

Image Retrieval – Podobnost obrázků metoda (A)SIFT

Filip Glazar
glazafil@fit.cvut.cz

Popis projektu

Cílem projektu bylo vypracovat softwarové řešení, které na vstupu přijme obrázek a porovná jeho podobnost s ostatními obrázky v databázi. Výstupem je seřazená database obrázku, dle podobnosti s obrázkem vstupním.

Způsob řešení

Řešení je rozděleno na tři části.

Knihovna

Za první část považuji samotnou knihovnu pro vyextrahování deskriptorů z obrázku. Pro tento účel jsem použil doporučenou knihovnu [ASIFT](#), kterou jsem mírně upravil tak, aby extrahovala deskriptory pouze z jednoho obrázku. K tomuto slouží samostatný modul asifator, kterému se předá jméno souboru, který chci zpracovat a on z něj vyextrahuje deskriptory a uloží do textového souboru.

Porovnání obrázků

Druhou částí řešení bylo implementovat samotné zpracování deskriptorů. Konkrétně v našem případě redukce počtu deskriptorů pomocí clusteringu a porovnání těchto bodů pomocí Signature Quadratic Form Distance (SQFD).

Clustering

Tomuto modulu je předán opět název souboru, podle kterého si program načte všechny vyextrahované body a pevně se určí epsilon. Poté se postupně projdou všechny body a dle maximální možné vzdálenosti definované hodnotou epsilon se seskupí do jednotlivých clusterů. Poté co je toto rozdělení dokončeno se pro každý cluster nalezne střed tzv. centroid, který reprezentuje celou skupinu bodů a společně s celkovým počtem bodů v clusteru se uloží do textového souboru.

SQFD

Tento modul přijme na vstupu clustery pro jednotlivé obrázky a pro všechny obrázky vzhledem k jednomu danému vypočítá skoré, řekněme míru podobnosti. Čím je tato míra podobnosti větší, tím menší je hodnota na výstupu algoritmu.

Frontend

Třetí a neméně důležitou částí je frontend, který umožňuje používat výše zmíněné algoritmy. Pro frontend jsem zvolil PHP Framework Symfony. To znamená, že je aplikace snadno rozšiřitelná a udržitelná. Frontend umožňuje především nahrávat obrázky do databáze a zobrazit samotný výstup po zpracování vstupních dat backendem.

Implementace

Pro implementaci jsem použil jazyk C++ a to především kvůli volbě dostupné knihovny a v neposlední řadě jeho rychlosti. Jak jsem již zmínil frontend běží na Symfony a to vyžaduje PHP verze minimálně 5.6 a jakýkoliv webový server. Zde byl použit Apache. Samotné obrázky a jejich miniatury včetně originálů jsou uloženy v přehledné adresářové struktuře. Jako textové soubory jsou uloženy i vyextrahované a dále zpracované deskriptory. Pro uložení

záznamů o obrázcích pro potřeby frontendu je využita MySQL databáze s velmi primitivním schématem.

compic.image	
id	int(11)
name	varchar(255)
status	int(11)
orig_name	varchar(255)
keysCnt	int(11)
duration_keys	double
duration_clusters	double


















compic.comparsion	
id	int(11)
image	int(11)
duration	double
time	datetime
level	int(11)

compic.subcomparsion	
id	int(11)
comparsion	int(11)
image	int(11)
ratio	double

Výstup

Výstup je rozdělen na dvě části a to stránka přehledu všech porovnání a samotný detail porovnání.

Rozcestník porovnání


Comparisons			
Image	Score	Date	Time
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10
	95	10.10.2020	10

Zde jsou přehledně vypsané všechny předchozí porovnání. Obrázky jsou seřazeny dle data provedení. Po kliknutí na obrázek přejde uživatel na detail porovnání.





