VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

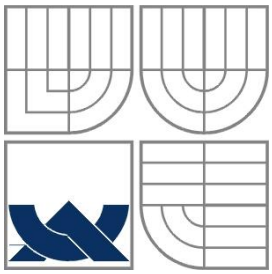
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AKCELERACE ZPRACOVÁNÍ OBRAZU V GP-GPU A OPENCL

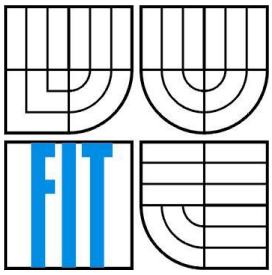
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE PAVEL GRIM
AUTHOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AKCELERACE ZPRACOVÁNÍ OBRAZU V GP-GPU A OPENCL

ACCELERATION OF IMAGE PROCESSING IN GP-GPU USING OPENCL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PAVEL GRIM

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Dr. Ing. PAVEL ZEMČÍK

BRNO 2012

Abstrakt

Tato práce studuje možnosti získání co nejkvalitnějšího obrázku z videa, které obsahuje konstantní záběr jediné věci, a navrhnutí a vytvoření programu, který by toto video takto zpracovával. V prácí se nachází popis několika metod pro zlepšení kvality obrázků a jsou zde uvedeny algoritmy, na kterých jsou některé z těchto metody závislé. Práce se také zaměřuje na využití standardu OpenCL. V práci je navrhnut a popsán program, který zpracovává video a vytváří z něj jediný kvalitnější snímek. Na konci práce jsou nastíněna další možná pokračování této práce.

Abstract

This work studies the possibility of obtaining the highest quality image from a video that contains a constant shot on one thing and proposes and creates a program that processes a video in this way. This work provides a description of several methods for improving the quality of images and algorithms on which some of these methods depend. The work also focuses on the usage of OpenCL standard. In this work there is a proposed and described program that processes the video and creates from it one better quality picture. At the end of this work there are outlined some possible continuations of this work.

Klíčová slova

Zpracování obrazu, super rozlišení, interpolace, registrace obrázků, OpenCL.

Keywords

Image processing, super resolution, interpolation, image registration, OpenCL.

Citace

Grim Pavel: Akcelerace zpracování obrazu v GP-GPU a OpenCL, bakalářská práce, Brno, FIT VUT v Brně, 2012

Akcelerace zpracování obrazu v GP-GPU a OpenCL

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením p. Doc. Dr. Ing. Pavla Zemčíka.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Grim
16. května 2012

Poděkování

Tímto bych chtěl poděkovat p. Doc. Dr. Ing. Pavlu Zemčíkovi za poskytnuté rady při řešení problémů a za cenné rady a pomoc při psaní práce.

Obsah

Seznam obrázků

1 Úvod

Během posledního desetiletí prošlo fotografování a pořizování filmů zásadními změnami. Tyto změny byly způsobeny rozvojem elektroniky a výpočetní techniky. Díky těmto změnám je nyní možné pořizovat fotografie a video přímo v digitální podobě. Digitální podobu fotografie a videa lze snadno ukládat, zobrazovat a zpracovávat počítačem. Počítače bývají také velmi často používány k archivaci, zobrazování fotografií a video sekvencí, případně jejich editaci a dalším drobným úpravám. Samotné zpracování ale může jít dále a umožňuje například skládání snímků tak, aby vznikly snímky nové, například větších rozměrů, ostřejší, s menším šumem a podobně. Je tedy možno složením několika méně kvalitních snímků vytvářet snímky kvalitnější. Tomuto tématu je věnována tato bakalářská práce.

Kvalitní obrázky mohou být potřebné v různorodých lidských oborech. Například v medicíně, kdy se vytváří snímky pacientů. Kvalitnější snímky umožní lékařům lépe prostudovat stav jednotlivých pacientů. Se stávajícími zařízeními nemusí být někdy možné již dosáhnout lepších kvalit. Další z možností může být také neúnosná pořizovací cena zařízení pro záznam kvalitnějšího snímku. Důvodů omezené kvality snímků může být více. Jedním z dalších důvodů, proč vylepšovat obrázky na počítači, je možnost využít i velmi levné zařízení pro záznam.

Práce se také zaměřuje na využití standardu OpenCL pro metody zpracovávání obrazu. Tento standard umožňuje maximalizaci počítačového výkonu. Díky standardu OpenCL je možné pro práci využít různorodá výpočetní zařízení, která se v počítači nacházejí.

Konkrétním cílem této práce je vytvoření programu na zpracování videa, který složí jednotlivé snímky a vytvoří tak jeden kvalitní snímek.

Vybral jsem si toto téma pro jeho aktuálnosti. Dodnes se každý snaží pořizovat stále kvalitnější snímky. Mnohdy jdou cestou velmi drahých zařízení, která nejsou samospasná. Někdo volí cestu „dokreslení" vad obrázku v některém z grafických editorů. Možnost kvalitních obrázků z videa je velmi zajímavá. Další věcí, která mě velmi zajímala na tomto tématu, bylo využití standardu OpenCL. Tento standard je stále populárnější a jeho výsledky jsou velmi zajímavé.

Ihned za tímto úvodem práce následuje kapitola 2. Tato kapitola je věnována dvou způsobům možnosti zvýšení kvality obrázků, kterým se tato práce zabývá. Obsahuje i algoritmus, který sice manipuluje s obrázky, ale nesnaží se měnit kvalitu těchto obrázků. Je ovšem pro jednu ze zmíněných metod nutný. Konec této kapitoly je věnován standardu OpenCL, který je v aplikaci využit. Kapitola 3 shrnuje stav algoritmů a metod zmíněných v předchozí kapitole. Je v ní navržena aplikace, která bude plnit záměr této práce. Kapitola 4 je popisem této aplikace. Zmiňuje strukturu programu a ukazuje výsledky. Následující kapitolou je závěr, který obsahuje shrnutí poznatků a výsledků z předchozích kapitol.

2 Zvyšování kvality obrázků

Tato kapitola obsahuje poznatky o některých metodách na zpracování digitálního obrazu. V žádném případě neobsahuje veškeré známé metody, protože to rozsah této práce ani neumožňuje, a pojednává pouze o těch, které mají vztah k této práci.

V podkapitole 2.1 se nachází popis obrazu jako takového a jeho vlastností. Interpolaci digitálních obrázků je věnována podkapitola 2.2. Podkapitola 2.3 je věnována metodě, která slučuje sérii snímků. Pro potřeby metody na slučování snímků je nutné registrovat snímky, aby se překrývaly, podkapitola 2.4 se věnuje tomuto problému. Podkapitola 2.5 se věnuje standardu OpenCL, který souvisí s potřebou výpočetního výkonu u zpracování obrazu.

2.1 Obraz

Pojmem obraz, obrázek nebo také snímek se rozumí digitální fotografie [1]. Obrázky na počítačích se dělí do dvou skupin, rastrové a vektorové. Vektorové obrázky jsou geometrickým popisem obrázku. Rastrové obrázky jsou v počítači uloženy v číslicové podobě do matice se dvěma dimenzemi, kde každý prvek číslem představuje určitou konkrétní barvu.

V přírodě, kde je obraz spojitým signálem, je možné popsat obrázek matematicky jako funkci dvou proměnných, $f(x, y)$, kde x je horizontální pozice a y je pozice vertikální. Lidské oko i každé zařízení pro záznam obrazu přijímá obraz jako tento spojitý signál. Zařízení tento obraz převádí na diskrétní signál vzorkováním [3]. U zařízení pro záznam snímků vzorkování znamená, uložení hodnoty spojitého signálu na určité ploše.

Fotografie jsou v digitální podobě popsány určitými informacemi. Každý digitální obrázek obsahuje informace jako například rozměry, barevnou hloubku nebo počet kanálů. Tou nejdůležitější informací jsou ovšem hodnoty jednotlivých obrázkových bodů. Barevná hloubka obrázku říká, kolik je na něm možné rozpoznat jednotlivých odstínů jedné barvy. Počet kanálů pak říká, kolik barev se na obrázku vyskytuje. Černobílé snímky obsahují kanál pouze jeden. Barevné obrázky pak mají tři barevné kanály. Každý snímek pak může obsahovat tzv. alfa kanál, který obsahuje informaci o průhlednosti.

Jednou z důležitých vlastností obrázků je jejich kvalita. Kvalitu obrázku může každý subjektivně určit pouhým pohledem na něj. V této práci je kvalita uvažována jako množství šumu a rozměry obrázku.

Kvalitní obrázky mohou být velmi potřebné v některých lidských oborech, jako například medicína [4]. V některých oblastech mohou být velkým přínosem, například satelitní snímky. Nebo také pro grafické účely, kde jsou kvalitní obrázky velmi žádané. V neposlední řadě stojí za zmínku i osobní snaha mít co nejkvalitnější obrázek. Samozřejmě existuje mnoho dalších situací, kde je žádán kvalitnější snímek. Ne vždy je ovšem možné dosahovat požadovaných nebo dokonce potřebných kvalit. Často je limitující snímající zařízení. Jindy může být velmi nákladné pořídit nové zařízení pro tvorbu kvalitnějších snímků. Důvodů může být i více.

