Using Python in Computer Vision: Performance and Usability

B. Thorne

University of Canterbury

Undergraduate Computer Vision Presentations, 2009

Outline

Outline

Why Not Use C or C++?

- C is low level and verbose
- C needs to be compiled and linked
- Static, unsafe, or nominative typing discipline
- having to deal with memory allocation when memory really isn't a critical concern
- pointers, hash defines, macros etc...
- when all you want to do is get something working
- ...quickly

Why Not Use C or C++?

- C is low level and verbose
- C needs to be compiled and linked
- Static, unsafe, or nominative typing discipline
- having to deal with memory allocation when memory really isn't a critical concern
- pointers, hash defines, macros etc...
- when all you want to do is get something working
- ...quickly

Why Not Use C or C++?

- C is low level and verbose
- C needs to be compiled and linked
- Static, unsafe, or nominative typing discipline
- having to deal with memory allocation when memory really isn't a critical concern
- pointers, hash defines, macros etc...
- when all you want to do is get something working
- ...quickly

Why Not Use C or C++?

- C is low level and verbose
- C needs to be compiled and linked
- Static, unsafe, or nominative typing discipline
- having to deal with memory allocation when memory really isn't a critical concern
- pointers, hash defines, macros etc...
- when all you want to do is get something working
- ...quickly

Why Not Use C or C++?

- C is low level and verbose
- C needs to be compiled and linked
- Static, unsafe, or nominative typing discipline
- having to deal with memory allocation when memory really isn't a critical concern
- pointers, hash defines, macros etc...
- when all you want to do is get something working
- ...quickly

Why Not Use C or C++?

- C is low level and verbose
- C needs to be compiled and linked
- Static, unsafe, or nominative typing discipline
- having to deal with memory allocation when memory really isn't a critical concern
- pointers, hash defines, macros etc...
- when all you want to do is get something working
- ...quickly

Why Not Use C or C++?

- C is low level and verbose
- C needs to be compiled and linked
- Static, unsafe, or nominative typing discipline
- having to deal with memory allocation when memory really isn't a critical concern
- pointers, hash defines, macros etc...
- when all you want to do is get something working
- ...quickly

So Why Python?

- Python is a high level language
- it is interactive
- it has a clear syntax
- comes with extensive libraries
- extensible with many more
- can call other languages
- multi paradigm
- dynamic OR strongly typed

So Why Python?

- Python is a high level language
- it is interactive
- it has a clear syntax
- comes with extensive libraries
- extensible with many more
- can call other languages
- multi paradigm
- dynamic OR strongly typed

So Why Python?

- Python is a high level language
- it is interactive
- it has a clear syntax
- comes with extensive libraries
- extensible with many more
- can call other languages
- multi paradigm
- dynamic OR strongly typed

So Why Python?

- Python is a high level language
- it is interactive
- it has a clear syntax
- comes with extensive libraries
- extensible with many more
- can call other languages
- multi paradigm
- dynamic OR strongly typed

So Why Python?

- Python is a high level language
- it is interactive
- it has a clear syntax
- comes with extensive libraries
- extensible with many more
- can call other languages
- multi paradigm
- dynamic OR strongly typed

So Why Python?

- Python is a high level language
- it is interactive
- it has a clear syntax
- comes with extensive libraries
- extensible with many more
- can call other languages
- multi paradigm
- dynamic OR strongly typed

So Why Python?

- Python is a high level language
- it is interactive
- it has a clear syntax
- comes with extensive libraries
- extensible with many more
- can call other languages
- multi paradigm
- dynamic OR strongly typed

So Why Python?

- Python is a high level language
- it is interactive
- it has a clear syntax
- comes with extensive libraries
- extensible with many more
- can call other languages
- multi paradigm
- dynamic OR strongly typed

Outline

Python In Computer Vision.

