

AGH

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ FIZYKI I INFORMATYKI STOSOWANEJ

Dokumentacja techniczna

Ukryte obrazy nr 47

Autor: Joanna Błaszka, Radosław Grabiec, Dawid Justyna
Kierunek studiów: Fizyka Medyczna
Przedmiot: Podstawy Grafiki Komputerowej

Kraków, 2022

Spis treści

1	Opis projektu	3
2	Założenia wstępne.	3
2.1	Problem	3
2.2	Algorytmy	3
2.3	Sprawdzenie poprawności działania algorytmów	4
3	Podział obowiązków między autorów.	4
3.1	Joanna Błaszka	4
3.2	Radosław Grabiec	4
3.3	Dawid Justyna	4
4	Analiza czasowa.	4
5	Realizacja zadania.	5
5.1	Użyte biblioteki.	5
6	Obsługa programu.	5
6.1	Interfejs użytkownika.	5
6.2	Specyfikacja danych wejściowych i wyjściowych.	6
6.3	Przycisk "Code"	6
6.4	Przycisk "Decode A"	6
6.5	Przycisk "Decode B"	6
6.6	Przycisk "Decode A + B"	7
7	Potrzebne algorytmy.	7
7.1	Algorytm kodowania metodą A.	7
7.2	Algorytm kodowania metodą B.	7
7.3	Kodowanie metodą mieszaną.	8
8	Kodowanie.	8
8.1	Main	8
8.2	GUI i GUIMyFrame1	8
8.3	ModalDialogue	9
8.4	MethodA	9
8.5	MethodB	9
9	Testowanie programu.	9
10	Podsumowanie.	13

1 Opis projektu

Zadaniem projektu było zaszyfrowanie informacji w obrazach, wykorzystując ich zmodyfikowane dane, a także odszyfrowywanie informacji z zakodowanych obrazów. Zadanie mieliśmy wykonać dwoma metodami.

- Metoda A - steganograficzna

Metoda ta polega na zwiększaniu wartości pikseli o 1 w miejscu, w którym mamy zaszyfrowaną wiadomość. Przy odkodowaniu, potrzebujemy obrazu wzorcowego, który porównujemy z obrazem zmodyfikowanym, a następnie zgodnie z kluczem, z którym kodowaliśmy porównujemy oba obrazy ze sobą.

- Metoda B - kryptograficzna

Ta metoda polega na stworzeniu dwóch obrazów dwa razy większych i zakodowaniu ich następującymi cegiełkami:



Rysunek 1: Klocki użyte w kodowaniu

Wiadomości są kodowane na czarno-biało. Gdy trafiamy na piksel biały losujemy jedną z cegiełek i dajemy ją do obu obrazków, natomiast, gdy trafiamy na piksel czarny do jednego z obrazków losujemy cegiełkę, a do drugiego dajemy tą co została. Przy odkodowywaniu nakładamy na siebie obrazy, gdy pojawiają się różnicę zamalowujemy ten piksel.

2 Założenia wstępne.

2.1 Problem

Napisanie jak najlepiej zoptymalizowanych algorytmów szyfrujących obraz (opisane w punkcie 5), oraz stworzenie dobrze działającej struktury opierającej się na współpracy różnych okien dialogowych.

2.2 Algorytmy

Algorytmy przygotowane jako osobne funkcję dla każdej metody, opierające się głównie na metodzie `getData()`, która daje dostęp do każdego kanału(R,G,B) pixela w danym obrazie, dzięki czemu mogliśmy bezproblemowo modyfikować każdy z nich.

2.3 Sprawdzenie poprawności działania algorytmów

Poprawność wykonanych algorytmów sprawdziliśmy, poprzez dodanie do programu przykładowych obrazów, zaszyfrowaniu ich, a następnie odszyfrowaniu. W momencie kiedy wynik wyjściowy operacji, był taki sam jak wyjściowy, wiedzieliśmy że algorytmy dobrze ze sobą współgrają.

