Seminarska naloga: Seam Carving

Amon Stopinšek
63150273

Vzporedni in porazdeljeni sistemi in algoritmi
Fakulteta za računalništvo in informatiko Univerze v Ljubljani

14. januar 2018

1 Opis problema

Rezanje šivov je algoritem za spreminjanje velikosti slik. Algoritem upošteva vsebino slike tako, da pride pri spreminjanju velikosti do čim manjšega popačenja. V vsakem koraku se odstrani šiv (povezano zaporedje pikslov od vrha do dna ali od levega do desnega roba slike) z najmanj pomembno vsebino.

Algoritem:

1. Izračunaj energijo vsakega piskla na sliki (npr. z uporabo sobelovega filtra za detekcijo robov).
2. Poišči najmanjši šiv (npr. izračunaj vse šive z dinamičnim programiranjem in poišči najmanjši šiv).
3. Odstani šiv.
4. Ponovi celoten postopek dokler slika ni želene velikosti.

2 Opis arhitekture in pristopa k reševanju

Implementirali smo dve verziji algoritma - serijsko in paralelno.

V obeh implementacijah smo za računanje energij uporabili sobelov filter, za iskanje najmanjšega šiva pa smo implementirali algoritem s pomočjo dinamičnega programiranja.

Posebnosti obeh implementacij so opisane v sekciji 2.2 in 2.3.

2.1 Splošne značilnosti

Za delo s slikami smo uporabili knjižico pgm. Knjižica podpira branje in shranjevanje slik v formatu pgm.

Za računanje energij smo uporabili sobelov filter.

Iskanje najmanjšega šiva smo implementirali s pomočjo dinamičnega programiranja. V vsaki vrstici za vsak piksel izračunamo skupno energijo najmanjšega možnega šiva do te vrstice. Po tem postopku v zadnji vrstici dobimo vrednosti najmanjših šivov. Izmed teh poiščemo najmanjši šiv.

2.2 Serijski algoritem

2.2.1 Iskanje in odstranjevanje najmanjšega šiva

Ob rekonstrukciji najmanjšega šiva sproti kopiramo piksle za novo sliko.

2.3 Paralelni algoritem

2.3.1 Sobelov filter

Za izračun sobelovega filtra poženemo toliko niti kolikor je pikslov na sliki. Vrednosti zunaj slike imajo vrednost 0.

2.3.2 Iskanje najmanjšega šiva

Zaradi potrebe po sinhronizaciji med nitmi poženemo samo en workgroup. Odvisno od velikosti slike vsaka nit za vsako vrstico v sliki izračuna vrednosti za nič ali več pikslov.

Najmanjši šiv nato poiščemo s pomočjo redukcije.

Rezultat iskanja je tabela z indeksi šiva.

2.3.3 Odstranjevanje šiva

Za odstranjevanje šiva poženemo toliko niti kolikor je število pikslov na originalni sliki. Vsaka nit, katere indeks ni čez širino nove slike skopira piksel iz originalne slike na ustrezno mesto na novi sliki.

3 Rezultati

Obe implementaciji smo testirali na 7 različnih slikah.

	serijski	paralelni
število meritev	245	286
povprečna pohitritev	0.6321	1.709
mediana pohitritev	0.5304	1.885
standardna deviacija pohitritev	0.2341	0.4072

Tabela 1: Povprečni rezultati

Na sliki 1 in 2 sta prikazana grafa pohitritve paralelnega algoritma v primerjavi s serijskim v odvisnosti od velikosti slike.

Na manjši sliki se je za hitrejšo izkazala serijska implementacija, na vseh preostalih slikah pa je bila hitrejša paralelna verzija.

3.1 Slika: cubes

- Velikost slike: 478px × 478px
- Število iteracij: 400

	serijski	paralelni
število meritev	50	50
povprečni čas izvajanja	0.6559	0.752
mediana	0.654	0.7497
standardna deviacija	0.003456	0.01374
pohitritev	1.1465	0.8722

Tabela 2: Rezultati za sliko cubes

3.2 Slika: tower

- Velikost slike: 1280px × 868px
- Število iteracij: 400

	serijski	paralelni
število meritev	50	50
povprečni čas izvajanja	4.597	3.079
mediana	4.596	3.07
standardna deviacija	0.007714	0.06314
pohitritev	0.6698	1.4930

