# Projekt 3

František Kajánek

# Ciele

Cieľom projektu 3 bolo pokračovať prácu z projektov 1 a 2, a to použitie AdaBoost algoritmu na detekciu objektov v obraze, s použitím vybraného deskriptora – Haarové vlnky. Taktiež bolo potrebné porovnať OpenCV Adaboost implementáciu s MultiBoost implementáciou a naimplementovať optimalizovanú metódu detect multi scale s použitím haarových vlniek.

## Predošlé zdroje

Z predošlých projektov sa prenášalo nasledovné:

1. Na projekte 1 boli nadobudnuté znalosti z Computer vision – hlavne Computer vision prezentácie od Mubarak Shah a analýza knižnice OpenCV
2. Na projekte 1 a 2 bol vytvorený dataset futbalistov a taktiež v niektorých obrázkoch z neho boli naanotované objekty – obrázky z datasetu
3. Na projekte 2 bola naimplementovaná naivná detekcia objektov pomocou OpenCV AdaBoost algoritmu a taktiež boli naimplementované podporné časti pre čítanie a zápis na disk. Naivná verzia detekcie nebola optimalizovaná a ešte nepoužívala žiadne vlastné deskriptory obrázkov.

## Postup

Prípravou pre ciele bola hĺbková analýza existujúcich algoritmov. Analyzoval som Viola-Jones kaskádu, ktorá mala naimplementovaný výpočet Haarových vlniek, HOG deskriptora v spojení s AdaBoost kaskádou, a taktiež obsahovala detectMultiScale s použitím spomínaných deskriptorov. Z tejto implementácie som hlavne čerpal pri implementácií Haarových vlniek, detectMultiScale a taktiež pri analýze dostupných nastavení pre 3.0 AdaBoost. Je potrebné spomenúť že Viola-Jones upravený AdaBoost, má zoptimalizovaný detekčný mód, dokonca až na úrovni rozhodovacích stromov.

Následne som taktiež analyzoval detekčný mód 3.0 AdaBoost a jeho vnútorné fungovanie, za cieľom zhodnotiť ako veľmi je ho možné zoptimalizovať. Tento algoritmus používa rozhodovacie stromy ako slabé klasifikátory.

Ako prvý som naimplementoval výpočet Haarových vlniek. Implementácia má niekoľko verzií, s ktorými som experimentoval. Prvá bola naivná verzia, ktorá vie z ľubovoľného integrálneho obrazu vypočítať ľubovoľnú Haarovú vlnku. Celkovo je naimplementovaných 17 rôznych vlniek. Následne som urobil zoptimalizovanú implementáciu pre zapojenie do detectMultiScale, ktorá vytvárala omnoho menej objektov a taktiež skoro vôbec nevyužívala dynamickú alokáciu a dealokáciu, ktorá výrazne spomaľovala naivnú verziu.

<typy vlniek>

Následne som implementoval benchmark algoritmus na testovanie natrénovaného modelu. Táto vezme obrázky na disku, kde sú rozdelené podľa toho či sú to pozitívne alebo negatívne vzorky, a dá ich OpenCV AdaBoostu aby rozhodol do ktorej triedy patria. Následne porovnávam očakávané a výsledné rozhodnutie a podľa toho určím koľko je pozitívny, negatívnych a nesprávnych pozitívnych a negatívnych výsledkov. Cieľom je zistiť či nálezy pozitívov a nesprávnych pozitívov je nad a pod určitú hranicu.

Nakoniec som sa venoval optimalizácií detectMultiScale algoritmu. Prvá verzia sa posúvala po obrázku a vysekávala okná, ktoré následne zväčšovala/zmenšovala podľa potreby pre natrénovaný AdaBoost model. Problém bol, že menenie veľkostí obrázkov je náročná operácia, a taktiež je náročná alokácia pamäte. Táto verzia bežala dlhšie ako hodinu pre jeden obrázok. (aj po hodine neboli hotové všetky mierky).

Finálna verzia po optimalizácií má trvanie od 15-20 sekúnd pre jeden QHD obrázok. Tento algoritmus je možné ďalej optimalizovať pomocou multithreadingu a taktiež je možné upraviť algoritmus tak aby zanalyzoval natrénovaný AdaBoost model a následne počítal len hodnoty, ktoré AdaBoost používa.