2.2 Interpolace digitálních obrázků

Interpolace obrázků je metoda na získávání nových obrázkových bodů, na základě těch původních [5][6]. Tato metoda je velmi důležitá v oblasti zpracování obrazu. Mnoho metod při zpracovávání obrazu potřebuje generovat nové obrázkové body, pixely. Použitím interpolační metody se nový obrázkový bod aproximuje na základě těch původních.

Potřeba generovat nové obrázkové body vzniká, kvůli manipulaci s obrázkovými body do polohy, kdy jsou posunuty do polohy, která je mimo obrázkovou matici. Dobrým příkladem metody, která způsobuje takovéto posunutí obrázkových bodů je rotace obrazu. Pokud není rotace o násobek kolmého úhlu, tak velká část obrázkových bodů se ocitne v oblasti někde mezi těmi normálními. Interpolací získáme body z rotovaného obrázku. Potřebu interpolace může ale vyvolat i mnohem jednodušší metoda jako je například posun obrázku. Tento posun může být v jediném směru, ale velikost posunu není v celých jednotkách. Další příčinou nutnosti interpolovat může být vytvoření nových obrázkových bodů, například při zvětšení velikosti obrazu. Opačně při zmenšení obrazu, kde je zase potřeba sloučit existující obrazové body. Ovšem toto je jen malý zlomek metod, které potřebují interpolaci. Nové body, které chceme získat interpolací, jsou zpravidla někde mezi těmi původními, ale mohou být i mimo obraz.

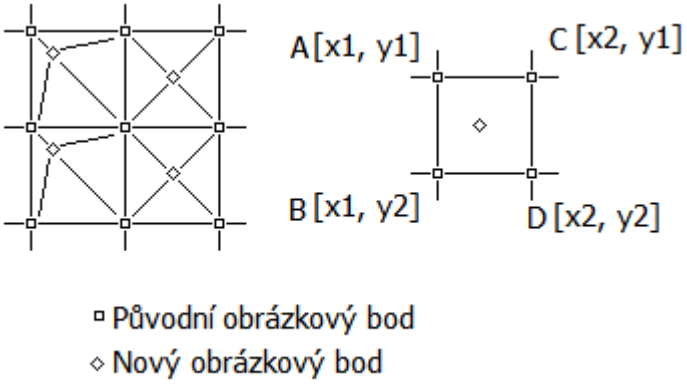
Metod interpolací existuje celá řada [6][7][5][8]. Od těch nejzákladnějších, jako je například metoda nejbližšího souseda. Až k těm složitějším, které se snaží analyzovat obrázek a podle získaných údajů provádí interpolaci. Nejčastěji se snaží identifikovat hra-

ny, které jsou poté interpolovány jinak. Metody, které detekují hrany nebo jakkoliv jinak analyzují obrázek, jsou nazývány adaptabilními metodami [6]. Hlavním rysem těchto metod je různý přístup ke každému obrázkovému bodu. Neadaptabilní metody zacházejí s každým obrázkovým bodem stejně. Adaptabilní metody zahrnují celou řadu proprietárních algoritmů z různých grafických editorů. V dalších odstavcích budou stručně popsány a ilustrovány některé neadaptabilní interpolační metody.

Metoda nejbližšího souseda je velmi jednoduchá [6]. Hodnotu nového bodu určí jako hodnotu nejbližšího bodu. Obrázek 2.1 znázorňuje tento princip. Při zvětšování obrázku je výsledným efektem zvětšení původních obrázkových bodů.

Obrázek 2.1 Interpolace nejbližšího souseda.

Jednou z dalších interpolačních metod je bilineární interpolace [6][5]. Tato metoda bere v potaz hodnotu čtyř nejbližších obrázkových bodů a novou hodnotu vypočítá z nich. Obrázek 2.2 ukazuje tento postup. Výpočet je proveden váhovým průměrem těchto bodů na základě jejich vzdálenosti, podle vzorce (2.1). Tato metoda produkuje hladší výsledné snímky než metoda nejbližšího souseda.



Obrázek 2.2 Bilineární interpolace.

Výpočet hodnoty bodu bilineární interpolace. Obrázek 2.2 ukazuje popis:

$$\begin{gathered} a = x - x1 \\ b = y - y1 \\ f(x,y) = A(1-a)(1-b) + B(1-a)b + Ca(1-b) + Dab \end{gathered} \tag{2.1}$$

Bikubická interpolační metoda jde o krok dále [6]. Nový pixel počítá z okolí šestnácti nejbližších bodů. Hodnota obrázkového bodu je opět počítána jako váhový průměr všech těchto bodů. Obrázek 2.3 znázorňuje tuto metodu.

Obrázek 2.3 Znázornění bikubické interpolace.

Dalšími interpolačními metodami, které jsou o mnoho složitější, ale o kterých stojí se aspoň zmínit, jsou například Lancsoz, Geniue Fractals, B-spline, Wavelet filter nebo SmartEdge [7][5][8].

2.3 Super rozlišení

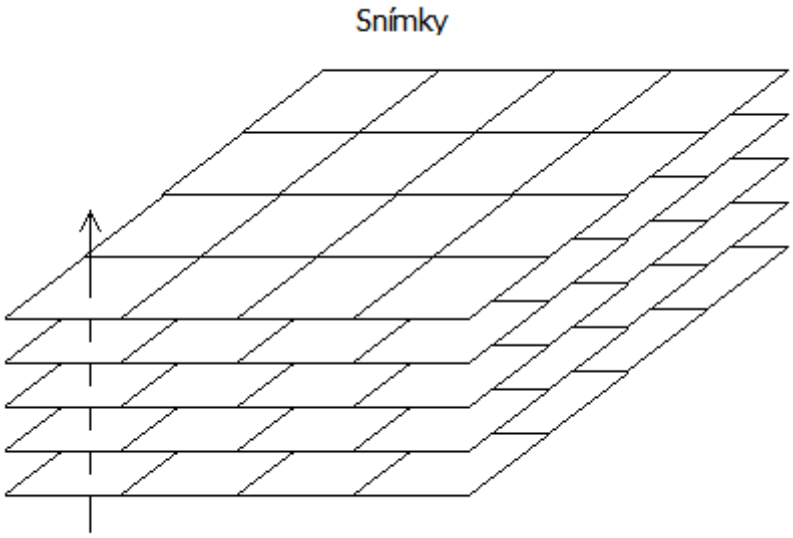
Metoda super rozlišení spočívá ve slučování snímků [4][9]. Nemá ustálené jméno a může být nazývána také jako rekonstrukce obrázku. Toto nejsou jediné názvy. Tato metoda je v angličtině nazývána „super resolution (SR) image reconstruction" [4], „high resolution (HR) image reconstruction" [4] nebo také jednodušeji „image enhancement" [4][9].

Sloučením se vytváří jediný obrázek ze série snímků. Jednotlivé snímky z řady zpravidla mají nízké rozlišení. Pro získání série takovýchto snímků stačí jakákoliv kamera nebo fotoaparát. Metoda má tedy velkou výhodu z hlediska zdrojů [4]. Každý ze série snímků na sloučení musí zabírat stejnou scénu. Toto je základní podmínka použitelnosti tohoto algoritmu. Je ovšem potřebné, aby se každý snímek minimálně lišil. Může to být různé pootočení snímku nebo také stačí jen posun snímku. Pouhý posun snímku ale nestačí. Posun nesmí být celé číslo, jinak jsou opět snímky totožné a proto nevyužitelné [4]. Obrázek 2.4 znázorňuje takovéto posunutí. Kdyby tak všechny snímky ze série obsahovaly úplně totožný záběr, tak všechny snímky nesou totožnou informaci a není možné jejich sloučením získat kvalitnější obraz. Snímky samozřejmě i tak sloučit jde. Ve výsledku by byl ovšem odstraněn maximálně přítomný šum. Když každý snímek ze série bude odlišný, tak každý z nich obsahuje trochu jinou informaci. Slučováním takovýchto snímků je velmi výhodné. Odstraní se tím velké množství vad. Tento fakt ovšem přináší velkou nevýhodu. Právě kvůli tolik žádaným drobným odlišnostem jednotlivých snímků je potřeba před sloučením tyto snímky registrovat, aby se překrývali [4][9]. Metoda na registraci snímků je popsána dále v podkapitole 2.4.

Obrázek 2.4 Posunutí dvou snímků.

Metoda pracuje se všemi snímky najednou [4][9]. Snímky již musí být zaregistrovány, aby se překrývaly. Musí také být zvýšeno rozlišení jednotlivých snímků. Ze všech snímků se vezme jeden konkrétní obrázkový bod. Tento bod se na všech snímcích musí překrývat. Obrázek 2.5 znázorňuje tento přístup. Ze všech snímků se porovnává hodnota tohoto bodu. Na některých snímcích může tento bod trpět některou vadou. Může se jednat o šum, o nějaký pohyb ve scéně nebo cokoliv jiného. Hodnoty jsou seřazeny

a od obou konců je odstraněn určitý počet extrémních hodnot. Tímto krokem se mimo jiné odstraní nelineární šum [9]. Ze zbylých hodnot se udělá průměr. Tento průměr ze zbylých hodnot odstraňuje lineární šum [9]. Výsledek průměru je nová hodnota obrázkového bodu na výsledném snímku.