3 Podział obowiązków między autorów.

3.1 Joanna Blaszka

Przygotowanie interfejsu użytkownika, przygotowanie obsługi błędów, napisanie funkcji ładujących obrazy oraz do zapisywania wyników. Implementacja algorytmów, stworzenie klas, przygotowanie sposobu na łączenie obu algorytmów. Optymalizowanie kodu. Przygotowanie dokumentacji.

3.2 Radosław Grabiec

Przygotowanie algorytmu, który umożliwił zaszyfrowanie obrazu metodą kryptograficzną oraz sporządzenie algorytmu odszyfrowania tej metody.

3.3 Dawid Justyna

Przygotowanie algorytmu, który umożliwił zaszyfrowanie obrazu metodą steganograficzną oraz sporządzenie algorytmu odszyfrowania tej metody, pomoc przy metodzie kryptograficznej, przygotowanie sposobu na łączenie obu algorytmów oraz pomoc przy optymalizacji kodu. Implementacja algorytmów. Przygotowanie dokumentacji.

4 Analiza czasowa.

Nie jesteśmy w stanie określić dokładnie, ile zajęły nam poszczególne zadania, jednak było to około:

- Przygotowanie interfejsu - 4 dni
- Napisanie algorytmów - 1 dzień
- Implementacja algorytmów - 7 dni
- Dodanie metody mieszanej - 2 dni
- Optymalizacja kodu - 3 dni
- Obsługa błędów - 0.5 dnia
- Przygotowanie dokumentacji - 3 dni

5 Realizacja zadania.

5.1 Użyte biblioteki.

Program został stworzony w C++, do jego stworzenia wykorzystaliśmy bibliotekę wxWidgets, której obsługę poznaliśmy dobrze na zajęciach laboratoryjnych.

6 Obsługa programu.

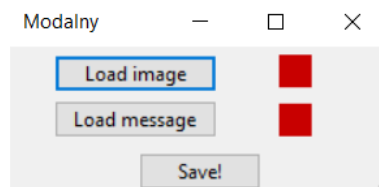
6.1 Interfejs użytkownika.

Po uruchomieniu programu pokazuje się okno przedstawione na rysunku (2).

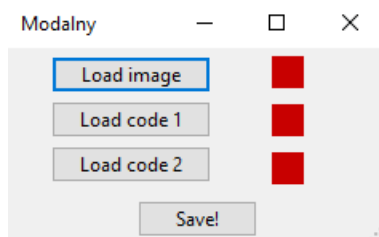


Rysunek 2: Okno programu.

W naszym programie wykorzystujemy dwa okna modalne przedstawione na rysunku (3) i (4).



Rysunek 3: Podstawowe okno modalne.



Rysunek 4: Okno modalne do dekodowania metody mieszanej.

6.2 Specyfikacja danych wejściowych i wyjściowych.

Program przyjmuje i zwraca plik w formacie ".png", ponieważ w naszym projekcie ważne było, żeby wykorzystywać format bezstratny. Poniżej przedstawiamy jakie pliki nam się wygenerują przy użyciu poszczególnych metod i pod jaką nazwą one będą.

Obrazek	Nazwa pliku
Zakodowany obraz metodą A	"methodA_encrypted.png"
Zakodowany obraz metodą B nr 1	"methodB_encrypted_1.png"
Zakodowany obraz metodą B nr 2	"methodB_encrypted_2.png"
Zakodowany obraz metodą mieszaną nr 1	"A1_and_B_1.png"
Zakodowany obraz metodą mieszaną nr 2	"A1_and_B_2.png"
Powiększony wzorzec do odkodowania metody mieszanej	"Reference_to_mixed_A_and_B.png"
Odkodowany szyfr metodą A	"methodA_decrypted.png"
Odkodowany szyfr metodą B	"methodB_decrypted.png"
Odkodowany szyfr metodą mieszaną	"A_and_B_decoded.png"

6.3 Przycisk "Code"

Po wciśnięciu przycisku "Code!" wyskakuje nam okno modalne przedstawione na rysunku (3).

