Rekonstrukcja obrazu z sygnału rzadkiego

Karol Olko, Piotr Wiśniewski

29 stycznia 2014

Abstrakt

Niniejszy raport stanowi opis pracy nad implementacją algorytmu rekonstrukcji obrazu rzadkiego na karcie graficznej, z wykorzystaniem biblioteki OpenCL. Projekt opiera się na osiągnięciach grupy badawczej uniwesytetu RICE, twórców tzw. jednopikselowej kamery [2]. Zasymulowano jej działanie w sposób software'owy oraz przy wykorzystaniu kamery. W ramach pracy zaimplementowano algorytm odtwarzania obrazu, wykorzystujący wyłącznie procesor CPU, kartę graficzną oraz rozwiązanie hybrydowe, które porównano ze względu na czas obliczeń. Implementację poprzedziły rozległe badania literaturowe.

Spis treści

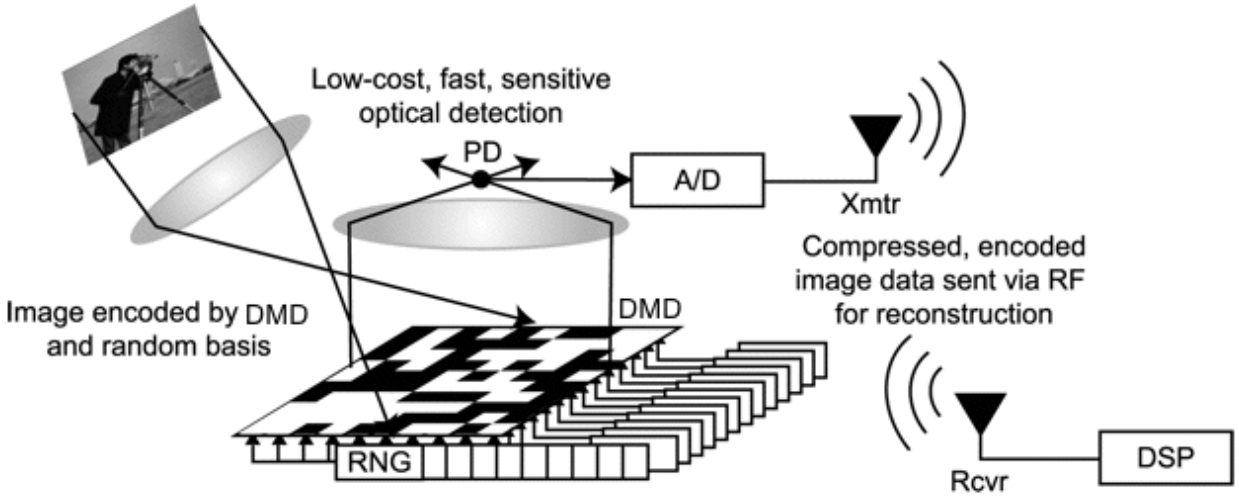
1 Wstęp

1.1 Wprowadzenie

Kierunek badań odtwarzania sygnału rzadkiego (ang. **CS** – *Compressed Sensing*) jest stosunkowo nową i bardzo ciekawym kierunkiem badań w dziedzinie przetwarzania sygnałów cyfrowych (ang. **DSP** – *Digital Signal Processing*). W klasycznym podejściu, aby dokładnie odtworzyć sygnał ciągły, próbkuje się go przynajmniej z częstotliwością Nyquista. Dla sygnałów szybkozmiennych jest to jednak bardzo kosztowne, gdyż wymaga to znacznego skomplikowania czujników (wyposażając je w mechanizm kompresji danych) lub wymusza wykorzystanie kanału transmisyjnego o dużej przepustowości. Często również wykonanie samego pomiaru jest drogie i powolne (jak w przypadku rezonansu magnetycznego) lub niebezpieczne (promienie rentgenowskie). Wyniki badań z dziedziny próbkowania rzadkiego dowodzą jednak, że często udaje się dokładnie odtworzyć sygnał ze znacznie mniejszej liczby pomiarów, wykorzystując jego nadmiarowość i rzadką reprezentację w odpowiednio dobranej przestrzeni [18]. Ta sama idea wykorzystywana jest w algorytmach kompresji danych: rzadka reprezentacja zdjęcia w bazie falkowej jest metodą zmniejszenia wielkości obrazu w standardzie JPEG2000 [6]. Algorytmy CS są zasadniczo różne - próbują one znaleźć taki sposób pomiaru, który pobiera konieczne informacje o sygnale już w skompresowanej formie. Proces odtwarzania sygnału możemy więc podzielić na dwie podstawowe czynności - rzadkie próbkowanie i rekonstrukcję. Drugi krok jest nietrywialny i kosztowny pod względem obliczeniowym. Zazwyczaj zadanie rekonstrukcji sygnału sprowadza się do liniowego lub kwadratowego problemu optymalizacji, w którym najwięcej czasu zajmuje obliczenie kroku Newtona [12].

1.2 Cele i założenia projektu

Głównym celem projektu była implementacja wybranego algorytmu odtwarzania sygnału rzadkiego, wykorzystując do obliczeń kartę graficzną. Zadaniem dodatkowym była symulacja jednopikselowej kamery, wykorzystując kamerę firmy Jai, dostępną w sali laboratoryjnej. Projekt miał umożliwić znaczne przyspieszenie istniejących rozwiązań, opartych głównie o skrypty środowiska Matlab/Simulink, a także umożliwić odtwarzanie obrazów o większej skali (docelowo obraz w pełnej rozdzielczości kamery, czyli 2560 x 2048 pikseli). Z uwagi na przyjęty sposób uzyskiwania pomiarów oraz przede wszystkim - ograniczenia pamięciowe, okazało się to kompletnie nierealne. Projekt zrealizowano w znacznie mniejszej skali, dając jednak możliwość na jego dalszą rozbudowę (co, według naszej wiedzy, byłoby możliwe przy innym sformułowaniu problemu).