Obrázek 2.5 Průsečík jednoho překrývajícího se bodu na všech snímcích.

Během procesu zaznamenávání obrazu, dochází k mnohým efektům, které zhoršují kvalitu výsledku. Jedná se například o šum, rozostření, aliasing [4]. Hlavním záměrem metody super rozlišení bylo provádět rekonstrukci obrazu ze snímků s nízkým rozlišením. Přesto tento algoritmus zvládá práci s rozostřenými snímky s šumem. Nejsou to ale zdaleka všechny možné vady, které se mohou na jednotlivých snímcích podepsat. Úkolem metody super rozlišení je zpracovávat snímky s nízkým rozlišením, ale také degradované snímky a snímky trpící na aliasing.

V článku [4] se píše, že metoda super rozlišení již byla prokázána jako užitečná v mnohých praktických případech, kde je možné získat sérii snímků ze stejné scény. Šlo například o obrázky v medicíně, satelitní snímky nebo také aplikace pro zpracování videa. V medicíně se může jednat například o počítačovou tomografii (CT) nebo o magnetickou rezonanci (MRI), kde rozlišení získaných obrázků je omezené, ale není problém získat sérii těchto snímků. U satelitních snímků je to podobné. Několik snímků stejné plochy je získáno a aplikovat na ni metodu super rozlišení je tedy možné.

2.4 Registrace obrázků

Různé algoritmy k registraci obrázků slouží ke zjištění vzájemné polohy dvou nebo více obrázků. To je zjištění polohy, kde se všechny obrázky vzájemně překrývají. Porovnávané obrázky musí obsahovat stejnou scénu. Jednotlivé snímky se od sebe mohou lišit drobnými odchylkami, malou rotací snímku, ale i jiným úhlem pohledu, jinou dobou zachycení snímku a jinými vadami [10][4][9]. Registrace obrázků je zásadním krokem v metodách na analýzu obrázků, které potřebují získat informaci z několika snímků [10]. Příkladem je metoda super rozlišení, která je rozebrána v podkapitole 2.3.

Každou z metod na registraci obrázků lze rozdělit na čtyři kroky [10]:

1) Detekce.
2) Přizpůsobení.
3) Odhad transformačního modelu.
4) Transformace a nové vzorkování obrázku.

Tato podkapitola se bude věnovat dále jen registraci snímků, kde je posun mezi obrázky jemnější než celý obrázkový bod. V angličtině „sub-pixel shift" [9][10][4]. Většina metod není tak přesná, aby zvládla takovýto posun [9].

Dva snímky, které chceme registrovat, bereme jako dvou rozměrné pole. Snímky označíme jako funkce f a g. Provádí se tedy mapování [9]:

$$g(x,y) = f(x^{'}, y^{'}) \tag{2.2}$$

V dalších vzorcích jsou použity tři koeficienty a, b, θ. Posun mezi snímky na ose x se označí jako a. Na ose y se potom značí b. Rotace mezi snímky se značí θ. Je to rotace od počátku. Vztah vyjadřující tyto koeficienty je pak [9]:

$$\begin{aligned} x^{'} &= x\cos\theta - y\sin\theta + a \\ y^{'} &= y\cos\theta + x\sin\theta + b \end{aligned} \tag{2.3}$$

Dosazením do původní rovnice (2.2) vznikne vztah:

$$g(x,y) = f(x\cos\theta - y\sin\theta + a, y\cos\theta + x\sin\theta + b) \tag{2.4}$$

Za předpokladu, že θ je malý úhel, můžeme rozložit $\sin\theta$ a $\cos\theta$ podle jejich Taylorovy řady [9]:

$$\begin{aligned} x' &\approx x + a - y\theta - \frac{x\theta^2}{2} \\ y' &\approx y + b + x\theta - \frac{y\theta^2}{2} \end{aligned} \tag{2.5}$$

Opětovným dosazením do původní rovnice (2.2) vznikne vztah:

$$g(x,y) \approx f(x + a - y\theta - \frac{x\theta^2}{2}, y + b + x\theta - \frac{y\theta^2}{2}) \tag{2.6}$$

Rozkladem funkce f podle vlastní Taylorovy řady vznikne [9]:

$$g(x,y) \approx f(x,y) + \left(a - y\theta - \frac{x\theta^2}{2}\right)\frac{\delta f}{\delta x} + \left(b + x\theta - \frac{y\theta^2}{2}\right)\frac{\delta f}{\delta y} \tag{2.7}$$

Chyba mezi funkcí f a g se počítá jako součet čtverců chyb [9]:

$$E(a,b,\theta) = \sum [f(x,y) + \left(a - y\theta - \frac{x\theta^2}{2}\right)\frac{\delta f}{\delta x} + \left(b + x\theta - \frac{y\theta^2}{2}\right)\frac{\delta f}{\delta y} - g(x,y)]^2 \tag{2.8}$$

Suma je přes překrývající se část f a g. Je možné použití i mnohem menší část než celé překrývající se části. Výpočtem minima z $E(a,b,\theta)$ derivací podle a, b, θ následným výpočtem k nule a drobnými dalšími úpravami, vznikne následující soustava rovnic [9]:

$$\begin{gathered} \left[\sum \left(\frac{\delta f}{\delta x}\right)^2\right] a + \left[\sum \left(\frac{\delta f}{\delta y}\right)\left(\frac{\delta f}{\delta x}\right)\right] b + \left[\sum \left(\frac{\delta f}{\delta x}\right) R\right] \theta = \sum \left(\frac{\delta f}{\delta x}\right)(f - g) \\ \left[\sum \left(\frac{\delta f}{\delta x}\right)\left(\frac{\delta f}{\delta y}\right)\right] a + \left[\sum \left(\frac{\delta f}{\delta y}\right)^2\right] b + \left[\sum \left(\frac{\delta f}{\delta y}\right) R\right] \theta = \sum \left(\frac{\delta f}{\delta y}\right)(f - g) \\ \left[\sum \left(\frac{\delta f}{\delta x}\right) R\right] a + \left[\sum \left(\frac{\delta f}{\delta y}\right) R\right] b + \left[\sum R^2\right] \theta = \sum R(f - g) \\ R = x\frac{\delta f}{\delta y} - y\frac{\delta f}{\delta x} \end{gathered} \tag{2.9}$$

Výpočet lze zachytit v maticové formě [10].

$$C = \begin{Bmatrix} \sum\left(\frac{\delta f}{\delta x}\right)^2 & \sum\left(\frac{\delta f}{\delta y}\right)\left(\frac{\delta f}{\delta x}\right) & \sum\left(\frac{\delta f}{\delta x}\right)R \\ \sum\left(\frac{\delta f}{\delta x}\right)\left(\frac{\delta f}{\delta y}\right) & \sum\left(\frac{\delta f}{\delta y}\right)^2 & \sum\left(\frac{\delta f}{\delta y}\right)R \\ \sum\left(\frac{\delta f}{\delta x}\right)R & \sum\left(\frac{\delta f}{\delta y}\right)R & \sum R^2 \end{Bmatrix}, X = \begin{Bmatrix} a \\ b \\ \theta \end{Bmatrix}$$

$$V = \begin{Bmatrix} \sum\left(\frac{\delta f}{\delta x}\right)(f-g) \\ \sum\left(\frac{\delta f}{\delta y}\right)(f-g) \\ \sum R(f-g) \end{Bmatrix}, R = x\frac{\delta f}{\delta y} - y\frac{\delta f}{\delta x} \tag{2.10}$$

$$R = x\frac{\delta f}{\delta y} - y\frac{\delta f}{\delta x}$$

$$CX = V$$

Výpočet koeficientů se potom získá:

$$X = C^{-1}V \tag{2.11}$$

Výpočet je ale přesný jen pro malé koeficienty a, b, θ. V dokumentu [10] je navrhnutý vylepšený algoritmus pro výpočet i když bude θ větší než současny algoritmus dovoluje. Toto omezení plyne z rovnice (2.5), kdy je použita Taylorova řada. Tento algoritmus používá čtyři koeficienty $\mathrm{a}_1, \mathrm{a}_2, \mathrm{a}_3, \mathrm{a}_4$ místo původních třech. Expanze funkce $\sin$ a $\cos$ je udělána jiným způsobem [10]: (2.12)

$$x' = x + a_1 x + a_2 y + a_3$$
$$y' = y + a_1 y - a_2 x + a_4$$

Dosazením do rovnice (2.2) se získá:

