycyjnej metody skalowania (Rysunek 4.13.) obiekty ulegają zniekształceniu poprzez zwężenie tylko w jednym wymiarze. Skalując tą metodą dobre efekty uzyskamy jedynie wtedy gdy współczynniki zmniejszenia lub zwiększenia zdjęcia są takie same zarówno w pionie jak i w poziomie. Kadrowanie zdjęcia (Rysunek 4.14.) również nie daje dobrego efektu. Rozmiary obiektów są co prawda nie zmienione ale ze względu na ich rozmieszczenie możliwa jest utrata istotnych informacji. Tak się dzieje w powyższym przykładzie gdzie odległość dzieląca człowieka i zamek jest zbyt duża w stosunku do rozmiaru docelowego zdjęcia.

Stosując metodę seam carving (Rysunek 4.15.) chronimy główne obiekty zdjęcia, które nie ulegają zmianom. Znacznie bardziej narażone na zmiany są tła charakteryzujące się małymi zmianami sąsiadujących pikseli w związku z czym otrzymujemy dobry efekt wizualny zdjęcia.

### 4.2.2. Występowanie w powszechnym oprogramowaniu

Algorytm seam carving znalazł już swoje implementacje w powszechnie używanym oprogramowaniu.

Jednym z przykładów jest funkcja *Content-aware scalling* w popularnym programie graficznym Adobe Photoshop. Szczegółowe informacje możemy znaleźć na stronie internetowej pomocy [12], gdzie opisana jest zasada działania algorytmu oraz opisane opcje dostępne w ramach działania tej funkcji.

Innym przykładem programu wykorzystujący algorytm seam carving jest Seam Carving GUI, któremu bardziej szczegółowo przyjrzymy się w kolejnym podrozdziale.

### 4.2.3. Opis i działanie wybranej aplikacji

Przykładową aplikacją wykorzystującą skalowanie seam carving jest program Seam Carving GUI.

Program Seam Carving GUI jest programem do obróbki grafiki nastawiony na stosowanie algorytmu skalowania seam carving.

Krótka charakterystyka kilku jego najistotniejszych funkcji:

- sekcja *Retain/Remove* – sekcja będąca rozwinięciem funkcji klasycznego seam carving. Dzięki niej mamy możliwość zarysowania obiektów na zdjęciu, które chcemy chronić

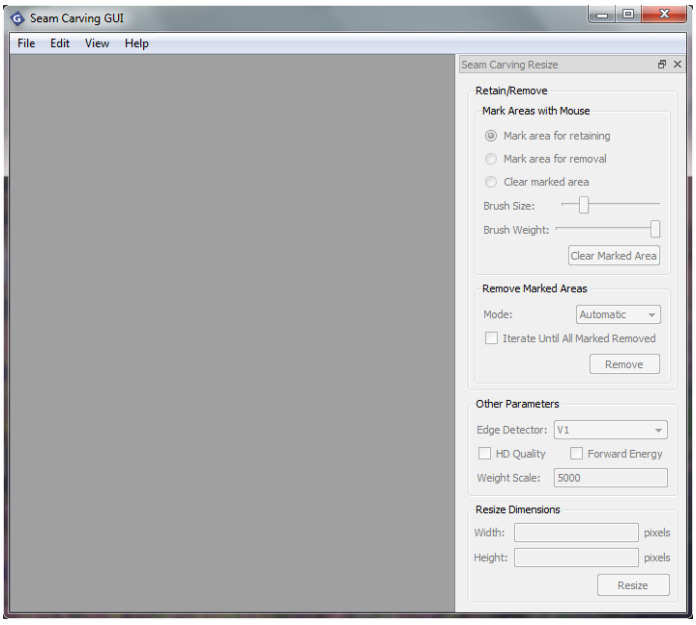
(zaznaczamy *Mark area for retaininig*) lub te, które chcemy usunąć (zaznaczamy *Mark area for removal*). Dane obszary zamalowujemy pędzlem (*brush*) – na zielono obszary chronione a na czerwono obszary przeznaczone do usunięcia. Możemy zmieniać rozmiar pędzla (suwak *Brush Size*) jak i jego wpływ na energię danych pikseli (suwak *Brush Weight*). Jeżeli uznamy, że chcemy jednak zaznaczyć inne obszary to mamy możliwość usunięcia wszystkich zaznaczeń (przycisk *Remove* w podsekcji *Clear Marked Area*) lub pędzlem wymazując to co chcemy (zaznaczamy *Clear marked area*) posługując się nim dokładnie w taki sam sposób jak przy zaznaczaniu obszarów do ochrony lub usunięcia.

Sekcja *Other* -> opcja *Edge detektor* – opcja ta umożliwia wykrywanie krawędzi, czyli obszarów o największej zmienności pikseli na kilka sposobów: *V_1*, *V_SQUARE*, *Prewitt*, *Sobel*, *Laplacian*.

Sekcja *Resize Dimension* – sekcja w której mamy możliwość zmiany rozmiarów obrazu. Po zmianach w tym miejscu jesteśmy w stanie zauważyć jakie wpływ na oryginalny obrazek mają zastosowane przez nas opcje.

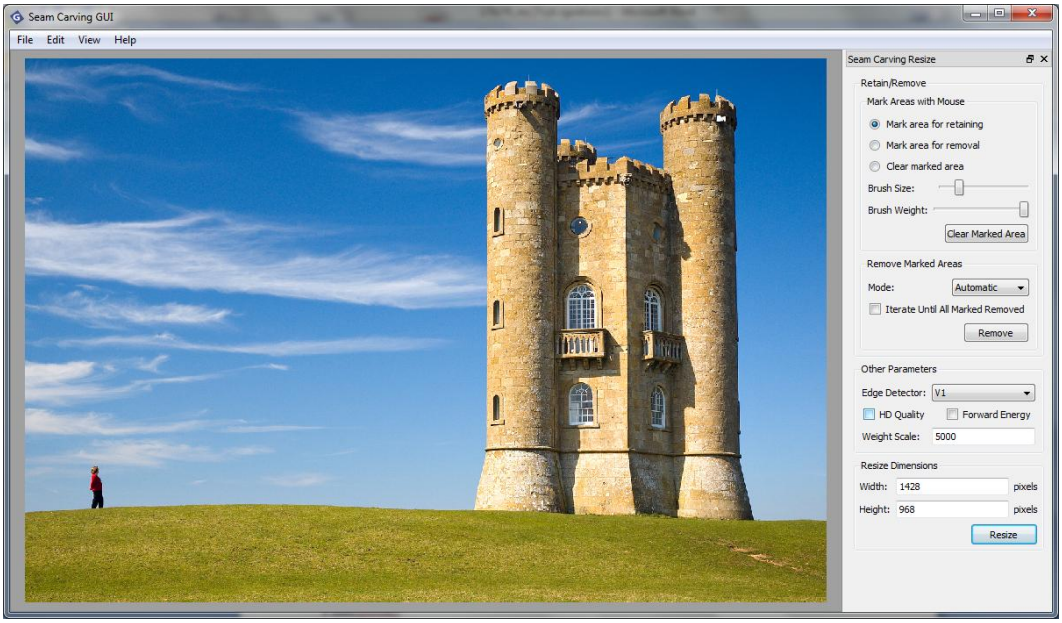
Opcje w menu pod zakładką *View* – zbiór różnych możliwości podglądu zdjęcia:

- *View Image* – widok zdjęcia w oryginale;
- *View Greyscale* – widok zdjęcia w odcieniach szarości;
- *View Edge* – widok wykrywania krawędzi odpowiednią metodą (wybór metody w sekcji *Other Parameters*), do wyboru: *V_1*, *V_SQUARE*, *Prewitt*, *Sobel*, *Laplacian*;
- *View Vertical Energy* – widok zmian energii w pionie;
- *View Horizontal Energy* – widok zmian energii w poziomie.



