

OpenCV

Andrej Lúčny

Katedra aplikovanej informatiky FMFI UK

lucny@fmph.uniba.sk

http://dai.fmph.uniba.sk/w/Andrej_Lucny

www.agentspace.org/opencv

Kompilácia OpenCV

- Skompilovanie knižnice priamo zo zdrojových kódov a 3rd party knižníc
- Prvý krát: práca na týždeň
- Ďalšie razy: práca na deň

Binárky OpenCV

- Binárky pre C++ sú dostupné bez contrib vo forme world dll
- Pre Python je možné stiahnuť aj contrib
- Ani v jednom prípade nemáme podporu CUDA

Dôvody kompilácie OpenCV

- Podpora CUDA
- Podpora HDF5
- Contrib pre C++
(contrib obsahuje veľa dobrých vecí)

Nástroje kompilácie

Windows

- Cmake
- Visual C++ 2017
- Python 2.7

Linux

- cmake a cmake-gui
- gcc
- Python 2.7

Java

- JDK
- Apache Ant

Python 3

- Python 3.5-3.7

Nástrahy

- OpenCV bežne obsahuje chyby a treba si s nimi vedieť poradiť

3rd party

- CUDA, CuDNN, OpenCL
- HDF5
- Intel SW Tools (MKL, TBB, IPP)
- Eigen
- Ceres, Glog, Gflags (pre opencv sfm)
- FFMPEG alebo GSTREAMER
- (VTK, Qt, Tesseract)

Inštalačný skript

- <https://github.com/spmallick/learnopencv/tree/master/InstallScripts>
- problém: vyskytujú sa chyby
- výhoda: skript poslúži aj v prípade zhavarovania: ako inštalačný návod
- Nevýhoda: chýba CUDA support
- Riešenie: každý má vlastný postup

Postup kompilácie

- Z `opencv.org` stiahneme zdrojáky
- Zdrojáky `opencv` sú priamo na stránke `opencv.org`, ale `opencv_contrib` sa hľadá komplikovane, je na https://github.com/opencv/opencv_contrib/releases
- Zdrojáky rozbalíme do
 - `I:\opencv450\opencv\`
 - `I:\opencv450\opencv_contrib\`

Postup kompilácie

- Spustíme Cmake / cmake-gui
- Zadáme cesty
 - Source: I:\opencv450\opencv\
Build I:\opencv450\build\
Build I:\opencv450\build\
- Configure
 - Vyberieme platform: native, 64 bit, VC15

Postup kompilácie

- **SHARED** – má byť výsledkom statická knižnica alebo dll / so ? Dll sú lepšie. Na následné bindings pre Java a Python však normálne potrebujeme statické knižnice, aj keď pre Python možné použiť **FORCE**, pyd potom dynamicky loaduje dll (ON)
- **world** alebo nie - má byť výsledkom jeden .lib alebo veľa? (OFF)

Postup kompilácie

- Aby sme kompilovali aj contrib:
OPENCV_EXTRA_MODULES_PATH
../../opencv_contrib/modules
- Configure

Postup kompilácie

- Unit testy a Performance testy
- **TEST** a **PERF** – kompilácia trvá hodiny aj bez nich, takže ich dávame OFF

Postup kompilácie

- Podpora videa:
 - Volíme iba jedno z nich
 - **WITH_GSTREAMER OFF**
 - **WITH_FFmpeg ON**
- Skôr sa používa ffmpeg ale napr. na edge zariadeniach sa zide gstreamer, ktorý umožňuje zdefinovať spracovateľskú pipeline

Postup kompilácie

- CUDA support:
 - **WITH_CUDA ON**
 - **OPENCV_DNN_CUDA ON**
- Configure, potom:
 - **CUDA_FAST_MATH ON**
 - **ENABLE_FAST_MATH ON**
 - **CUDA_ARCH_BIN 5.2 5.3 6.0 6.1 7.0 7.5**
 - **CUDA_ARCH_PTX 7.5**

Bežný GPU target v našich končinách je 5.2, 5.3

5.1 už nestačí pre DNN

Postup kompilácie

- Podpora HDF5 (potrebné pre Keras .h5)
 - **BUILD_opencv_hdf** ON
 - **HDF5_C_LIBRARY=**
C:/Program Files/HDF_Group/HDF5/1.12.0/lib/hdf5.lib
 - **HDF5_INCLUDE_DIRS=**
C:/Program Files/HDF_Group/HDF5/1.12.0/include
- Pozor: pri importe takéhoto OpenCV v Pythone, treba najprv importovať keras a až potom cv2

Postup kompilácie

- Nie je nevyhnutné, ale osoží rýchlosti:
BLAS, CBLAS & LAPACK
(Intel implemenoval ako MKL)
 - **MKL ON**
 - **MKL_INCLUDE_DIRS**=c:/Program Files
(x86)/IntelSWTools/compilers_and_libraries_2020/windows/mkl/include
 - **MKL_ROOT_DIR**=c:/Program Files
(x86)/IntelSWTools/compilers_and_libraries_2020/windows/mkl

- OpenCV si samo stiahne IPP od Intelu
- TBB: nie je nevyhnutné a spôsobuje závislosť na tbb.dll, ale umožňuje programovať v OpenCV paralelne (cez foreach() a TBB, nielen cez OMP):
 - **WITH_TBB ON, BUILD_TB OFF**
 - Configure
 - **TBB_DIR**=c:/Program Files (x86)/IntelSWTools/compilers_and_libraries_2020/windows/tbb
 - **TBB_ENV_INCLUDE**=c:/Program Files (x86)/IntelSWTools/compilers_and_libraries_2020/windows/tbb/include
 - **TBB_ENV_LIB**=c:/Program Files (x86)/IntelSWTools/compilers_and_libraries_2020/windows/tbb/lib/intel64/vc14/tbb.lib
 - **TBB_ENV_LIB_DEBUG**=c:/Program Files (x86)/IntelSWTools/compilers_and_libraries_2020/windows/tbb/lib/intel64/vc14/tbb_debug.lib

