IMAGE PROCESSING LIBRARY WRAPPER FOR LUA

Jiří Prymus
Master Degree Programme (2), FEEC BUT
E-mail: xprymu00@stud.feec.vutbr.cz

Supervised by: Petr Petyovský
E-mail: petyovsky@feec.vutbr.cz

Abstract: This paper deals with OpenCV library wrappered to Lua language thus providing fast image processing library to non-type interpreted language. For easier compiling of library and distributing are used open source applications CMake and NSIS. Whole project is stored at SourceForge repository as open source project.

Keywords: Image processing, OpenCV library, Lua, C++ wrapper, CMake, NSIS, Sourceforge

1 ÚVOD

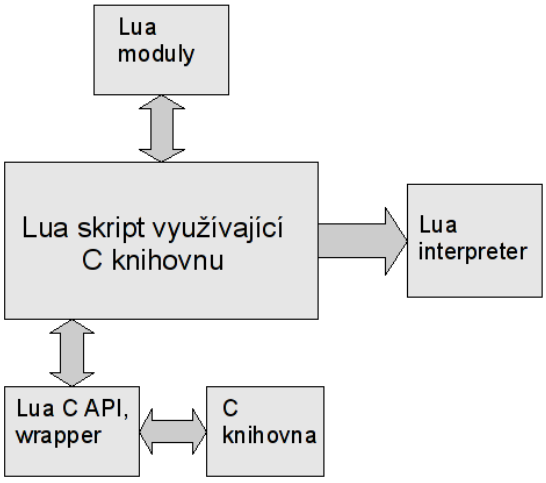
Knihovna OpenCV obsahuje často používané algoritmy pro zpracování obrazu a strojového učení v jazyce C a C++. Proto bylo výhodné zpřístupnit tuto funkcionalitu do skriptovacího netypového jazyka Lua s nízkými systémovými nároky, který byl původně určen pro úlohy zpracování obrazu. Toto přemostění je nazváno LuaCV a implementuje téměř celé C API knihovny OpenCV do jazyka Lua, čímž zpřístupňuje maticové počty, zpracování obrazu, kalibraci kamer, detektory objektů, analýzu obrazu nebo jednoduché grafické rozhraní dostupné pod licencí *LGPLv3*.

Použití knihovny je značně zjednodušeno pomocí internetového instalátoru implementovaného v NSIS frameworku, který stahuje komponenty knihovny z repositáře SourceForge. Volitelná možnost kompilace projektu je pak zajištěna pomocí aplikace CMake, která generuje podpůrné soubory v závislosti na platformě a překladači (*GCC* - Makefile, *Visual Studio* - VS projekt), proto je LuaCV možno jednoduše přeložit na všech platformách, kde funguje OpenCV.

2 IMPLEMENTACE

Přemostění LuaCV je implementováno v jazyce C++ pomocí Lua C API a vytváří modul načítaný interpretrem Lua před vykonáváním skriptu. Na Obrázku 1 můžeme vidět schématické znázornění fungování knihovny LuaCV a její spolupráce s interpretrem Lua a knihovnou OpenCV. Ze schématu je patrné, že LuaCV funguje jako sdílená knihovna dynamicky linkovaná s knihovnou OpenCV a jazykem Lua.

Lua obsahuje pouze základní datové typy (číslo, řetězec, tabulka, uservalue, funkce)[1], proto všechny datové typy (struktury) používané knihovnou OpenCV musí být do Lua exportovány jako netypové odkazy (*void**) s přiřazenou metatabulkou. Tato tabulka tvoří mechanismus dynamického přetypovávání ukazatele na správný typ a dovolí Lua správně pracovat s pamětí, přičemž se využívá callback funkcí metatabulek



Obrázek 1: Schéma fungování LuaCV.

__index a *__newindex*. Proto při exportu objektu musíme explicitně definovat, kterou vlastnost vrátíme na zásobník jazyka Lua, protože pro každou vlastnost objektu bude zavolána tatáž indexovací funkce. Pro účely LuaCV byl navržen mechanismus, jenž ukládá ukazatele na funkce vracející hodnoty proměnných struktury a následně při zavolání indexovací funkce dynamicky vyhledá správnou funkci. Optimalizace je zajištěna pomocí Dijkstrova binárního vyhledávání nad seřazeným polem, tím se časová složitost vyhledávacího algoritmu stala logaritmickou, čímž dochází k výrazné úspoře výpočetního výkonu. Lua využívá inkrementální *garbage collector*, a proto musí mít jednotlivé objekty registrovanou i callback funkci pro uvolnění uživatelských dat. Tato callback funkce se nazývá *__gc* a pokud ji daný objekt neobsahuje, nebude uvolněn z paměti a dojde k tzv. *memory leaku*.