OpenCV

- Provides well tested, optimized and open source code for image processing and computer vision
- Written in C, ensuring both fast and portable.
- Has multiple language wrappers including for Python

NumPy/SciPy

- Gives strongly typed N-dimensional arrays to Python
- Well used and tested libraries for scientific computing
- implemented in C and FORTRAN

Outline

Acquiring & Display Of An Image

- live image acquisition is such a crucial role in the majority of CV applications.
- tested getting and showing a frame as a most basic, but necessary test
- also serves to compare the syntax of the languages

Acquiring & Display Of An Image

- live image acquisition is such a crucial role in the majority of CV applications.
- tested getting and showing a frame as a most basic, but necessary test
- also serves to compare the syntax of the languages

Acquiring & Display Of An Image

- live image acquisition is such a crucial role in the majority of CV applications.
- tested getting and showing a frame as a most basic, but necessary test
- also serves to compare the syntax of the languages

Syntax of C and Python.

Example

```
#include "cv.h"
#include "highgui.h"
int main(){
IplImage *frame;
CvCapture *capture;
capture = cvCreateCameraCapture(0);
cvNamedWindow( "Snapshot", 0 );
frame = cvQueryFrame( capture );
cvShowImage( "Snapshot", frame );
}
```

Syntax of C and Python.

Example

```
from opencv import highgui as hg
capture = hg.cvCreateCameraCapture(0)
hg.cvNamedWindow("Snapshot")
frame = hg.cvQueryFrame(capture)
hg.cvShowImage("Snapshot", frame)
```

Image Capture Performance.

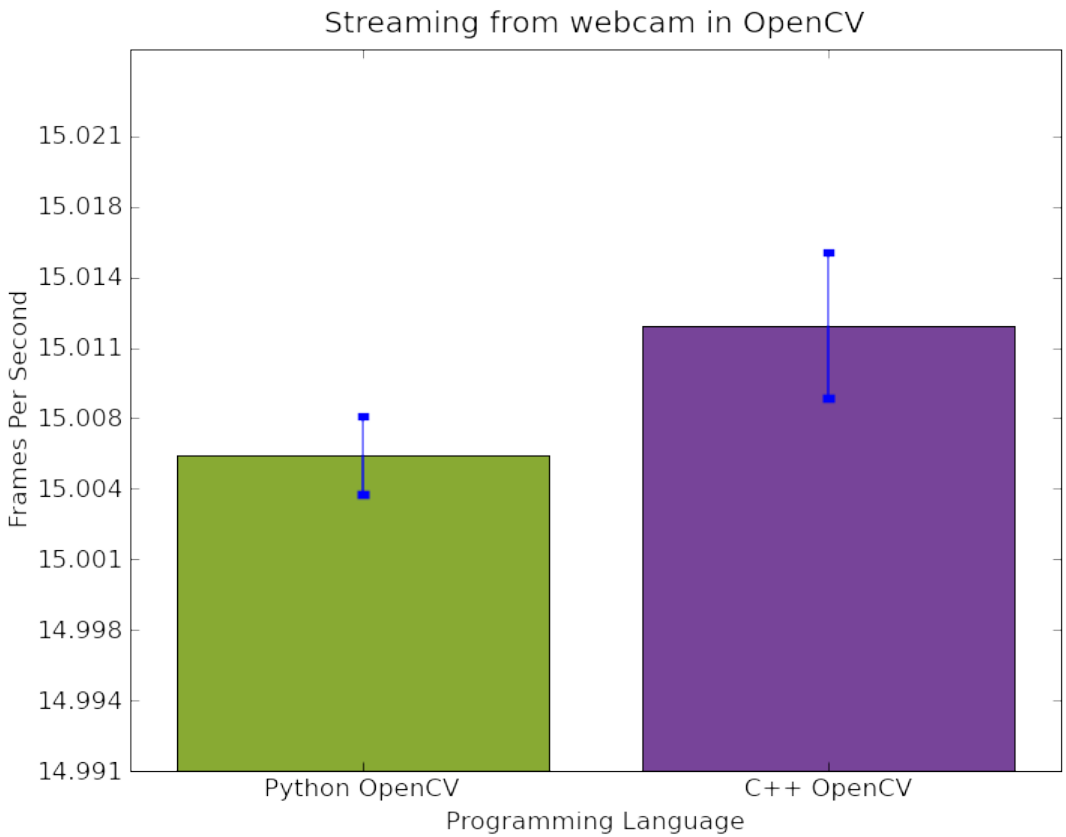

Figure: Comparison of Python and C++ performance using OpenCV for webcam capture and display.