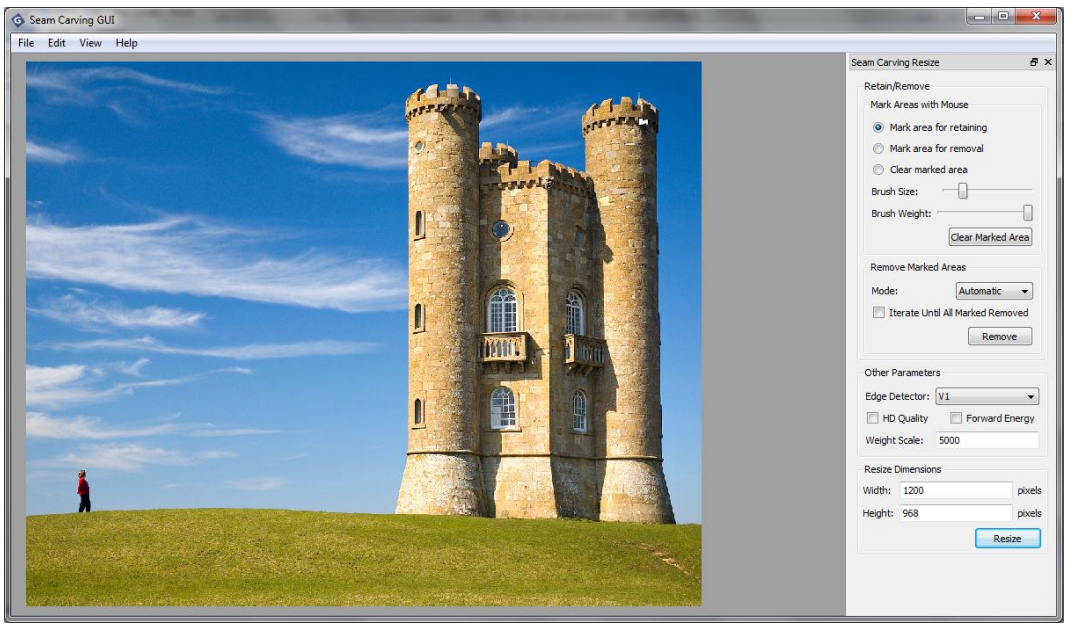

*Rys. 4.16. Okno programu Seam Carving GUI (bez wczytanego zdjęcia)*

Przykładowe działanie programu prześledźmy na wykorzystanym już w rozdziale 4.2.1. zdjęcia przedstawiającego człowieka i zamek, nazwa pliku to *brd.jpg*. Na początku wczytujemy obraz w oryginalnym rozmiarze, który wynosi 1428x968 pikseli. Ze względu na rozdzielczość większą niż rozdzielczość ekranu, w celu dobrej widoczności całego zdjęcia dwukrotnie zmniejszamy widok o 25 %. Efekt jest widoczny na Rysunku 4.17.




*Rys. 4.17. Wczytanie pliku brd.jpg*

Oryginalną szerokość obrazu zmniejszamy z 1428 na 1200 pikseli. Na Rysunku 4.18 jest przedstawiony wynik przeskalowania.




*Rys. 4.18. Plik brd.jpg po zmianie szerokości obrazu z 1428 na 1200 pikseli*

Spróbujmy też zmienić wysokość zdjęcia. Zatem w kolejnym kroku zmieniamy wysokości zdjęcia z 968 na 850. Rozmiary zmieniamy w stosunku do oryginalnego zdjęcia o rozdzielczości 1428 na 968 pikseli. Efekty przedstawia Rysunek 4.19.


*Rys. 4.19. Plik brd,jpg po zmianie wysokości obrazu z 968 na 850*

W obu przypadkach możemy zaobserwować, że zmiany rozmiarów obrazu nie wpłynęły znacząco na zmiany kluczowych elementów zdjęcia – człowieka oraz zamku. Wobec tego efekty jakie możemy uzyskać tym programem możemy uznać jako dobre. Działanie programu Seam Carving GUI wykorzystamy w dalszej analizie, porównując z efektami własnej implementacji algorytmu seam carving.

## 5. Implementacja własnego algorytmu

### *5.1. Założenia algorytmu*

W trzech poprzednich rozdziałach zostały opisane główne elementy wiążące całe zadanie w jedną całość. Elementami składającymi się na to są:

- rodzaj przetwarzanych sygnałów – obrazy rastrowe;
- funkcja wykorzystywana do analizy – falka Haara;
- sposób realizacji skalowania zdjęć – algorytm seam carving.

Wszystkie te elementy połączymy w środowisku Matlab (wersja R2011b) w jeden program, pozwalający na realizację skalowania zdjęć, a dokładnie:

- zmniejszania szerokość obrazu;
- zwiększania szerokość obrazu;
- zmniejszania wysokość obrazu;
- zwiększania wysokość obrazu.

Do przetestowania programu wybierzemy kilka obrazów rastrowych w formacie *jpg*. Dla ujednolicenia analizy, rozmiary początkowe wszystkich obrazów wyniosą 512 pikseli wysokości i 512 pikseli szerokości. Wyselekcjonowane obrazy zostaną poddane zmniejszeniu i zwiększeniu zarówno wysokości jak i szerokości. Zwiększenia i zmniejszenia rozmiarów będą dokonywane o tę samą wartość pikseli: 50 (ok. 10 % początkowego wymiaru), 100 (ok. 20 % początkowego wymiaru) i 150 (ok. 30 % początkowego wymiary).

Wyniki analizy działania własnego algorytmu zostaną porównane z wynikami działania programu Seam Carving GUI, a porównanie działań algorytmów zostanie opisane i poparte obrazami wynikowymi.

### *5.2. Opis działania*

Proces wykonywania skalowania składa się z kilku części:

I – wczytanie obrazu i wykrywanie jego rozmiarów;

II – wykonanie czarno-białej kopii;

III – wybór opcji skalowania;

IV – właściwe skalowanie;

V – wyświetlenie efektów skalowania.

Charakterystyka każdego z tych etapów:

I – wczytanie obrazu i wykrywanie jego rozmiarów.

Obraz o odpowiedniej nazwie zostaje skopiowany do nowopowstałej w ten sposób macierzy *M1* i wyświetlony. Za pomocą funkcji *size(M1)* pobieramy jego rozmiary (na potrzeby tej analizy każdy z obrazów ma rozmiary 512x512). Równocześnie mamy możliwość podania rozmiarów na które chcemy dany obraz przeskalować, z zastrzeżeniem, że muszą to być liczby parzyste.

II – wykonanie czarno-białej kopii.

Każdy z nowo-wczytanych obrazów zamieniamy na czarno-białą kopię. Ze względu na to, że Matlab po przekopiowaniu robi to automatycznie, wystarczy dwukrotne przekopiowanie – najpierw do macierzy pomocniczej *MX* a następnie z powrotem do macierzy *M1*. To właśnie dane z macierzy *M1* będą nam służyły do analizy.

III – wybór opcji skalowania.