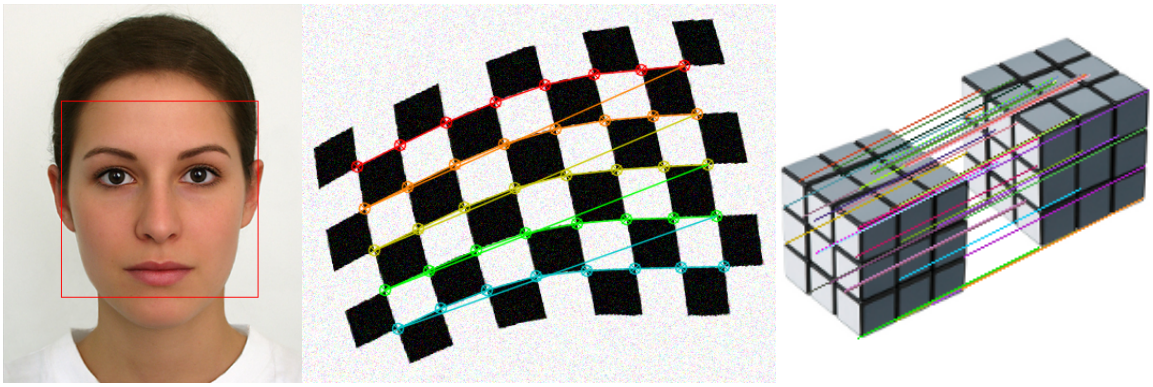
Jednotlivé funkce z OpenCV C API jsou implementovány v LuaCV téměř vždy se shodným funkčním prototypem a mají implementované ochrany typů parametrů (šablonová funkce *checkObject*) a také počet parametrů na zásobníku dané funkce. Tím vzniká první vrstva ochrany před nesprávným použitím funkce a následným pádem programu. Druhá vrstva je implementována v OpenCV a nesprávné použití funkce ve většině případů vyvolá neobslouženou výjimku a tedy ukončení procesu pomocí signálu *SIGABRT*[2]. Knihovna LuaCV zachytává tyto výjimky a vypisuje na obrazovku doplňující informace ze zásobníku interpretru Lua, čímž uživateli udává aktuální pozici programu a název funkce, která chybu vyvolala. Poté se uvolní všechna paměť a program se řádně ukončí. Obsluha výjimek je implementována pomocí tzv. *hooks* neboli háčků. V tomto případě jde o háček, který je spouštěn vždy, když interpretr zavolá libovolnou funkci.

3 POUŽITÍ

Na otestování správné implementace OpenCV funkcí do knihovny LuaCV byly vypracovány téměř všechny oficiální příklady využívající OpenCV C API. Ve většině případů nebylo nutno kód nijak výrazně modifikovat, a proto stačilo pouze přepsat syntax jazyka C do jazyka Lua.

Pro otestování systémových nároků přemostění byly realizovány orientační testy časových náročností mezi nativním programem, přemostěním do jazyka Python a přemostěním do jazyka Lua. Výsledky ukázaly, že implementace LuaCV v jazyku Lua byla průměrně rychlejší o 47% než přemostění pro Python.

Na obrázcích níže můžeme vidět některé z implementovaných aplikací pro přemostění LuaCV. Jde o aplikace detektorů objektů pomocí Haarových kaskád, kalibraci kamery a výpočet optického toku pohybu objektu.

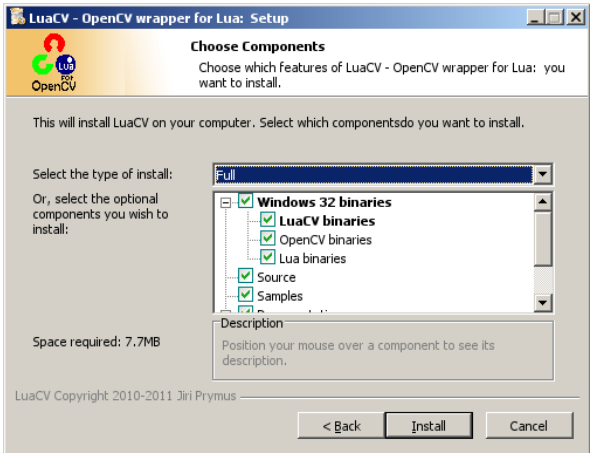


Obrázek 2: Příklady využití knihovny LuaCV.

