Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji

Kierunek: **Informatyka techniczna**
Specjalność: **Grafika i systemy multimedialne**

PRACA DYPLOMOWA MAGISTERSKA

Autorska implementacja algorytmów transformacji Hough'a dla wybranych metod akceleracji obliczeń w środowiskach języka JavaScript

Propietary implementation of Hough transformation algorithms for selected calculation acceleration methods in JavaScript language environments

Damian Koper

Opiekun pracy
Dr inż. Marek Woda

WROCŁAW, 2022

Streszczenie

blabla

Słowa kluczowe: raz, dwa, trzy, cztery

Abstract

blabla in english

Keywords: one, two, three, four

Spis treści

Spis rysunków

Spis tabel

Spis listingów

Skróty

EXAMPLE (ang. *Example*)

Rozdział 1

Wstęp

Prawo Moore'a mówi, że liczba tranzystorów w układach scalonych podwaja się co około dwa lata. Prawo Koomey'a opisuje natomiast trend wzrostu liczby obliczeń na jeden dżul energii, która podwaja się co 1.57 lat. Choć w ostatnich latach, w związku ze zmniejszającym się tempem miniaturyzacji tranzystorów, wartości te przestały być aktualne to wciąż mamy do czynienia ze zjawiskiem ustawicznego wzrostu mocy obliczeniowej. Dodatkowo, zgodnie z obserwacją nazwaną prawem Huang'a - prezesa firmy NVIDIA, wzrost wydajności układów graficznych wzrasta więcej niż dwukrotnie co dwa lata[3], co świadczy o obecnym rozwoju możliwości optymalizacji architektur i programów wykorzystujących przetwarzanie masowo równoległe. Wzrost wydajności z kolei zwiększa możliwości wykorzystania algorytmów, które wcześniej były zbyt intensywne obliczeniowo i nie mogły być wykorzystane w rozwiązaniach produkcyjnych. Powszechnie dostępne wydajne maszyny dają również możliwości szybszego prototypowania rozwiązań większej liczbie osób, co z kolei wpływa na rozwój samych algorytmów i ich zastosowań. Algorytmy i obliczenia opierają się na osiągnięciach analizy numerycznej oraz matematyki dyskretnej.

Analiza numeryczna zajmuje się opisywaniem i analizą metod pozyskiwania wyników dla problemów matematycznych. Dzięki jej osiągnięciom możliwe jest budowanie algorytmów, które są kompletnym i jednoznacznym opisem metody konstruowania rozwiązania owych problemów. Konstruowane algorytmy mogą mieć różne poziomy skomplikowania zaczynając od obliczania wartości funkcji, wielomianów, czy też znajdować rozwiązania układów równań. Dzięki nim możemy aproksymować wartości funkcji - obliczać pochodne oraz całki korzystając z metod prostokątów, trapezów, czy Simpson'a. Możemy obliczać przybliżenia funkcji trygonometrycznych korzystając z szeregów Taylor'a. Dzięki obliczeniom opartym o algebrę liniową i rachunek różniczkowy mogły rozwinąć się pola związana z uczeniem maszynowym. Znajdowanie wektorów i wartości własnych macierzy w metodzie PCA w celu redukcji wymiarowości danych, a w końcu algorytmy propagacji wstecznej szczególnie wykorzystywane w intensywnie rozwijającym się obszarze uczenia głębokiego[4].

Niezależnie od skali zaawansowania algorytmów dążą one do stanowienia rozwiązania konkretnych problemów świata rzeczywistego. Przykładem ich zastosowania jest przewidywanie pogody w oparciu o modele meteorologiczne, które zaczęło intensywnie rozwijać się wraz z dostępem do coraz większej mocy obliczeniowej i udoskonalaniem samych modeli. W ciągu 15 lat, od 1971 roku, spełnialność prognoz 36-godzinnych zrównała się ze spełnialnością prognoz 72-godzinnych[2]. Kolejnym wartym przytoczenia przykładem jest obszar analizy obrazów z wyszczególnieniem zagadnienia ich klasyfikacji, gdzie wzrost mocy obliczeniowej pozwolił na rozwój algorytmów. W ciągu 8 lat metryka dokładności klasyfikacji obrazów na zbiorze danych ImageNet wzrosła z 63% do 90%[2, 1].

Dynamiczny rozwój algorytmów napędzany jest przede wszystkim przez poszerzanie i budowanie wiedzy domenowej, specyficznej dla rozwiązywanego problemu. Jednak, aby uczynić taki rozwój możliwym, musi być on oparty na niezbędnych filarach jakimi są oprogramowanie, które służy do prototypowania, a następnie wdrażania rozwiązań oraz środowisko sprzętowe, które umożliwia przeprowadzanie obliczeń, wielokrotnych testów, prototypów na małą oraz wdrożeń na dużą skalę. Wraz ze wzrostem złożoności problemu liczba obliczeń niezbędnych do jego rozwiązania również rośnie i w przypadku ich większości ten wzrost jest wykładniczy lub większy. Niezbędnym zatem jest, aby oprogramowanie potrafiło wykorzystać wszelkie dostępne metody akceleracji zarówno te związane z optymalizacją samego algorytmu, jak i te związane z mechanizmami akceleracji sprzętowej.