W programie istnieje zmienna *opcja*, do której przypisujemy jedną z wartości: 1, 2, 3 lub 4, w zależności od tego jaki rodzaj skalowania wybieramy. Kolejne wartości odpowiadają:

1 – zmniejszeniu szerokości;

2 – zwiększeniu szerokości;

3 – zmniejszeniu wysokości;

4 – zwiększeniu wysokości.

IV – właściwe skalowanie.

W zależności od wybranej opcji, każde skalowanie ma bardzo zbliżony algorytm postępowania. W ogólnym zarysie kolejne kroki algorytmu skalowania to:

a – sprawdzenie czy nowy rozmiar jest taki sam z pożądanym – jeśli tak to pętla programu nie wykona się ani razu, w przeciwnym przypadku pętla będzie wykonywana aż do uzyskania równości;

b – obliczenie wartości poszczególnych pikseli z wykorzystaniem funkcji Haara (wzory 3.4-3.7) i przypisanie ich do macierzy *M2*;

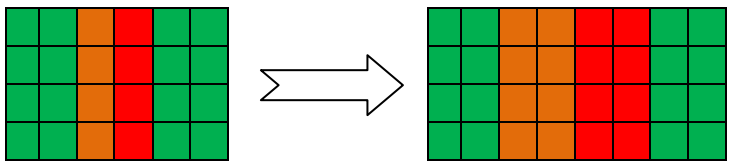
c – skopiowanie średniego poziomego lub pionowego gradientu w zależności od tego czy chcemy zmieniać szerokość czy wysokość to do macierzy pomocniczej *MG* kopiujemy odpowiednią część macierzy powstałej z wzorów 3.4-3.7;

d – wyznaczanie ścieżek i energii w pionie lub w poziomie – do wyznaczenia pionowych szwów wykorzystamy macierze pomocnicze *ScPn* i *EnPn*, a do poziomych *ScPz* i *EnPz*. W macierzach *ScPn* i *ScPz* wstawiamy współrzędne odpowiednio kolumny lub rzędu piksela, dla którego sumaryczne wartości energii są najmniejsze. Stosujemy zasadę z Rys. 4.3. Dla skrajnego górnego rzędu macierzy *ScPn* i skrajnej lewej kolumny macierzy *ScPz* macierzy wypełniamy wartościami 0. W macierzach *EnPn* i *EnPz* przechowujemy wartości całkowitych energii fragmentów szwów, zgodnie z zasadą z Rys. 4.3. wykorzystując kolejne współrzędne kolumn i wierszy z macierzy odpowiednio *ScPn* i *ScPz*;

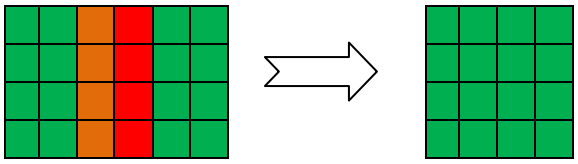
e – znalezienie najmniejszej sumy energii – ze skrajnego dolnego wiersza macierzy EnPn lub skrajnej prawej kolumny macierzy EnPz wyszukujemy najmniejszej energii aby równocześnie wyznaczyć początek szwu. W przypadku równych wartości algorytm wybiera skrajne szwy – naprzemiennie skrajnie lewy lub prawy (*EnPn*) lub naprzemiennie skrajnie górny lub dolny (*EnPz*). Następnie zapamiętywana jest współrzędna początku szwu;

f – wyznaczenie szwu – za pomocą ścieżek wyznaczonych w macierzy *ScPn* lub *ScPz* i sumy energii wyznaczonych w macierzy *EnPn* lub *EnPz* i początku wyznaczonym w podpunkcie d, wyznaczamy współrzędne punktów tworzących szew. Współrzędne zapamiętujemy w macierzy *SzPn* dla szwu pionowego lub *SzPz* dla szwu poziomego. Pamiętajmy o tym, że szew wyznaczamy z gradientu pionowego lub poziomego, który przechowujemy w macierzy *MG*. Rozmiary tej macierzy są dokładnie dwukrotnie mniejsze od rozmiarów macierzy, w której przechowujemy obraz oryginalny. W związku z tym każdy piksel szwu wyznaczony na podstawie gradientu reprezentuje 4 piksele na obrazie oryginalnym (w ułożeniu 2x2);

g – usunięcie lub dodanie pikseli do zdjęcia – w zależności od tego czy dany szew usuwamy lub dodajemy to stosujemy zasadę zobrazowaną na Rysunkach 5.1. i 5.2 Oba sposoby stosujemy zarówno do szerokości jak i wysokość. Na obu rysunkach szwy oznaczone są kolorami pomarańczowym i czerwonym a pozostałe piksele kolorem zielonym. Przy zwiększaniu rozmiaru zdjęcia piksele szwu są podwajane a podczas zmniejszania rozmiaru, cały szew jest po prostu usuwany;



*Rys. 5.1. Zasada dodawania pikseli w miejscu szwu (kolor pomarańczowy i czerwony)*



*Rys. 5.2. Usuwanie szwu z obrazu (kolor pomarańczowy i czerwony)*

h – przypisanie efektów działania ponownie do początkowej macierzy *M1*;

i – zmiana szerokości lub wysokości obrazu – w zależności od tego czy zwiększyliśmy lub zmniejszyliśmy, wysokość lub szerokość to zmianie ulega zmienna, w której dany rozmiar przechowujemy. Następnie powracamy do punktu a.

V – wyświetlenie efektów skalowania

Działanie skalowania możemy zaobserwować na utworzonym i wyświetlonym obrazie. W związku z tym, że domyślnie zostaje wyświetlony również obraz początkowy, możemy porównać zmiany jakie zaszły poprzez stosowanie algorytmu.

# 6. Doświadczenia

## *6.1. Dobór materiału badawczego*

Do sprawdzenia działania własnego algorytmu zostało wybranych 6 obrazów przedstawiającej ludzi, przedmioty oraz zjawiska. Charakteryzują się różną szczegółowością, kolorystyką i rozmieszczeniem poszczególnych elementów na nich przedstawionych. Różnorodność w doborze materiału badawczego pozwoli na sprawdzenie działania własnego algorytmu w szerszym zakresie i pozwoli na pełniejsze wyszukanie wad i zalet w jego stosowaniu. Dla ujednolicenia analizy wszystkie wykorzystane obrazy są w jednakowym rozmiarze 512x512 pikseli. Każdy z obrazów przeznaczony do analizy został krótko scharakteryzowany.

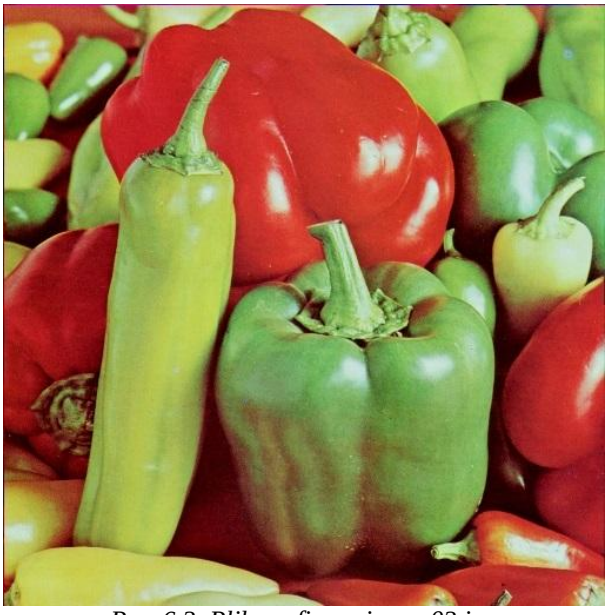
Obraz z Rysunku 6.1. Nazwa pliku – *image01.jpg*. Zdjęcie przedstawia pole z charakterystycznymi żółtymi kwiatami rzepaku oraz niebo łagodnie przechodzące z barwy jasno-błękitnej do granatowej. Możemy tu wydzielić 2 obszary – jeden o dużej zmienności obiektów oraz drugi o małej zmienności.

