# Cíl práce

Cílem této práce je naimplementovat detektor objektů na GPU a následně i rozšíření pro rozpoznání obličeje a jeho sledování. Vstupem jsou obrazová data (fotografie, video) a výstupem obrazová data a statistiky s vyznačenými detekcemi odpovídající sledovaným/detekovaným objektům.

# Detekce objektů

## Waldboost

Zde doplnit (PM)

## LBP příznaky

Zde doplnit (PM)

# Implementace detektoru na GPU

GPU se oproti CPU liší především tím, že obsahuje mnohonásobně více výpočetních jader. Není vhodné tedy zpracovávat data sekvenčně, kdy by se jedním jádrem, kterých můžou být v případě moderních grafických karet stovky až tisíce, zpracovával celý výpočet, ale paralelně, kdy využijeme všech.

Detektor má 2 základní fáze:

* Vytvoření pyramidového obrazu
* Detekce obličejů na pyramidovém obrazu

## Vytvoření pyramidového obrazu

Vláken se rozeběhne tolik, kolik je pixelů původního obrazu. Následně se z každého pixelu vytváří zmenšeniny.

Jednotlivé zmenšeniny se vytvářejí po oktávách a úrovních. V jednoduchosti lze říci, že zmenšeniny jsou omezeny na maximální a minimální velikost. Vytvoří tolik obrazů kolik je oktáv \* úrovní a nevytvářejí se již zmenšeniny, které jsou menší, než určitá mez.

Každé vlákno vezme tedy jeden pixel původního obrazu a sekvenčně prochází jednotlivé zmenšeniny dané mírou zmenšení a v nich ho umísťuje.

Výstupem je jeden obraz, ve kterém jsou jednotlivé zmenšeniny původního obrazu. Tento obraz se následně nabinduje jako textura.

## Detekce objektů

Vstupem do samotné detekce je pyramidový obraz uložený jako textura. Vláken se rozeběhne tolik, kolik je pixelů pyramidového obrazu.

Každý pixel je počátkem vzorku daného velikostí, na kterou je natrénován detektor. V našem případě 26x26. Následně prochází jednotlivými stages detektoru a snižuje nebo zvyšuje odezvu. V případě, že odezva spadne pod mez danou konkrétní stagí, je výpočet pro daný vzorek ukončen. V případě, že výpočet projde přes všechny stages, je zkontrolována ještě konečná mez, na odstranění parazitních vzorků s velmi malou odezvou. Pokud je odezva větší i než konečná mez, jedná se o hledaný objekt.

Takovýchto detekcí je tradičně více na jednom místě a to pro jednotlivé zmenšeniny.

## Uspořádání paměti

Podstatné součásti detektoru jsou rozděleny na následující části:

* Stages – constant memory
  + K jednotlivým stages je přistupováno sekvenčně pro každý pixel (výpočetní vlákno) obrazu. Jsou tak broadcastováný do všech výpočetních jader.
* Alphas – texture (global) memory
  + Náhodné přístupy do constant memory by dělali přístup k alphám velmi neefektivní, protože by se zbytečně broadcastovaly. Navíc se jedná o data, kterých je více, než-li constant memory na většině grafických karet. Texturovací pamět je optimalizována pro náhodný read-only přístup.
* Původní obraz – texture (global) memory
  + Texturovací pamět, krom toho, že je optimalizována pro náhodné read-only přístupy umožňuje bilineární interpolaci 2D dat. Lze tak za pomocí hardwarových jednotek počítat zmenšeniny obrazu
* Pyramidový obraz – texture (global) memory
* Obrazové parametry – constant memory
  + Parametry obrazu, se kterými se hojně pracuje, např. velikost obrazu jsou taktéž ukládány v constant memory. Vzhledem k tomu, že se pro jednotlivá vlákna nemění, je vhodné je broadcastovat a využívat rychlého přístupu.

# Trackování a rozpoznávání obličejů

*Zde doplnit (JB).*

# Rozdělení práce

Pavel Macenauer – detektor objektů na GPU (zároveň diplomová práce), rozpoznání objektů a kostra aplikace

Jan Bureš – sledování objektů a jejich rozpoznávání. *Zde případně doplnit (JB).*

# Ovládání programu

## Překlad

Závislost na knihovnách: OpenCV 2.4.\* (testováno na 2.4.9, 2.4.10), CUDA 6.5

Přiložen projekt pro Microsoft Visual Studio.

Nejaktuálnější verze je k nalezení na: https://github.com/mmaci/vutbr-fit-pov-face-tracking

## Spuštění

Program se spouští přes příkazovou řádku s následujícími parametry:

-ii [input file] or -di [dataset] or -iv [input video] and -ot [output track info]

Input file – vstupní obrázek

Dataset – textový soubor obsahující seznam obrázků

Input video – vstupní video

Output track info – soubor pro textový výstup

Výstupem je textový soubor s výsledky trackování a výřezy jednotlivých obličejů. Vše je uloženo ve stejné složce, ve které je spuštěna aplikace.

# Závěr a vize do budoucnosti

Naimplementovaný detektor objektů je funkční a na mobilní grafické kartě Nvidia Quadro K1000M zvládá detekovat obličeje na HD obraze průměrně za 100ms, což odpovídá 10 FPS. Na zvýšení rychlosti by bylo třeba omezit výpočet např. zredukováním počtu vyhodnocovaných stagí z 2048 na ½ nebo ¼, čímž snížíme jeho přesnost, ale snížímé délku výpočtu pod úroveň FPS v běžných videích.

Další možné optimalizace do budoucnosti jsou:

* Zpřesnění generování zmenšenin
  + Aktuálně jsou sice generovány z texturovací paměti, ale v ní jsou uložena obrazová data jako unsigned char, což vede ke zmenšeninám pomocí alg. Nearest neighbour. Převod na float by umožnil bilineární interpolaci pomocí hardwaru.
* Rozdělení vyhodnocování stagí na více kernelů a reorganizace vláken
  + Aktuálně probíhá vyhodnocování stagí od začátku do konce v rámci jednoho kernelu. Většina vzorků je však zamítnuta již na začátku. Šlo by tak zjistit, pro které vzorky stále ještě běží výpočet a spustit nový kernel pouze pro ně.
* Optimalizace zmenšenin na velikost videa
  + Velikost videa je často standardizována na velikost a poměr stran (16:9, 4:3, HD, FULL HD, …). Šlo by tak generovat zmenšeniny pro daný standard, kde by se využilo maximum obrazové plochy, zmenšila její i urychlit výpočet zmenšenin rozbalením cyklů.

*Zde případně doplnit (JB).*