$$g(x, y) = f(x + a_1 x + a_2 y + a_3, y + a_1 y - a_2 x + a_4) \tag{2.13}$$

Následné kroky jsou stejné jako v rovnicích (2.7), (2.8) a (2.9). Expanze funkce f podle vlastní Taylorovy řady, vyčíslení chyby E a derivace chyby podle jednotlivých koeficientů. Jelikož jsou ale koeficienty čtyři, vznikne soustava čtyř rovnic o čtyřech neznámých. Rovnice vyjádřeny rovnou v maticovém tvaru:

$$C = \begin{Bmatrix} \sum R_1^2 & \sum RR_1 & \sum\left(\frac{\delta f}{\delta x}\right) & \sum\left(\frac{\delta f}{\delta y}\right)R_1 \\ \sum R_1 R & \sum R^2 & \sum\left(\frac{\delta f}{\delta x}\right) & \sum\left(\frac{\delta f}{\delta y}\right)R \\ \sum R_1\left(\frac{\delta f}{\delta x}\right) & \sum R\left(\frac{\delta f}{\delta x}\right) & \sum\left(\frac{\delta f}{\delta x}\right)^2 & \sum\left(\frac{\delta f}{\delta y}\right)\left(\frac{\delta f}{\delta x}\right) \\ \sum R_1\left(\frac{\delta f}{\delta y}\right) & \sum R\left(\frac{\delta f}{\delta y}\right) & \sum\left(\frac{\delta f}{\delta x}\right)\left(\frac{\delta f}{\delta yx}\right) & \sum\left(\frac{\delta f}{\delta y}\right)^2 \end{Bmatrix}$$

$$V = \begin{Bmatrix} \sum R_1(g-f) \\ \sum R(g-f) \\ \sum\left(\frac{\delta f}{\delta x}\right)(g-f) \\ \sum\left(\frac{\delta f}{\delta y}\right)(g-f) \end{Bmatrix}, X = \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{Bmatrix} \tag{2.14}$$

$$R = x\frac{\delta f}{\delta y} - y\frac{\delta f}{\delta x}, R_1 = x\frac{\delta f}{\delta x} + y\frac{\delta f}{\delta y}$$

$$CX = V$$

Pro výpočet koeficientů se získá:

$$X = C^{-1}V \tag{2.15}$$

Vztah koeficientů a_1, a_2, a_3, a_4 a a, b, θ je vyjádřen vztahem:

$$\begin{Bmatrix} a \\ b \\ \theta \end{Bmatrix} = \begin{Bmatrix} a_3 \\ a_4 \\ -\sin^{-1} a_2 * \frac{180}{\pi} \end{Bmatrix} \tag{2.16}$$

2.5 OpenCL

Rostoucí potřeba heterogenních výpočtů a paralelních architektur vytvořila velkou potřebu pro programovací jazyky a knihovny, které mohou podporovat tento přístup, na zařízeních od různých výrobců [11]. Zpočátku si vývojáři adaptovali již existující vědecká řešení, pro využití více jádrových procesorů a silně paralelních grafických procesorů. Využívali při tom pomůcky jako například CUDA, neboli Computer Unified Device Architecture, a další. Tyto dosavadní pomůcky, přestože byly velmi užitečné, byly většinou limitovány na jedinou rodinu výpočetních jednotek, nepodporovaly heterogenní výpočty nebo také obsahovaly různá další omezení.

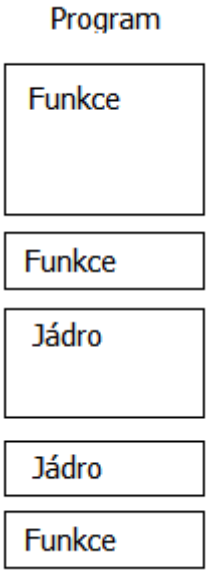
Nástupcem těchto pomůcek se stal standard OpenCL. Standard OpenCL je reakcí na tuto se stále zvyšující potřebu výpočetního výkonu [11]. Poskytuje prostředí s podporou pro paralelní heterogenní výpočty na systémech, složených z různých výpočetních jednotek [11][12][13]. Vytváří velmi lehkou abstrakci od výpočetního zařízení pro snadné použití. Programové rozhraní (API) pro aplikace bylo inspirováno již existujícími pomůckami. Definuje základní funkčnost, kterou všechny zařízení podporují. Zároveň ale podporuje rozšiřitelnost od různých výrobců, pro použití různorodých vylepšení na každém ze zařízení nebo pro využití sofistikovanějších zařízení [11]. Příkladem rozšíření může být dvojnásobná přesnost čísel s plovoucí desetinou čárkou nebo podpora atomických operací nad skalárními typy.

Různé zařízení podporující OpenCL nemusejí sdílet paměť s procesorem a většinou mají i vlastní instrukční sadu [11][12][13]. OpenCL se stará o případné potřebné přenosy dat mezi zařízeními. Přestože nemůže úplně zakrýt rozdíly v různých architekturách, garantuje přenositelnost a správnost kódu. Negarantuje ovšem stejný výkon kódu na jednotlivých zařízeních.

Klíčové vlastnosti programového rozhraní OpenCL jsou [11]:

- Výčet dostupných zařízení pro využití OpenCL.
- Správa kontextu vytvořeného pro zařízení.
- Správa alokování paměti.
- Přenos dat do zařízení a z něj.
- Kompilace OpenCL programů a jejich jader.
- Spouštění jader na zařízení.
- Dotazování vývoje.
- Kontrola chyb.

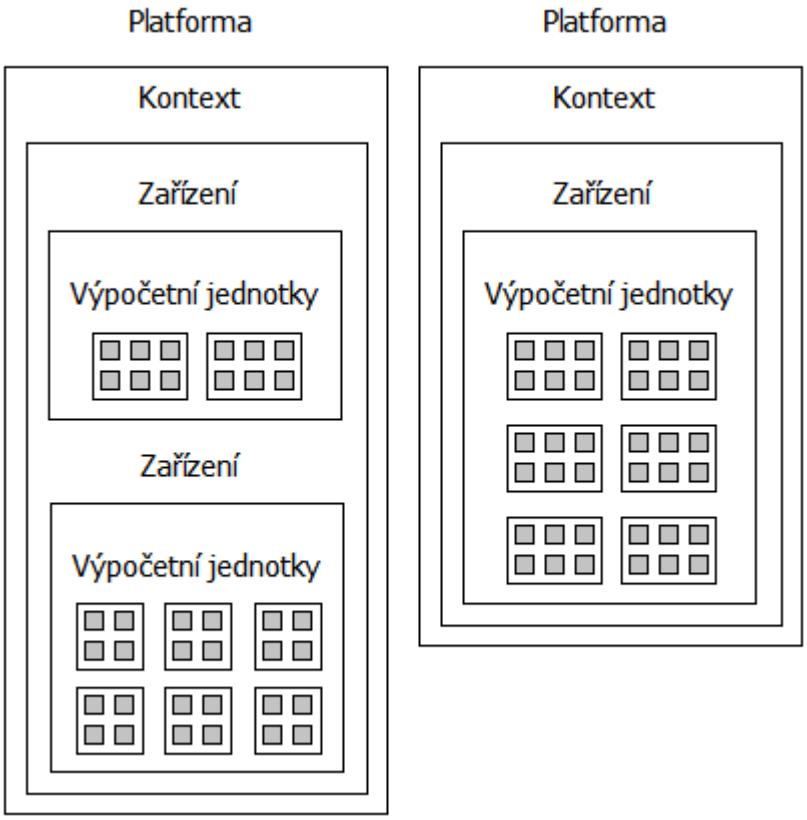
Programy psané pro OpenCL se nacházejí v separátních souborech. Každý soubor obsahuje alespoň jednu funkci, která je označena jako jádro, v angličtině „kernel". Obrázek 2.6 znázorňuje tento přístup. Z programu se volá právě jedna z takto označených funkcí. Funkce označená jako jádro nemá žádnou návratovou hodnotu. Jazyk používaný v těchto programech bývá označován jako OpenCL C. Už jen podle označení je tento jazyk podobný jazyku C. K překladu ovšem dochází až v době spuštění. Díky tomuto jsou kódy nezávislé na spouštěném zařízení. Je možné kompletně předělat instrukční sadu některého ze zařízení a program na něm půjde stále spouštět.



Obrázek 2.6 OpenCL program.

Prvním krokem programu, který chce využít OpenCL, je získat seznam platforem [12][11]. Každá platforma se liší výrobcem, což v důsledku znamená různorodou implementaci OpenCL. Pokud je seznam platforem prázdný, tak se v počítači nemusí nacházet žádné zařízení s podporou OpenCL. Další možností prázdného seznamu platforem může být absence patřičných ovladačů k zařízení, které by podporovaly OpenCL. V takovém případě se zařízení s podporou OpenCL v počítači sice nachází, ale není možné ho využít.

Druhým základním krokem je vytvoření kontextu [12][13][11]. Velká část dalších akcí je vykonávána nad určitým kontextem. Pro vytvoření kontextu se vybírá typ zařízení a platforma. Jeden kontext ale může obsahovat více zařízení. Typ zařízení může být procesor, grafický procesor nebo také nějaké další zařízení, které podporuje OpenCL. OpenCL popisuje hardware jako hierarchii [11]. Obrázek 2.7 znázorňuje tuto hierarchii.