Rysunek 1: Schemat jednopikselowej kamery

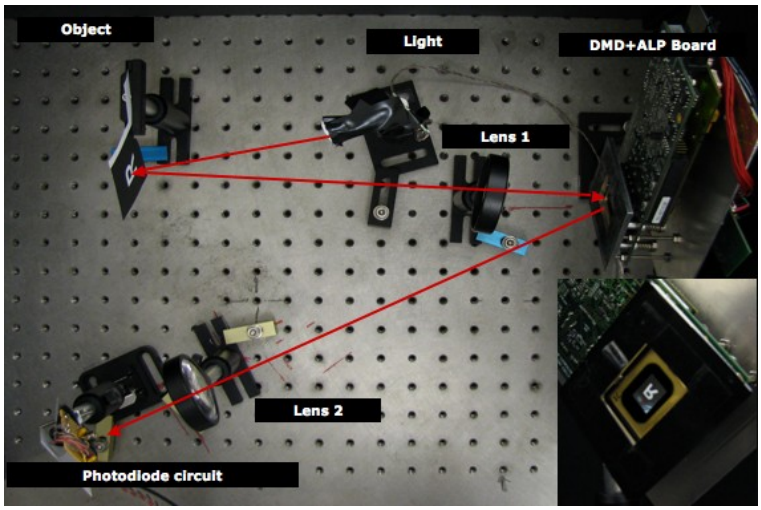
1.2.1 Budowa jednopikselowej kamery

Pierwszym krokiem w realizacji projektu było dokładne zaznajomienie się z urządzeniem, którego działanie mieliśmy symulować. Pomogło nam to zrozumieć praktyczny kontekst przy zapoznawaniu się z publikacjami naukowymi oraz poznać ograniczenia przy projektowaniu algorytmów odtwarzania obrazu narzucone przez sprzęt.

Jednopikselowa kamera to prototyp urządzenia pomiarowego, skonstruowany przez naukowców uniwersytetu RICE [2]. Składa się ona z jednej fotodiody ("piksela"), dwóch soczewek i matrycy mikroluster (ang. **DMD** - *Digital Micromirror Device*) 2. Ustawienie luster macierzy jest sterowane komputerowo. Podanie 1 na poszczególny element DMD powoduje jego skierowanie w kierunku drugiej soczewki, zaś 0 - z dala od niej. Wartości między 0 a 1 mogą być uzyskane przez wielokrotne sterowanie elementem matrycy w czasie całkowania sygnału na odbiorniku. Zgromadzone na fotodiodzie napięcie jest konwertowane do postaci cyfrowej i wysyłane drogą radiową do odbiornika 1. Ze względu na długi proces pomiarowy, trudno wyobrazić sobie, by powyższa konstrukcja była w stanie wyprzeć zwykłe kamery w codziennych zastosowaniach. Może jednak stanowić atrakcyjną cenowo alternatywę dla czujników światła w zakresach niewidzialnych dla oka.

1.2.2 Formalne sformułowanie problemu

W swym doskonałym artykule [8] Emmanuel Candes, jeden z pionierów dziedziny przetwarzania rzadkiego, przedstawił podstawowe obserwacje naukowców umożliwiające zastosowanie metody, dokonał pewnych oszacowań ilościowych, dotyczących potrzebnej liczby pomiarów i ograniczeń metody na dzisiejszym etapie rozwoju. Dziedzina prze-



Rysunek 2: Zestawienie jednopikselowej kamery na stanowisku laboratoryjnym

twarzania rzadkiego wykorzystuje fakt *rzadkości* sygnału oraz *niekoherencji* [11] różnych baz przestrzeni liniowych. Pierwsze z pojęć oznacza możliwość przechowania większości użytecznych informacji o sygnale w niewielkim obszarze danych. Ściślej rzecz biorąc, reprezentacja $x = \phi\alpha$ w bazie ϕ jest k-rzadka, gdy $K \ll N$ współczynników α dokładnie opisuje x. Niekoherencja wyraża z kolei ideę, że mając rzadką reprezentację sygnału w jednej bazie, jest ona znacznie rozłożona w innej. Przykładem może być delta Diraca, która choć w dziedzinie jest reprezentowana przez jeden pik, jest funkcją stałą w całej dziedzinie częstotliwości. Wykorzystując dwie powyższe obserwacje, można skonstruować taki sposób pomiaru, który próbkuje go od razu w skompresowanej formie, bez potrzeby jego zrozumienia.

Przyjmujemy, że informacja o sygnale f_t jest uzyskiwana poprzez liniowe funkcjonały

$$y_k =< f, \varphi_k >, k = 1, ..., m. \tag{1}$$

Gdzie φ_k wyraża sposób pomiaru. Przykładowo, dla matrycy CCD będzie to funkcja charakterystyczna jej danego piksela, a dla rezonansu magnetycznego - sinusoida o określonej częstotliwości.

Sposób wykonania m pomiarów przechowujemy w macierzy pomiarowej $A \in R^{M*N}$. Tym samym, problem CS może zostać opisany jako nieoznaczone równanie liniowe:

$$y = A \cdot x \tag{2}$$

Powyższe równanie posiada nieskończoną liczbę rozwiązań z $N - M$ stopniami swobody. Rekonstrukcja x polega na takim wybraniu rozwiązania, które najlepiej spełnia postawione przed nim dodatkowe założenia. Najbardziej oczywistym jest znaleźć najrzadsze rozwiązanie, tzn.

$$\hat{x} = \arg\min_z \|x_0\| \text{spełniający} \Phi x = y \tag{3}$$

[8, 7, 37], gdzie $\|x\|_0$ = ilość niezerowych współczynników x. Użycie normy zero cechuje się jednak kombinatoryczną złożonością obliczeniową i nie jest realizowalne w dużej skali. Najczęściej jednak udaje się dokładnie odzyskać rzadki sygnał, rozwiązując problem prostego dopasowania (**BP** - *basis pursuit*):

$$\min_{x \in R^n} \|x_1\| \tag{4}$$

, gdzie $\|x\|_1 = \sum_{i=1}^{n} |x_i|$. Problem 4 ma wyraźną przewagę nad 3, gdyż daje się rozwiązać metodami programowania liniowego [23]. Część algorytmów odtwarzania sygnałów rozwiązuje zmodyfikowaną wersję 4. Istnieje również wiele metod rekonstrukcji sygnału, wykorzystując algorytmy zachłanne, stochastyczne lub wariacyjne.

Warto przy tej okazji wspomnieć, że przetwarzanie rzadkie wykorzystywane jest także dla sygnałów ciągłych [9], zaszumionych i nie całkowicie rzadkich [5]. W takim wypadku α stanowi aproksymację x. Jak pokazują wyniki rekonstrukcji zdjęć, są one wciąż bardzo skuteczne.

Istotnym osiągnięciem CS jest jego zdolność do oszacowania liczby pomiarów tak, by rekonstrukcja przebiegła pomyślnie z prawdopodobieństwem bliskim pewności [8]. Jest ona zależna od rzadkości reprezentacji sygnału w danej bazie oraz niekoherencji bazy, w której próbkujemy sygnał z bazą rozrzedzającą x. Wielkość tę definiujemy następująco:

$$\mu(\Phi, \Psi) = \sqrt{n} \max_{i,j} | < \phi_i, \psi_j > | \tag{5}$$

Okazuje się, że bezpiecznym wyborem jest losowe próbkowanie sygnału, gdyż jest ono niekoherentne z dowolną, ustaloną bazą ϕ. Pozostaje więc odpowiednio dobrać sposób jego reprezentacji.