Detail porovnání





ComPic Home Comparisons Upload Stats





Comparsion #17





Result









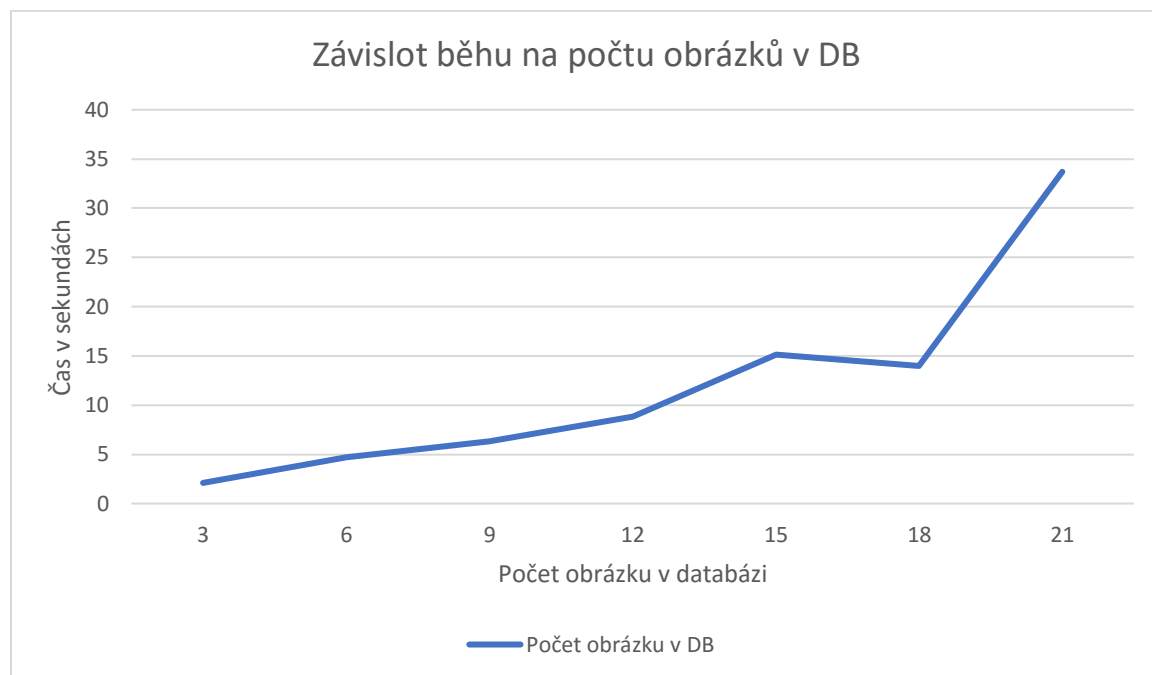
Stats

Operation	Duration
Keyp extraction	45s
Clustering	12s
SGFD	38s

Detail je samotné srdce frontendu. Nahoře je zobrazen obrázek, který jsme porovnali a pod ním miniatury obrázků seřazený dle podobnosti. Na konci stránky nalezneme jednoduché statistiky toho kolik jaká akce zabrala času při výpočtu.

Experimentální sekce

Zde se nabízí popsat na čem čas výpočtu aplikace nejvíce závisí. Vše se odvíjí od parametru epsilon v clusteringu. Čím menší je tato konstanta, tím více clusterů je vygenerováno a tím delší je potom porovnání se všemi ostatními clustery pro všechny obrázky. Velmi jsem se snažil algoritmus clusteringu optimalizovat, bohužel asymptotická složitost je neúprosná a vzhledem k tomu, že aplikace byla původně plánovaná k použití na velkém množství dat je toto řešení nevhodné. Je potřeba zahrnout i časovou náročnost SQFD, která je $O(N \cdot M)$, kde N a M jsou počty clusterů porovnávaných obrázků. Jako další jsem zkoumal závislost délky výpočtu na počtu obrázků v databázi.



Z tohoto grafu je vidět, že čím více obrázků v DB máme tím delší je jejich porovnání. Dobré je, že tento nárůst není nikdy větší, než lineární. To nám zaručuje dobrou škálovatelnost až po vyšší počty objektů.

Diskuze

Vzhledem k dosaženým výsledkům a měřením si dovoluji tvrdit, že takto koncipovaná aplikace je v praxi použitelná pouze jako doplněk k jiným metodám porovnání. Například srovnání histogramů a tak podobně.

Závěr

Věřím, že cíle projektu byly splněny, ale pro reálné nasazení je potřeba provést jisté úpravy. Například vůbec nefunguje rozpoznání na rotacích objektů, což je zřejmě vzhledem k použitým algoritmům. Velmi dobrých výsledků jsem dosáhl při použití velmi podobných obrázků. Bohužel se spíše stávalo, že i při použití velmi rozdílných obrázků bylo skóre velmi nízké. Při volbě nižších epsilon u clustering se výsledky zlepšují ovšem určitě ne adekvátně k prodlužující se době běhu.