*Rys. 6.1. Plik graficzny image01.jpg*

Obraz z Rysunku 6.2. Nazwa pliku – *image02.jpg*. Zdjęcie przedstawia drogę znajdującą się w dolinie pagórkowatego terenu, otoczoną przez liczną roślinność. Występuje bardzo dużo szczegółów.

*Rys. 6.2. Plik graficzny image02.jpg*

Obraz z Rysunku 6.3. Nazwa pliku – *image03.jpg*. Zdjęcie przedstawia różnokolorową paprykę. Zdjęcie w określonych obszarach ma więcej szczegółów, w innych zaś możemy zaobserwować obszary niemal jednolite. Występują światło-cienie.

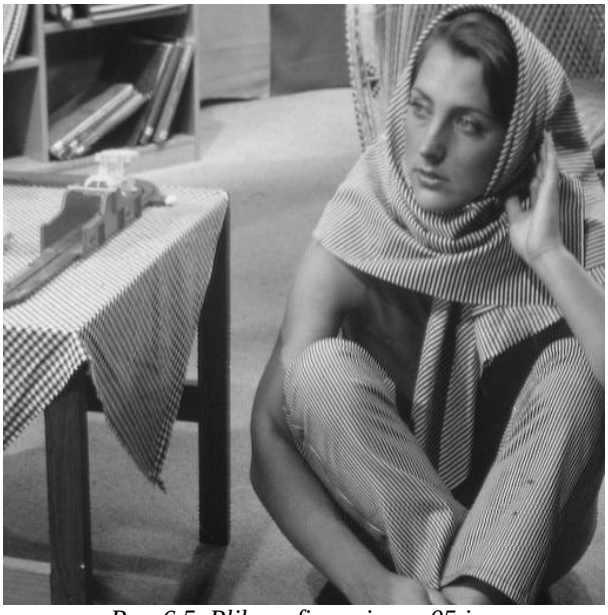


*Rys. 6.3. Plik graficzny image03.jpg*

Obraz z Rysunku 6.4. Nazwa pliku – *image04.jpg*. Zdjęcie przedstawia fragment kamiennego muru, składającego się z wielu różnej wielkości elementów. Zdjęcie charakteryzuje się dużo szczegółowością i jednocześnie monotonną kolorystyką.

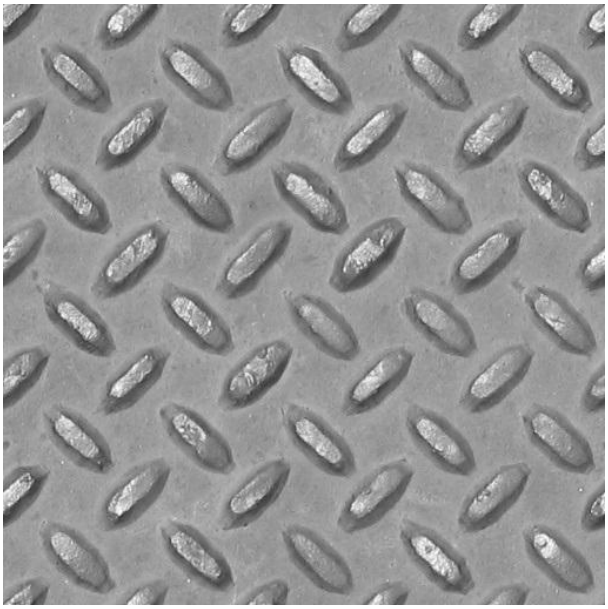
*Rys. 6.4. Plik graficzny image04.jpg*

Obraz z Rysunku 6.5. Nazwa pliku – *image05.jpg*. Zdjęcie przedstawia kobietę znajdującą się w zamkniętym pomieszczeniu. Na pierwszym planie obok kobiety znajduje się stolik, a innymi ważnymi elementami zdjęcia jest chusta znajdująca się na głowie kobiety oraz nakrycie stolika. Oba przedmioty mają określone, charakterystyczne wzornictwo.



*Rys. 6.5. Plik graficzny image05.jpg*

Obraz z Rysunku 6.6. Nazwa pliku – *image06.jpg*. Zdjęcie przedstawia metalową płytę, zawierającą charakterystyczne, prostopadłe względem siebie wypustki.

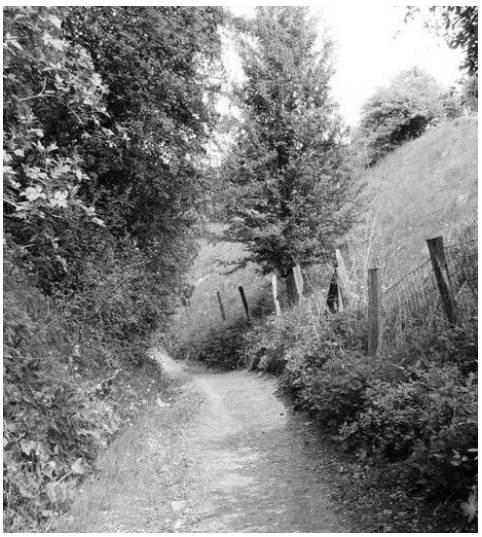


*Rys. 6.6. Plik graficzny image06.jpg*

Do analizy wszystkie wymienione obrazy zostały przekonwertowane do postaci obrazów czarno-białych.

### *6.2. Zmniejszenie szerokości obrazów*

Analizy obrazów dokonano po zmniejszeniu ich szerokości. Szerokość obrazów została zmieniana trzykrotnie w stosunku do oryginalnego obrazu. Zmiany wyniosły: 50, 100 i 150 pikseli. Skrót w. a. oznacza zdjęcie wynikowe własnego algorytmu a skrót S. C. GUI zdjęcie wynikowe z programu Seam Carving GUI.

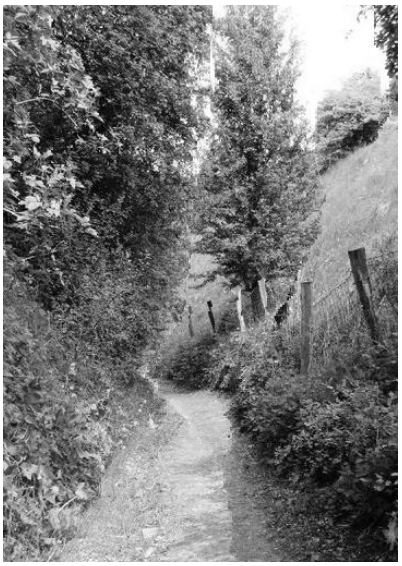


*Rys. 6.7. Plik image02.jpg po zmianie szerokości z 512 na 462 pikseli (w. a.)*

*Rys. 6.8. Plik image02.jpg po zmianie szerokości z 512 na 462 pikseli (S.C. GUI)*