Postup kompilácie

- Eigen
 - **WITH_EIGEN ON**
 - **EIGEN_INCLUDE_PATH=**
c:/Eigen3/include/eigen3
 - **Eigen3_DIR=**
c:/Eigen3/share/eigen3/cmake

Python

- OpenCV potrebuje Python 2.7 pre spúšťanie kompilačných skriptov
- OpenCV kompilujeme pre verziu 3.x
- Musíme mať nainštalované aj 2.7 aj 3.x

- **OPENCV_FORCE_PYTHON_LIBS ON**
- **OPENCV_PYTHON3_VERSION ON**
- **PYTHON3_EXECUTABLE=**
C:/Python37/python.exe
- **PYTHON3_INCLUDE_DIR=**
C:/Python37/include
- **PYTHON3_LIBRARY=**
C:/Python37/libs/python37.lib
- **PYTHON3_NUMPY_INCLUDE_DIRS=**
C:/Python37/Lib/site-packages/numpy/core/include
- **PYTHON3_PACKAGES_PATH=**
c:/Python37/Lib/site-packages

- V súčasných verziách je v cmake fileoch OpenCV v prípade Pythonu chyba, kvôli ktorej Cmake pochopí **OPENCV_PYTHON3_VERSION** ako ON/OFF, očakáva tam ale napr. 3.7
- Fix: v OpenCVDetectPython.cmake na riadku 280 treba priamo uviesť verziu
- `find_python("3.7" "${MIN_VER_PYTHON3}"
PYTHON3_LIBRARY PYTHON3_INCLUDE_DIR`
- Configure

(Po kompilácii)

- Výsledkom kompilácie sa v `c:/Python37/Lib/site-packages` objaví adresár `cv2` s `pyd` `opencv`
- Do `c:\Python37\Lib\site-packages\cv2\python-3.7\` treba prikopírovať dllky `opencv_videoio_ffmpeg450_64.dll`, `hdf5.dll` a `tbb.dll`
- Do `config.py` v ňom treba pridať cesty k dllkam `opencv`, `CUDA`, `CuDNN`

takže vyzerá napr. takto:

```
import os
```

```
BINARIES_PATHS = [  
    os.path.join('c:/opencv450', 'x64/vc15/bin'),  
    os.path.join(os.getenv('CUDA_PATH', 'C:/Program  
Files/NVIDIA GPU Computing Toolkit/CUDA/v10.2'), 'bin')  
] + BINARIES_PATHS
```


Postup kompilácie

- **JAVA**
 - **BUILD_JAVA ON**
- Teraz to už ide aj pri **SHARED**
- Ale musí byť nainštalovaný nielen **JDK** (napr 1.8.0) ale aj **Ant**

- Pre sfm (structure from movement) treba ručne nastavit'
 - **BUILD_opencv_sfm** ON
- **GLOG**
 - **GLOG_INCLUDE_DIR** = c:/glog/include
 - **Glog_LIBS** = c:/glog/lib/glog.lib
- **CERES**
 - **CERES_DIR** = c:/ceres/CMake
- **GFLAGS**
 - **GFLAGS_DIR** = C:/gflags/lib/cmake/gflags
- Configure. „Checking SFM deps... TRUE“

Postup kompilácie

- warning o setupvars scripts ignorujeme
- Inak venuje pozornosť každej chybe a opravujeme ako sa dá
- Generate
- Otvoríme .sln
- zvolíme Release
- Build

Postup kompilácie

- Opravíme chyby. Momentálne je len jedna: v projekte multiview chýba include `c:\ceres\include`, znovu Build
- Build project only na INSTALL
- V build/install máme výsledok pre Release

- Podobne môžeme skompilovať Debug Opravíme chyby. Momentálne sú dve:
 - multiview treba doplnit include pre ceres
c:\ceres\include
 - v opencv_sfm treba zmeniť ceres, glog a gflags
knížnice na debug verzie:
 - C:\ceres\lib\ceres-debug.lib
 - C:\glog\lib\glogd.lib
 - C:\gflags\lib\gflags_staticd.lib
- Build project only na INSTALL
- V build/install máme výsledok pre Debug

Použitie

- V pythone: `import cv2`
- V C++ spravime empty projekt
 - s include `c:\opencv450\include\`
 - s input: všetky cesty k libs v `c:\opencv450\x64\vc15\lib\`
 - s environment
`PATH=c:\opencv450\x64\vc15\bin\`

```
cmake_minimum_required(VERSION 2.8)
```

```
set( NAME_PROJECT  
      colorizeImage  
    )
```

```
set( NAME_SRC  
      ${NAME_PROJECT}.cpp  
    )
```

```
set( NAME_HEADERS  
    )
```

```
project ( ${NAME_PROJECT} )  
find_package( OpenCV REQUIRED )  
include_directories( ${OpenCV_INCLUDE_DIRS}  
${CMAKE_CURRENT_SOURCE_DIR} )  
add_executable( ${NAME_PROJECT} ${NAME_SRC} ${NAME_HEADERS} )  
target_link_libraries( ${NAME_PROJECT} ${OpenCV_LIBS} )
```