Matlab Code Generator for OpenCV

Google Summer of Code 2013 Proposal

Hilton Bristow

1 Summary

The computer vision community is largely divided between those who code in Matlab and those who code in C++ using the OpenCV library. This division has added a barrier to easy, transparent sharing of code amongst researchers. My project aims to address this issue, by adding a set of auto-generated Matlab bindings and tools to OpenCV so that Matlab users can easily access the core functionality of OpenCV as well as applications written by researchers more familiar with C++.

2 Introduction

The ability to verify the claims of other authors is critical to creating a respectable, progressive scientific community. The idea of *reproducible research* is that the ultimate product of research should be a paper along with the full computational environment used to produce the results in that paper. A significant component of this environment is the *code*.

Within the computer vision community, there exist two dominant programming languages: Matlab[1] and C++. The appeal of the former is its reduced syntax and lack of peripheral requirements such as compilation. This is a boon for researchers who are not intimate with programming languages. Matlab comes prepackaged with a large library of functionality (called "toolboxes") dedicated to image manipulation and the absence of compilation means running someone else's code is normally as easy as adding the folder to the PATH and hitting run.

C++, on the other hand, is lightning fast in comparison to Matlab. For computationally intensive applications – and many computer vision applications certainly fall into this category – this can mean the difference between running a detector at 30fps rather than 1fps, or waiting to learn a classifier for a day rather than a week. The defacto standard computer vision library for C++ is OpenCV (`http://opencv.org`). The library has powerful matrix expressions and functionality similar to Matlab.

But herein lies the problem. The presence of these two languages has added a barrier to the sharing of code within the community. Matlab's dynamic typing and interpreter make it excellent for prototyping, while C++ is faster and safer for large applications.

3 The Project

This project aims to bring together the best of Matlab and C++ for computer vision researchers, by providing a service for auto-generating Matlab mex[2] files from C++ code using OpenCV. The main functionality this will provide will be (i) to make all of OpenCV's base functionality available natively in Matlab, (ii) to allow Matlab users to use OpenCV expressions in their mex files, and (iii) for researchers who use C++ to easily provide Matlab hooks so users of Matlab can run their code too.

This project contains 5 principal components, each detailed below.

[1] Matlab is a trademark of The MathWorks, Inc.

[2] mex files are dynamic C++ libraries with Matlab hooks.

3.1 cmake

cmake is the default build system for OpenCV. Adding Matlab bindings first requires compile support and detection of the Matlab installation. KitWare distributes a version of `FindMatlab.cmake` but it is sorely out of date and contains a lot of hard coded parameters. This part of the project involves rewriting that module to be more cross platform and robust to version changes.

3.2 Parsing OpenCV Declarations

OpenCV includes a C++ header parser in the `python` module to parse C++ declarations, including namespaces, classes, functions, constants and enums. So the hard work here has been done. However, the output of the parser in its raw form is not very amenable to template population. This part of the project involves writing a `Refactorer` python class to take the output of the parser and refactor it into a semantic tree.

3.3 Type Conversions

The major component of this project requires translating OpenCV types to Matlab types and visa versa. This will be handled transparently by a C++ `Bridge` class providing either explicit or implicit type conversions. The appeal of a bridging class is that it is easy to extend when new types are used (either by the user, OpenCV or new standards).

Some care must be taken in passing C++ class instances back to Matlab due to the way memory is handled. Fortunately the Matlab community has discussed this in depth and has devised a set of best practices.

3.4 Populating Templates

Matlab mex gateway functions have a characteristic form that is well suited to template population. In this project I will use Jinja (http://jinja.pocoo.org/), a python template engine derived from the django web framework. Jinja has powerful filters and implements a model-view-controller paradigm to separate logic and syntax. Coupled with the refactored parse tree and type converter, this will produce templates for documentation, standalone functions and entire classes that are highly maintainable and modular.

3.5 Ancillaries

With the code generator framework in place, this will enable me to easily write functionality to:

- generate wrappers for all of OpenCV,
- write a custom mex "compiler" that can take an arbitrary C++ file containing OpenCV definitions, and automagically create a mex gateway, and
- encapsulate the entire public API of a C++ library (say, for face tracking) so that its methods can be called from within Matlab

One fringe component I will also write is support for reading and writing Matlab's `.mat` file format version 7 and 7.3 to `cv::FileStorage`. I have already written some of the code as part of a personal project.

4 Incumbents

The excellent mexopencv toolbox (https://github.com/kyamagu/mexopencv) already provides a large amount of the functionality described in this project. This, however, introduces a number of difficulties for the maintainers of OpenCV. Specifically:

- The wrappers are written by a 3rd party, which divides the experience for users. Matlab users are often less familiar with the intricacies of dependencies, and would prefer a solution that *Just Works*^{TM}. A fully integrated Matlab generator means 1 download, 1 bug tracker, 1 maintaining body and conveys the image of core functionality.

- The wrappers are hand-written, so any changes to the opencv C++ API will take time to propagate to the mexopencv bindings.
- Auto-generated wrappers reflect *exactly* the functionality available dictated by the OpenCV version, compiled modules, extras, etc. API changes or optional features will not influence or break old versions of the generator.
- mexopencv does not come with some of the utilities we would like to offer.

5 Additional Info

Project discussion: http://answers.opencv.org/question/9487/matlab-api-for-opencv/