Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra Kybernetiky

Dokumentace

Queetech LBP knihovna

2012 verze 0.9

Obsah

1 Úvod

Tato knihovna se zabývá metodou LBP[1], která je v současné době hojně využívána v počítačovém zpracování obrazu k rozpoznávání, klasifikaci či případně segmentaci textur objektů. Předkládaný kód je výsledkem řady dřívějších pokusů a testů. Obsahuje všechny nejpoužívanější verze metody LBP pro 2D data včetně speciální real time implementace, která je oproti klasické mnohonásobně rychlejší. Přes svojí rychlost má tato implementace také své nevýhody - viz část 2 tohoto dokumentu.

V dalších částech budou postupně probrány všechny součásti knihovny. Od imlementace v C++, přes wrapper pro matlab až po ukázkové aplikace, které jsou součástí řipraveného balíku knihovny. Součástí popisů je i návod k přeložení knihovny a přehled jednotlivých verzí.

2 LBP

Jedná se o statistickou metodu, která se snaží popsat texturu objektu na základě lokálních charakteristik. To v praxi znamená, že pro každý pixel obrazu je z jeho předem vybraného okolí o n bodech vypočítáno *n-bitové* číslo. Toto číslo je následně uloženo na pozici pixelu, který byl pro dané okolí středovým elementem. Princip je ukázán v následujícím přikladu

$$\mathbf{G} = \begin{bmatrix} g_1 & g_2 & g_3 \\ g_8 & g_0 & g_4 \\ g_7 & g_6 & g_5 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 3 & 1 \\ 7 & 2 & 1 \\ 9 & 1 & 4 \end{bmatrix} \Rightarrow \sum_{i=1}^{n} sg(g_i - g_0) \cdot 2^{n-1} \Rightarrow b = [11010010],$$

kde G je okolí bodu g_0, funkce $sg(x) = 1$ pro $x \geq 0$ a $sg(x) = 0$ pro $x < 0$ a b je výsledné binární číslo.

Uvedený proces se vztahuje k výpočtu jedné LBP hodnoty. K popsání celého obrazu je potřeba provést tento postup pro každý pixel zdrojových dat, kolem kterého existuje definované okolí. Výsledná LBP reprezentace by tedy v ideálním případě měla být stejně velká jako původní obraz. Většinou však existují body, které ohodnotit nemůžeme a to na okrajích zdrojového obrazu. To však není velký problém, neboť se při porovnávání texturních reprezentací založených na této metodě nepoužívá celá reprezentace, nýbrž jen její histogram. I přes redukci informace o pozici dosahuje metoda LBP velice dobrých výsledků.

Metodou LBP se po celém světě zabývá mnoho vědců a vědeckých skupin. Proto není divu, že existuje velké množství úprav a rozšíření. Dvěma nejznámějšími jsou takzvané uniform patterns a rotmin LBP.

[1] Local Binary Patterns

2.1 Uniform patterns

První uvedené slouží k omezení skupiny hodnot, které může LBP produkovat. Toho je docíleno tak, že se počítají jen ty hodnoty, jejichž bitová reprezentace obsahuje maximálně dva přechody $0 \rightarrow 1$ resp. $1 \rightarrow 0$ a ostatní sečte do jedné předem určené hodnoty. V této práci budou tyto hodnoty ukládány na pozici 59. Při 8-bitovém okolí je pak počet hodnot histogramu omezen z 256 na 59.

2.2 Rotmin LBP

Rotmin LBP pracuje tak, že každé binární číslo LBP reprezentace zrotuje (respektive zrotuje bity tohoto čísla) tak, aby bylo ze všech rotovaných čísel nejmenší. Tím se dosáhne invariance vůči natočení textury a zároveň k výraznému snížení počtu hodnot v histogramu.