1.3 Zarys proponowanego rozwiązania

Rozwiązanie problemu CS wymaga podjęcia kilku kluczowych decyzji. Po pierwsze, należy starannie wybrać sposób próbkowania sygnału, bazę, w której będzie on reprezentowany oraz algorytm jego rekonstrukcji. W rozważaniach należy również wziąć pod uwagę docelową architekturę sprzętową i jej możliwie pełne wykorzystanie.

1.3.1 Wybór macierzy pomiarowej

Punktem wyjścia w badaniach był pakiet ℓ_1-magic [1], zawierający kod źródłowy wybranych algorytmów odtwarzania sygnałów oraz dokument [12] je opisujący. Pierwszym ważnym wnioskiem płynącym z publikacji jest konieczność wyrażenia macierzy pomiarowej w sposób ukryty, nie wymagający jawnego przechowywania w pamięci. Jej wielkość bardzo szybko urasta bowiem do ogromnych rozmiarów. Dla przykładu, macierz pomiarowa eksperymentu, w którym dokonano 20 % pomiarów zdjęcia o wielkości

256x256 pikseli, zapisana w formacie *float* będzie zajmować ponad 3GB pamięci, stanowczo za dużo, zmieścić ją w RAM GPU. Powyższy wniosek stanowi to problem z dwóch powodów. Po pierwsze, jak w skompresowany sposób zapisać losowe ciągi liczb wysyłane do DMD? Po drugie, zastosowanie implikowanej formy macierzy znacznie utrudnia przyspieszenie obliczeń za pomocą karty graficznej. Nie próbując jednak od razu odpowiedzieć na wszystkie stawiane wyzwania, rozpoczęto badania bogatej literatury CS. We wstępie [23] dostępny jest imponujący przegląd dostępnych metod próbkowania i odzyskiwania sygnału. Oprócz braku konieczności przechowywania macierzy pomiarowej w pamięci, autor narzuca ciekawe ograniczenie - możliwie szybkie mnożenie macierzy z wektorem (np. $\mathcal{O}(nlog(n))$ lub $\mathcal{O}(n)$). Przyspieszenie tej operacji znacząco skróci czas odtwarzania sygnału, bez względu na wykorzystaną architekturę sprzętową.
Ciekawą propozycją autorów jest niepełna macierz Fouriera. Jej konstrukcja polega na losowym wybraniu m rzędów macierzy Fouriera o rozmiarze n x n. Nie wymaga przechowywania w pamięci, a jej iloczyn z wektorem daje się szybko obliczyć wykorzystując algorytm FFT (cechuje się on złożonością $\mathcal{O}(nlog(n))$). Niestety, jej postać spełnia 4 z ogromnym prawdopodobieństwem dla znacznie większej liczby pomiarów niż wykorzystując zwykłą macierz losową o rozkładzie Gaussa [29]. Postanowiono więc poszukać innych rozwiązań. W swym doskonałym artykule [27], Gabriel Peyre ocenia kilka ortonormalnych baz, które mogą posłużyć jako sposoby rzadkiej reprezentacji sygnału. W praktyce zdjęcia są skutecznie rozrzedzane przez bazę falkową, kosinusową czy curvelet [10, 4]. Według autora, każda z powyższych posiada swe wady i zalety, zatem aby móc odtwarzać sygnał o różnej charakterystyce, bazę reprezentacji należy dobierać dynamicznie. Jest to niewątpliwie ciekawe podejście, ale wymagające ogromnego nakładu pracy i zostało porzucone. Z tego samego powodu porzucono również ideę poszukiwania najrzadszej bazy sygnału poprzez estymację macierzy Q transformacji Karhunena-Loeve'a [17]. [1] Zainteresowało nas jednak próbkowanie sygnału za pomocą podmacierzy bazy Hadamarda-Walsha. Jej definicja jest prosta [2], a będąc mocno związaną z bazą Fouriera, pozwala na mnożenie wektorów ze złożonością $\mathcal{O}(nlog(n))$. Ten sam pomysł, choć w bardziej przejrzysty sposób, przedstawiono również w [7] i w ramach praktycznego przykładu odniesiono go do jednopikselowej kamery. Macierz pomiarową możemy generować poprzez

$$\Phi = \left(\frac{1}{2}\sqrt{N} I_\Gamma W_B + \frac{1}{2}\right) D \tag{6}$$

Gdzie I_Γ oznacza identycznościową podmacierz MxN, uzyskaną przez losowy wybór M wierszy, tak, by $I_\Gamma W_B$ była podmacierzą W_B indeksowaną przez Γ. Ponadto, D oznacza

[1] Odzyskujemy $s = Q^H x$, gdzie $R_x = E[xx^H]$ i $R_x = Q\Lambda Q^H$. Kolumny Q to wektory własne R_x, Λ to diagonalna macierz wartości własnych.

[2] $W_0 = 1$, $W_j = \frac{1}{\sqrt{2}} \begin{Bmatrix} W_{j-1} & W_{j-1} \\ W_{j-1} & -W_{j-1} \end{Bmatrix}$

losową macierz permutacji o rozmiarze NxN. Wyrażenie $\frac{1}{2}\sqrt{N}I_\Gamma W_B + \frac{1}{2}$ przeskalowuje elementy macierzy Φ tak, by mieściły się w zakresie $(0, 1)$. Metoda generowania pomiarów wygląda na bardzo atrakcyjną. Niestety, algorytm szybkiego mnożenia macierzy W_B istnieje tylko dla rozmiarów 2^N. Ponadto, wynik pomiarów sygnału wyrażonego w bazie falkowej $\Phi\Psi$ nie daje teoretycznych gwarancji odtworzenia x z ogromnym prawdopodobieństwem (choć przedstawiona metoda w praktyce daje dobre, powtarzalne rezultaty). Z tego powodu postanowiono poszukać innego rozwiązania. W ciekawym artykule naukowców z uniwersytetu RICE [28] opisano możliwość założenia pewnej struktury sygnału, co ogranicza przestrzeń rozwiązań, a co za tym idzie - zmniejsza liczbę potrzebnych pomiarów i złożoność obliczeniową algorytmów. Dodatkowo, posiadanie pewnych założeń co do odzyskiwanych danych ułatwia rozróżnienie ich od szumu, co zwiększa stabilność procesu. Niestety, autorzy pracy nie zaproponowali takiego rozwiązania, które z jednej strony znacznie ułatwiałaby odzyskanie sygnału, a z drugiej - było dostatecznie elastyczne.