"Load image" pozwoli nam załadować obraz do zakodowania metodą A, a "Load message" wiadomość, którą chcemy zaszyfrować.

"Save" oznacza zatwierdzenie załadowanych obrazów. Otrzymamy trzy pliki typu "png", jeden z metody A i dwa z metody B.

6.4 Przycisk "Decode A"

Wyskakuje nam okno modalne z rysunku (3).

"Load image" służy do załadowania obrazka wzorcowego. "Load message" do załadowania obrazu z zaszyfrowaną wiadomością. Kliknięcie "Save" oznacza zatwierdzenie i otrzymujemy plik ".png" z wiadomością, która była ukryta w zdjęciu.

6.5 Przycisk "Decode B"

W tym przypadku wyskakuje inne okno modalne, które jest przedstawione na rysunku (4). Przyciski służą do załadowania dwóch obrazów, które otrzymaliśmy za pomocą kodowania metodą B.

"Save" oznacza zatwierdzenie, otrzymujemy plik ".png" zawierający ukrytą wiadomość.

6.6 Przycisk "Decode A + B"

W tym przypadku wyskakuje okno modalne, które jest przedstawione na rysunku (4). Wyświetlają nam się trzy przyciski.

Pierwszy z nich służy do załadowania wzorca, drugi do załadowania jednego z obrazów zakodowanego obiema metodami i trzeci przycisk do załadowania drugiego obrazu zakodowanego.

W dekodowaniu mieszanej metody ważne jest, żeby w "Load image" **załadować wzorzec, który uzyskaliśmy podczas kodowania**. Jest to plik o nazwie "Reference_to_mixed_A_and_B.png". Jest to spowodowane tym, że podczas kodowania metodą B zmieniane są rozmiary obrazka.

"Save" oznacza zatwierdzenie, otrzymujemy plik ".png" zawierający ukrytą wiadomość.

7 Potrzebne algorytmy.

7.1 Algorytm kodowania metodą A.

W celu wykonania algorytmu dla metody A stworzyliśmy klasę zawierającą funkcję kodowania i dekodowania obrazów. W poleceniu mieliśmy za zadanie kodować obrazy czarno-białe zwiększając piksel na kanale 'Red' lub kodując obrazy w 8 odcieniach szarości, wykorzystując wszystkie trzy kanały RGB. W poniższej tabeli przedstawiamy wybrany przez nas "klucz do kodowania".

Tabela 1: Klucz do kodowania w 8 odcieniach szarości

Odcień szarości	R	G	B
227 - 255	0	0	0
192 - 227	0	0	1
160 - 191	0	1	0
128 - 159	0	1	1
96 - 127	1	0	0
64 - 95	1	0	1
32 - 63	1	1	0
0 - 31	1	1	1

Jako, że mamy 3 kanały RGB i możliwość każdego zwiększenia o 1 lub pozostawienia, mogliśmy rozważyć 8 przypadków. Do zmian wartości pikselów korzystaliśmy z funkcji **GetData()** z biblioteki **wxWidgets**. Przy odkodowywaniu ponownie posłużyliśmy się tabelką z kluczami, sprawdzaliśmy różnicę między obrazem wzorcowym a zaszyfrowanym i następnie malowaliśmy piksel obrazu, na którym wyświetli się szyfr na odpowiedni kolor w skali szarości. W naszym projekcie do stworzenia tego algorytmu stworzyliśmy klasę **MethodA** oraz funkcje kodującą i dekodującą nazwane odpowiednio **codeWithMethodA** oraz **decodeWithMethodA**

