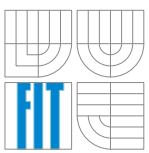
Detekce objektů na CUDA, jejich rozpoznání a sledování

Pavel Macenauer
xmacen02@stud.fit.vutbr.cz

Jan Bureš
xbures19@stud.fit.vutbr.cz

Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

12. ledna 2015



Obsah

Cíl práce

- Detektor obličejů na CUDA
- Trackování obličejů
- Rozpoznání obličeje

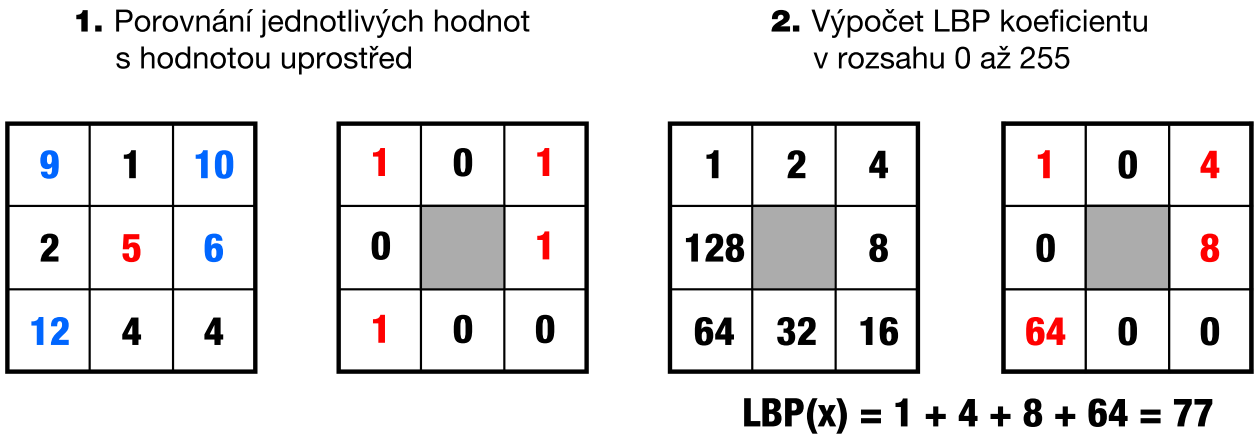
Detekce objektů

- **Waldboost**
  Metaalgoritmus, který skládá slabé klasifikátory v jeden silný. V každém kroku kontroluje, zda-li nepřesáhl danou mez a případně skončí.
- slabý klasifikátor - **LBP** příznaky

Waldboost

Data: $h^{(t)}$, $\theta_A^{(t)}$, $\theta_B^{(t)}$, γ, x
Result: +1, -1
begin
  for 1 **to** T **do**
    if $H_T \geq \theta_B^{(t)}$ **then**
      classify x to the class +1 and terminate
    end
    if $H_T \leq \theta_A^{(t)}$ **then**
      classify x to the class -1 and terminate
    end
  end
  if $H_T > \gamma$ **then**
    classify x to the class +1
  else
    classify x to the class -1
  end
end

LBP příznaky



- Pro každý pixel prochází 1 až N stages
- Pro daný stage spočte LBP = kód a z tabulky určí odezvu
- Ukončí výpočet, pokud odezva přesáhne mez
- Dojde-li nakonec stages, zkontroluje konečnou mez a rozhodne

Implementace

2 fáze (kernely):

1. Postavení pyramidového obrazu
   Pro každý pixel původního obrazu se rozběhne vlákno a generuje N zmenšenin
2. Detekce objektů
   Pro každý pixel pyramidového obrazu se rozběhne vlákno a vyhodnocuje příznaky.
3. Detekce je většinou nalezena na podvzorkovaném obraze - přepočítání na původní