Ostatecznie, naszym wyborem stała się rzadka macierz pomiarowa, opisana w [3]. Cechuje ją niezwykła prostota. W każdej kolumnie macierzy A wybieramy d jedynek. Dbamy przy tym, by żadne dwa wektory kolumnowe nie były identyczne (co przy $d \ll N$ jest praktycznie niemożliwe). Zaletą tego rozwiązania jest udowodniona odporność na szum, efektywne mnożenie macierzy (wymaga ono nie więcej niż nd dodawań) i niewielka ilość zajmowanego miejsca w pamięci. Przedstawione wyniki empiryczne dla zdjęć o rozmiarze 256x256 potwierdziły, że wykorzystując tę macierz pomiarową, można rozwiązywać problemy w odpowiednio dużej skali. Po decyzji związanej z macierzą pomiarową, przystąpiono do wyboru algorytmu rekonstrukcji zdjęcia.

1.3.2 Dobór algorytmu rekonstrukcyjnego

Odpowiedni dobór algorytmu odtworzenia sygnału ma niebagatelne znaczenie na szybkość, stabilność i dokładność procesu. Wyzwaniem stało się odpowiednie dobranie metody dla implementacji na karcie graficznej. Temat ten jest bowiem rzadko podejmowany w publikacjach naukowych. Natrafiono jedynie na jedną bibliotekę implementującą algorytmy odtwarzania sygnału z użyciem kart graficznych [22], zbudowaną na bazie biblioteki CUDA [25]. Grupa metod optymalizacji tam wykorzystanych nosi miano zachłannych. Natrafiono na doskonałe przeglądy metod z tego obszaru w [24, 20]. Przykładowo, przedstawiony tam algorytm iteracyjnego progowania (ang. **IHT** – *Iterative hard thresholding*) [35] opisany jest następującą, iteracyjną formułą:

$$\begin{aligned} x_0 &= 0 \\ x_{n+1} &= H_s\left(x_n + \Phi^T(y - \Phi x_n)\right) \end{aligned} \tag{7}$$

Gdzie $H_s(a)$ zeruje wszystkie poza s największymi współczynniki x. Algorytm cechuje

niezwykła prostota. Jedna jego iteracja wymaga dwóch mnożeń macierz-wektor i częściowego posortowania elementów $a_n = H_s\left(x_n + \Phi^T(y - \Phi x_n)\right)$. Łatwo wyobrazić sobie równoległą implementację takiego rozwiązania, np. przy użyciu algorytmu *mergesort* [36]. Dodatkową zaletą rozwiązania jest niewielka ilość przechowywanych danych. Niestety, prostota i szybkość algorytmu odbija się na jego dokładności. W kontekście zdjęć ich wyjście może stanowić pierwsze przybliżenie rozwiązania problemu 4. W szczególności, metody zachłanne słabo aproksymują niezbyt rzadki sygnał ($\frac{k}{n} \geq 0.2$, gdzie k - liczba znaczących elementów x, n - rozmiar x). Z tego powodu zrezygnowano z tej klasy algorytmów i zwrócono się w stronę klasycznych metod optymalizacyjnych rozwiązujących problem 4 lub pokrewny.

Ponownie, punktem wyjścia w naszych rozważaniach były algorytmy zaimplementowane w pakiecie $\ell 1$-MAGIC [1]. Grupa metod programowania liniowego minimalizujących normę $\ell 1$ okazały się jednak skutecznie głównie dla sygnałów prawdziwie rzadkich. Odtworzenie zdjęcia algorytmem $\ell 1$-pd [31] w aplikacji współtworzonej przez jednego autorów projektu [19] było skuteczne, gdy $\frac{m}{n} \geq 0.6$. Chcąc znacząco poprawić ten wynik, należało poszukać innej metody rozwiązania. Pamiętając również o akceleracji obliczeń przy pomocy karty graficznej, przyglądnięto się klasie algorytmów zwykle implementowanych na tych urządzeniach.

Za niezwykle interesujący uznano algorytm **TVQC** [14, 13]. Traktując x_{ij} jako piksel znajdujący się w i-tym rzędzie oraz j-tej kolumnie zdjęcia x o wymiarach $n \times n$, oraz po zdefiniowaniu operatorów:

$$D_{h;i,j}x = \begin{cases} x_{i+1,j} - x_{i,j} & , i < n \\ 0 & , i = n \end{cases} \qquad D_{v;i,j}x = \begin{cases} x_{i,j+1} - x_{i,j} & , j < n \\ 0 & , j = n \end{cases} \tag{8}$$

Algorytm ten można opisać jako rozwiązanie następującego problemu optymalizacyjnego:

$$\begin{gathered} \min_{TV} \ \text{spełniający} \ \|Ax - y\|_{\ell 2} \leq \epsilon \\ \text{gdzie:} \\ \|x\|_{TV} = \sum_{i,j} \sqrt{\left(D_{h;i,j}\right)^2 + \left(D_{v;i,j}\right)^2} = \sum_{i,j} |\left(D_{i,j}\right)| \end{gathered} \tag{9}$$

Podstawowym założeniem tej metody jest rzadkość (lub kompresowalność) *gradientu* zdjęcia. Jest ono słuszne dla dużej części naturalnych ekspozycji i tylko nieznacznie zmniejsza uniwersalność metody. Baza gradientu sygnału wydaje się być wysoce niekoherentna z rzadką macierzą pomiarową. Dokonano bowiem udanych rekonstrukcji zdjęć przy użyciu ok. 8% pomiarów [3]. Jedyną zagadką pozostała ocena wydajności algorytmu zaimplementowanego na karcie graficznej. Nie odnaleziono bowiem żadnych prób równoległej implementacji algorytmu. Ustalono jednak[33, 21], że możliwe jest wyraźne przyspieszenie najbardziej kosztownego etapu algorytmu - obliczenie kroku Newtona przy

pomocy iteracyjnej metody gradientu sprzężonego. Po długich rozważaniach przyjęto w końcu sposób realizacji algorytmu i przystąpiono do oceny jego złożoności obliczeniowej oraz możliwości stanowiska laboratoryjnego. Miało to bezpośrednie przełożenie na skalę rozwiązywanych później problemów.

1.4 Omówienie zawartości pracy

Poniższy raport składa się z sześciu rozdziałów. Po wstępie, w którym opisano kontekst pracy i wykonane badania literaturowe, następuje rozdział poświęcony złożoności czasowej i pamięciowej postawionego problemu. Rozdział trzeci przedstawia koncepcję jego rozwiązania, a czwarty - procedury testowe. Rozdziały 5 i 6 opisują odpowiednio wyniki przeprowadzonych eksperymentów oraz podsumowanie. Szczegółowy opis zadania znajduje się w załączniku.