Tabela 3: Rezultati za sliko tower

3.3 Slika: bike

- Velikost slike: 2592px × 1456px
- Število iteracij: 400

	serijski	paralelni
število meritev	50	50
povprečni čas izvajanja	17.11	9.076
mediana	17.1	8.952
standardna deviacija	0.02976	0.2548
pohitritev	0.5304	1.8852

Tabela 4: Rezultati za sliko bike

3.4 Slika: wires

- Velikost slike: 2592px × 1728px
- Število iteracij: 400

	serijski	paralelni
število meritev	50	50
povprečni čas izvajanja	20.31	10.4
mediana	20.3	10.39
standardna deviacija	0.04101	0.05947
pohitritev	0.5121	1.9529

Tabela 5: Rezultati za sliko wires

3.5 Slika: texture

- Velikost slike: 5160px × 3870px
- Število iteracij: 400

	serijski	paralelni
število meritev	50	25
povprečni čas izvajanja	123.4	67.85
mediana	123.7	68.52
standardna deviacija	1.602	5.775
pohitritev	0.5498	1.8187

Tabela 6: Rezultati za sliko texture

3.6 Slika: paris

- Velikost slike: 7600px × 5066px
- Število iteracij: 400

	serijski	paralelni
število meritev	10	25
povprečni čas izvajanja	249.4	122.5
mediana	249.3	121.3
standardna deviacija	0.9592	4.425
pohitritev	0.4912	2.0359

Tabela 7: Rezultati za sliko paris

3.7 Slika: nature

- Velikost slike: 15360px × 8640px
- Število iteracij: 400

	serijski	paralelni
število meritev	10	11
povprečni čas izvajanja	868	455.5
mediana	454.4	868.6
standardna deviacija	3.193	5.847
pohitritev	0.5248	1.9056

Tabela 8: Rezultati za sliko nature

3.8 Opis sistema

- CPU: Intel i5 3570k @4.2 GHz
- GPU: Nvidia GTX 1070 8 Gb
- RAM: 8Gb
- Disk: SSD Samsung 840 Pro 128 Gb
- Operacijski sistem: Manjaro Linux 4.14.2-1

4 Možne pohitritve

Nekaj hitrosti bi lahko pridobili pri računanju energij. Trenutno se po vsaki odstranitvi šiva na novo izračuna energije za celotno sliko. Hitreje bi bilo, če bi na novo izračunali le del v okolici šiva, ki smo ga odstranili.