Pyramidový obraz

Uspořádání detektoru na GPU - v CUDA paměti

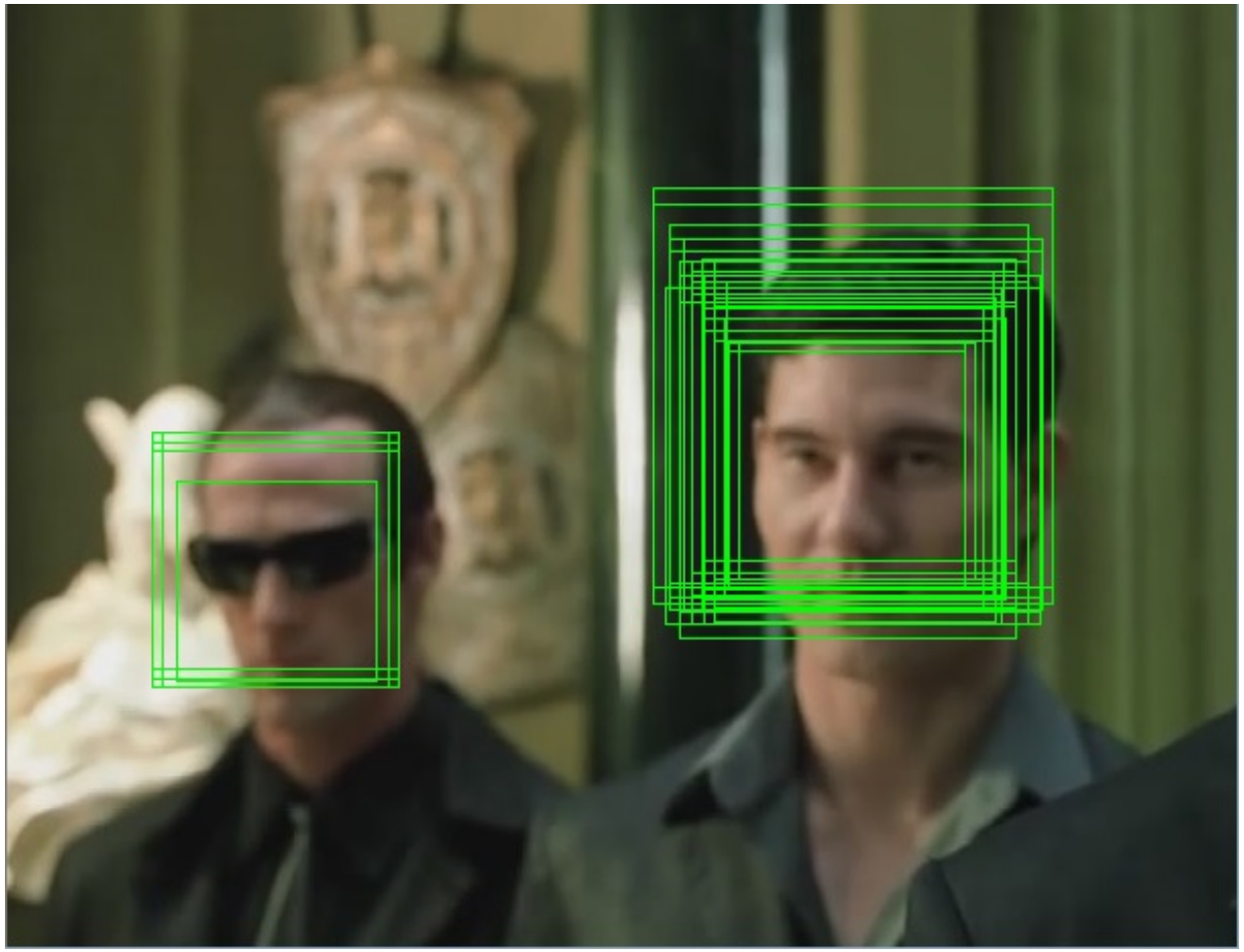
- **Stages** – constant memory
- **Alphas** – texture (global) memory
- **Původní obraz** – texture (global) memory
- **Pyramidový obraz** – texture (global) memory
- **Obrazové parametry** – constant memory

Optimalizace

- Detektor vyexportován z XML do `C++` headeru
- Příznaky jsou max. velikosti 2x2
  Bilineární interpolace jednotlivých LBP hodnot na HW pomocí texturovací paměti
- Stages v konstantní paměti - umožňuje broadcast
- Slabá podpora mipmap přímo pomocí CUDA. Pyramidový obraz? Bilineární interpolace pomocí texturovací paměti na HW
- Seskládání pyramidového obrazu, aby zabíral méně prostoru