*Rys. 6.9. Plik image02.jpg po zmianie szerokości z 512 na 412 pikseli (w. a.)*

*. Rys. 6.10. Plik image02.jpg po zmianie szerokości z 512 na 412 pikseli (S.C. GUI)*



*Rys. 6.11. Plik image02.jpg po zmianie szerokości z 512 na 362 pikseli (w. a.)*

*Rys. 6.12. Plik image02.jpg po zmianie szerokości z 512 na 362 pikseli (S.C. GUI)*

*Rys. 6.13. Plik image05.jpg po zmianie szerokości z 512 na 462 pikseli (w. a.)*

*Rys. 6.14. Plik image05.jpg po zmianie szerokości z 512 na 462 pikseli (S.C. GUI)*

*Rys. 6.15. Plik image05.jpg po zmianie szerokości z 512 na 412 pikseli (w. a.)*

*Rys. 6.16. Plik image05.jpg po zmianie szerokości z 512 na 412 pikseli (S.C. GUI)*

*Rys. 6.17. Plik image05.jpg po zmianie szerokości z 512 na 362 pikseli (w. a.)*

*Rys. 6.18. Plik image05.jpg po zmianie szerokości z 512 na 362 pikseli (S.C. GUI)*

Do zobrazowania działania własnego algorytmu posłużyły obrazy *image02.jpg* oraz *image05.jpg*.

Pierwszy z wymienionych obrazów przedstawia ścieżkę znajdującą się w otoczeniu pagórków. Charakterystyczne jest liczna roślinność zgromadzona w bezpośrednim sąsiedztwie ścieżki. Rysunki 6.7. oraz 6.8. dają zbliżone efekty. Nieznaczne różnice możemy zauważyć na Rysunkach 6.9. i 6.10. Widzimy, że na Rysunku 6.9. zostało zabrane nieco więcej obszarów z lewej strony niż ma to miejsce na Rysunku 6.10. Generalnie jednak nadmiarowy obszar został usunięty w miarę równomiernie przez oba algorytmy. Tak jak chociażby ścieżka znajdująca się na pierwszym planie jest niemal identyczna. Nieco inaczej jest w górnej części obu obrazów. Różnice te uwidaczniają się jeszcze bardziej na Rysunkach

6.11. i 6.12. Drzewo znajdujące się w centralnej części zdjęcia wygląda identycznie w wyniku działania obu algorytmów. Własny algorytm najmniejszą energię znajdował w lewej części zdjęcia. Wynik tego możemy zobaczyć na Rysunku 6.11. gdzie przestrzeń widoczna w prawym górnym rogu pozostała mniej zmieniona w stosunku do oryginalnego obrazu niż zrobił to algorytm z Seam Carving GUI (Rysunek 6.12). Nierównomierne znajdowanie energii może powodować mniejsze lub większe zniekształcenia obrazu. W tym przypadku, w związku z bardzo dużą szczegółowością sceny jest to mało widoczne a efekt wizualny możemy uznać za dobry.

Drugi z wykorzystanych obrazów to *image05.jpg*, przedstawiający kobietę w zamkniętym pomieszczeniu. W przypadku tego obrazu możemy już w pierwszym kroku (Rysunki 6.13. i 6.14.) zauważyć różnicę w usuwaniu szwów zastosowanych przez oba algorytmy. W przypadku własnego algorytmu nie została „ochroniona" prawa ręka kobiety, przez co już w pierwszym kroku uzyskaliśmy niepożądane zniekształcenie. Wykrycie i usunięcie szwu nastąpiło w miejscu granicy pomiędzy częścią ręki oświetlonej a częścią ręki będącej w cieniu. Bardzo niewielkim zmianom uległ za to kształt stołu znajdującego się na pierwszym planie. W przypadku Rysunku 6.14. jest dokładnie odwrotnie tj. ręka kobiety została maksymalnie pominięta przy usuwaniu pikseli, za to zniekształceniu uległa noga stołu. Kolejne obrazy przedstawiające efekt własnego algorytmu (Rysunki 6.15. i 6.17.) nie powodują dużych zmian w geometrii osoby (poza lewą dłonią kobiety) i obiektów na nim się znajdujących w stosunku do pierwszego przekształcenia (Rysunek 6.13). Usunięto obszary w miarę równomiernie na całej szerokości obrazu. Inaczej jest jeśli chodzi o efekty programu Seam Carving GUI. W kolejnych krokach zmniejszania szerokości (Rysunki 6.16 i 6.18). W tym przypadku w znacznie mniejszym stopniu w porównaniu do oryginału został zmieniony wizerunek kobiety. Zdecydowanie bardziej za to został zniekształcony stolik znajdujący się w lewej stronie obrazu. Zarówno kształt nogi jak i kantu stolika zostały bardzo mocno zniekształcone w stosunku do oryginału. Na ostatnich zdjęciach uwidaczniają się różnice pomiędzy działaniem obu algorytmów. W dalszym etapie zwężania, własny algorytm szwy znajdował w prawej części obrazu, przez co wizerunek kobiety jest zniekształcony (Rys. 6.17.), natomiast stolik pozostał niewiele zmieniony w stosunku do oryginału. Dokładnie odwrotnie zachował się drugi z algorytmów. Istotną sprawą są również zmiany zachodzące na twarzy kobiety. Własny algorytm w kolejnych etapach wyszukiwał szwy przechodzące przez twarz kobiety a po ich usunięciu twarz ulegała dużym zniekształceniom. Seam Carving GUI delikatnie zwężał twarz więc zmiana jest słabo widoczna W tym przypadku własny algorytm poradził sobie gorzej niż algorytm z Seam Carving GUI.

### *6.3. Zwiększenie szerokości obrazów*

Analizy obrazów dokonano po zwiększeniu ich szerokości. Szerokość obrazów została zmieniana trzykrotnie w stosunku do oryginalnego obrazu. Zmiany wyniosły: 50, 100 i 150 pikseli.

*Rys. 6.19. Plik image03.jpg po zmianie szerokości z 512 na 562 pikseli (w. a.)*

*Rys. 6.20. Plik image03.jpg po zmianie szerokości z 512 na 562 pikseli (S.C. GUI)*

*Rys. 6.21. Plik image03.jpg po zmianie szerokości z 512 na 612 pikseli (w. a.)*

*Rys. 6.22. Plik image03.jpg po zmianie szerokości z 512 na 612 pikseli (S.C. GUI)*

*Rys. 6.23. Plik image03.jpg po zmianie szerokości z 512 na 662 pikseli (w. a.)*

*Rys. 6.24. Plik image03.jpg po zmianie szerokości z 512 na 662 pikseli (S.C. GUI)*

Do zobrazowania działania własnego algorytmu posłużył obraz *image03.jpg*. Obraz ten przedstawia różnej wielkości i różnokolorowe (po zamianie obrazu mówimy o różnych odcieniach szarości) warzywa papryki ułożone w niełądną stertę. Porównanie zwiększania szerokości uwypukla największą, jak się okazuje bolączkę własnego algorytmu – wyszukiwanie szwów ciągle w tych samych miejscach. Różnice możemy zauważyć już na pierwszych obrazach (Rysunki 6.19. i 6.20.). Podczas gdy program Seam Carving GUI rozszerza obraz równomiernie (Rysunek 6.20) to własny algorytm po znalezieniu szwu wciąż zwiększa w tym miejscu ilość pikseli. Bierze się to stąd, że algorytm ciągle widzi najmniejszą energię tylko w miejscu pierwotnie wyznaczonych szwów. Powoduje to rozciągnięcie tych samych pikseli na długość odpowiadającą ilość pikseli o którą zwiększyliśmy szerokość obrazu. Zatem dodanie jeszcze większej ilości pikseli (Rysunki 6.21. i 6.23.) jeszcze bardziej uwydatnia to zjawisko. Równocześnie w pozostałej części obrazu nie zachodzą żadne zmiany. Seam Carving GUI na zróżnicowanym obrazie dodaje piksele równomiernie co powoduje, że obiekty wciąż wyglądają naturalnie (Rysunki 6.22. i 6.24.).