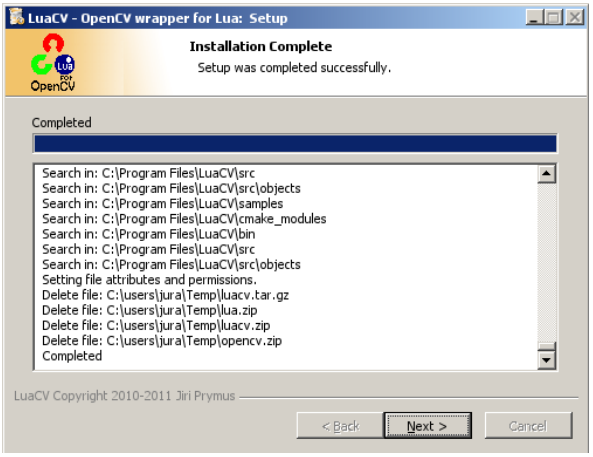
Funkčnost knihovny byla také otestována v kurzu počítačového vidění MPOV na ÚAMT FEKT VUT v Brně, kde ji studenti využili k vypracování zápočtového projektu.

4 INSTALACE

Pro snadnější instalaci na platformě MS Windows byl vyvinut internetový instalátor ve frameworku NSIS, který umožňuje modulární rozložení komponent knihovny a jejich stažení z repositáře SourceForge. Na Obrázku 3 můžeme vidět dialog výběru komponent a na Obrázku 4 dialog průběhu instalace. Tento instalátor umožňuje stáhnout, jak zdrojové kódy knihovny, tak i binární soubory zkompilované v MS Visual Studiu a dokumentaci vygenerovanou pomocí zdrojových kódů LuaCV, OpenCV a doxygen Latexové šablony.



Obrázek 3: Dialog výběru komponent.



Obrázek 4: Dialog výpisu průběhu instalace.

Kompilace knihovny pro vývojáře je značně zjednodušená pomocí aplikace CMake, která generuje MakeFile nebo Projekt v závislosti na detekovaném kompilátoru a platformě. Díky tomu je zajištěna kontrola závislostí přemostění včetně kontroly verzí a vyhledávání cest k binárním souborům, ke kterým se bude LuaCV linkovat.

5 ZÁVĚR

Knihovna LuaCV v sobě implementuje téměř kompletní funkcionalitu OpenCV knihovny (cca. 600 funkcí a 100 objektů), mezi zbývajícími algoritmy jsou C++ objekty a na jejich implementaci se dále pracuje. Přemostění LuaCV si klade za cíl obsáhnout celou knihovnu OpenCV.

Díky přepracovaným oficiálním příkladům OpenCV knihovny je možno porovnat styl zápisu, funkčnost a rychlost implementace v jazyku Lua. Implementovány byly téměř všechny příklady, které využívají C API knihovny.

LuaCV se ukázala být v praxi použitelná, jak ukázalo její využití desítkami studentů kurzu počítačového vidění MPOV. Ambicemi LuaCV je také být konkurence schopná s prostředím Matlab a jeho *Image processing toolboxem*.

Knihovna LuaCV je psána pod svobodnou licencí LGPLv3, a proto je licenčně kompatibilní s open source aplikacemi. Dalším vývojem knihovny bude snaha o začlenění do oficiální distribuce OpenCV knihovny, jakožto přemostění pro jazyk Lua. Přemostění LuaCV je volně dostupné z repositáře SourceForge.

REFERENCE

[1] Ierusamlischy, R., De Figueiredo, L. H., Celes, W.: *Lua 5.1 Reference Manual*. Lua.org, 2006, 112 s., ISBN 85-903798-3-3

[2] Bradski, G., Kaehler, A.: *Learning OpenCV*, Sebastopol, O'Reilly Media, 2008, 543 s., ISBN 978-0-596-51613-0