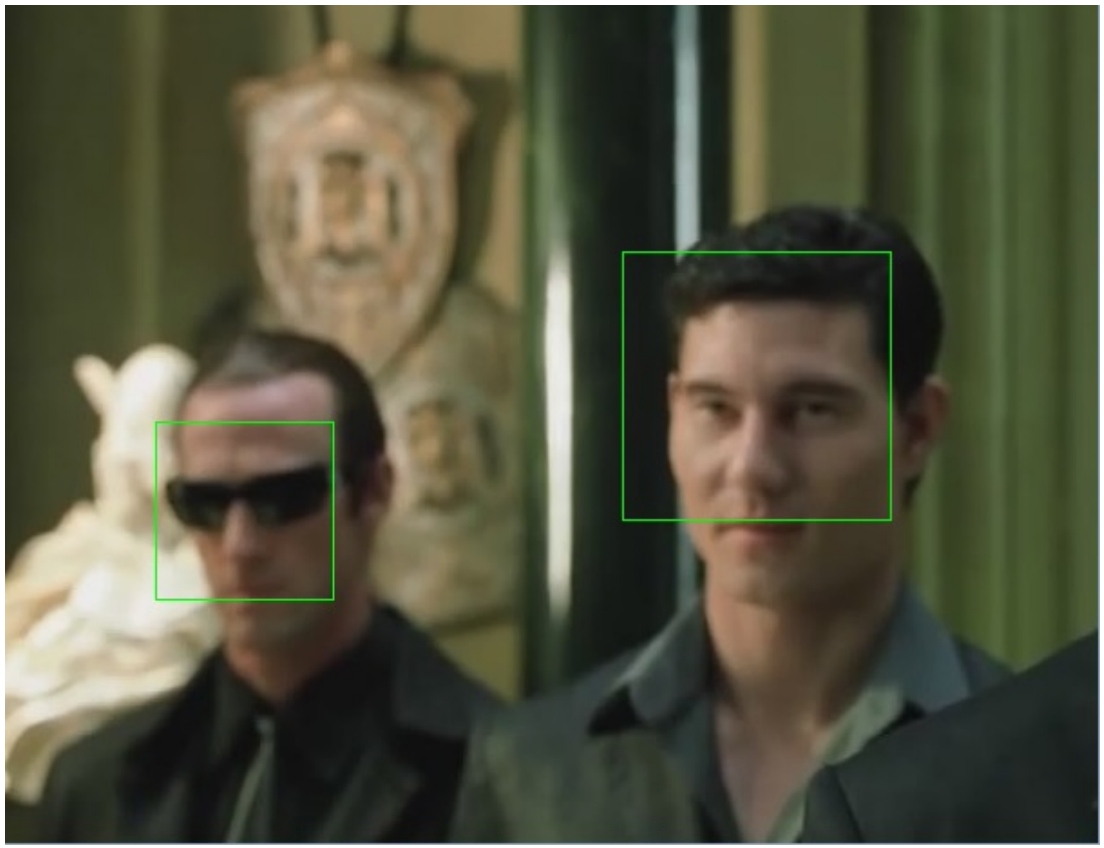
Výstup z detektoru

- Několik detekcí na jeden obličej



Předzpracování

- Porovnání detekcí po dvojicích
- Společná plocha `> 50%`
- Vyberu tu s lepším hodnocením



Rozpoznání pomocí klíčových bodů

- Většinou 0-3 body na jednu oblast.
- Nelze použít.