7.2 Algorytm kodowania metodą B.

Aby zakodować obraz metodą B, potrzebujemy dwóch nowych obrazów, które uzupełniamy płytkami z rysunku 1. Do stworzenia tego algorytmu stworzyliśmy dwa obrazy,

o rozmiarach dwa razy większych niż obraz kodowany. Parametry obrazu szyfrowanego pobieraliśmy z funkcji **GetHeight()** oraz **GetWidth()**. Następnie sprawdzaliśmy czy w szyfrze w danym pikselu znajduje się kolor czarny czy biały. Użyliśmy do tego funkcji **GetData()**. Jeśli piksel był niepokolorowany wybieraliśmy losowo płytkę C1 lub C2 z rysunku 1 korzystając z funkcji **rand()** i w zależności od tego co wylosowaliśmy malowaliśmy 4 piksele w ten sam wzór na obu obrazach. Przy odszyfrowaniu podobnie jak w metodzie A, przyrównywaliśmy do siebie oba obrazy, jeśli okazały się inne, malowaliśmy piksel na czarno. W przypadku, gdy piksele były takie same zostawialiśmy kolor piksela biały. Jako, że nasze szyfry były rozmiarów dwa razy większych odszyfrowany obraz zmniejsziliśmy korzystając z funkcji **resize()**. Do tej metody stworzyliśmy klasę **MethodB** oraz funkcję kodującą i dekodującą nazwane odpowiednio **codeWithMethodB** oraz **decodeWithMethodB**.

7.3 Kodowanie metodą mieszaną.

W tym przypadku skorzystaliśmy z gotowych już algorytmów na kodowanie metodą A i B. Konieczne było poprawne wykorzystanie funkcji **resize()**, ponieważ algorytm A i B kodowały ten sam szyfr w innych rozmiarach. Przy kodowaniu najpierw użyliśmy algorytmu B, a następnie oba powstałe obrazki, zakodowaliśmy metodą A. Przy odkodowywaniu zastosowaliśmy odwrotną kolejność. Załadowaliśmy wzorzec i dwa obrazki zakodowane metodą A, następnie odkodowaliśmy je otrzymując dwa obrazki w metodzie B i na końcu odczytaliśmy z nich szyfr. W tej metodzie konieczne było również zapisanie nowego wzorca, którego rozmiary są dwa razy większe od wczytywanego.

8 Kodowanie.

Nasz program jest zbudowany z 6 klas.

- Main
- GUI
- GUIMyFrame1
- ModalDialogue
- MethodA
- MethodB

8.1 Main

Jest to klasa rozruchowa programu.

8.2 GUI i GUIMyFrame1

Są to klasy służące do stworzenia interfejsu użytkownika. Przy tworzeniu jego korzystaliśmy z pomocy "wxFormBuilder". W GUIMyFrame1 znajdują się funkcje opisujące, co wydarzy się po wciśnięciu danego przycisku.

Każda z funkcji ma swoje własne okno modalne oraz obsługę błędów, na wypadek gdy

użytkownik przerwie załadowywanie obrazków.

Funkcja *m_button_code_click* powoduje wywołanie klas metody A i B, zakodowanie załadowanych przez okno modalne obrazków. Znajdę się w niej również kodowanie obrazu metodą mieszaną. Wywołujemy na obrazie zwróconym z funkcji metody B metodę A. Dla następnych przycisków powstały analogiczne funkcje, zawierające funkcje do poszczególnych dekodowań. Na końcu znajduję się w niej funkcja *Repaint* służąca do wyświetlania przekazanych do niej obrazków na panelu.

8.3 ModalDialogue

Jest to klasa, która stwarza okno dialogowe. Dodaliśmy w niej dodatkowo funkcję umożliwiającą zmiany napisów na poszczególnych przyciskach (*setFirstButtonText*, *setSecondButtonText*), przyjmują one jako argument wxString oraz funkcje do zmiany kolorów paneli (*setIndicatorCode*, *setIndicatorImage*, *setIndicatorMessage*), które mają sygnalizować załadowanie obrazu. Stworzyliśmy również gettery (*getImage*, *getMessage*, *getCode*), które zwracają poszczególne obrazki.