2 Analiza złożoności i estymacja zapotrzebowania na zasoby

Pierwszym i niezbędnym krokiem przy pracy na stanowisku laboratoryjnym było zapoznanie się z dostępną architekturą sprzętową oraz wybór środowiska programistycznego. Następnie, zapoznano się i opisano standard OpenCL, w ramach którego implementowano algorytm rekonstrukcji obrazu na kartach graficznych. Należało bowiem poznać szczególne wymagania oraz ograniczenia wynikające z jego użycia. Dalszą część rozdziału stanowi oszacowanie złożoności pamięciowej i obliczeniowej wybranego algorytmu odtwarzania obrazu.

2.1 Sprzęt uruchomieniowy

Dostępny w laboratorium komputer składał się z wielordzeniowego procesora klasy Intel i5 oraz dwóch kart graficznych firmy Nvidia GeForce GTX670. W poniższej tabeli zebrano ich główne parametry:

	Nvidia GeForce GTX 670
Liczba rdzeni	1344
Częstotliwość bazowa	915 [MHz]
Pojemność pamięci	2048 [MB]
Magistrala	256 [bit]

Tablica 1: Wybrane parametry kart graficznych Nvidia GTX 670

Liczba rdzeni wykorzystywanej przez nas karty graficznej jest wprost imponująca. Potencjalnie, wykorzystanie tej ilości jednostek obliczeniowych może dać kolosalne korzyści czasowe. Program, który w skuteczny sposób wykorzystuje powyższe zasoby, musi dokonać kilku niskopoziomowych optymalizacji. Doskonałą lekturę w tym zakresie stanowiła książka opisująca bibliotekę OpenCL [32]. Po pierwsze, należy minimalizować liczbę rozgałęzień algorytmu. Po drugie, zadania na karcie graficznej należy dzielić tak, by zależności między wątkami były jak najmniejsze - konieczność synchronizacji znacznie zmniejsza wydajność programu. Po trzecie, należy minimalizować liczbę odwołań wątków karty graficznej do pamięci globalnej - podręczne zasoby lokalne charakteryzują się znacznie mniejszym opóźnieniem. W końcu, należy liczyć się z każdorazowym opóźnieniem obliczeń na GPU, ze względu na dwukrotny transfer danych między hostem a urządzeniem.

Zdecydowano się na implementację programu w języku C++, z wykorzystaniem środowiska Visual Studio. Wynikało to z kilku przyczyn. Po pierwsze, wybór był kon-

sekwencją doświadczenia zawodowego autora. Pisanie kodu w znanym sobie języku i środowisku znacznie przyspieszyło pracę. Po drugie, przygotowany SDK kamery Jai zawierał binding w tym właśnie języku, co znacznie uprościło integrację tego modułu programu. C++ pozwolił twórcom z jednej strony zdekomponować problem obiektowo, a z drugiej - skompilować program natywnie, licząc na optymalną wydajność. Z perspektywy czasu trzeba jednak rozważyć, czy nie lepiej wykorzystać Pythona, z uwagi na duże wsparcie biblioteczne tego języka, a także czytelność i zwartość kodu. Ostatecznym celem programu było bowiem maksymalne wykorzystanie karty graficznej, z minimalną ingerencją komputera-hosta. Ponadto, krytyczne dla wydajności procedury można by zawrzeć w formie skompilowanej biblioteki języka C. Ciekawym byłoby zatem porównać nakład pracy przygotowanych programów i ich wydajność.

2.2 Standard OpenCL

Jednym z głównym wyzwań projektu było zaznajomienie się i wykorzystanie standardu OpenCL [16] do programowania w heterogenicznym środowisku obliczeniowym. W przeciwieństwie do rozwijanego przez firmę Nvidia standardu CUDA, OpenCL może być wykorzystywany na różnych platformach sprzętowych, o ile tylko standard jest wspierany przez producenta. Choć OpenCL definiowuje tylko interfejs programistyczny w języku C oraz wrapper w C++, dostępne są również jego realizacje w Javie, Pythonie i innych. W chwili pisania raportu, opublikowano wersję 2.0 tego dokumentu. Niestety, żaden z producentów kart graficznych jeszcze go nie wspiera. Posługiwano się więc interfejsem programistycznym standardu w wersji 1.2.

Na podstawowe pojęcia w OpenCL składają się: *host* i *device*, na których wykonywana jest aplikacja. *Host* to zwykle CPU, wysyłające rozkazy do GPU (*device*), w szczególności w celu wykonywania *kerneli*, czyli dynamicznie kompilowanych programów. Pojedynczy rdzeń procesora karty graficznej zwany jest *work-itemem*. Pewien zbiór tychże stanowi *work-group*. Jednostki obliczeniowe rozróżniane są globalnym identyfikatorem *globalID*, a grupy - *localID*. Pamięć *device* możemy podzielić na 4 typy: private, local, constant, global. Ich dostęp możliwy jest odpowiednio: *work-item*, *work-group* oraz cały *device*. Cechują się one kolejno rosnącą pojemnością. Niestety, wielkość pamięci idzie w parze z coraz dłuższym czasem dostępu.

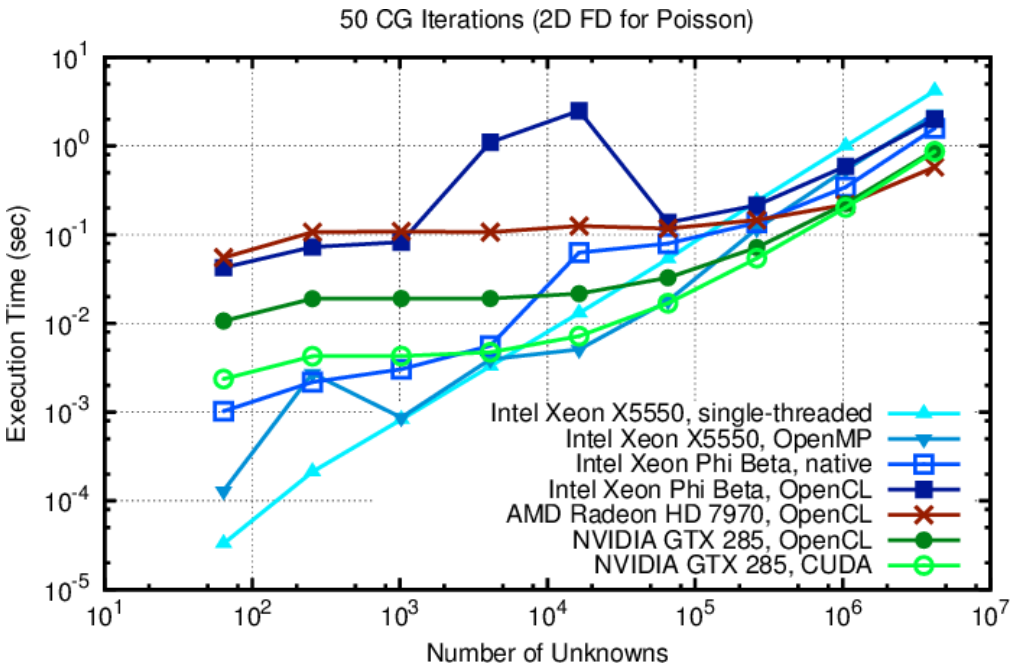
2.3 Wykorzystanie biblioteki ViennaCL w projekcie