Porovnání histogramu

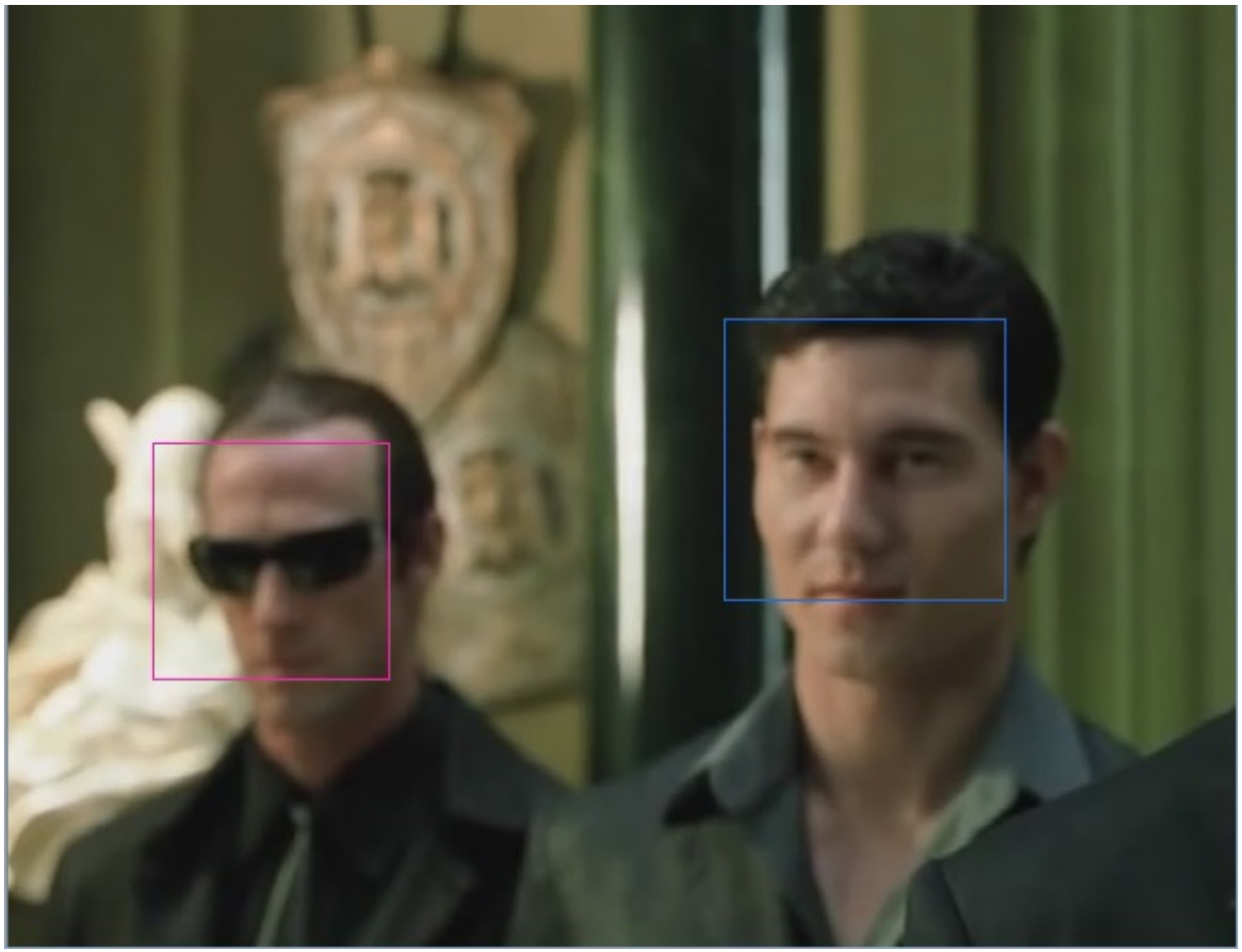
- Převod oblasti do HSV
- Histogram ze složek H a S
- Porovnání pomocí Bhattacharyya distance
- Hodnota (vzdálenost) od 0 do 1
- Problém s nastavením prahu pro nový obličej (stejný obličej 0,1 až 0,35)

Bhattacharyya distance:

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\overline{H_1}\,\overline{H_2}N^2}} \sum_I \sqrt{H_1(I) \cdot H_2(I)}} \qquad (1)$$

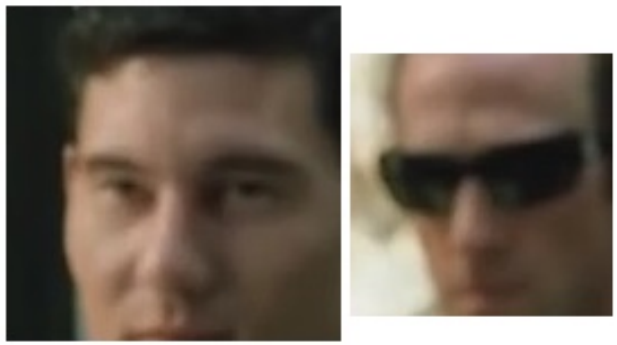
Vzdálenost od posledního výskytu

- Normalizována do rozsahu $\langle 0; 0,5)$
- Výsledné score tedy $\langle 0; 1,5)$ práh $0,6$



Výstup programu

- Textový soubor se seznamem unikátních obličejů a pozicí obličejů v každém snímku
- Unikátní obličeje v samostatném souboru



Možná vylepšení

Další metriky:

- Vzdálenost očí-nos-ústa
- Směr a rychlost pohybu

Děkujeme za pozornost.
Otázky?