2.3 Zrychlení výpočtu LBP reprezentace

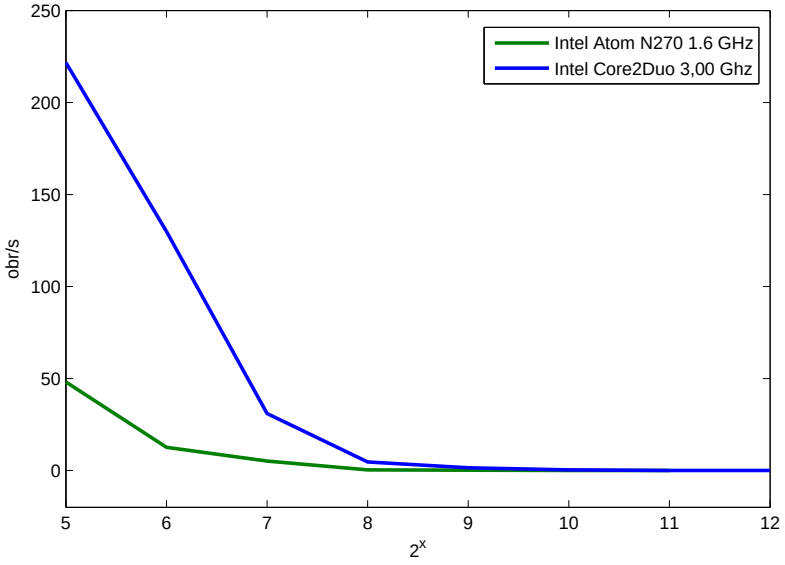
Výpočet jednoho bodu LBP reprezentace naznačený výše je spolehlivý, ale výsledný algoritmus pracuje značně pomalu. Proto se hledají způsoby jak výpočet urychlit. Možností existuje samozřejmě více. Například použití výpočtů na grafické kartě pomocí technologií CUDA a OpenCl. Druhou možností je využití procesoru v PC na maximum.

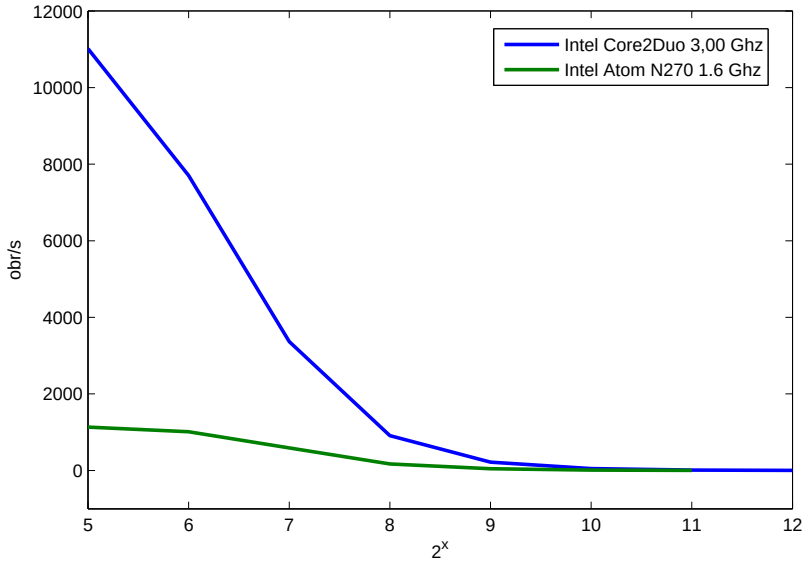
Právě druhá možnost byla zvolena i v tomto případě. Princip spočívá v eliminaci podmíněných skoků ve výpočtu algoritmu. Postup výpočtu jednoho bodu texturní informace je následující (ukázka počítá se 4-okolím počítaného bodu).

$$\begin{bmatrix} & 3 & \\ 7 & 2 & 1 \\ & 1 & \end{bmatrix} \Rightarrow \begin{array}{cccc} i & g_0 - g_i - 1 & b & Znamenko \\ 1 & 2-3-1=-2 & 1110 & 0001 \\ 2 & 2-1-1=0 & 0000 & 0000 \\ 3 & 2-1-1=0 & 0000 & 0000 \\ 4 & 2-7-1=-6 & 1010 & 1000 \\ & & LBP_{4,1} & 1001 \end{array}$$

V tomto případě je výpočet omezen na sčítání a bitové operace. Postup je takový, že se provede rozdíl $g_0 - g_i$. Z tohoto výsledku se pomocí operace AND s číslem 1000 zjistí znaménko rozdílu. Toto znaménko se následně orotuje doprava o $3 - i$. Tím je získána jeden bit LBP hodnoty. Po získání všech čtyř se pomocí operace OR sečtou.