1.1. Istota rzeczy

Wykonywanie obliczeń wśród obecnie popularnych rozwiązań można podzielić na dwie grupy. Pierwsza z nich używa specjalnie do tego celu stworzonego języka programowania, często również w połączeniu ze zintegrowanym środowiskiem programistycznym (Integrated Development Environment, ang. IDE). Przykładem takiego rozwiązania jest środowisko i język MATLAB[6] oraz R[9], czy też środowisko Mathematica z językiem Wolfram[5].

Druga grupa używa języka ogólnego przeznaczenia do wykonywania obliczeń w oparciu o zewnętrzne biblioteki w zdecydowanej większości udostępniane jako oprogramowanie open-source, które dostarczają wymagany zestaw funkcjonalności niwelując potrzebę ich ręcznej implementacji. Języki takie możemy podzielić na te niskiego poziomu, zapewniające wysoką wydajność, oraz te wysokiego poziomu, interpretowane, zapewniające większą wygodę użytkowania. Najbardziej popularnym tego typu środowiskiem jest język Python, którego społeczność stworzyła liczne biblioteki (w postaci pakietów) do przetwarzania danych, obliczeń numerycznych i statystycznych, analizy obrazów, czy uczenia maszynowego. Najpopularniejsze z nich podane zostały w tabeli 1.1. W nawiasach została ujęta liczba pobrań danego pakietu w przeciągu ostatniego tygodnia na dzień 2022-04-27. Dla punktu odniesienia warto dodać, że w przeciągu tego samego tygodnia pobrano łącznie 3.409.997.407 pakietów [8].

Biblioteki takich języków, czego przykładem jest biblioteka OpenCV, często implementowane są w językach kompilowanych bezpośrednio do kodu maszynowego takich jak C++ czy Rust udostępniając jednolity interfejs, którego metody poprzez powiązania wywoływać mogą języki skryptowe wysokiego poziomu takie, jak już wspomniany Python. Takie rozwiązania zapewnia możliwość zbudowania wersji biblioteki kompatybilnej z wieloma środowiskami i językami na podstawie jednego kodu bazowego, poddając adaptacji tylko na linii biblioteki niskopoziomowej i języka wysokiego poziomu. Skompilowana biblioteka poddana może być również procesom optymalizacji, co zwiększyć może jej wydajność. Wadę takiego rozwiązania stanowi konieczność kompilowania biblioteki niskiego poziomu dla wielu systemów operacyjnych, czy architektur procesora. Taka kompilacja odbywać może się przed publikacją samej biblioteki, a gotowe artefakty pobierane są przez użytkownika w momencie instalacji. Kompilacja może odbywać się również bezpośrednio na maszynie użytkownika w momencie instalacji.

Opisany podział nie skłania do uznania przewagi jednej z grup nad drugą w żadnym z aspektów. W środowiskach specyficznych i zintegrowanych brakujące funkcjonalności mogą zostać dodane przez twórców jako biblioteki standardowe języka oraz zaimplementowane przez społeczność w bibliotekach języków ogólnego przeznaczenia. Obie grupy rozwiązań z reguły są w stanie działać w środowisku tego samego systemu operacyjnego i wchodzić w interakcje z tym samym sprzętem, i wykorzystać idące za tym możliwości akceleracji obliczeń.

1.1.1. Duża i rosnąca rola technologii webowych

Technologie webowe zdefiniować można jako narzędzia i techniki umożliwiające wymianę danych pomiędzy różnymi urządzeniami przez internet. Technologie webowe mogą występować i spełniać różne zadania na wielu poziomach architektury aplikacji. W przypadku architektury klient-serwer środowiskiem frontend'owym umożliwiającym wyświetlanie i obsługę graficznego interfejsu użytkownika zwykle jest przeglądarka internetowa, gdzie za jego implementację na najniższym poziomie abstrakcji odpowiadają języki HTML, CSS i JavaScript. Serwerem może być aplikacja komunikująca się z klientem za pomocą API zaimplementowanego w architekturze REST, czy też za pomocą języka zapytań GraphQL uruchamiana w środowisku NodeJS.

Warto zaznaczyć, że serwer, jak i klient, nie muszą być zaimplementowane z wykorzystaniem języka JavaScript by być uznawanym jako aplikacja webowa. Języki takie jak PHP, Python, Ruby, Java, czy C# także oferują zaawansowane frameworki, jednak to język JavaScript, ze względu na historię swojego rozwoju ściśle powiązaną z przeglądarką internetową, uważany jest jako główne narzędzie i czynnik rozwoju technologii webowych zarówno w części klienckiej jak i serwerowej. Potwierdzają to badania, gdzie JavaScript w 2021 roku dziewiąty rok z rzędu został wyłoniony jako najpopularniejsza technologia wśród developerów [10].

Tab. 1.1: Popularne biblioteki do przetwarzania danych w języku Python i ich odpowiedniki w języku JavaScript. Dane pochodzą z serwisów kolejno PyPI Stats oraz NPM, a tygodniowa liczba pobrań znajduje się w nawiasach.