Obrázek 2.7 Hardwarové hierarchie podle OpenCL.

Různé úkony jsou v OpenCL vykonávány přes fronty. Ty jsou vytvořeny z kontextu k určitému zařízení. Fronty představují virtuální rozhraní vždy k jedinému zařízení. Některé úkony mohou fronty využít, ale nemusí. Například nahrání dat do alokované paměti pro OpenCL program. Zařazení požadavku do fronty neblokuje program v dalším běhu. Programátor si sám může zvolit, kde dojde k synchronizaci. Další důležitou metodou, která je ovšem vždy volána přes frontu, je volání jádra v OpenCL programu.

OpenCL dělí paměť na čtyři typy [11]:

- Velkou globální paměť s pomalou odezvou.
- Malou paměť s rychlou odezvou pro konstantní data.
- Lokální paměť, která je sdílená v rámci zařízení.
- Privátní paměť nebo registry jedné výpočetní jednotky.

Každý jednotlivý proces představuje v OpenCL jednu pracovní jednotku, tzv. „work-item". Tyto jednotky mohou byt uskupeny do skupin, tzv. „work-groups". Při invokaci funkce jádra z OpenCL programu se definuje přesný počet pracovních jednotek, které mají danou funkci vykonat. Samotná invokace je prováděna přes frontu.

K jakémukoliv přenosu dat mezi jádrem a aplikací dochází přes dočasnou paměť, tzv. „buffer". Při vytváření dočasné paměti dochází zároveň k jejímu alokování. Dočasná paměť není vázána na zařízení, ale na kontext. Kvůli tomuto je maximální velikost alokované paměti závislá na zařízení v kontextu s nejmenší pamětí [11]. Funkce v OpenCL programu, označená jako jádro, dostává jako vstupní parametry tyto dočasné paměti. Vypočítané hodnoty se z OpenCL vrací také přes dočasné paměti. Při vytváření dočasných pamětí lze specifikovat, jestli jsou jen pro zápis, pro čtení nebo pro obojí.

3 Zhodnocení

V této kapitole se nachází rozbor jednotlivých algoritmů, metod a technologií, zmíněných v předchozí kapitole. Každá z těchto položek bude trochu rozebrána podle účelnosti pro cíl této práce. Další věcí je vytyčení určitého záměru práce. Vyjádření potřebných kroků k dosažení cíle. Na konci této kapitoly bude navržena výsledná aplikace, která by plnila vytyčený záměr práce.

V dnešní době existuje velká řada algoritmů pro zpracování obrazu. Vhodným výběrem těchto algoritmů lze dosáhnout mnoha různých cílů. V této práci je vybrán ovšem jen zlomek z nich a tyto vybrané algoritmy by měly posloužit ke splnění cíle práce.

3.1 Zhodnocení aktuálního stavu

První vybranou metodou vylepšení snímku je interpolace digitálního obrázku. Tato metoda je velmi známá a nepochybně je velmi používaná. Metoda je použita při zvyšování rozměrů obrázku. Při zvýšení rozměrů se vytváří potřeba doplnit chybějící body a přesně k tomu je tato metoda určena. V podkapitole 2.2 je popis některých metod a u některých interpolačních metod i názorná ukázka jak pracují.

Interpolační metoda nejbližšího souseda neprovádí v podstatě žádné výpočty. Je tedy velmi výhodná na výpočetní výkon a tím velmi rychlá. Výsledný obrázek ale není v podstatě vůbec žádným způsobem zlepšený a veškeré vady ve snímku přetrvávají. Jelikož je cílem práce zvyšovat kvalitu, není žádný důvod si zvolit tuto metodu.

Druhou detailněji popsanou metodou je bilineární interpolace. Tato metoda už počítá výsledný pixel podle čtyř nejbližších obrázkových bodů. Je tedy více náročná na výpočetní výkon. Na druhou stranu už ale vylepšuje kvalitu výsledného obrázku a je vhodná pro použití v této práci.

Další z popsaných metod je metoda bikubická. Tato metoda se hodně podobá metodě bilineární. Jediná a zásadní změna je v počtu obrázkových bodů, ze kterých se počítá výsledný obrázkový bod. Tato metoda používá k výpočtu nového bodu šestnáct nejbližších bodů. Tato metoda je tedy ještě více výpočetně náročná než metoda bilineární. Vylepšuje ovšem kvalitu obrázku a je vhodná pro použití v této práci.

Další metodou na zvýšení kvality obrázku je metoda super rozlišení. Ve srovnání s ostatními metodami a algoritmy je tato metoda pro tuto práci klíčová. I když zůstává závislá na jiné metodě, je to nejdůležitější metoda v této práci. Tato metoda bere sérii snímků a sloučí je v jediný snímek [9][4].

Algoritmus pro registraci obrázků je klíčový pro metodu super rozlišení. Jelikož je pro metodu super rozlišení nutné, aby video sekvence obsahovala jedinou scénu, nemusí algoritmus pro registraci snímků řešit větší než malé posuny a rotace snímků. Jestli jsou posuny minimální, lze předpokládat, že se algoritmus nemusí starat ani o jiné úhly pohledu. Popsaný algoritmus se nachází v podkapitole 2.4 se stará přesně jen o tyto odchylky v podobě posunu obrázků o necelé obrázkové body a drobné rotace [9][10].

Standard OpenCL poskytuje vysoce paralelní prostředí [11][12][13]. Jelikož bude v programu zapotřebí zpracovávat obrázky, je standard OpenCL ideálním řešením. Zpracování obrázků přináší vysoký potenciál ve využití paralelních architektur či prostředí. Pro ideální, intuitivní a jednoduché využití ale zůstává podmínka pro algoritmus na zpracování obrázků. Vstupem toho algoritmu pro zpracování obrázků je konstantní vstup a závislost jednotlivých výsledných obrázkových bodů je jenom na vstupu. Neboli, je možné spustit nezávisle algoritmus pro jakýkoliv bod. Výpočty lze pak provádět pro každý pixel, popřípadě i pro každý kanál v obrázku nezávisle na okolí. Kdyby tato podmínka splněna nebyla, nešel by algoritmus spustit pro jakýkoliv bod a využitelnost OpenCL by se mohla výrazně omezit. Takový algoritmus by se musel nejdříve nějakým způsobem ošetřit, aby šel využít paralelně.

3.2 Návrh aplikace

Tato práce je zaměřena na vytvoření programu, které zpracovává video pro získání co nejkvalitnějšího snímku. Toto video obsahuje jeden konkrétní záběr jediného místa, věci nebo čehokoliv jiného. Hlavní podmínkou je, aby se jednotlivé snímky ve video sekvenci od sebe příliš nelišily. Přesto se musí snímky lišit, jinak zpracováním takovéto video sekvence nedošlo k žádnému zlepšení [4][9]. K tomuto záměru jsou v práci vybrány a popsány různé algoritmy v kapitole 2. Zhodnocení a shrnutí těchto metod a algoritmů je před v podkapitole 3.1.

Vstupem programu tedy bude video soubor. Program musí zpracovat nějakým způsobem video do snímků. K tomuto účelu jsem vybral knihovnu OpenCV [14], anglicky

„Open Source Computer Vision". Tato knihovna umožňuje načtení videa, které je zakódováno v určitém kodeku [15]. Po načtení videa je možné extrahovat snímky po jednom. Jakmile bude mít aplikace všechny snímky, provede na nich úpravy pro zlepšení kvality. Vhodnou volbou bude zvýšení rozlišení a použití některé z interpolačních metod a popřípadě ještě některé další úpravy. Nyní bude muset aplikace všechny snímky registrovat. Jakmile budou všechny snímky registrovány, může je program sloučit a vytvořit tím jeden kvalitní snímek. Nakonec se vyberou vhodné části programu, které by šly spouštět v OpenCL. Tyto části se vhodně upraví a algoritmy přepíší do OpenCL C.