Outline

Output Images from Gaussian Blur Examples

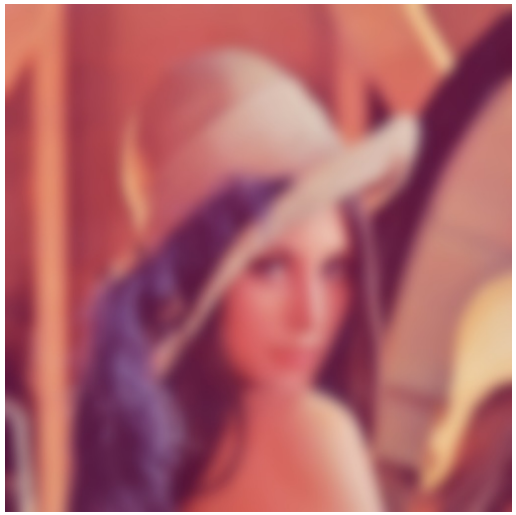

Figure: C++

Output Images from Gaussian Blur Examples

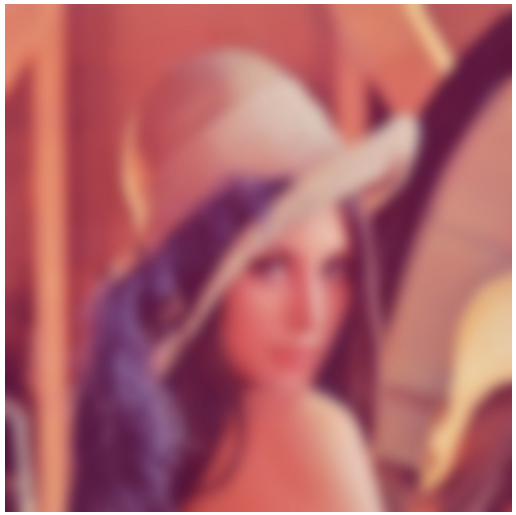

Figure: Python OpenCV

Output Images from Gaussian Blur Examples

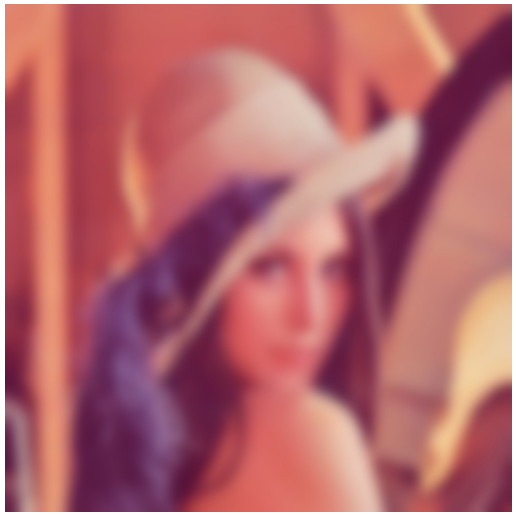

Figure: Python SciPy