Pokrótce analizując kod źródłowy pakietu ℓ_1-MAGIC, szybko zrozumiano, że napisanie i przetestowanie *kerneli* dla karty graficznej będzie zadaniem karkołomnym. Standard w swej naturze jest bardzo niskopoziomowy, co z jednej strony zapewnia elastyczność - ale z drugiej, niską efektywność pracy. Zdolność do debugowania kodu przeznaczonego

na kartę graficzną jest ograniczona. W dodatku, współbieżna praca wielu wątków programu jest częstym błędem występowania trudnych w zreprodukowaniu i zrozumieniu błędów. Przygotowanie efektywnych solverów układów liniowych nie jest zadaniem trywialnym. Z tego powodu zdecydowano się wykorzystać bibliotekę **ViennaCL** [15], która oferuje wysokopoziomowy interfejs programistyczny do przygotowania i uruchamiania zadań na karcie graficznej, zawiera wiele przydatnych, gotowych funkcji, takich jak solvery układów liniowych, metody dekompozycji macierzy, a w końcu - jest niezwykle elastyczna, pozwalając na uruchamianie własnoręcznie napisanych *kerneli*.

2.4 Oszacowanie złożoności algorytmu TVQC

Dokładna ocena złożoności czasowej i pamięciowej algorytmu TVQC wydaje się być trudna. Tym niemniej, wnioski płynące z krótkiej analizy problemu wydają się oczywiste. Najbardziej kosztownym czasowo i obliczeniowo krokiem algorytmu jest rozwiązanie formy kwadratowej. Rozważania ze wstępu tłumaczą konieczność wykorzystania iteracyjnej metody rozwiązywania tego układu. W swym doskonałym artykule, autor biblioteki *ViennaCL* porównał teoretyczną i praktyczną wydajność architektur CPU i GPU. Poza specyficznymi wyjątkami (operacje mnożenia dwóch gęstych macierzy), wydajność kart graficznych zależy w głównej mierze od przepustowości magistrali pamięci. Uzyskane od autora artykułu informacje wskazują, że mnożenie rzadkich macierzy na GPU osiąga praktyczną wydajność ok. 1% piku teoretycznego. Obliczenia arytmetyczne uzyskiwane na karcie graficznej są więc źa darmo- karta głównie czeka na dane. Okazuje się jednak, że wysokiej klasy karty graficzne osiągają w praktycznej implementacji metody gradientu sprzężonego lepsze rezultaty dla problemów dużych rozmiarów, z uwagi na większą przepustowość pamięci, która w pewnym momencie kompensuje konieczność kopiowania danych *host-device*. Najlepiej ilustrują to pomiary wydajności metody gradientu sprzężonego, wykonane przez twórców ViennaCL i przedstawione na rysunku 3.



Rysunek 3: Porównanie wydajności algorytmu CG dla wybranych platform sprzętowych [15]

Wnioski płynące z analizy złożoności TVQC są następujące: kluczowy wydaje się taki podział prac między CPU a GPU, by karta graficzna wykonywała tylko operacje w dużej skali. Ponadto, należy zadbać o jak najlepsze wykorzystanie przepustowości pamięci karty graficznej, np. poprzez dostęp do niej w sposób ćoalesced"[26], tzn. by work-item'y w obrębie work-group'y żądały jednoczesnego dostępu do pamięci globalnej.

3 Koncepcja proponowanego rozwiązania

Jednym z głównych celów przyświecających autorowi programu było stworzenie jasnej, przejrzystej i modularnej struktury aplikacji, umożliwiającej łatwą analizę funkcjonalną i późniejszą możliwość rozszerzeń. Zdefiniowano zatem szereg interfejsów, które są jego ŝtopniami swobody". Poniżej przedstawiono diagramy klas, zależności między nimi oraz ich wykorzystanie w trakcie działania programu.

3.1 Interfejsy programu

3.1.1 Interfejs macierzy

(...)

3.1.2 Interfejs kamery

(...)

3.1.3 Interfejs Solvera

(...)

3.1.4 Interfejs Algorytmu

(...)

3.2 Działanie aplikacji

4 Symulacja i testowanie

4.1 Modelowanie i symulacja

4.2 Testowanie i weryfikacja

4.2.1 Walidacja

4.2.2 Wydajność

5 Rezultaty i wnioski

(...)

6 Podsumowanie

Spis rysunków

Literatura

[1] Kod źródłowy l1-magic. `http://dsp.rice.edu/files/cs/cscameradata.zip`, 2014.

[2] Strona internetowa projektu single pixel camera. `http://dsp.rice.edu/cscamera`, 2014.

[3] Radu Berinde and Piotr Indyk. Sparse recovery using sparse random matrices, 2008.

[4] Emmanuel J. Candès and David L. Donoho. Curvelets: a surprisingly effective nonadaptive representation of objects with edges. In *IN CURVE AND SURFACE FITTING: SAINT-MALO*. University Press, 2000.

[5] Volkan Cevher, Piotr Indyk, Lawrence Carin, and Richard G. Baraniuk. A tutorial on sparse signal acquisition and recovery with graphical models, 2009.

[6] JPEG Comittee.

[7] Mark A. Davenport. Random observations on random observations: Sparse signal acquisition and processing.

[8] M.Wakin E. Candes. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 2008.

[9] Yonina C. Eldar. Compressed sensing of analog signals in shift-invariant spaces. 2009.

[10] David Donoho Lexing Ying Emmanuel Candes, Laurent Demanet.

[11] Justin Romberg Emmanuel Candes. Sparsity and incoherence in compressive sampling. 2006.

[12] Justin Romberg Emmanuel Candes. l1-magic: Recovery of sparse signals via convex programming. `http://users.ece.gatech.edu/~justin/l1magic/downloads/l1magic.pdf`, 2014.

[13] Terence Tao Emmanuel Candes, Justin Romberg. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information, 2005.

[14] Terence Tao Emmanuel Candes, Justin Romberg. Stable signal recovery from incomplete and inaccurate measurements, 2005.

[15] Karl Rupp et al.

[16] Khronos group.