4 Implementace

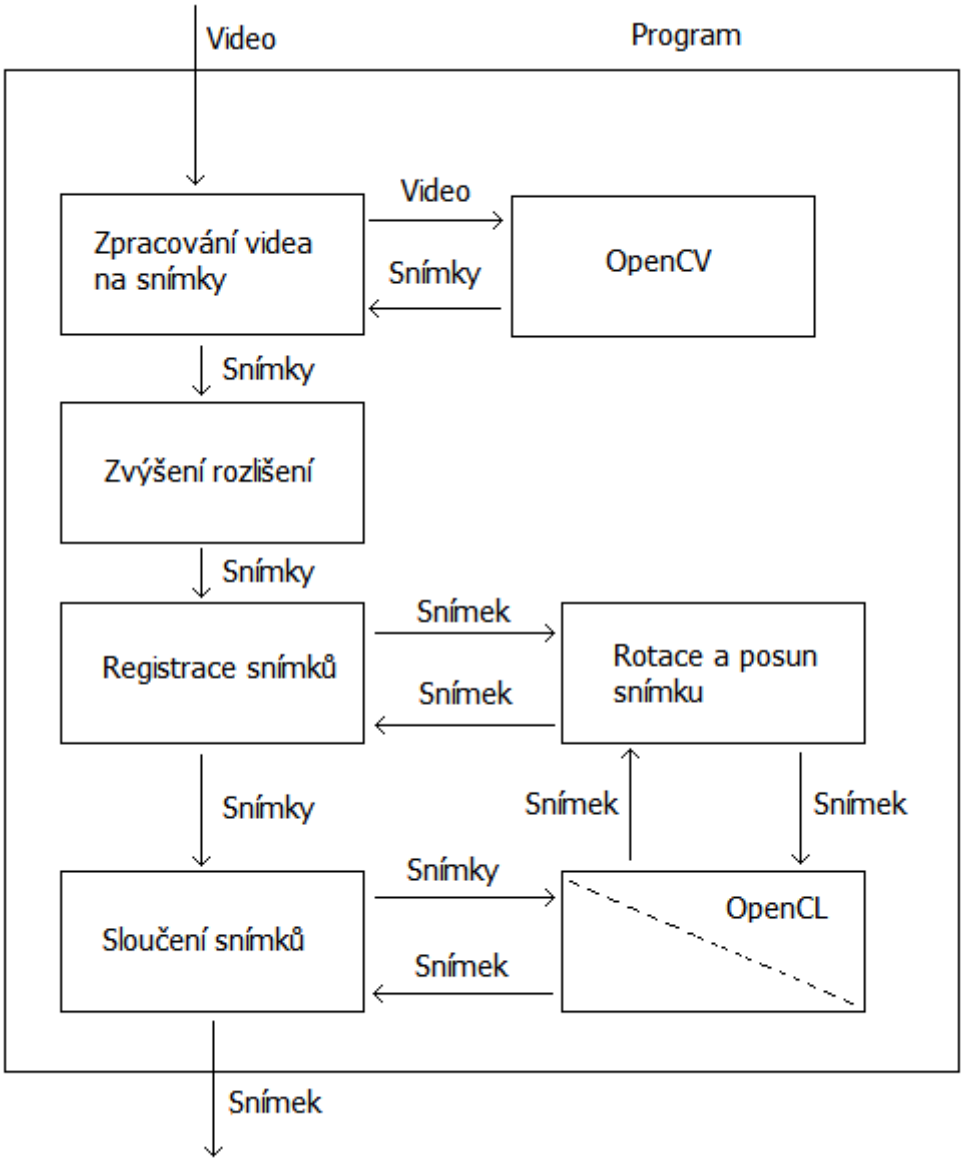
Tato kapitola popisuje aplikaci. Popisuje její strukturu, způsob ovládání a jakým způsobem konkrétně věci fungují. Předvádí její výsledky dílčích částí i jako celku. Na konci se nachází vyhodnocení aplikace.

Jako vývojové prostředím jsem si zvolil Microsoft Visual Studio 2010. Microsoft Windows je tedy operační systém, na kterém budu program vyvíjet. I přes tuto volbu jsem dbal, aby vybrané knihovny byly pokud možno podporované na více operačních systémech, než je jen Microsoft Windows. Jako jazyk jsem si zvolil C++. Zdrojové kódy jsou členěny do souborů podle jejich funkce. Jednou z použitých knihoven je OpenCL, protože je práce zaměřena na využití tohoto standardu. Další zvolenou knihovnou je OpenCV. Tato knihovna obstarává zpracování videa a obrázků.

4.1 Popis programu

Obrázek 4.1 znázorňuje princip fungování tohoto programu. Vstupem programu je video. Výstup programu je pak jeden snímek, který by měl být kvalitnější než jednotlivé snímky z videa.

Program zpracuje vstupní video soubor do sekvence snímků. Rozdělení videa na snímky provede za pomoci knihovny OpenCV. Tato část programu je popsána v podkapitole 4.2. Následuje zpracování jednotlivých snímků, které je podrobněji popsáno v podkapitole 4.3. Jednotlivým snímkům zvýší rozlišení a registruje je na sebe. Při registraci je nutno posunovat obrázky podle výsledků algoritmu popsaného v podkapitole 2.4. Algoritmus na popis obrázků je přepsán do OpenCL. Výkonnostní porovnání algoritmu v OpenCL oproti algoritmu psanému v C++ je v podkapitole 4.7. Po registraci snímků jsou snímky sloučeny. Tento algoritmus je také přepsán do OpenCL a výsledky výkonnostního testu jsou v podkapitole 4.7. Výsledný snímek je uložen do souboru.



Obrázek 4.1 Schéma programu

4.2 Práce s OpenCV

Program zpracovává video soubor. K tomuto účelu byla vybrána knihovna OpenCV [14]. Pro jednotlivé hledání funkcí, tříd, struktur nebo možností knihovny OpenCV jsem používal [16]. Na některých operačních systémech může tato knihovna video otevřít, ale jinde ne, OpenCV totiž nepodporuje na všech operačních systémech stejné kodeky [15]. Některé kodeky jsou ovšem podporovány všude. Seznam doporučených kodeků

je nalezení na webových stránkách OpenCV [15]. Pokud je potřeba změnit kodek videa, lze k tomuto účelu vybrat například program VirtualDub [17], který je vydán pod licencí GNU GPL.

Knihovna OpenCV obsahuje třídu, která se stará o zpracování videa. Tato třída otevře videa a následně je možné z videa extrahovat jednotlivé snímky. Tato třída bohužel neobsahuje žádné informace o délce zpracovávaného videa nebo počtu snímků. Program tedy musí extrahovat snímky, dokud to lze, aby zjistil jejich počet. Jednotlivé extrahované snímky vkládám do listu. Jakmile funkce, které extrahuje jednotlivé snímky, vrátí chybový kód, je videa zpracované celé a je možné začít provádět další úpravy.

Všechny obrázky jsou v knihovně OpenCV uchovávány v instancích jedné třídy, `cv::Mat`. Tato třída slouží nejen k uchovávání obrázků, ale matic obecně. Třída ukládá veškerá data do jednorozměrného pole. Pokud je uložen barevný obrázek, jsou barevné kanály uloženy v pořadí, modrá, zelená a červená. Třída obsahuje abstraktní přístupové metody k jednotlivým řádkům a sloupcům [16]. Nikoliv však k jednotlivým kanálům obrázku. K tomuto účelu je v programu dopsána funkce, která se stará i o přístup ke kanálu.

Tuto třídu používám v celém programu pro uložení jednotlivých snímků. Je velmi výhodné ji použít v programu, když už ji musím použít minimálně při čtení videa. Navíc nabízí vhodné možnosti manipulace se snímky.

4.3 Zpracovávání série snímků

Dalším krokem programu je zvětšení všech snímků. Pro zvětšování snímků jsem zvolil bilineární interpolační algoritmus. Funkce pro zvětšení obrázků má dva parametry. Prvním je obrázek na zvětšení a druhým je číslo s plovoucí desetinnou čárkou. Toto číslo představuje násobek velikosti jedné ze stran obrázku. Obrázek je tedy zvětšen o druhou mocninu tohoto čísla.

Po zvětšení všech snímků přichází na řadu registrace snímků. Pro tento účel se používá tzv. gaussova pyramida [9]. Ze všech snímků se vytvoří řada jejich zmenšenin. Poté se iterativně na sebe snímky registrují. Nejdříve se na sebe mapují tyto zmenšené obrázky. Až už je odchylka velmi malá nebo po určitém počtu iterací se algoritmus zastaví, přejde se k registraci obrázků s vyšším rozlišením. Tento postup zvyšuje robustnost toho algoritmu a zároveň ho zrychluje [9]. Implementačními detaily registrací obrázků se poté detailněji věnuje podkapitola 4.4.

Jakmile jsou všechny snímky na sebe registrovány je možné je začít slučovat. Algoritmus prochází po jednom obrázkovém bodu všechny snímky. Konkrétně prochází jedním bodem na všech snímcích podle počtu barevných kanálů. Po každém průchodu tedy projde jedním obrázkovým bodem, jedním kanálem barvy a všemi snímky. Z tohoto průchodu si uloží jednotlivé hodnoty. Tyto hodnoty seřadí a odstraní extrémy, jak je popsáno v podkapitole 2.3 [9]. Konkrétně program zahodí čtvrtinu hodnot na obou koncích. Ze zbylých hodnot udělá aritmetický průměr a výslednou hodnotu uloží do výstupního obrázku. Konkrétní výsledky metody si lze prohlédnout v podkapitole 4.5.

4.4 Registrace obrázků

Algoritmus na registraci obrázků vrací tři koeficienty, a, b, θ. Tyto koeficienty v tomto pořadí určují posun snímku na ose x, na ose y a rotaci od počátku. Implementoval jsem funkci, která brala tyto tři koeficienty a obrázek podle nich posunula nebo provedla rotaci. Implementaci jsem postavil na rovnici (2.4). Rotaci jsem ale implementoval, aby se prováděla podle středu obrázku.