8.4 MethodA

Jest to klasa mająca za zadanie zakodować i odkodować obraz metodą steganograficzną. Zawiera ona funkcję *codeWithMethodA*, która zawiera zaimplementowany algorytm opisany w podpunkcie 7.1. Funkcja ta zwraca obiekt wxImage, a konkretnie obraz zaszyfrowany. Klasa posiada także, funkcję *save_MethodA_encrypted*, która zapisuje obraz zwracany w funkcji *codeWithMethodA* do pliku. Mamy również funkcję *decodeMethodA*, która przyjmuje za argumenty dwa obrazki. Pierwszy z nich to wzorzec drugi to szyfr ukryty we wzorcu. Funkcja zwraca obraz wxImage. Tym razem jest to odkodowany obraz. Ostatnia funkcją jest *save_MethodA_decrypted*, która zapisuje obraz zwrócony w funkcji *decodeMethodA* do pliku.

8.5 MethodB

Klasa ta jest analogicznie stworzona do klasy *MethodA*. Zawiera te same funkcję tylko z końcówką nazwy "B" zamiast "A". Różnice są również w funkcji kodującej i dekodującej, w tym przypadku korzystają one z algorytmu opisanego w punkcie 7.2. W przypadku funkcji kodującej *codeWithMethodB* zwraca ona obiekt wxVector, który zawiera dwa zakodowane obrazki.

9 Testowanie programu.

Dla sprawdzenia czy nasz algorytm koduje i odkodowuje poprawnie sprawdziliśmy kilka przykładów.

Na początku załadowaliśmy wzorzec strusia. Kliknięcie **Load image** jak na rysunku 3.

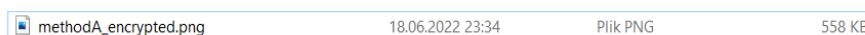


Rysunek 5: Wzorzec do kodowania metodą A
i szyfr, który chcieliśmy zakodować. Kliknięcie **Load message**

AU!!

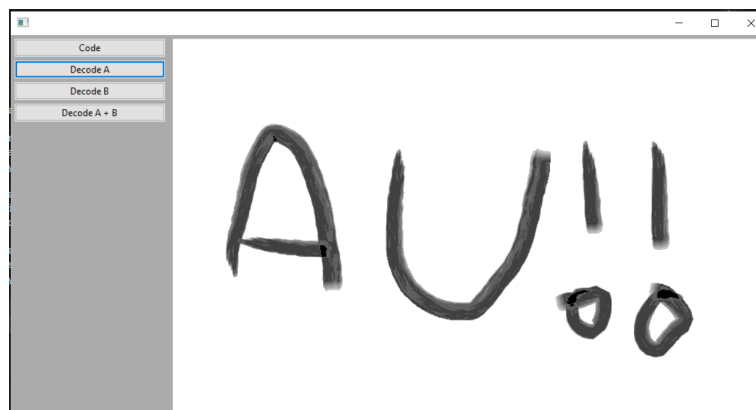
Rysunek 6: Szyfr

Następnie klikając przycisk **save** otrzymaliśmy w folderze programu zapisany w formie png zakodowany obraz.



Rysunek 7: Zapisany obraz w folderze



Aby sprawdzić czy nasza metoda jest poprawna używamy funkcji **Decode A** ładując najpierw wzorzec, a następnie zakodowany obraz wygenerowany w wywołaniu funkcji **Code**. Otrzymaliśmy następujący efekt:



Rysunek 8: Odkodowany szyfr

Widzimy, że otrzymaliśmy tę samą wiadomość co zakodowaliśmy. Skala szarości również została zachowana.