Takto upravený algoritmus dokáže celý výpočet výrazně urychlit. Například sada 83 obrazů o rozměrech 512x512 pixelů se v původním tvaru výpočtu počítá 65 vteřin (počítáno na domácím počítači s dvoujádrovým procesorem o taktu 3,33 GHz) a při použití nového výpočtu reprezentace se stejná sada vypočítala za 0,7 vteřiny. Následující grafy ukazují závislost počtu vypočítaných LBP reprezentací za vteřinu na velikosti obrázku. První graf ukazuje počty při klasickém výpočtu a druhý při rychlém výpočtu.





3 Popis knihovny

V této části bude popsána implementace knihovny v programovacím jazyku C++. Bude zde uveden seznam funkcí včetně jejich parametrů a chování.

3.1 Změny ve verzích

- **0.1.** - První funkční verze obsahující základní verzi LBP algoritmu
- **0.2.** - Přidána podpora rotmin algoritmu

- **0.3.** - Přidána podpora uniform patterns
- **0.4.** - Došlo k překopání jádra knihovny a rozdělení na více samostatných souborů
- **0.5.** - Přidání možnosti výběru počtu okolních bodů a poloměru masky
- **0.6.** - Přidány funkce pro práci ukládání/načítání obrázků a dat
- **0.7.** - Vytvořen wrapper pro Matlab a ukázkové aplikace
- **0.8.0** - Přidána podpora realTime LBP algoritmu do knihovny
- **0.8.1** - Přidána podpora realTime LBP algoritmu do wrapperu pro Matlab
- **0.9.0** - První verze dostupná na internetu

3.2 Minimální požadavky

Pro přeložení knihovny a běh aplikací je potřeba následující programové vybavení

- Překladač C/C++
- Cmake od verze 2.8.
- ITK od verze 2.4.(v současné době nepodporuje ITK 4)
- soubor ITKCommon.dll dostupný v systému Windows

3.3 Překlad staženého balíčku

K překladu kódů je potřeba mít v počítači zkompilovaný ITK toolkit od verze 2.4 do verze 3.2 a program cmake, který sám vytvoří potřebné knihovny. Soubor *CMakeLists.txt* obsahuje celý proces vytvoření projektu ve vybraném IDE. Vše je vyzkoušeno ve Visual studiu 2008 (Windows) a makefile (Linux). Po vytvoření projektu by mělo být možné celou knihovnu zkompilovat. Uživatel tak získá dva ukázkové programy a dynamickou knihovnu, kterou lze připojit například k prostředí Matlab pomocí přiloženého wraperu.Pro podrobnější pochopení překladu LBP knihovny nahlédněte do okomentovaného souboru *CMakeLists.txt*.

3.4 Popis funkcí

Na začátek této části je potřeba říct, že hlavní částí knihovny je třída imageToLbpFilter, která je začleněna do bnalíku ITK tím, že je vytvořena jako filtr tohoto balíku. Tudíž se s tímto filtrem pracuje stejně jako i s ostatními filtry ITK toolkitu. Filtr však navíc obsahuje několik funkcí

- **SetMaskProperties(radius, samples)** - Nastavení poloměru a počtu bodů v okolí středového bodu masky.
- **SetMappingType(type)** - Nastavení typu LBP algoritmu (viz soubor imageToLbpFilter.h).
- **GetMappingType()** - Vrátí číselně nastavený typ algoritmu.
- **GetSamples()** - Vrátí číselně počet bodů v okolí středového bodu masky.
- **GetRadius()** - Vrátí nastavený poloměr masky.