W tym momencie uwidoczniło się ułatwienie w implementacji wykrywania i dodawania szwów. Wprowadzając w przyszłości poprawki należy zmodyfikować proces tak aby w miarę możliwości wyszukiwać szwów na całej szerokości obrazu. W tej chwili po wykryciu tych o najmniejszych sumach energii ciągle skupia się przemiennie na skrajnych szwach wobec czego dodawanie pikseli występuje ciągle w tym samym obszarze.

### *6.4. Zmniejszenie wysokości obrazów*

Analizy obrazów dokonano po zmniejszeniu ich wysokości. Wysokość obrazów została zmieniana trzykrotnie w stosunku do oryginalnego obrazu. Zmiany wyniosły: 50, 100 i 150 pikseli.

*Rys. 6.25. Plik image01.jpg po zmianie wysokości z 512 na 462 pikseli (w. a.)*

*Rys. 6.26. Plik image01.jpg po zmianie wysokości z 512 na 462 pikseli (S.C. GUI)*

*Rys. 6.27. Plik image01.jpg po zmianie wysokości z 512 na 412 pikseli (w. a.)*

*Rys. 6.28. Plik image01.jpg po zmianie wysokości z 512 na 412 pikseli (S.C. GUI)*

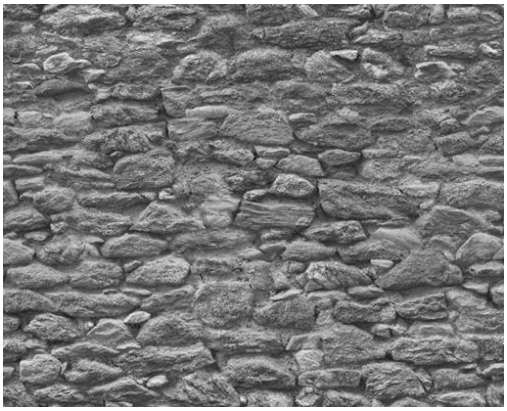
*Rys. 6.29. Plik image01.jpg po zmianie wysokości z 512 na 362 pikseli (w. a.)*

*Rys. 6.30. Plik image01.jpg po zmianie wysokości z 512 na 362 pikseli (S.C. GUI)*

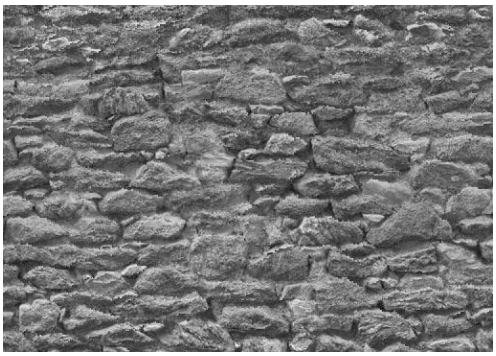
*Rys. 6.31. Plik image04.jpg po zmianie wysokości z 512 na 462 pikseli (w. a.)*

*Rys, 6.32. Plik image04.jpg po zmianie wysokości z 512 na 462 pikseli (S.C. GUI)*

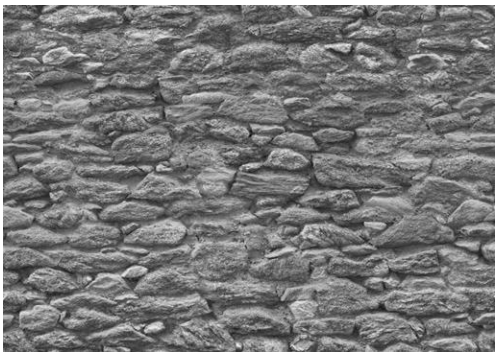
*Rys. 6.33. Plik image04.jpg po zmianie wysokości z 512 na 412 pikseli (w. a.)*



*Rys. 6.34. Plik image04.jpg po zmianie wysokości z 512 na 412 pikseli (S.C. GUI)*



*Rys. 6.35. Plik image04.jpg po zmianie wysokości z 512 na 362 pikseli (w. a.)*



*Rys. 6.36. Plik image04.jpg po zmianie wysokości z 512 na 362 pikseli (S.C. GUI)*

Do zobrazowania działania własnego algorytmu posłużyły obrazy *image01.jpg* oraz *image04.jpg*.

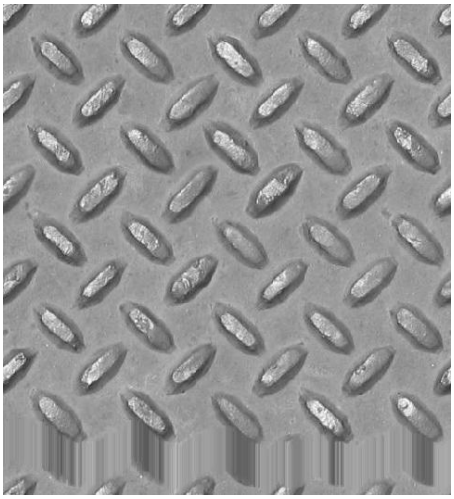
Plik *image01.jpg* przedstawia część pola rzepaku na pierwszym planie a w dalszym skraj lasu oraz niebo. Charakterystyczną cechą tego obrazu jest wyraźny podział na dwie różniące się części. Dolna część obrazu zawiera liczne bardzo dużo szczegółów, natomiast w górnej części znajduje się obszar, który jest bardzo monotonny. Już od początku zmniejszania wysokości obrazów możemy zauważyć, że oba algorytmy słusznie skupiają się na usuwaniu obszarów nieba (Rysunki 6.25 i 6.26.), i początkowo robią to niemal identycznie. Większe różnice pomiędzy algorytmami możemy zauważyć na kolejnych zmniejszeniach (Rysunki 6.27. i 6.28.). Seam Carving GUI po raz kolejny (Rysunek 6.28.) usuwa obszary w miarę równomiernie i to widoczne jest na obrazie. Przejście od jasnego koloru nieba znajdującego się na horyzoncie do ciemnego koloru znajdującego się na górnej krawędzi obrazu, jest łagodne i równomierne. W przypadku własnego algorytmu (Rysunek 6.27.) większe usunięcie pikseli nastąpiło w ten sposób, że na niebie pojawiły się „skoki" pomiędzy kolejnymi, różnymi odcieniami koloru nieba. Na kolejnym zmniejszaniu wysokości obrazu własnym algorytmem (Rysunek 6.29.) zjawisko to uwidacznia się jeszcze bardziej. Na kolejnych

obrazach występuje coraz więcej krzywych, które rozgraniczają obszary o różnych odcieniach jasności. Jest to szczególnie widoczne w prawej, górnej części obrazu. Sprawia to wrażenie takie, jakby niebo miało różnokolorowe „warstwy", co na pewno nie jest pożądanym efektem. Znacznie lepiej z tym obrazem poradził sobie program Seam Carving GUI, dzięki równomiernemu usuwaniu pikseli z obrazu (Rysunki 6.28. i 6.30). Przy bardzo dużym usunięciu obszaru nieba efekt wizualny jest bardzo dobry i nie ma wyraźnych „skoków" pomiędzy różnymi odcieniami koloru nieba.