Następnie chcieliśmy sprawdzić poprawność metody B. Ponieważ nasza funkcja **Code** koduje od razu wszystkimi dostępnymi metodami to od razu wygenerowały nam się również zakodowane obrazy metody B:

 methodB_encrypted_1.png	19.06.2022 00:12	Plik PNG	276 KB
 methodB_encrypted_2.png	19.06.2022 00:12	Plik PNG	276 KB

Rysunek 9: Obrazy zakodowane zapisane w folderze

Wyglądają one następująco:

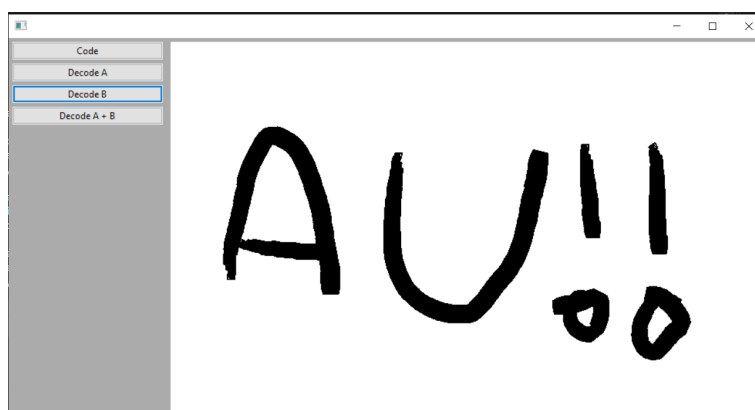


Rysunek 10: Obraz kodowany nr 1



Rysunek 11: Obraz kodowany nr 2

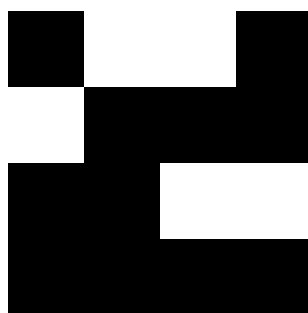
Następnie używając funkcji **Decode B** załadowaliśmy oba powyższe obrazki i otrzymaliśmy następujący wynik:



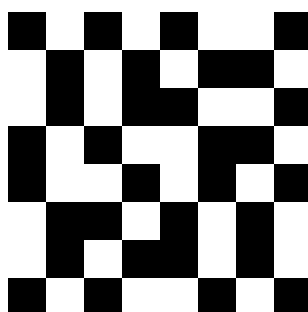
Rysunek 12: Odkodowany obraz metodą B

W tym przypadku nasz program również poprawnie rozszyfrował kod. Otrzymany obraz jest czarno-biały. Tego się spodziewaliśmy, ponieważ metoda B wykrywa jedynie czy jest pokolorowany piksel czy też nie.

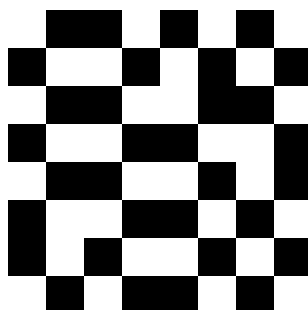
W celu sprawdzenia metody B dodaliśmy jeszcze 1 obrazek o wymiarach 4x4, aby upewnić się czy nie popełniliśmy tego samego błędu przy kodowaniu i dekodowaniu otrzymując przypadkowo poprawny wynik:



Rysunek 13: Szyfr w wymiarze 4x4



Rysunek 14: Zaszyfrowany obraz nr 1



Rysunek 15: Zaszyfrowany obraz nr 2

Widzimy, że obrazki są poprawnie zakodowane zgodnie ze schematem w opisie projektu dla metody kryptograficznej.

10 Podsumowanie.

W naszym projekcie udało nam się spełnić większość wymagań. W przyszłości moglibyśmy ulepszyć aplikację o możliwość kodowania obrazu tylko metodą A lub B. Chcielibyśmy stworzyć bardziej nowoczesny interfejs użytkownika. Obecnie nasz program wymaga, aby wzorzec i szyfr były tych samych rozmiarów, nie wiemy czy jest to najlepszy pomysł, być może w przyszłości spróbujemy zmodyfikować obraz, aby niezależnie od rozmiarów obrazów był w stanie zakodować je. Aplikacja wymaga również bardziej szczegółowej ob-

sługi błędów. Zdajemy sobie sprawę z niedociągnięć naszego projektu, jednak mimo to jesteśmy z niego zadowoleni i sądzimy, że aplikacja spełnia swoje zadanie.