Uwierzytelnianie obrazów cyfrowych z wykorzystaniem kodowania fraktalnego

Antoni Grzanka, Adam Niedziałkowski

I. Wstęp

Celem projektu jest implementacja algorytmu do samo-rekonstrukcji obrazów cyfrowych z wykorzystaniem kodowania fraktalnego. Samo-rekonstrukcja (ang. self-recovery lub self-embedding) pozwala na weryfikację integralności zdjęcia oraz na odtworzenie jego oryginalnej treści w oparciu o cyfrowy znak wodny. Opracowaną w ramach projektu aplikację należy wyposażyć w graficzny interfejs użytkownika pozwalający na dostosowanie parametrów algorytmu oraz na zabezpieczanie i weryfikację integralności obrazów cyfrowych.

II. Instrukcja konfiguracji oraz uruchomienia

Aby przygotować nasz program do uruchomienia, należy uruchomić skrypt konfiguracyjny poleceniem

```
./config.sh
```

Zainstaluje on wszystkie niezbędne do działania pakiety: środowisko Python 2, biblioteki NumPy, SciPy, PIL i inne. Do jego poprawnego działania wymagane jest połączenie z internetem.

Aby tego dokonać, może okazać się potrzebne nadanie odpowiednich praw naszemu plikowi

```
chmod +x config.sh
```

lub uruchomienie go z prawami administratora

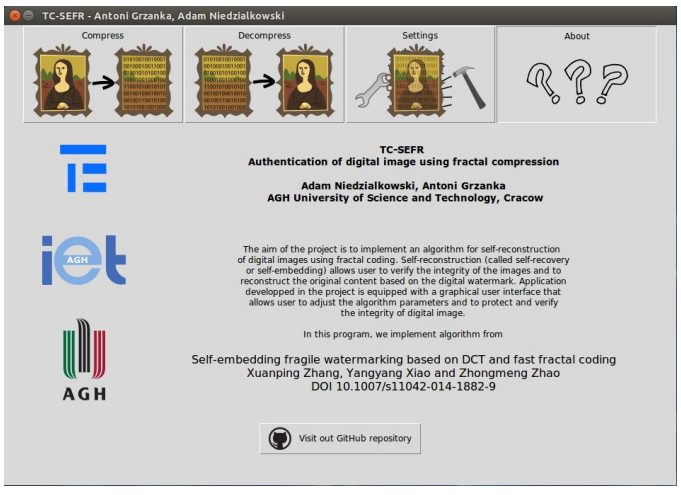
```
sudo ./config.sh
```

Po chwili, gdy skrypt konfiguracyjny zakończy się wykonywać oraz wszystkie pakiety zostaną zainstalowane, główny program będzie można uruchomić poleceniem

```
python gui_tabs.py
```

III. Instrukcja użytkowania

Po uruchomieniu naszym oczom ukaże się graficzny interfejs użytkownika. Kliknięcie na dowolną z zakładek przeniesie nas do odpowiedniej części programu. Każda z funkcjonalności prezentowana jest przez prostą grafikę.



A Zakładka „Compress”

W tej części programu możemy skompresować wybrany przez nas obraz w formacie PGM. Kliknięcie „Load file” wywoła okno dialogowe pozwalające wybrać obraz. Po jego wybraniu, zostanie on wyświetlony w programie oraz uaktywni się przycisk „Perform compression and coding”. Jego kliknięcie spowoduje rozpoczęcie procesu kompresji fraktalnej – plik wynikowy zostanie zapisany w katalogu z plikiem wejściowym z dopiskiem *_compressed*.

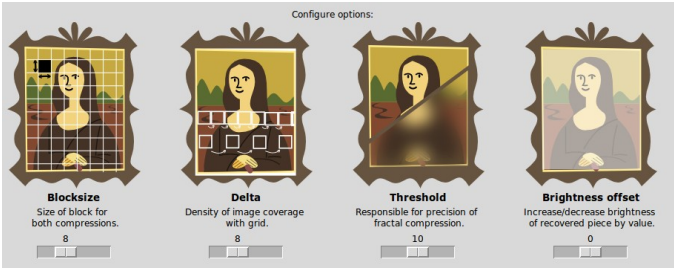
B Zakładka „Decompress”

Ta zakładka spełnia funkcję odwrotną do poprzedniej – pozwala zdekompresować skompresowany wcześniej obraz. Plik wynikowy zostanie zapisany w katalogu z plikiem wejściowym z dopiskiem *_uncompressed*.