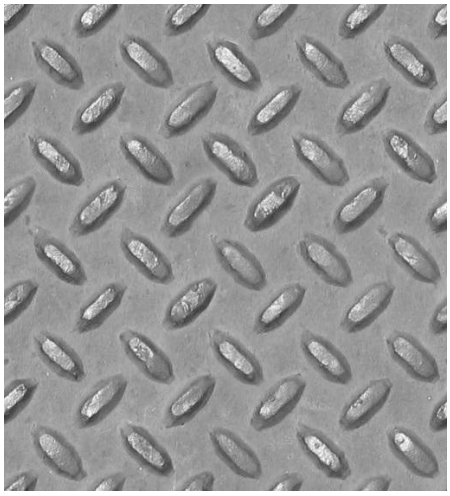
Drugi z wykorzystanych obrazów to *image04.jpg*. Przedstawia on fragment kamiennej ściany. Z racji przedstawionej sceny, zmniejszenie wysokości dla obu algorytmów wydaje się dość łatwym zadaniem. Analizę monotonnie wyglądającego muru najlepiej oprzeć na porównaniu pozycji kilku określonych kamieni. Charakterystycznym obiektem jest kamień znajdujący się niemal w samym środku sceny. Na wszystkich obrazach prezentuje się on niemal identycznie, razem z sąsiednimi kamieniami. Zatem dla obu algorytmów centralna część zdjęć została zmieniona w bardzo zbliżonym stopniu. Różnice możemy zauważyć w lewej górnej części obrazów, gdzie znajduje się inny charakterystyczny kamień. Seam Carving GUI w bardzo małym stopniu zmieniał jego wygląd i położenie (Rysunki 6.32., 6.34., 6.36.). Nieco inaczej dzieje się w przypadku własnego algorytmu. Początkowo co prawda możemy zauważyć, że jest on bardzo podobny (Rysunek 6.31.) w stosunku do tego co wyznaczył Seam Carving GUI (Rysunek 6.32.). W dalszych etapach (Rysunki 6.33. i 6.35.) możemy zauważyć, że zostaje on znacznie spłaszczony względem początkowego. Pozwala to stwierdzić, że algorytm własny w górnej części obrazu wyszukał więcej szwów niż algorytm z Seam Carving GUI. Ze względu jednak na rodzaj i położenie obiektów przedstawionych na zdjęciu zmiany te nie są aż tak istotne. Zbyt mało jest na tym obrazie elementów charakterystycznych, które zdecydowanie wyróżniają się na tle innych. Wobec tego ocenę skalowania obrazu traktujemy wobec całokształtu jaki zaprezentowały oba algorytmy. W obu przypadkach efekty są bardzo zbliżone i dobre.

### *6.5. Zwiększenie wysokości obrazów*

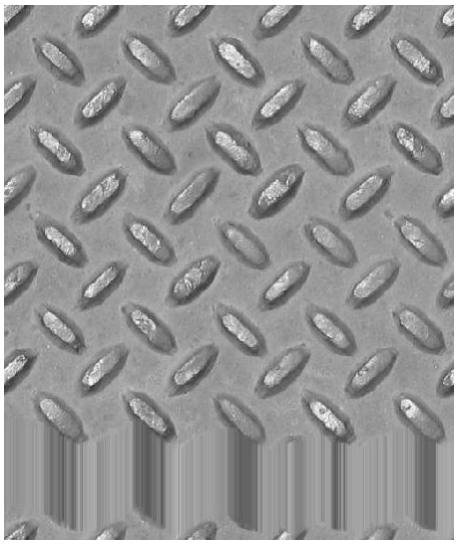
Analizy obrazów dokonano po zwiększeniu ich wysokości. Wysokość obrazów została zmieniana trzykrotnie w stosunku do oryginalnego obrazu. Zmiany wyniosły: 50, 100 i 150 pikseli.



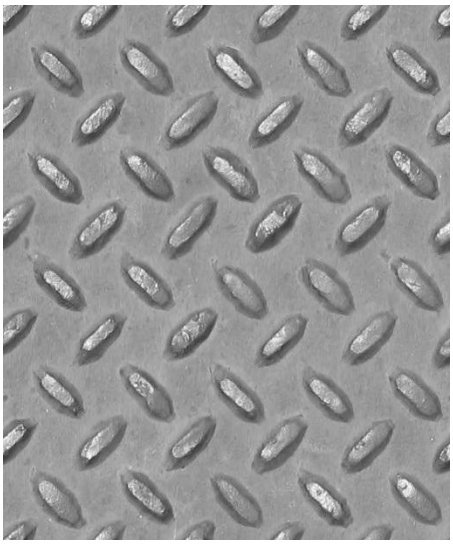
*Rys. 6.37. Plik image06.jpg po zmianie wysokości z 512 na 562 pikseli (w. a.)*



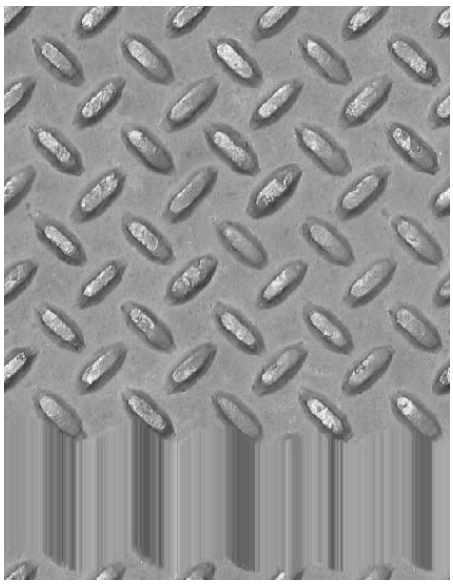
*Rys. 6.38. Plik image06.jpg po zmianie wysokości z 512 na 562 pikseli (S.C. GUI)*



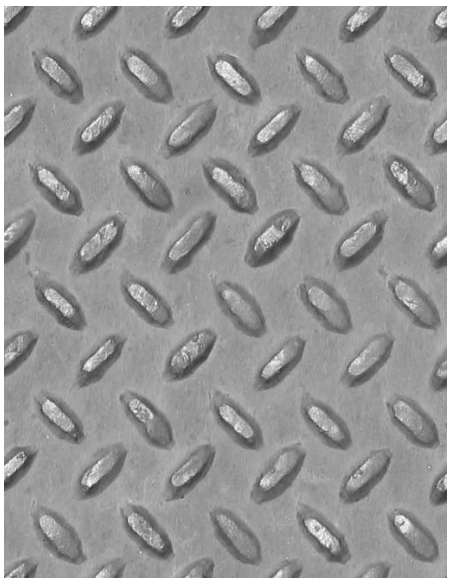
*Rys. 6.39. Plik image06.jpg po zmianie wysokości z 512 na 612 pikseli (w. a.)*



*Rys. 6.40. Plik image06.jpg po zmianie wysokości z 512 na 612 pikseli (S.C. GUI)*



*Rys. 6.41. Plik image06.jpg po zmianie wysokości z 512 na 662 pikseli (w. a.)*



*Rys, 6.42. Plik image06.jpg po zmianie wysokości z 512 na 662 pikseli (S.C. GUI)*