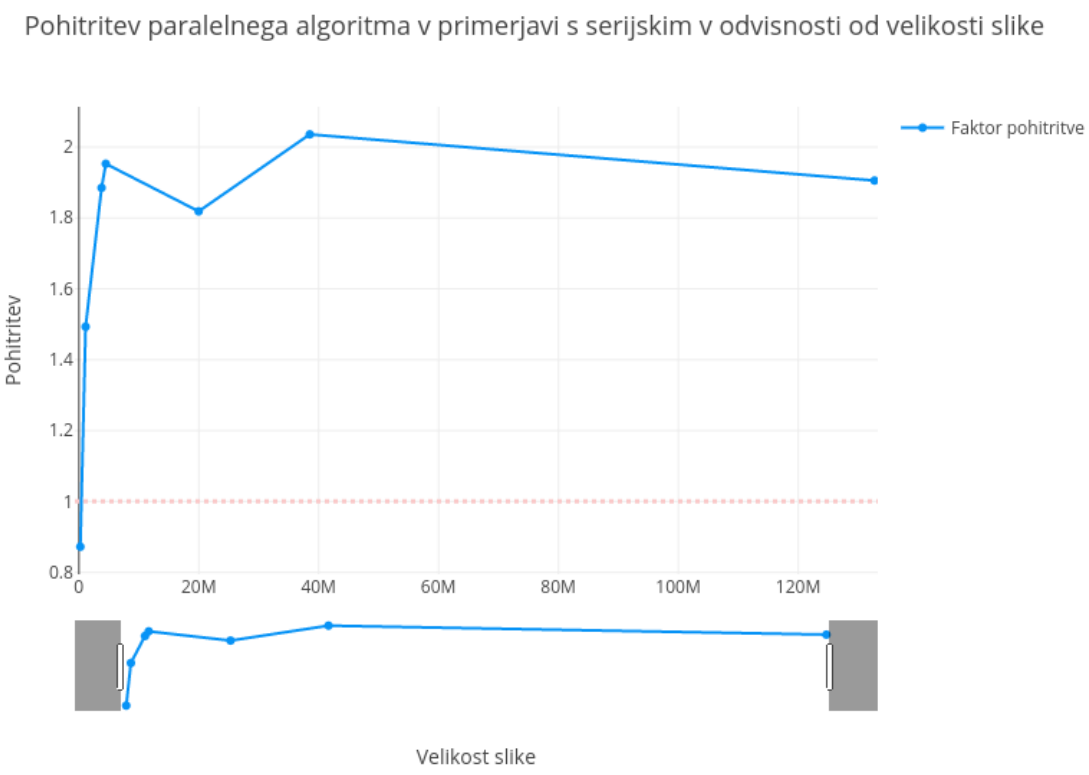
4.1 Serijski algoritem

Ob odstranjevanju šiva v zanki kopiramo vsak piksel iz prvotne slike v novo. Hitreje bi bilo če bi za to uporabili operacijo memcpy.

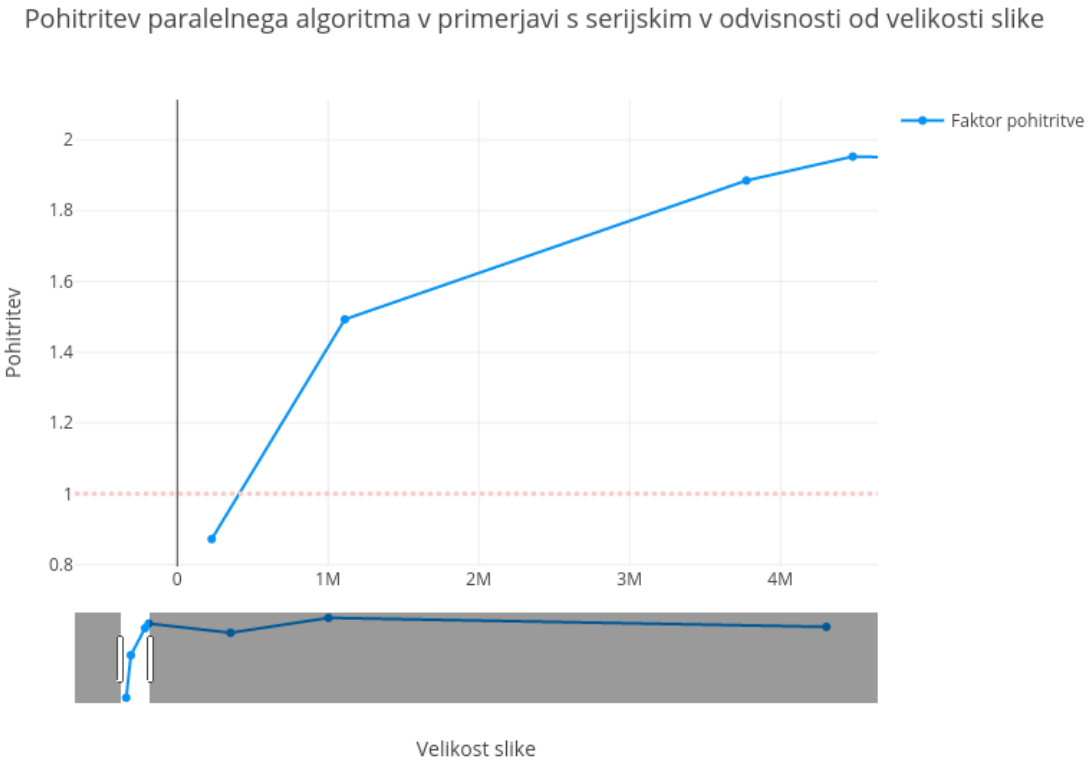
4.2 Paralelni algoritem

Zaradi sinhronizacije pri računanju šivov trenutno poganjamo le en workgroup. Dobro bi bilo preizkusiti še drugo možnost - klic kernela za izračun vsake vrstice.

5 Dodatki



Slika 1: Faktor pohitritve paralelnega algoritma.



Slika 2: Faktor pohitritve paralelnega algoritma na majhnih slikah.