C Zakładka “Settings”

W założeniu ma pozwalać na dostosowanie parametry wykorzystywanych w naszym algorytmie. Dostępne do manipulacji są:

- **Blocksize** – rozmiar bloków na które dzielimy obrazki (domyślnie 8x8px, możliwe od 4x4 do 16x16)
- **Delta** – odległość kolejnych siatek bloków od siebie; gęstość pokrycia obrazu siatką oraz blokami (domyślnie 8px)



- **E Threshold** – wartość odpowiadająca za precyzję kompresji fraktalnej
- **DCT Threshold** – wartość odpowiadająca za precyzję kompresji DCT

Opcje domyślne zostały przez nas dobrane tak, aby przedstawić maksymalne możliwości zaimplementowanego algorytmu.

W przyszłości przewidujemy możliwość zmiany opcji, Wymaga to jednak przeprowadzenia licznych testów, które pozwoliłyby zdefiniować podział puli bitowej na odpowiednie własności.

D Zakładka "About"

Zawiera krótkie zestawienie informacji o naszym projekcie oraz link do jego repozytorium na platformie GitHub.

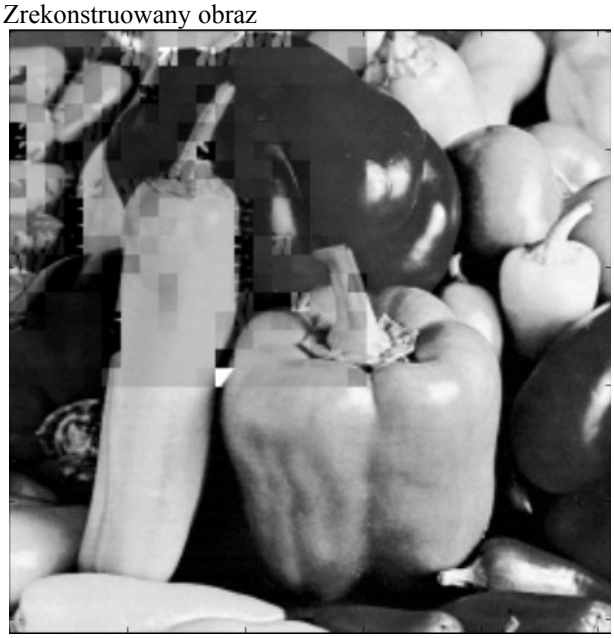
IV. Wyniki testów

Do testów wybraliśmy klasyczny obrazek wykorzystywany do demonstracji kompresji obrazów – pepper.pgm.

1) Oryginalny obraz: pepper.pgm

2) Uszkodzony przez nas obraz

3) Zrekonstruowany obraz



V. Podsumowanie

Podsumowując, zadanie okazało się trudniejsze niż się spodziewaliśmy. Z czasem zauważyliśmy niedoskonałości i liczne problemy z przyjętym przez nas modelem pracy – często niemożliwe okazało się zsynchronizowanie naszych osobno pisanych kodów tak, aby działały razem.

Jednakże, jesteśmy bardzo zadowoleni z faktu, że udało nam się doprowadzić program do funkcjonalnej i estetycznej postaci. Wyniki testów wyglądają obiecująco i jesteśmy przekonani, że przy większej ilości czasu będziemy w stanie lepiej zoptymalizować wykorzystywany algorytm.

Wielu założonych początkowo funkcjonalności nie udało nam się zrealizować z uwagi na brak czasu, m.in. obsługa innych formatów plików niż PGM. Byliśmy też zmuszeni tymczasowo wyłączyć opcję dobierania współczynników.

Podział prac wyglądał następująco:

Adam Niedziałkowski –

- implementacja DCT
- implementacja kompresji fraktalnej
- implementacja autentykacji
- implementacja rekonstrukcji
- liczne testy i optymalizacja

Antoni Grzanka –

- wstępny podział obrazu na bloki
- częściowa implementacja kompresji fraktalnej
- obliczanie wartości znaku wodnego
- graficzny interfejs użytkownika
- niniejsza dokumentacja

Całość naszej pracy, wraz z poszczególnymi „kamieniami milowymi" można prześledzić w zakładce „Issues" w naszym repozytorium na platformie GitHub.