Transformace digitalizovaného obrazu

KIV/PPR
Martina Málková (tina.malkova@centrum.cz) *15.10.1984

1 Zadání

Realizujte transformaci digitalizovaného obrazu zadaného jako matice (m,n) s celočíselnými prvky. Hodnota prvku je v rozmezí <0,k> a udává jas příslušného pixelu. Transformace spočívá v "roztažení" jasového rozsahu (šedivého) obrazu do celého intervalu <0,k>.

Povinnými parametry jsou matice (načítat ze souboru) a hodnota k.

Nejprve je nutné nalézt nejmenší/největší prvek (první paralelní běh), následné roztažení se provede podle vzorce new(i,j) = (old(i,j)-Min)*(k/(Max-Min)) (druhý paralelní běh).

Úlohu vypracujte ve dvou verzích:

1) Paralelní program pro systém se sdílenou pamětí (použitelné prostředky: prog. jazyk Java nebo vlákna POSIX).
2) Paralelní program pro systém s distribuovanou pamětí (použitelné prostředky: PVM nebo MPI) .

1.1 Doplněk zadání

Matice bude obsahovat hodnoty 0-255 reprezentující jasové hodnoty (0 = černá, 255 = bílá).

Pro systém se sdílenou pamětí byl použit jazyk Java. Zde se načítají přímo obrázky ve formátu *jpg,bmp* nebo *png*. Není možno zde vygenerovat matici náhodných čísel. Je možno ovlivňovat hodnotu *k*, počet vláken a část zpracovanou v jednom průchodu vlákna (přidělovanou hlavním vláknem pracujícím vláknům).

Pro systém s distribuovanou pamětí bylo použito MPI. Obrázky jsou buď generovány náhodně (zde je zadán konstantní rozměr 800x600), nebo načítány ze zjednodušeného formátu souboru. Ten bude popsán v uživatelském manuálu. Lze zde ovlivňovat pouze počet procesů a hodnota *k*.

2 Řešení – obecná část & programová dokumentace

2.1 Java

Obrázek je nejprve načten jako pole rgb hodnot uložených v typu *int*. To je umožněno použitím knihovny *java.awt.Image*. Načtené hodnoty převedeme do odstínů šedi pomocí vzorce:

*jas = 0.299 * červená + 0.587 * zelená + 0.114 * modrá*

Pro správné zobrazení obrázku je pak nutné tuto hodnotu přiřadit každé složce barvy. Pro správný převod obrázku je ale potřeba pole obsahující pouze hodnoty jasu. Protože chceme šetřit operační paměť, vytvoříme tedy pouze pole s jasovými hodnotami, a až při zobrazení obrázku bude pomocí lokálního pole převedeno na obrázek.

2.1.1 Převod matice obsahující jasové hodnoty

Převod je implementován pomocí modelu Farmer-Worker. Je vytvořen jeden Farmer (Farmer.java), který reprezentuje monitor. Obsahuje dvě synchronized metody *SetMinMax(int min, int max)* a *inc()*.
Výpočet probíhá tak, že na začátku spustí Farmer daný počet vláken (Workerů – třída WorkerMinMax.java) hledajících minimum a maximum v poli. Postupně podle toho, který už dopočítal, přiděluje další práci, dokud není projité celé pole. Vlákna se sejdou na bariéře v metodě SetMinMax(), kterou volají při každém nalezení lokálního minima a maxima. Poté, co se sejdou, jsou tyto vlákna ukončeny. Vytvoří se nová vlákna (třída Worker.java) počítající nové jasové hodnoty v obrázku. Ta se synchronizuje pomocí metody *inc()*, pracuje stejným způsobem jako první skupina vláken.

Nakonec se překreslí obrázek na obrazovce. Původní hodnoty obrázku jsou přepisovány, aby nebylo alokováno příliš mnoho paměti (samotná matice obrázku zabírá velký díl paměti). Efekt bohužel je, že se nejde vrátit k minulé verzi jasových hodnot obrázku, při neúspěchu (např. zadáme maximální jas 0 – celý obrázek se vynuluje) je uživatel nucen znovu načíst obrázek.

2.2 MPI

Protože narozdíl od Javy bylo MPI novinkou přinesenou tímto předmětem, na úvod stručně několik slov jak pracuje.