Differances In Output Images from Gaussian Blur Examples

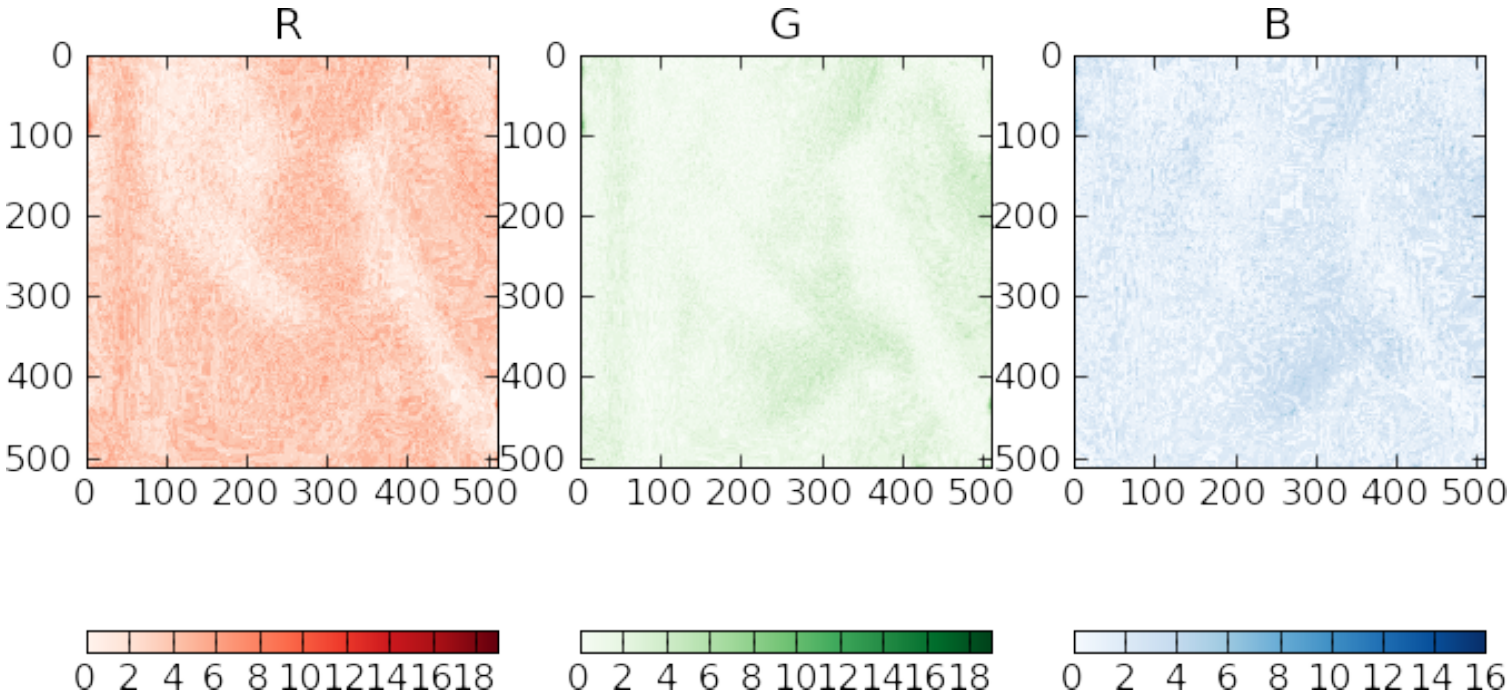

Figure: Difference of each channel after gaussian blur in OpenCV and SciPy.

Note the full scale is 255, Pixel for pixel difference image stats: maximum intensity difference = 20, avg diff = 2.1, std dev = 1.4