Krom samotného filtru knihovna obsahuje soubory s pomocnými funkcemi a s funkcemi, které filtr využívá. Nejdříve budou uvedeny funkce, které využívá filtr. Dají se ovšem použít i samostatně případně se pro filtr dají lehce přepsat, neboť se nachází v souboru mimo samotný filtr. Následuje jejich výpis. Pro podrobnosti o typech parametrů se prosím podívejte do příslušných souborů (convolution.h, mapping.h, mask.h)

- **convolutionFFT(*image, *filters)** - Provede konvoluci zdrojovych dat a pripravenych filtru. Vrati pole vyslednych konvoluci.
- **basic(*LbpImage, *filters)** - Vytvoří LBP obraz ze skupiny odezev na filtry získaných konvolucí dle základního algoritmu LBP.
- **rotmin8(*LbpImage, *filters)** - Vytvoří LBP obraz ze skupiny odezev na filtry získaných konvolucí dle algoritmu rotmin LBP. Funguje maximálně do 8 okolních bodů.
- **rotmin16(*LbpImage, *filters)** - Vytvoří LBP obraz ze skupiny odezev na filtry získaných konvolucí dle algoritmu rotmin LBP. Funguje maximálně do 16 okolních bodů.
- **rotmin32(*LbpImage, *filters)** - Vytvoří LBP obraz ze skupiny odezev na filtry získaných konvolucí dle algoritmu rotmin LBP. Funguje maximálně do 32 okolních bodů.
- **uniform(*LbpImage, *filters)** - Vytvoří LBP obraz ze skupiny odezev na filtry získaných konvolucí dle algoritmu uniform patterns LBP.
- **generate2D(filters, radius, samples, length)** - Vytvoří masky pro konvoluci dle zadaných parametrů - poloměr, počet okolních bodů, velikost strany zdrojových dat.
- **generateEight2D(filters, length)** - Vytvoří masky pro konvoluci dle zadaných parametrů - poloměr = 1, počet okolních bodů = 8, velikost strany zdrojových dat.

- **generateFour2D(filters, length)** - Vytvoří masky pro konvoluci dle zadaných parametrů - poloměr = 1, počet okolních bodů = 4, velikost strany zdrojových dat.

Dále knihovna disponuje funkcemi pro ulehčení práce se soubory, histogramy a pro porovnávání výsledků a tedy zjišťování podobnosti zdrojových dat.

- loadImage(*image, filename) - Načte obrázek z umístění zadaného v parametru filename.
- rescaleImageIntensity(*image, *image2, min, max) - Převede hodnoty pixelu v obrazu do zadaných mezí.
- saveImage8bit(image, filename) - Uloží data do 8 bitové reprezentace na umístění zadané v parametru filename.
- saveImage16bit(image, filename) - Uloží data do 16 bitové reprezentace na umístění zadané v parametru filename.
- createSimpleHistogram(image, *histogram, hSize) - Vytvoří histogram obrazu (musí být v ITK reprezentaci), délka histogramu se předává v parametru hSize.
- euclideanDistance(histogram1, histogram2) - Porovná histogramu pomocí euklidovy vzdálenosti.
- minMaxDistance(histogram1, histogram2) - Porovná podobnost histogramů pomocí min max kritéria.

4 Wrapper pro Matlab

Jak již bylo řečeno je součástí knihovny také wrapper pro Matlab. Při kompilaci knihovny se vytvoří soubory, které jsou připravené pro daný systém a korespondují s názvy knihoven a hlavičkových souborů, které využívají. V základu je to lbp.dll a lbp.h pro windows a liblbp.a a lbplinux.h pro linuxové distribuce. Součástí wrapperu jsou následující matlabovské skripty.

- **loadLbpLibrary.m** - Načte dynamickou knihovnu s LBP algoritmem.
- **unloadLbpLibrary.m** - Ukončení práce s dynamickou knihovnou.
- **imageToLbp.m** - Převede data do LBP reprezentace dle zadaných parametrů
- **realTimeLbp.m** - Převede data do LBP reprezentace pomocí real time LBP algoritmu

Použití knihovny může vypadat následovně :
Pro podrobnější informace jsou všechny soubory okomentovány a dále je k dispozici několik ukázkových programů a skriptů, které je možné k pochopení práce s wrapperem použít.

Algoritmus 1: Ukázka práce s wrapperem

```
% Načtení LBP knihovny
loadLbpLibrary();
% Vytvoření LBP reprezentace
LBP = imageToLbpR(image,1,1,8);
% Odpojení LBP knihovny
unloadLbpLibrary();
```

5 Ukázkové aplikace

6 Závěr

7 Příloha