

ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Semestrální práce z předmětu KIV/PPR

Transformace šedotónového obrázku do negativu pomocí MPI

Martin Sloup, A08N0111P

msloup@students.zcu.cz

20.12.1984, zadání č. 2

Zadání

Realizujte transformaci digitalizovaného šedotónového obrazu zadaného jako matice (m, n) s celočíselnými prvky. Hodnota prvků je v rozmezí **<0,k>** a udává jas příslušného pixelu. Transformace spočívá ve vytvoření "negativu" původního obrazu.

Zadanou úlohu individuálně vypracujte jako paralelní program pro systém s distribuovanou pamětí (použitelné prostředky: PVM nebo MPI)

Povinné parametry

- počet vláken (není nutné u MPI, tam se zadává počet vláken při spuštění prostředí)
- velikost kusu práce přidělovaného vláknům (opět není nutné u některých úloh v MPI)
- matice (není nutné načítat opravdové obrázky (plus body))
- hodnota **k** (nemusí být zadána explicitně - může být např. v souboru s daty)

Poznámky k řešení

- negativ se vytváří podle vzorce: new(x,y) = k - old(x,y)

Řešení

Účelem programu, je podle zadání spočítat v několika pracovních procesech, inverzi šedotónové matice vstupního obrázku. V našem případě byl použit pro matici formát **PGM** (Portable Graymap). Použití obrázku jako vstupní matice do našeho paralelního programu zajistí, že je možné testovat na reálných datech. Formát **PGM** byl zvolen z důvodu jednoduchého načítání a velmi dobré podpoře napříč konvertory obrázků. Obrázek je načítán do jednorozměrného pole, čímž zajistíme pozdější rozložení tohoto pole na bloky. U obrázku známe výšku a šířku, takže nepotřebujeme mít obrázek v dvojrozměrném poli.

Po načtení vstupního obrázku do paměti je vypočítána velikost bloku, který každý výpočetní proces dostane ke své práci. Rozdělení obrázku mezi procesy je zajištěno pomocí funkce **MPI_Gather**. Podmínkou této funkce je ale, že velikost obrázku (výška krát šířka) musí být dělitelná beze zbytku počtem výpočetních procesů. Na toto je potřeba pamatovat již při načítání obrázku a zvolit vhodnou velikost pole při alokaci paměti obrázku. Jakmile každý paralelní proces dostane svůj díl bloku, provede inverzi pomocí vzorce:

```
new(i) = k - old(i)
```

Parametr ***i*** nám reprezentuje pixel v bloku, a parametr ***k*** je zadán již v souboru s **PGM** obrázkem. Tato hodnota reprezentuje maximální hodnotu světlosti pixelu (nula zde reprezentuje černou barvu). Po výpočtu nové hodnoty pixelu pro všechny pixely bloku rodič získá pomocí **MPI_Scatter** výslednou matici celého obrázku. Tu nakonec zpět uloží do výstupního souboru, taktéž ve formátu PGM.

Formát **PGM** má dvě možné způsoby zápisu jasové matice. Jeden z nich je v **ASCII formátu**, kde na jedné řádce (odpovídá řádce obrázku) se vyskytují číselné hodnoty šedi oddělené bílým znakem (mezerou, tabulátorem, novou řádkou, atd...). Druhý možný způsob je v **binárním formátu**. Jeden pixel šedotónové matice je pak reprezentován jako jeden bajt (pro rozsah 0-255, u 0-65535 to jsou dva bajty). Aplikace je naprogramovaná tak, že ukládá ve stejném formátu, jako byl vstupní obrázek. Pokud tedy v parametru příkazové řádky není řečeno jinak.

Níže se nachází zjednodušený popis formátu obrázku:

```
P2          # Formát souboru. P2 pro ASCII, P5 pro binární (RAW) formát
3  4        # Rozměry obrázku: šířka a výška
8           # Maximální hodnota jasu
1  2  6     # Jasy jednotlivých pixelů (u binárního formátu by hodnota byla zapsána
3  0  4     # co pixel to jeden bajt, případně dva bajty – závisí na maximální
5  7  8     # hodnotě jasu)
8  1  2
```

Programátorská dokumentace

Zdrojový kód paralelního programu je obsažen v souboru invimg.c. Tento soubor obsahuje následující metody:

- **main** – hlavní metoda programu, která zpracovává příkazovou řádku a volá ostatní metody programu
- **load_file** – načítá obrázek ze souboru ve formátu PGM.
- **save_file** – ukládá obrázek do souboru ve formátu PGM.
- **random_image** – generuje náhodný obrázek o zadaných rozměrech a maximálního jasu
- **worker** – provádí paralelní výpočet inverzního obrázku

Uživatelská dokumentace

Než je možné spustit výpočet, je nutné zkompilovat program. V následující kapitole je popsán návod, jak to lze provést.