Python	**JavaScript**	**Zastosowanie**
numpy (26.067.844)	numjs (533)	Operacje na macierzach
pandas (19.778.648)	danfojs (1.029)	Operacje na strukturach danych
scipy (9.986.376)	simple-statistics (87.882) fft.js (8.027)	Operacje związane z analizą numeryczną, przetwarzanie sygnałów, algebra liniowa.
scikit-learn (7.705.438)	ml (156)	Uczenie maszynowe
matplotlib (6.457.099) plotly (1.632.246)	plotly.js (149.542) c3 (83.564)	Wizualizacja danych
tensorflow (3.348.986)	@tensorflow/tfjs (91.233)	Sieci neuronowe
opencv-python (1.236.711)	OpenCV.js (b.d [7]) jimp (1.479.783) image-js (4.103)	Operacja na obrazach, computer vision

JavaScript w obliczeniach numerycznych

Jednak pomimo swojej popularności, język JavaScript nie zyskał popularności wśród zadań obliczeń inżynierskich, naukowych, statystycznych, obróbki i analizy obrazów w przeciwieństwie do języka Python. Jest on starszy od języka JavaScript i w przeciwieństwie do niego od początku mógł pełnić zadanie narzędzia do implementacji obliczeń, podczas gdy JavaScript ograniczony był jedynie do środowiska przeglądarki internetowej. Dopiero w 2009 roku, wraz z pojawieniem się środowiska NodeJS, możliwe stało się uruchamianie kodu JavaScript po stronie serwera. Innym środowiskiem umożliwiającym uruchamianie kodu języka JavaScript po stronie serwera jest powstały w 2018 roku Deno.

Innym, czynnikiem

Obecnie, z technicznego punktu widzenia, nie istnieją zasadnicze różnice pomiędzy środowiskami języków Python i JavaScript, które hamowałyby w znacznym stopniu rozwój bibliotek

służącym do prototypowania i wdrażania aplikacji zajmującymi się obliczeniami numerycznymi w środowiskach języka JavaScript.

W tabeli 1.1 przedstawiono porównanie popularnych bibliotek stosowanych przy obliczeniach numerycznych języka Python z ich odpowiednikami z języka JavaScript. Widać wyraźnie dysproporcję w liczbie pobrań bibliotek pomiędzy językami. W wielu przypadkach, kiedy autorowi nie udało się znaleźć dokładnego odpowiednika danej biblioteki, a znalezione zestawy bibliotek języka JavaScript nie pokrywają w całości funkcjonalności biblioteki języka Python.

- Czy nadają się do obliczeń
- - pod kątem istniejących rozwiązań
- - pod kątem community, bibliotek i metod interakcji
- problem wielu środowisk i niekompatybilności
- - API
- - budowanie

1.2. Cel badań i zawartość pracy

1.2.1. Analiza obrazów

1.2.2. Analiza obrazów

1.2.3. Analiza obrazów

1.2.4. Analiza obrazów

- ...jako podzbiór algorytmów intensywnych obliczeniowo
  - - sieci neuronowe,
  - - algorytmy deterministyczne
  - potrzeby, zastosowania, ograniczenia

Rozdział 2

Język JavaScript

2.1. Model wykonywania

2.2. Środowiska

2.3. Metody akceleracji

Rozdział 3

Transformacja Hough'a

3.1. Standard Hough Transform

3.2. Circle Hough Transform

Rozdział 4

Metodologia pomiarów

Rozdział 5

Zaimplementowane algorytmy

Rozdział 6

Wyniki pomiarów

Rozdział 7

Podsumowanie

asdsad

asdsad

Literatura

[1] Z. Dai, H. Liu, Q. V. Le, and M. Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34:3965–3977, 2021.

[2] P. Lynch. The origins of computer weather prediction and climate modeling. *Journal of computational physics*, 227(7):3431–3444, 2008.

[3] C. Mims. Huang's law is the new Moore's law, and explains why Nvidia wants arm. *The Wall Street Journal*, Sep 2020.

[4] G. M. Phillips and P. J. Taylor. *Theory and applications of numerical analysis*. Elsevier, 1996.

[5] Mathematica. `https://www.mathematica.pl/` Na dzień 2022-04-24.

[6] Matlab. `https://www.mathworks.com/products/matlab.html` Na dzień 2022-04-24.

[7] Opencv.js. `https://docs.opencv.org/4.x/d4/da1/tutorial_js_setup.html` Na dzień 2022-04-27.

[8] Pypi stats. `https://pypistats.org/packages/__all__` Na dzień 2022-04-27.

[9] R. `https://www.r-project.org/about.html` Na dzień 2022-04-24.

[10] Stackoverflow 2021 developer survey. `https://insights.stackoverflow.com/survey/2021` Na dzień 2022-04-27.

Dodatek A

Opis załączonej płyty CD/DVD

Dołączona płyta zawiera wszystkie pliki wykorzystane do stworzenia projektu, dokumentacji i niniejszej pracy. Pliki podzielono na następujące katalogi:

1. `.vscode` - ustawienia środowiska VsCode,