[17] H.T. Kung Dario Vlah Gwon, Youngjune. Compressive sensing with optimal sparsifying basis and applications in spectrum sensing. *IEEE Globecom conference report*, 2012.

[18] M. Davenport i inni. *Introduction to Compressed Sensing in Compressed Sensing: Theory and Applications*.

[19] P. Wiśniewski I. Kułakowska. Próbkowanie sygnałów rzadkich - model.

[20] Erald Vucini Jasper van de Gronde. Compressed sensing overview.

[21] Eitan Grinspun Peter Schroeder Jeff Bolz, Ian Farmer. Sparse matrixs solvers on gpu: Conjugates gradients and multigrid. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.5723&rep=rep1&type=pdf`.

[22] Jared Tanner Jeffrey Blanchard. Gpu accelerated greedy algorithms. `http://www.gaga4cs.org/`, 2014.

[23] Pavel Zhlobich Kimon Foutoulakis, Jacek Gondzio. Matrix-free interior point method for compressed sensing problems. 2012.

[24] Deanna Needell. Topics in compressed sensing.

[25] NVIDIA Corporation. *NVIDIA CUDA Compute Unified Device Architecture Programming Guide*. NVIDIA Corporation, 2007.

[26] Cornell Univeristy of Advanced Computing.

[27] Gabriel Peyre. Best basis compressed sensing.

[28] Marco F. Duarte Chinmay Hedge Richard G. Baraniuk, Volkan Cevher. Model-based compressive sensing.

[29] M. Rudelson and R. Vershynin. On sparse reconstruction from fourier and gaussian measurements. 2008.

[30] Karl Rupp.

[31] L. Vanderberghe S. Boyd. *Convex Optimization*. Cambridge University Press.

[32] Matthew Scarpino. *OpenCL in Action - How to accelerate graphics and computation*. Manning.

[33] H.T. Kung Stephen J. Tarsa, Tsung-Han Lin. Performance gains in conjugate gradient computation with linearly connected gpu multiprocessors.

[34] Zespół Innovative Computing Laboratory Uniwersytetu Tennessee. Matrix algebra on gpu and multicore architectures. `http://icl.utk.edu/magma/`, 2014.

[35] Mike E. Davis Thomas Blumensath. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 2009.

[36] Krishnahari Thouti. An opencl method of parallel sorting algorithms for gpu architecture.

[37] Yin Zhang. Theory of compressive sensing via l1 minimization: a non-rip analysis and extensions.

Appendices

A Szczegółowy opis zadania

A.1 Specyfikacja projektu

Celem projektu było zaproponowanie realistycznego modelu jednopikselowej kamery oraz przygotowanie co najmniej jednej metody odtwarzania rzadko spróbkowanego sygnału. Zadanie to wydaje się mieć wiele potencjalnych zastosowań, szczególnie w dziedzinie medycyny lub telekomunikacji. Oczekiwanym celem projektu jest maksymalne przyspieszenie działania algorytmu, poprzez wykorzystanie zasobów karty graficznej.

A.2 Szczegółowy opis algorytmu odtwarzania obrazu TVQC

Algorytm TVQC został dokładnie zdefiniowany w dokumentacji pakietu ℓ_1-MAGIC [12]. Metoda optymalizacji wykorzystana do odtworzenia zdjęcia została z kolei opisana w [31]. Poza przepisem matematycznym, przedstawiono również funkcjonowanie algorytmu na schemacie blokowym.

A.2.1 Zdefiniowanie problemu

Algorytm TVQC należy do grupy kwadratowych problemów optymalizacji (ang. **SOCP** - *Second-order Cone Programming*). Przepiszmy więc problem 9 do ich generycznej postaci:

$$\begin{aligned} \min_{x,t} \sum_{ij} t_{i,j} \text{ spełniający } & \|D_{ij}x\|_2 < t_{ij}, i, j = 1, ..., n \\ & \|Ax - b\|_2 \leq \epsilon. \end{aligned} \tag{10}$$

Przyjmując

$$f_{t_{ij}} = \frac{1}{2}\left(\|D_{ij}\|_2^2 - t_{ij}^2\right) \quad i,j = 1,\ldots,n \tag{11}$$

oraz

$$f_\epsilon = \frac{1}{2}\left(\|Ax-b\|_2^2 - \epsilon^2\right),$$

. Mając

$$\nabla f_\epsilon = \begin{pmatrix} A^T r \\ \mathbf{0} \end{pmatrix}, \quad \nabla f_\epsilon \nabla f_\epsilon^T = \begin{pmatrix} A^T r r^T A & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad \nabla^2 f_\epsilon = \begin{pmatrix} A^* A & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

gdzie $r = Ax - b$. Także,

$$\nabla^2 f_{t_{ij}} = \begin{pmatrix} D_{ij}^* D_{ij} & \mathbf{0} \\ \mathbf{0} & -\delta_{ij}\delta_{ij}^T \end{pmatrix} \quad \nabla^2 f_\epsilon = \begin{pmatrix} A^* A & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

Rozważana forma kwadratowa przedstawia się następująco:

$$\begin{pmatrix} H_{11} & B\Sigma_{12} \\ \Sigma_{12}B^T & \Sigma_{22} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta t \end{pmatrix} = \begin{pmatrix} D_h^T F_t^{-1} D_h x + D_v^T F_t^{-1} D_v x + f_\epsilon^{-1} A^T r \\ -\tau\mathbf{1} - t f_t^{-1} \end{pmatrix} := \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}.$$

gdzie $(tf_t^{-1})_{ij} = t_{ij}/f_{t_{ij}}$, oraz

$$\begin{aligned} H_{11} &= D_h^T(-F_t^{-1})D_h \;+\; D_v^T(-F_t^{-1})D_v \;+\; BF_t^{-2}B^T \;- \\ &\qquad f_\epsilon^{-1}A^T A \;+\; f_\epsilon^{-2}A^T r r^T A, \\ \Sigma_{12} &= \;-TF_t^{-2}, \\ \Sigma_{22} &= F_t^{-1} + F_t^{-2}T^2, \end{aligned}$$

Rugując Δt

$$\Delta t = \Sigma_{22}^{-1}(w_2 - \Sigma_{12}\Sigma_{\partial h}D_h\Delta x - \Sigma_{12}\Sigma_{\partial v}D_v\Delta x),$$

kluczową do rozwiązania formą kwadratową jest

$$H'_{11}\Delta x = w_1 - (D_h^T\Sigma_{\partial h} + D_v^T\Sigma_{\partial v})\Sigma_{12}\Sigma_{22}^{-1}w_2$$

gdzie

$$\begin{aligned} H'_{11} &= H_{11} - B\Sigma_{12}^2\Sigma_{22}^{-1}B^T \\ &= D_h^T(\Sigma_b\Sigma_{\partial h}^2 - F_t^{-1})D_h \;+\; D_v^T(\Sigma_b\Sigma_{\partial v}^2 - F_t^{-1})D_v \;+ \\ &\quad D_h^T(\Sigma_b\Sigma_{\partial h}\Sigma_{\partial v})D_v \;+\; D_v^T(\Sigma_b\Sigma_{\partial h}\Sigma_{\partial v})D_h \;- \\ &\quad f_\epsilon^{-1}A^T A \;+\; f_\epsilon^{-2}A^T r r^T A, \\ \Sigma_b &= F_t^{-2} - \Sigma_{12}^2\Sigma_{22}^{-1}. \end{aligned}$$

Powyższy układ równań jest dodatnio określony, co pozwala rozwiązać go metodą gradientu sprzężonego.