MPI využívá SPMD model paralelního výpočtu, tedy vyrobí se pouze jeden spustitelný soubor programu a ten se zavede do všech procesorů. V každém procesoru běží jen jeden proces. Všechny procesy běží podle téhož programu, lze je rozlišovat pomocí jejich ID.

MPI lze využít v programovacích jazycích C nebo Fortran. Zde byl využit jazyk C a knihovna *mpi.h.* Z *mpi.h* byly využity následující funkce (vypsány už s použitými parametry):

MPI_Init(&argc, &argv);	inicializace
MPI_Comm_rank(MPI_COMM_WORLD, &id_proc);	zjistí id procesu (sebe sama)
MPI_Comm_size(MPI_COMM_WORLD, &n_proc);	zjistí počet všech procesů
MPI_Finalize();	ukončení výpočtu
MPI_Bcast(&k, 1, MPI_INT, 0, MPI_COMM_WORLD);	rozešle hodnotu *k* všem procesům
MPI_Scatter(img, step, MPI_INT, img_part, step, MPI_INT, 0, MPI_COMM_WORLD);	rozdělí data z pole img jednotlivým procesům (do jejich lokálního pole img_part)
MPI_Allreduce(&tmp_min, &min, 1, MPI_INT, MPI_MIN, MPI_COMM_WORLD);	sesbírá lokální hodnoty minima od všech procesů, tím zároveň zajistí jejich synchronizaci
MPI_Gather(img_part, step, MPI_INT, img, step, MPI_INT, 0, MPI_COMM_WORLD);	sesbírá vypočítané hodnoty v img_part zpět do globální proměnné (pole) img

2.2.1 Výpočet

Matice je reprezentována jako jednorozměrné celočíselné pole, jak je u obrázků obvyklé. V souboru je ale zapsána běžným maticovým zápisem (na řádce souboru řádek matice).
Nejprve se podle zadaných parametrů inicializují hodnoty *k*, matice, vstupního a výstupního souboru. To provádí pouze proces ROOT (ten má vždy ID 0).

Následně ROOT distribuuje parametr *k* ostatním procesům a všichni (včetně ROOTa) začínají pracovat. Rozdělí si práci a počítají hodnoty min, max, které se pak sloučí do globálních extrémů pomocí jmenované funkce *MPI_Allreduce*. Pak všechny procesy počítají nové hodnoty matice, které se sesbírají do původní matice (opět dochází k přepisování hodnot) pomocí funkce *MPI_Gather*.

Na závěr proces ROOT uloží výsledné hodnoty do výstupního souboru a všechny procesy ukončí svou činnost.