Do zobrazowania działania własnego algorytmu posłużył obraz *image03.jpg*. Obraz ten przedstawia szarą metalową płytę z charakterystycznym wzorem. Na wzór ten składają się małe, podłużne metalowe elementy leżące prostopadle względem siebie. Już przy najmniejszym powiększeniu zdjęcia (Rysunki 6.37. i 6.38.) uwidoczniła się różnica pomiędzy własnym algorytmem a działaniem programu Seam Carving GUI. Przy powiększaniu własnym algorytmem pojawił się ten sam problem jaki został zauważony w podrozdziale 6.3 gdzie mieliśmy do czynienia ze zwiększaniem szerokości zdjęcia. Ponownie algorytm po znalezieniu szwu powiela go i nie zajmuje się szwami w innym obszarze zdjęcia. Jest to wypaczone przez założenie algorytmu, które mówi, że w przypadku znalezienia kilku szwów, pod uwagę bierze ciągle skrajne z obu stron. W tym przypadku zostało to sprowadzone do jednego szwu, zatem został znaleziony tylko jeden szew o minimalnej energii. Zwiększenie obszaru następuje poprzez powielenie pikseli szwu i w tym momencie powstaje nam „wzmocnienie" małej energii obrazu w poziomie. Wobec tego wyjście poza tym szew i wyszukiwanie dalszych jest już właściwie niemożliwe. Oczywiście, zmiany energii w tych miejscach są jak najbardziej słusznie przeliczane jako najmniejsze. Jednak poza obliczeniami istotny w grafice jest efekt wizualny. Ciągłe powielanie tych samych pikseli powoduje, że na zdjęciach pojawiają się nienaturalne obszary kopiowanych pikseli (Rysunki 6.39. i 6.41.). Seam Carving GUI w tej kwestii ponownie poradził sobie lepiej i równomiernie wydłużając zdjęcie nie spowodował żadnych widocznych zniekształceń obrazu.

## 7. Podsumowanie

Zadaniem jakie przyświecało tworzenie tego projektu to własne implementacja algorytmu seam carving w oparciu o transformatę falkową. W tym celu dobrano falkę, na podstawie której realizowana była kluczowa kwestia algorytmu czyli wyszukiwanie szwów. Wykorzystano falkę Haara, dla której wzory na wartości poszczególnych pikseli sąsiadujących ze sobą były przystępne pod względem trudności do zaimplementowania. Przy okazji daje tez dobre efekty jeśli chodzi o wyszukiwanie szwów. Następnie po wyznaczeniu szwu określone piksele zostawały przeznaczone do usunięcia lub dodania. Tutaj mieliśmy do czynienia z dużym rozrzutem jeśli chodzi o skuteczność algorytmu. Jeśli chodzi o usuwanie pikseli to zdarzało się, że w większym stopniu były to obszary z jednej strony zdjęcia. Było to efektem ułatwienia w implementacji a dokładniej w przypadku znajdowania szwów o tej samej sumie energii, preferowany był szew z określonej strony zdjęcia. Mimo to zdarzały się wśród analizowanych takie przypadki, w których zastosowanie własnego algorytmu dawało dobre efekty. Znacznie gorzej działo się w przypadku gdy należało dodać piksele. Wówczas owe ułatwienie w implementacji było zgubne. Działo się tak dlatego, że po znalezienie szwu, piksele były powielane z pikseli szwu. Wobec tego w miejscu szwu powiększył się obszar o najmniejszej energii szwu na zdjęciu. Za każdym kolejnym przeszukiwaniem, algorytm ciągle natrafiał na to samo miejsce, wobec czego powielane były wciąż te same piksele. Doprowadzało to do sytuacji gdzie następowało „rozciągnięcie” zdjęcia w określonym fragmencie. Powodowało to bardzo duże zniekształcenia i zakłamania w stosunku do obrazu oryginalnego. Porównując to do efektów prezentowanych przez program Seam Carving GUI, algorytm własnej implementacji wygląda zdecydowanie słabiej. O ile w przypadku zmniejszania rozmiarów zdjęcia mógł on w niektórych przypadkach również dać satysfakcjonujące rezultaty, to niestety w kwestii dodawania obszarów na zdjęciu własny algorytm się po prostu nie sprawdził.

Efekty działania własnego algorytmu można polepszyć odpowiednio zmieniając fragment kodu odpowiedzialny za znajdowanie szwów. O ile faktycznie zawsze znajdował ten o najmniejszej wadze, to równocześnie mógł zostać pominięty szew o dokładniej takiej samej sumie energii, znajdujący się w innym fragmencie zdjęcia. Pomysłem na zmianę sposobu wyszukiwania szwów może być chociażby położenie większego nacisku na lepsze określenie sposobu usuwania i dodawania pikseli.

W ogólnym rozrachunku funkcja Haara okazała się bardzo przydatna w algorytmie skalowania seam carving. Dzięki niej możliwe było dokładne wyznaczanie obszarów

przeznaczonych do usunięcia lub dodania. Nie mniej jednak, w przetwarzaniu obrazów oprócz wyniku analitycznego ważny jest też efekt wizualny. Ten, na tym etapie analizy nie okazał się zbyt konkurencyjny dla przykładowej, powszechnie używanej aplikacji Seam Carving GUI. Program ten, nastawiony ściśle swoim działaniem na stosowanie w swoim działaniu algorytmu seam carving, był zdecydowanie lepszym rozwiązaniem niż własny algorytm. To skłania do znalezienia sposobów na polepszenie własnego algorytmu na tyle ile to możliwe, aby inteligentne skalowanie, jakie reprezentuje metoda seam carving, było możliwe.

## 8. Spis rysunków

## 9. Bibliografia


[1] Hennel J., Olejniczak Z., *Jak zrozumieć falki – podstawy falkowej analizy sygnałów*, ZamKor, Kraków 2010

[2] Zieliński T. P., *Cyfrowe przetwarzanie sygnałów – od teorii do zastosowań*, Wydawnictwa Komunikacji i Łączności, Warszawa 2005

[3] Mokrzycki W,. *Wprowadzenie do przetwarzania informacji – II. Dyskretyzacja obrazu, operacje pikslowe, morfologiczne i przekształcenia obrazowe*, Akademicka Oficyna Wydawnicza Exit, Warszawa 2012

[4] Misiti M., Misiti Y., Oppenheim G., Poggi J., *Wavelet Toolbox for use with Matlab*

[5] Białasiewicz J., *Falki I aproksymacje*, Wydawnictwo Naukowo-Techniczne, Warszawa 2004

[6] Hsi-Chin Hsin, Tze-Yun Sung, Chin-Wei Su, *A Fast wavelet-based seam carving algorithm for image resizign*, Asian Journal of Computer and Information Systems, tom 2- wydanie 5, Październik 2014

[7] Jong-Woo Han, Kang-Sun Choi, Tae-Shick Wang, Sung-Hyun Cheon, Sung-Jea Ko, *Improved seam carving using a modified energy function based on wavelet decomposition*, The 13th IEEE International Symposium on Consumer Electronics, 25-28 Maja 2009

[8] Weiming Dong, Ning Zhou, Jean-Claude Paul, Xiaopeng Zhang, *Optimized image resizing using seam carving and scaling*

[9] http://www.jpeg.org/jpeg2000/index.html

[10] http://homepage.math.uiowa.edu/~jorgen/Haar.html

[11] http://en.wikipedia.org/wiki/Seam_carving

[12] http://helpx.adobe.com/photoshop/using/content-aware-scaling.html