Kompilace programu

Předpokladem pro úspěšné zkompilování programu je mít funkční prostředí pro **MPI**. Program byl vyvíjen a testován na referenčním stroji hydra, kde už **MPI** bylo rozběhnuto. Program využívá ke své funkci knihovnu **libnetpbn**, která je přibalena k programu. Knihovna je již nakonfigurována pro potřeby stroje hydra, ale pokud je program použit na jiném prostředí, je nutné provést konfiguraci buildících skriptů knihovny.

V adresáři *src/netpbn/* se spustí konfigurační skript pomocí příkazu configure. Skript se zeptá na pár otázek. Pro stroj hydra byly použity výchozí hodnoty. Důležité je u otázky, zda **libnetpbn** má být linkován staticky nebo má být vytvořena sdílená knihovna, **uvést volbu static**, tedy staticky. Pokud bychom knihovnu vytvořili jako sdílenou, bylo by ji potřeba překopírovat (případně vytvořit link) do repositáře se sdílenými knihovnami. Na to jsou potřeba práva administrátora. Vystačíme si ale s volbou static, kdy knihovna libnetpbn bude přilinkována přímo do spustitelného souboru našeho paralelního programu. To má nevýhodu ve větší velikosti spustitelného souboru, který se při spuštění prostředí **MPI** déle přenáší na další stroje.

Pokud máme nakonfigurováno **libnetpbn** knihovnu, tak již stačí v adresáři *src/* spustit příkaz **make**, který zkompiluje nejen knihovnu libnetpbn, ale i samotný paralelní program.

Spuštění výpočtu

Paralelní program se spustí v **MPI** prostředí následujícími příkazy:

```
> mpirun -np <počet_procesů> invimg [-a | -b] <vstup.pgm> <výstup.pgm>
```

Provede inverzi obrázku <vstup.pgm> a uloží ho do <výstup.pgm>.

```
> mpirun -np <počet_procesů> invimg [-a | -b] <šířka> <výška> [<maximální_jas>] <výstup.pgm>
```

Vygeneruje obrázek o zadaných rozměrech, maximálním jasem (pokud není uvedeno, bere se výchozí hodnota jasu 255) a uloží ho do souboru <výstup.pgm>

V obou případech je možné uvést parametry -a či -b, které definují formát výstupního souboru (-a pro **ASCII** a -b pro binární soubor). Pokud není uveden ani jeden z těchto parametrů, volí se výstupní formát souboru dle formátu vstupního souboru. Při generování obrázku je výchozí formát **ASCII**.

K paralelnímu programu je přiloženo několik ukázkových obrázků, které je možné použít jako vstupní obrázky programu.

Závěr

Podle zadání byl vytvořen nástroj pro transformaci dvourozměrného pole (obrázku) do negativu. Pro vytvoření paralelního zpracování bylo použito prostředí MPI. Zároveň s tím byla použita knihovna libnetpbn k načítání a ukládání obrázku. V aplikaci nechybí implementace generátoru náhodného obrázku pro případ, že uživatel nebude mít schopnost vytvořit si obrázek svépomocí. Práce na této semestrální práce byla pro mě zajímavá a poučná.

Použité zdroje

- Libnetpbm manual: Netpbm projekt na Source Forge [cit. Prosinec 2003]. Dostupný z WWW: <http://netpbm.sourceforge.net/doc/libnetpbm.html>
- Herout, Pavel: Učebnice jazyka C. IV přepracované vydání. České Budějovice: Nakladatelství Kopp, 2005. ISBN 80-7232-220-6
- Herout, Pavel: Učebnice jazyka C, 2. díl. II přepracované vydání. České Budějovice: Nakladatelství Kopp, 2004. ISBN 80-7232-221-4

Přílohy

Vývojový diagram