Od vzorce (2.7), se v rovnicích začala objevovat parciální derivace na obrázku. Podle [18][19][20] jsem implementoval v programu numerickou derivaci.

$$\frac{\delta f}{\delta x} = f(x+1, y) - f(x, y) \tag{4.1}$$

Derivace v rovnici (4.1) [18][19][20] se ale ukázala jako velmi nepřesná. Přistoupil jsem tedy k přesnější numerické derivaci, která je v rovnici (4.2).

$$\frac{\delta f}{\delta x} = \frac{f(x+1, y) - f(x-1, y)}{2} \tag{4.2}$$

Výsledky ale stále nedosahovaly požadované přesnosti. Zvolil jsem tedy možnost počítání derivace přes diskrétní Fourierovy transformace. Rovnice (4.3) [21] zobrazuje rovnici pro výpočet koeficientů Fourierovy řady.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi kn}{N}} \tag{4.3}$$

Tyto koeficienty poté používám pro výpočet derivace z inverzní diskrétní Fourierovy transformace. Rovnice (4.4) [21] obsahuje vzorec pro výpočet inverzní diskrétní Fourierovy transformace.

$$X_n = \frac{1}{N}\sum_{k=0}^{N-1} x_n e^{\frac{i2\pi kn}{N}} \tag{4.4}$$

Derivace inverzní diskrétní Fourierovy transformace je v rovnici (4.5).

$$x_n^{'} = \frac{1}{N}\sum_{k=0}^{N-1} x_n \, i2\pi k \, e^{\frac{i2\pi kn}{n}} = \frac{1}{N}\sum_{k=0}^{N-1} x_n \frac{i2\pi k}{N} \left(-\sin\left(\frac{2\pi kn}{N}\right) + i\cos\left(\frac{2\pi kn}{N}\right)\right) \tag{4.5}$$

4.5 Ukázka super rozlišení

Následující sada obrázků, Obrázek 4.2 až Obrázek 4.4, předvede metodu super rozlišení na prvním testu.

Obrázek 4.2 Jeden ze série zašuměných snímků.

Obrázek 4.3 Výsledek metody super rozlišení po 10 snímcích.

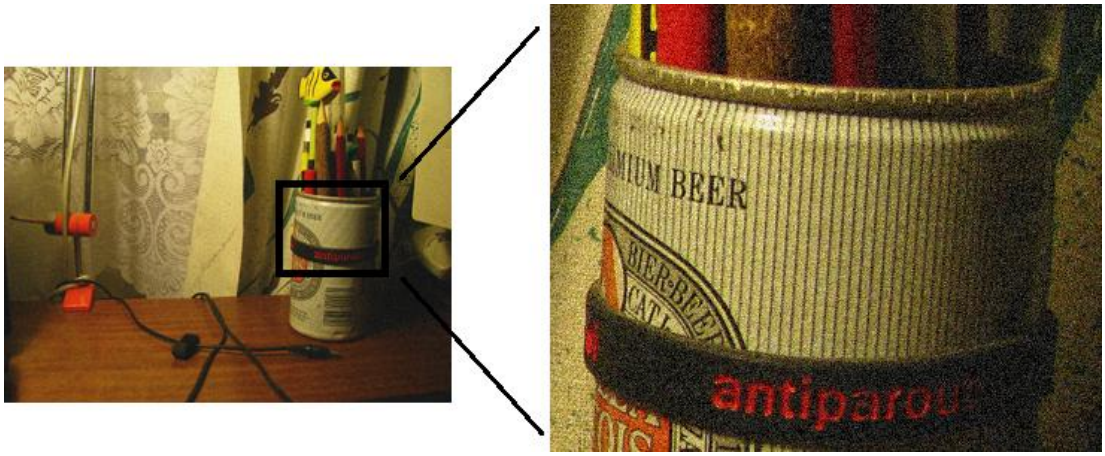
Obrázek 4.2 je jedním ze série zašuměných snímků. Obrázek 4.3 obsahuje výsledek metody super rozlišení při zpracování 10 snímků. Jak jde vidět, Obrázek 4.3 již má velmi potlačený šum. Přesto zůstává pouhým okem viditelný.

Obrázek 4.4 Výsledek metody super rozlišení po 100 snímcích.

Obrázek 4.4 ukazuje výsledný snímek pro metodu super rozlišení při zpracování 100 snímků, je velmi hladký. Rozdíl mezi obrázky Obrázek 4.3 a Obrázek 4.4 je již jen velmi malý. Jak ukazuje Obrázek 4.3, metoda je účinná už i pro malý počet snímků.

Test jsem provedl ještě pro jednu sérii hodně zašuměných obrázků. Obrázek 4.5 je jedním z řady takto zašuměných obrázků.

Obrázek 4.5 Jeden ze série zašuměných snímků.



Obrázek 4.6 Výsledek metody super rozlišení po 30 snímcích.

Obrázek 4.6 ukazuje výsledek po zpracování série třiceti takto zašuměných snímků. Algoritmus si podle tohoto obrázku poradí i s hodně zašuměnými obrázky. Šum ale není zdaleka odstraněn všechen. Oproti předchozímu testu je kvalita snímku o mnoho horší i při větším počtu složených snímků, jak ukazuje v porovnání Obrázek 4.3.

4.6 Využití OpenCL

Při spouštění programu na OpenCL se zadává počet pracovních jednotek, které budou program vykonávat [13][12]. Každá spuštěná pracovní jednotka si v programu zjistí své identifikační číslo, podle kterého by se měla rozhodnou, kterou část úlohy bude počítat. Pracovní jednotky je také možné spouštět ve více dimenzích. Identifikační čísla pracovních jednotek jsou vztažena k jednotlivým dimenzím.

Než jsem cokoliv začal psát pro OpenCL, tak jsem algoritmus implementoval v C++ a otestoval, jestli vůbec funguje. Do začátku psaní programů pro OpenCL se velmi hodila názorná ukázka na jednom konkrétním příkladu [12]. Po prostudování a vyzkoušení tohoto příkladu a dalším prostudováním dalších materiálů [11][13]. Pro potřeby psaní v tomto programovacím jazyce jsem i hodně využil referenční stánky [22]. Prvním algoritmem, který jsem se rozhodl přepsat do OpenCL byl algoritmus super rozlišení.

Pro využití maxima z OpenCL jsem se rozhodl, že se tento algoritmus bude spouštět pro každý obrázkový bod a i pro každý barevný kanál na obrázku. Tímto se algoritmus bude zabývat jedinou hodnotou z každého obrázku a po výpočtu skončí. Program tedy budu spouštět ve třech dimenzích, pro každý sloupec, řádek a každý barevný kanál.

Před spuštěním je také potřeba nahrát veškeré potřebné informace. Rozhodl jsem se, že celou sérii obrázků na zpracování předám OpenCL programu jako jednorozměrné pole. Dalšími parametry tedy musely být rozměry obrázku, aby se program mohl v poli orientovat. Jedním z dalších parametrů je i výstupní obrázek, kam jednotlivé pracovní jednotky OpenCL ukládají výsledky.

Jako druhý algoritmus, který převedu do OpenCL, jsem vybral rotaci a posun obrázku. Tento algoritmus jsem se rozhodl spustit jen ve dvou dimenzích. Jelikož se jedná o posun celého obrázkového bodu, je zbytečné ho spouštět i pro každý barevný kanál. Algoritmus jsem implementoval, aby rotoval obrázek podle středu. V tomto algoritmu jsem implementoval bilineární interpolaci. Tento algoritmus se ukázal jako velmi výkonný oproti implementaci v C++ jak je předvedeno v podkapitole 4.7.

4.7 Vyhodnocení

Záměr práce byl splněn. V práci je podrobně popsán možný způsob zpracování videa, pro účel získání co nejkvalitnějšího snímku. Jeho jednotlivé kroky před slučováním snímků i nutné úpravy snímků pro použití algoritmu na sloučení snímků. Obsahuje i podrobný popis, jak algoritmus na sloučení snímků pracuje. Popisuje i možné využití OpenCL v práci. Výsledkem práce je také aplikace. Tato aplikace splňuje požadovaný záměr práce. Zpracovává video soubor na vstupu a sloučením jeho snímků vytvoří jeden kvalitnější obrázek. Přepsané části programu do OpenCL se ukázaly být jako velmi výkonné, oproti stejnému algoritmu psanému v C++.

Sloučení třiceti snímků trvá programu, při využití standardu OpenCL, v průměru 14 sekund. Pokud program nevyužije OpenCL, trvá to v průměru 110 sekund. Snímky v tomto testu byly barevné a rozměrově 1600 bodů široké a 1200 bodů vysoké.

Další test jsem provedl s rotací a posunem jednoho snímku. Tyto operace na jednom obrázku zabraly v průměru s využitím OpenCL 2,5 sekundy. Bez využití OpenCL to zabralo průměrně 44 sekund. Rotace s posunem byla testována na barevném obrázku 3264 bodů širokém a 2448 bodů vysokém.