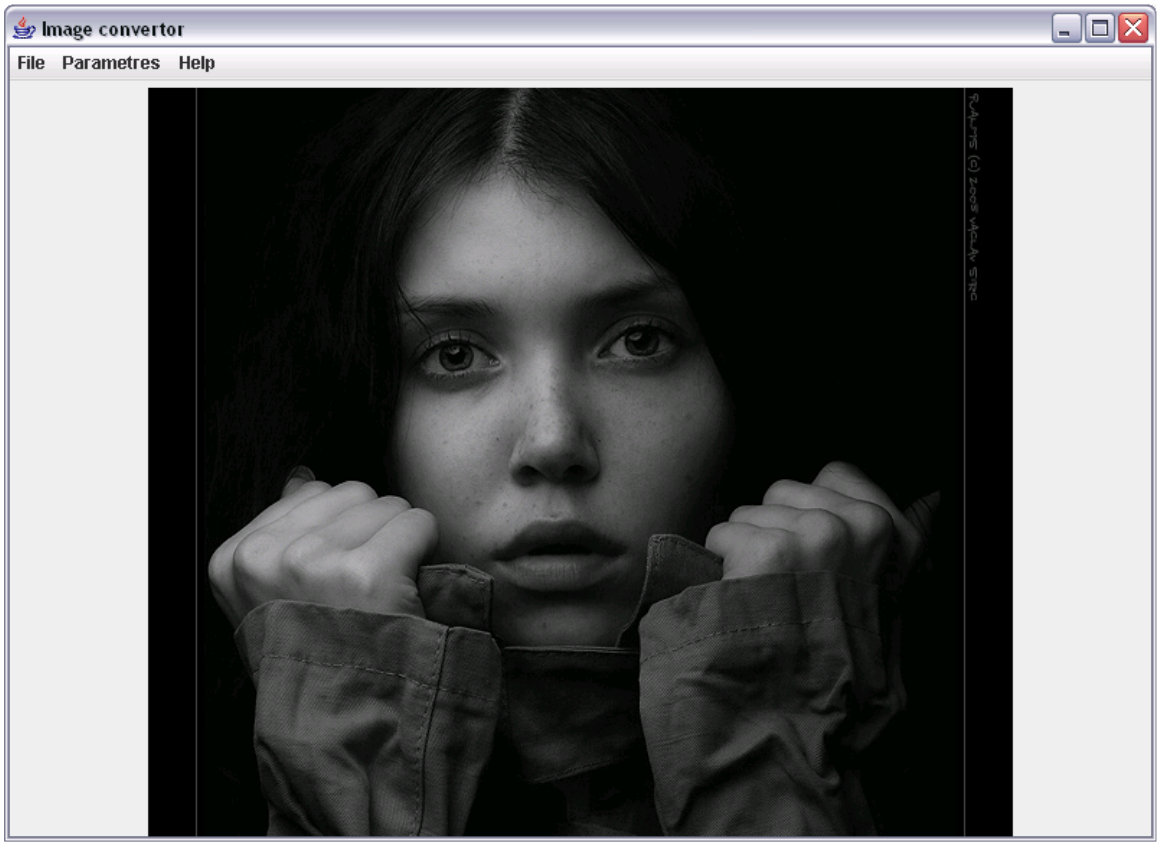
3 Uživatelská dokumentace

3.1 Java – convert.jar

Po spuštění *convert.jar* se zobrazí prázdné okno s názvem „Image convertor“ a nabídkou menu obsahující položky *File*, *Parametres*, *Help*.

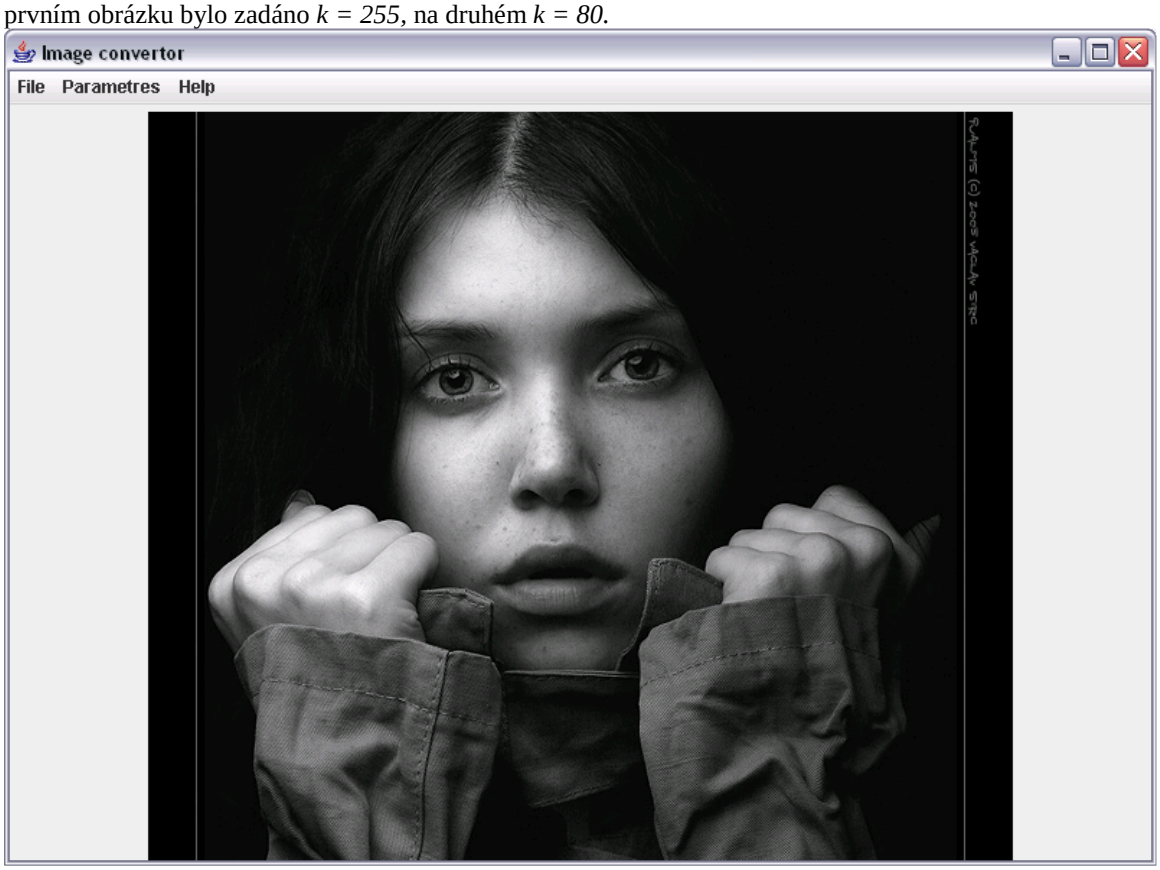
- **File –** slouží pro otevírání, převod a ukládání souborů, ukončení programu.
  - **convert** – převod obrázku. Pro použití této volby je nutné mít načtený nějaký obrázek, jinak volba nefunguje.
- **Parametres –** zde lze zadat parametry ovlivňující průběh a výsledek převodu obrázku (*convert*).
  - **No. of processes** – počet vláken, které budou přepočítávat obrázek.
  - **Amount of work** – velikost dat, které bude zpracovávat jeden proces v jednom průběhu (při dokončení práce dostane nový kus k zpracování – viz model *Farmer-Worker*).
  - **k** – maximální hodnota jasu. Při volbě *convert* se obrázek převede z původního jasového rozsahu do rozsahu <0,k> dle této zadané hodnoty.
- **Help** – pro zobrazení informací o programu a autorovi.