Performance Of Gaussian Blur

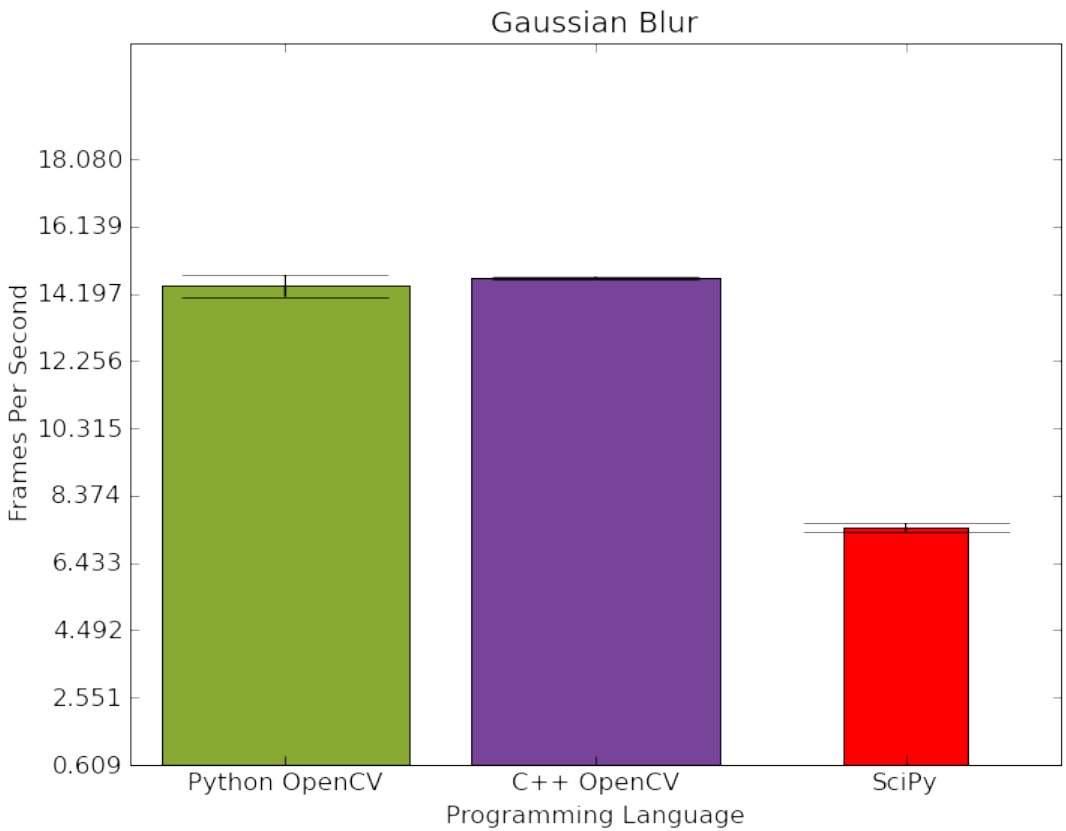

Figure: Performance carrying out Gaussian Blur

Outline

Background Subtraction

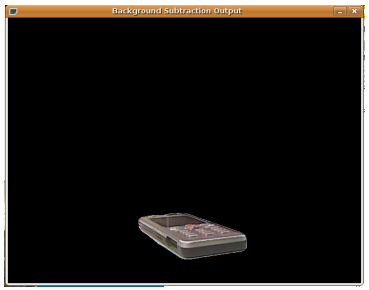

Figure: Adding an Item

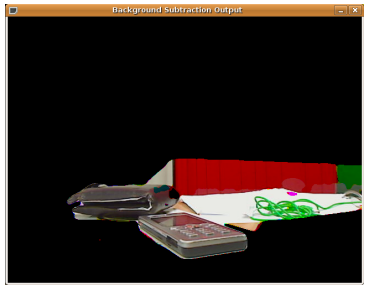

Figure: Addition and Removal of Items

Background Subtraction Performance

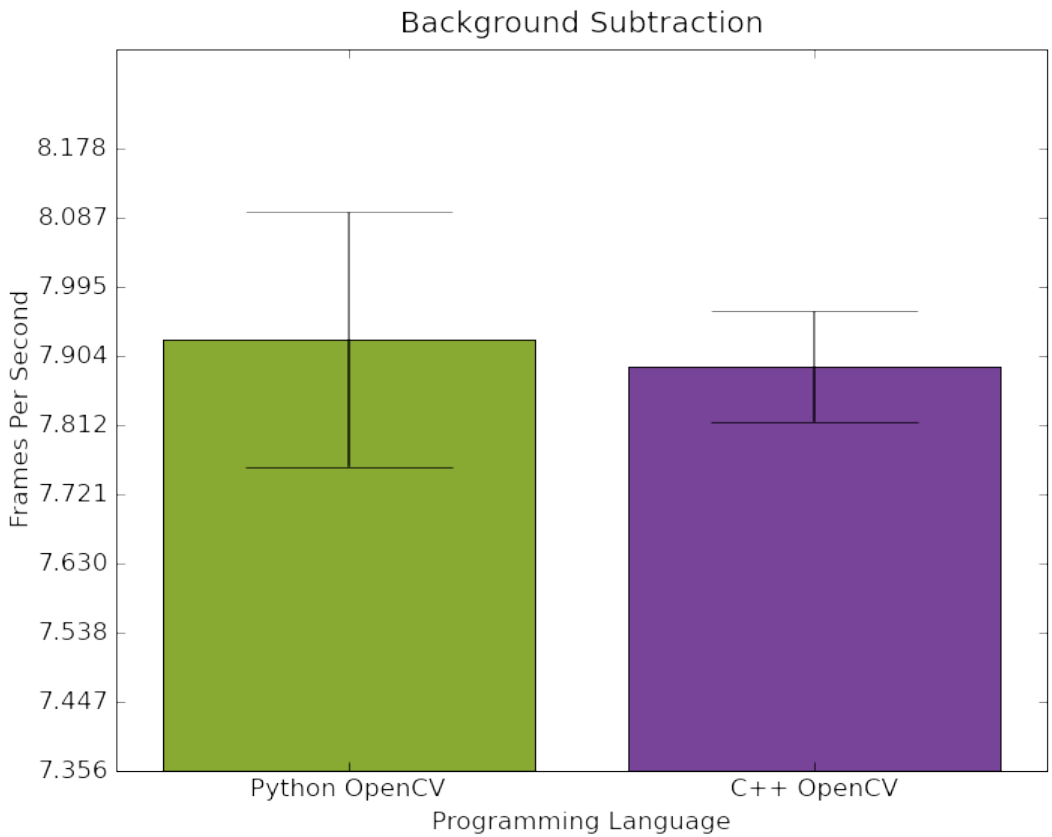

Figure: Background subtraction performance

Feature Point Detection

- Feature point detection was implemented in SciPy and OpenCV
- OpenCV had a built in function so is being called at a much much lower level
- OpenCV from C or Python was ~3 times faster than SciPy when doing Harris & Stephens feature detection

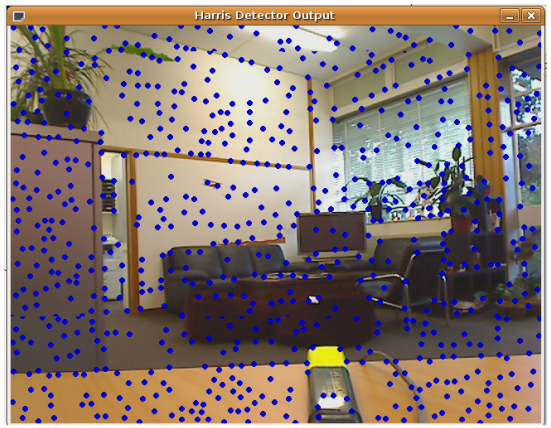

Figure: SciPy Feature Detection

Outline

IPython & MatPlotLib

- Using IPython, an interactive shell can be used from deep inside a nested loop in a running program. In place timing and plotting with access to local variables and functions... this speeds up development/debugging time.

Example

```
In [1]: from opencv import cv
In [2]: cv.cvAnd(diffImage,image, temp)
In [3]: timeit cv.cvAnd(diffImage,image, temp)
1000 loops, best of 3: 229 µs per loop
In [4]: from pylab import imshow, show
In [5]: imshow(temp)
Out[5]: <AxesImage object at 0x42489d0>
In [6]: show()
```

Outline

Meaningful Quantifiable Statements

- shorter than C versions
- easier to create
- easier to debug with the use of an interactive interpreter
- easier to test

Documentation & Support

The documentation in both SciPy and OpenCV was found to be complete.

Remember Python is Free

Documentation is not as extensive as for a professional package like Matlab.

Support for these open source packages is almost entirely reliant on experienced members of the community responding to requests on message boards or mailing lists.

Summary

- Python isn't actually slowing things down significantly when calling OpenCV
- Python is much more forgiving than C or C++, it can be used interactively so you can happily make 10 mistakes in a row, without having to recompile and start execution again for each attempted fix.
- For the scholar of computer vision research using Python can help in trying out new algorithms very quickly. The breadth of the additional libraries available and the ease of integrating, make new and novel solutions quickly realizable.

Limitations

- Limitations/Future Work
  - Automated suite of timed tests for benchmarking
  - A major limitation of using Python would be when the application is being developed for special embedded hardware or when the best possible performance is required at YOUR expense.

For Further Reading I


G. Bradski, A. Kaehler
Learning OpenCV.
O'Reilly Media, September 2008.


T. Oliphant
Guide to NumPy.
UT, Trelgol Publishing, 2006.