Testy byly prováděny na dvou-jádrovém procesoru od firmy AMD s frekvencí jednoho jádra 1,8 GHz. OpenCL bylo spouštěno také na tomto procesoru. Počítač, na kterém byly testu prováděny má k dispozici RAM o velikosti 1788MB.

5 Závěr

Cílem této práce bylo prostudovat možnosti získání co nejkvalitnějšího obrázku z videa, které obsahuje konstantní záběr jediné věci, a navrhnout a vytvořit program, který by toto video takto zpracovával. Cíl této práce byl splněn.

Při řešení práce jsem se seznámil s technologií OpenCL a prostudoval jsem její možnosti. S vedoucím práce jsem probral různé algoritmy, které by mohly být vytvořeny. Vybráno bylo zpracování série snímků z videa za účelem vytvoření co nejkvalitnějšího snímku z nich. Navrhl jsem vhodný způsob implementace algoritmů. Implementoval jsem algoritmy do aplikace. Ukázka výsledků je v podkapitole 4.5.

Výsledkem je program očekávající při spuštění na vstupu soubor, který obsahuje video. Výstupem programu je jediný obrázek, který je seskládán ze snímků videa. Tento výstupní obrázek je kvalitnější než jednotlivé snímky zpracovávaného videa.

Jednotlivé části program jsem také porovnal výkonnostně. Při testování sloučení třiceti snímků, byla doba běhu programu, při využití standardu OpenCL, v průměru 14 sekund. Pokud program nevyužije OpenCL, trvá to v průměru 110 sekund. Další test jsem provedl s rotací a posunem jednoho velkého snímku. Tyto operace na jednom obrázku zabraly v průměru s využitím OpenCL 2,5 sekundy. Bez využití OpenCL to zabralo průměrně 44 sekund. Testy byly prováděny na dvou-jádrovém procesoru od firmy AMD s frekvencí jednoho jádra 1,8 GHz. OpenCL bylo spouštěno také na tomto procesoru.

Nejvíce času jsem strávil implementací algoritmu pro registraci jednotlivých snímků. Tento algoritmus mi stále nechtěl fungovat, i když jsem v mezičase dokončil celý zbytek aplikace. Nakonec se mi ho konečně podařilo zprovoznit a aplikace byla dokončena.

V této práci je možnost pokračování v několika směrech. Je možné vytvořit přívětivé uživatelské rozhraní pro práci s tímto programem. Přidání možnosti vstupu složku obrázků. Je možné rozšířit program o kvalitnější interpolační algoritmy. Před samotným sloučením snímků lze přidat metody pro úpravy samostatných snímků. Po sloučení snímků je možné provést další úpravy na výsledném snímku. Další z možností úprav je optimalizace. Jak samotného programu, tak přídavných programů psaných pro OpenCL. Popřípadě přesunutí dalších operací nad obrázky z programu do OpenCL.

6 Literatura

[1] JORGE STOLFI, aj. *Digital Image* [online]. 13. března 2007, 9. května 2012 [cit. 10. května 2012] Dostupné na: < http://en.wikipedia.org/wiki/Digital_image >.

[2] BAKAL06, *Obraz jako signál* [online]. 2006 [cit. 10. května 2012] Dostupné na: < http://www.bakal06.chytrak.cz/2006_K44.pdf >.

[3] RS2, aj. *Discrete signal* [online]. 30. bžezna 2004, 14. ledna 2012 [cit. 10. května 2012] Dostupné na: < http://en.wikipedia.org/wiki/Discrete_signal >.

[4] SUNG CHEOL PARK, MIN KYU PARK, MOON GI KANG. Super-resolution image reconstruction: a technical overview. *Signal Processing Magazine, IEEE* [online]. May 2003 [cit. 3. října 2011], vol. 20, no. 3. S. 21-36. ISSN: 1053-5888. doi: 10.1109/MSP.2003.1203207. Dostupné na: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1203207&isnumber=27099 >.

[5] DUMIC E., GRGIC S., GRGIC, M. Hidden influences on image quality when comparing interpolation methods. *Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference on* [online]. 25-28 June 2008 [cit. 25. března 2012]. S. 367-372. ISBN: 978-80-227-2856-0. E-ISBN: 978-80-227-2880-5. doi: 10.1109/IWSSIP.2008.4604443. Dostupné na: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4604443&isnumber=4604343 >.

[6] MCHUGH S. *Digital Image Interpolation* [online]. [cit. 25. září 2011] Dostupné na: < http://www.cambridgeincolour.com/tutorials/image-interpolation.htm >.

[7] MCHUGH S. *Digital Image Enlargement* [online]. [cit. 25. září 2011] Dostupné na: < http://www.cambridgeincolour.com/tutorials/digital-photo-enlargement.htm >.

[8] ALEXEY LUKIN. *Image resampling algorithms* [online]. 1. ledna 2004, 23. dubna 2007 [cit. 25. září 2011] Dostupné na: < http://audio.rightmark.org/lukin/graphics/resampling.htm >.

[9] KEREN D., PELEG S., BRADA R. Image sequence enhancement using sub-pixel displacements. *Computer Vision and Pattern Recognition, 1988. Proceedings CVPR '88., Computer Society Conference on* [online]. 5-9 Jun 1988 [cit. 18. Prosince 2011]. S. 742-746. ISBN: 0-8186-0862-5. doi: 10.1109/CVPR.1988.196317. Dostupné na:
< http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=196317&isnumber=5032 >.

[10] FAN C., GONG J., ZHU J., ZHANG L. An improvement approach based on keren sub-pixel registration method. *Signal Processing, 2006 8th International Conference on* [online]. 2006 [cit. 12. ledna 2012], vol. 2. ISBN: 0-7803-9736-3. E-ISBN: 0-7803-9737-1. doi: 10.1109/ICOSP.2006.345563. Dostupné na:
< http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4129044&isnumber=4128962 >.

[11] STONE J.E., GOHARA D., GUOCHUN SHI. OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. *Computing in Science & Engineering* [online]. May-June 2010 [cit. 28. ledna 2012], vol. 12, n. 3. S. 66-73. ISSN: 1521-9615. doi: 10.1109/MCSE.2010.69. Dostupné na:
< http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5457293&isnumber=5457278 >.

[12] BENEDICT R. GASTER. *Introductory Tutorial to OpenCL* [online]. [cit. 22. září 2011] Dostupné na:
< http://developer.amd.com/SDKS/AMDAPPSDK/documentation/pages/TutorialOpenCL.aspx >.

[13] JUSTIN HENSLEY. What is OpenCL. *ACM SIGGRAPH ASIA 2010 Courses* [online]. 2010 [cit. 20. října 2011]. S. 1-61. ISBN: 978-1-4503-0527-3.
doi: 10.1145/1900520.1900529. Dostupné na:
< http://doi.acm.org/10.1145/1900520.1900529 >.

[14] RELAX18, aj. *OpenCV* [online]. 13. února 2006, 24. srpna 2011 [cit. 25. září 2011] Dostupné na: < http://opencv.willowgarage.com/wiki/ >.

[15] DEEPAK NAYAK*, aj. VideoCodecs – OpenCV Wiki* [online]. 27. dubna 2006, 14. dubna 2011 [cit. 18. ledna 2012] Dostupné na:
< http://opencv.willowgarage.com/wiki/VideoCodecs >.

[16] DEEPAK NAYAK*, aj. OpenCV 2.1 C++ Reference Page* [online]. [cit. 18. ledna 2012] Dostupné na: < http://opencv.willowgarage.com/documentation/cpp/index.html >.

[17] AVERY LEE. *VirtualDub* [online]. [cit. 18. ledna 2012] Dostupné na: < http://www.virtualdub.org/ >.

[18] CARLO TOMASI. *Convolution, Smoothing, and Image Derivatives* [online]. 5. února 2003 [cit. 19. prosince 2011]. Dostupné na: < http://www.cs.duke.edu/courses/cps196.1/spring04/handouts/Image%20Processing.pdf >.

[19] JAKEVORTEX, aj. *Numerical differentiation* [online]. 8. prosince 2004, 22. února 2012 [cit. 24. března 2012] Dostupné na: < http://en.wikipedia.org/wiki/Numerical_differentiation >.

[20] SUPERALE85, aj. *Finite difference coefficient* [online]. 5. dubna 2010, 28. ledna 2012 [cit. 24. března 2012] Dostupné na: < http://en.wikipedia.org/wiki/Finite_difference_coefficients >.

[21] GARETH OWEN, aj. *Discrete Fourier transform* [online]. 13. listopadu 2001, 15. dubna 2012 [cit. 1. května 2012] Dostupné na: < http://en.wikipedia.org/wiki/Discrete_Fourier_transform >.

[22] THE KHRONOS GROUP, *OpenCL 1.2 C++ Reference Pages* [online]. [cit. 18. prosince 2011] Dostupné na: < http://www.khronos.org/registry/cl/sdk/1.2/docs/man/xhtml/ >.