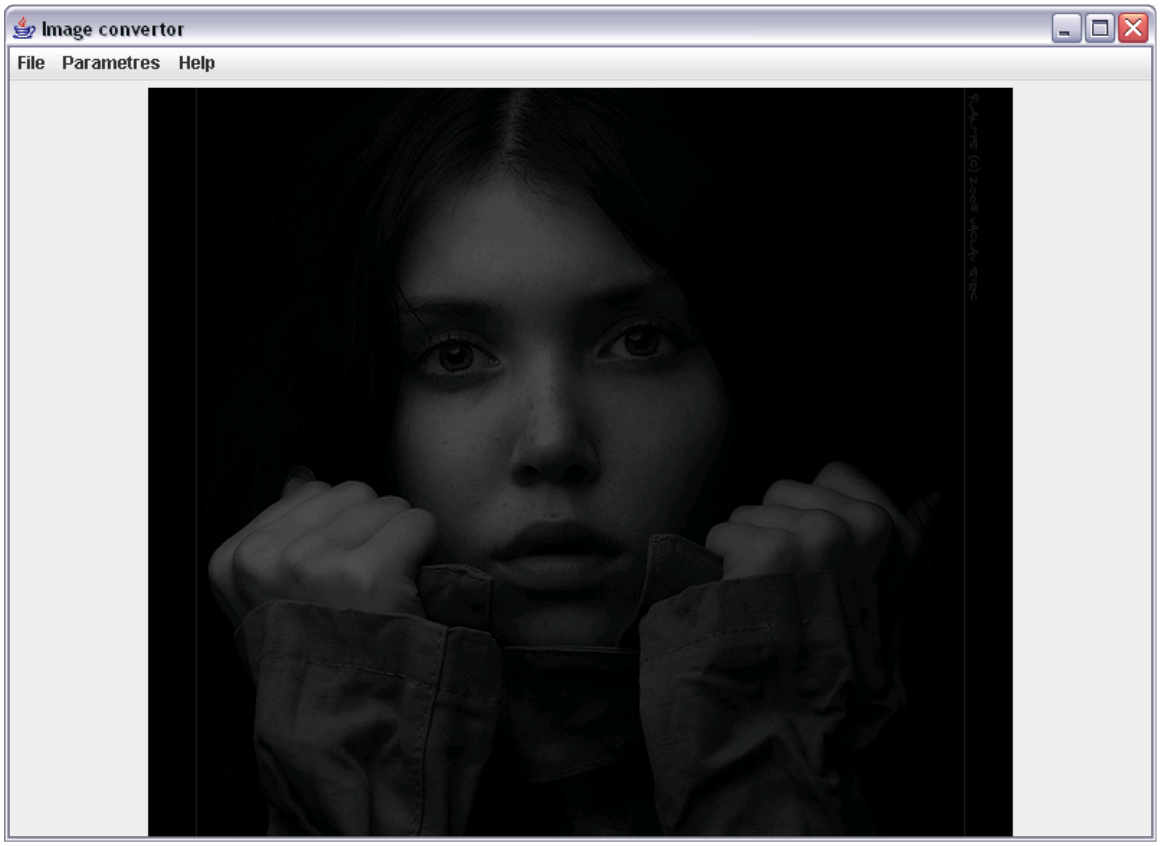
Při volbě otevření obrázku by se měl obrázek zobrazit v okně. V případě, že se tak nestane nebo je zobrazena zpráva o nepovedeném čtení, je nutné zvolit jiný obrázek. Následující screenshot ukazuje správně načtený obrázek:



Okno se bohužel nepřizpůsobuje obrázku, má stále stejnou velikost, tedy je možné, že při zobrazení velkého obrázku uvidíte pouze levý horní roh.

Hodnota *k* ovlivňuje maximální jas na obrázku (minimální je vždy 0). Maximální hodnota *k* je 255, tedy bílá barva. Na následující sérii obrázků lze vidět výsledek při zadání *k* a následujícím kliknutím na *convert*. Na prvním obrázku bylo zadáno *k = 255*, na druhém *k = 80*.





3.2 MPI – convert.c

Program se spouští pomocí příkazu

```
mpirun -np <n_proc> convert <k> [<in_file>] [<out_file>]
```

Parametr *n_proc* je povinný, určije počet procesů použitých při výpočtu. Parametr *k* určuje již jmenovanou maximální hodnotu jasu. Dále jsou volitelné parametry *in_file* a *out_file*, které určují vstupní a výstupní soubor. Pokud je zadán jen jeden soubor, je považován za výstupní, matice je vygenerována náhodně a uložena do souboru *‚in.kro'*. Při nezadání žádného souboru je vygenerována opět náhodně matice, navíc je uložen výsledek do souboru s názvem *‚out.kro'*. Parametr *k* musí být mezi 0 a 255, jinak je použita implicitní hodnota 150.

Ukázka formátu souboru:

```
8 4

19 19 19 19
19 141 91 82
19 91 82 135
19 82 135 129
19 82 36 148
19 14 0 133
19 135 148 68
19 84 57 67
```

První řádek obsahuje počet řádek a počet sloupců matice, za oběma čísly je mezera. Následuje jeden volný řádek a výpis matice po řádkách. Opět je za každým číslem (i posledním na řádku) mezera.

Na příponě souboru nezáleží, ale musí být v textovém formátu.

4 Závěr

Program v Javě byl spouštěn na operačním systému Windows, verze javy 1.5.0_09, ale byl otestován i v systému Linux. Byly zjištěny problémy nedostatku paměti při velkém obrázku, které se nepodařily odstranit.
Program s použitím MPI byl spouštěn na serveru *hydra.fav.zcu.cz.*
Oba programy (alespoň se zdá) splňují základní zadání, program v Javě umí navíc pracovat se skutečnými obrázky.

Práce mi přinesla spousty trápení s vlákny v Javě, přestože jsem je už znala z předchozích předmětů. Naopak MPI, které bylo pro mě novinkou, bylo snadno pochopitelné i neprogramovatelné.
Také jsem se naučila pracovat s obrázky v Javě, což jsem si chtěla již dlouho vyzkoušet. Objevila zajímavou skutečnost s převáděním obrázků do stupňů šedi, kdy v rovnici nejsou složky zastoupeny rovnoměrně.
Všeobecně mi toto téma jako člověku z oboru počítačové grafiky přišlo dostatečně zajímavé (i přes svou jednoduchost).