W celu rozwiązania powyższego problemu użyto metody *log-barrier* [31]. Przekształca ona *SOCP* w szereg problemów z ograniczeniami liniowymi:

$$\min_z \langle c_0, z\rangle + \frac{1}{\tau^k}\sum_i -\log(-f_i(z)) \quad \text{spełniający} \quad A_0 z = b, \tag{12}$$

gdzie $\tau^k > \tau^{k-1}$. Ograniczenia nierównościowe zostały włączone do funkcjonału w postaci *funkcji kary* (wybór $-\log(-x)$ dla funkcji bariery nie jest przypadkowy, gdyż posiada ona pożądaną własność (samozgodność [3]), która ułatwia szybką zbieżność (12) do (10) zarówno w teorii, jak i praktyce), która jest nieskończona gdy ograniczenie jest przekroczone i gładka na pozostałym obszarze. Gdy τ^k rośnie, rozwiązanie z^k problemu (12) zmierza do rozwiązania $z^\star$ (10): Można wykazać, że $\langle c_0, z^k\rangle - \langle c_0, z^\star\rangle < m/\tau^k$, tzn. znajdujemy się o mniej niż m/τ^k od optymalnej wartości w k-tej iteracji (m/τ^k jest zwany *odstęp dualności*). Ideą powyższej metody jest to, że każdy z gładkich podproblemów może być rozwiązany z dużą dokładnością przy jedynie kilku iteracjach metody Newtona, szczególnie, że możemy wykorzystać rozwiązanie z^k jako punkt startowy podproblemu $k+1$.

W k-tej iteracji metody *log-barrier*, metoda Newtona tworzy szereg kwadratowych aproksymacji równania (12) i minimalizuje je rozwiązując układ liniowy. Aproksymacja kwadratowa funkcjonału

$$f_0(z) = \langle c_0, z\rangle + \frac{1}{\tau}\sum_i -\log(-f_i(z))$$

w (12) wokół punktu z jest dana jako:

$$f_0(z+\Delta z) \;\approx\; z + \langle g_z, \Delta z\rangle + \frac{1}{2}\langle H_z \Delta z, \Delta z\rangle \;:=\; q(z+\Delta z),$$

gdzie g_z jest gradientem

$$g_z = c_0 + \frac{1}{\tau}\sum_i \frac{1}{-f_i(z)}\nabla f_i(z)$$

a H_z jest macierzą Hessego

$$H_z \;=\; \frac{1}{\tau}\sum_i \frac{1}{f_i(z)^2}\nabla f_i(z)(\nabla f_i(z))^T \;+\; \frac{1}{\tau}\sum_i \frac{1}{-f_i(z)}\nabla^2 f_i(z).$$

Przyjmując, że z jest rozwiązaniem dopuszczalnym (tzn. $A_0 z = b$), Δz które minimalizuje $q(z+\Delta z)$ spełniające $A_0 z = b$ jest rozwiązaniem układu równań liniowych:

$$\tau \begin{pmatrix} H_z & A_0^T \\ A_0 & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Delta z \\ v \end{pmatrix} = -\tau g_z. \tag{13}$$

[3] $|f'''(x)| \le 2f''(x)^{3/2}$

(Wektor v, który może być interpretowany jako mnożniki Lagrange'a ograniczeń jakościowych w kwadratowym problemie optymalizacji, nie jest bezpośrednio wykorzystywany.)

Ponieważ w postawionym wyżej nie problemie nie występują ograniczenia równościowe, zachodzi $A_0 = \mathbf{0}$.

Mając Δz, znamy kierunek kroku metody Newtona. Długość kroku $s \leq 1$ jest tak wybrana, by

1. $f_i(z + s\Delta z) < 0$ dla każdego $i = 1, \ldots, m$,
2. Funkcjonał dostatecznie zmalał:

$$f_0(z + s\Delta z) < f_0(z) + \alpha s \Delta z \langle g_z, \Delta z \rangle,$$

   gdzie α jest arbitralnym parametrem (przyjęto $\alpha = 0.01$). To wymaganie implikuje, że rozwiązanie musi być bliskie wyznaczonemu przez model liniowy w z.

Ustawiamy $s = 1$ i zmniejszamy o wielokrotność β aż oba z tych warunków zostaną spełnione (przyjęto $\beta = 1/2$).

Ostatecznie, implementacja metody log-barrier przedstawia się w następującej postaci algorytmicznej:

1. Wejścia: rozwiązanie dopuszczalne z^0, tolerancja η, parametry μ i początkowe τ^1. Ustaw $k = 1$.
2. Rozwiąż (12) metodą Newtona, wykorzystując z^{k-1} jako punkt startowy. Rozwiązanie nazwij z^k.
3. Jeżeli $m/\tau^k < \eta$, zakończ i zwróć z^k.
4. W przeciwnym wypadku, ustaw $\tau^{k+1} = \mu\tau^k$, $k = k + 1$ i idź do kroku 2.

W praktyce możemy z góry przewidzieć jak wiele iteracji będzie potrzebnych:

$$\text{liczba iteracji} = \left\lceil \frac{\log m - \log \eta - \log \tau^1}{\log \mu} \right\rceil.$$

Ostatnim problemem jest wybranie τ^1. Dostało ono ustawione tak, by m/τ^1 po pierwszej iteracji było równe $\langle c_0, z^0 \rangle$.

B Dokumentacja techniczna projektu

B.1 Dokumentacja oprogramowania

B.2 Środowisko programistyczne

B.3 Kompilacja programu

B.4 Użycie funkcji

C Spis zawartości dołączonego nośnika(płyta CD ROM)

D Historia zmian

Autor	Data	Opis zmian	Wersja
Karol Olko	23.01.2014	Pierwsza wersja raportu	0.0
Karol Olko	29.01.2014	Uzupełnienie wstępu i analiza, zalążek koncepcji	0.1