Embedded computing for scientific and industrial imaging applications

Lecture 11 - BLAS and LAPACK

Algorithm

Software

Architecture

SOC Hardware

- **Algorithm**
- Software
- Architecture
- SOC Hardware

Correlation Filters with Limited Boundaries

Hamed Kiani Galoogahi
Istituto Italiano di Tecnologia
Genova, Italy
hamed.kiani@iit.it

Terence Sim
National University of Singapore
Singapore
tsim@comp.nus.edu.sg

Simon Lucey
Carnegie Mellon University
Pittsburgh, USA
slucey@cs.cmu.edu

Abstract

Correlation filters take advantage of specific properties in the Fourier domain allowing them to be estimated efficiently: $\mathcal{O}(ND \log D)$ in the frequency domain, versus $\mathcal{O}(D^3 + ND^2)$ spatially where D is signal length, and N is the number of signals. Recent extensions to correlation filters, such as MOSSE, have reignited interest of their use in the vision community due to their robustness and attractive computational properties. In this paper we demonstrate, however, that this computational efficiency comes at a cost. Specifically we demonstrate that only $\frac{1}{D}$ proportion of shifted examples are unaffected by boundary effects which has a dramatic effect on detection/tracking performance. In this paper, we propose a novel approach to correlation filter estimation that: (i) takes advantage of inherent computational redundancies in the frequency domain, (ii) dramatically reduces boundary effects, and (iii) is able to implicitly exploit all possible patches densely extracted from training examples during learning process. Impressive object tracking and detection results are presented in terms of both accuracy and computational efficiency.

$$Ax = b$$

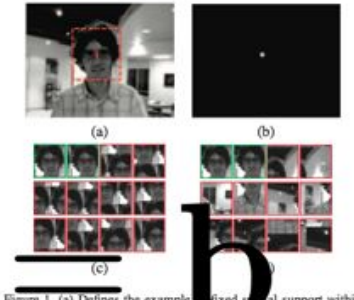


Figure 1. (a) Defines the example fixed spatial support within the image from which the peak correlation output should occur. (b) The desired output response, based on (a), of the correlation filter when applied to the entire image. (c) A subset of patch examples used in a canonical correlation filter where green denotes a non-zero correlation output, and red denotes a zero correlation output in direct accordance with (b). (d) A subset of patch examples used in our proposed correlation filter. Note that our proposed approach uses all possible patches stemming from different parts of the image, whereas the canonical correlation filter simply employs circular shifted versions of the same single patch. The central dilemma in this paper is how to perform (d) efficiently in the Fourier domain. The two last patches of (d) show that $\frac{D-1}{T}$ patches near the image border are affected by circular shift in our method which can be greatly diminished by choosing $D << T$, where D and T indicate the length of the vectorized face patch in (a) and the whole image in (a), respectively.

1. Introduction

Correlation between two signals is a standard approach to feature detection/matching. Correlation touches nearly every facet of computer vision from pattern detection to object tracking. Correlation is rarely performed naively in the spatial domain. Instead, the fast Fourier transform (FFT) affords the efficient application of correlating a desired template/filter with a signal.

Correlation filters, developed initially in the seminal work of Hester and Casasent [15], are a method for learning a template/filter in the frequency domain that rose to some prominence in the 80s and 90s. Although many variants have been proposed [15, 18, 20, 19], the approach's central tenet is to learn a filter, that when correlated with a set of training signals, gives a desired response, e.g. Figure 1 (b). Like correlation, one of the central advantages of the approach is that it attempts to learn the filter in the frequency domain due to the efficiency of correlation in that domain.

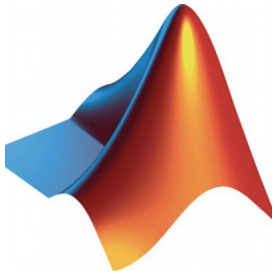
Interest in correlation filters has been reignited in the vision world through the recent work of Bolme et al. [5] on Minimum Output Sum of Squared Error (MOSSE) correlation filters for object detection and tracking. Bolme et al.'s work was able to circumvent some of the classical problems

Algorithm

Software

Architecture

SOC Hardwar



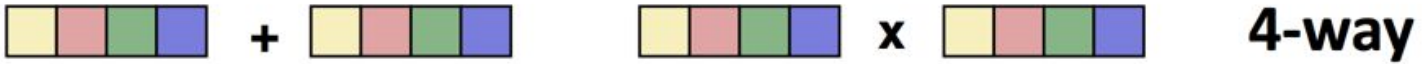

```
// 5. Now apply some OpenCV operations
cv::Mat gray; cv::cvtColor(cvImage, gray,
    CV_RGBA2GRAY); // Convert to grayscale
cv::GaussianBlur(gray, gray, cv::Size(5,5), 1.2, 1.2); //
    Apply Gaussian blur
cv::Mat edges; cv::Canny(gray, edges, 0, 50); // Estimate
    edge map using Canny edge detector
```

Algorithm

Software

Architecture

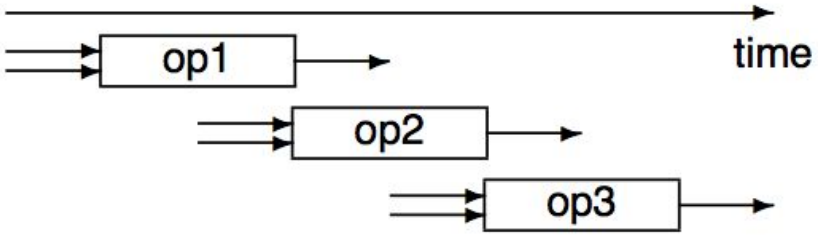
SOC Hardware



SIMD (Single Instruction, Multiple Data)

Ideal Von Neumann Processor

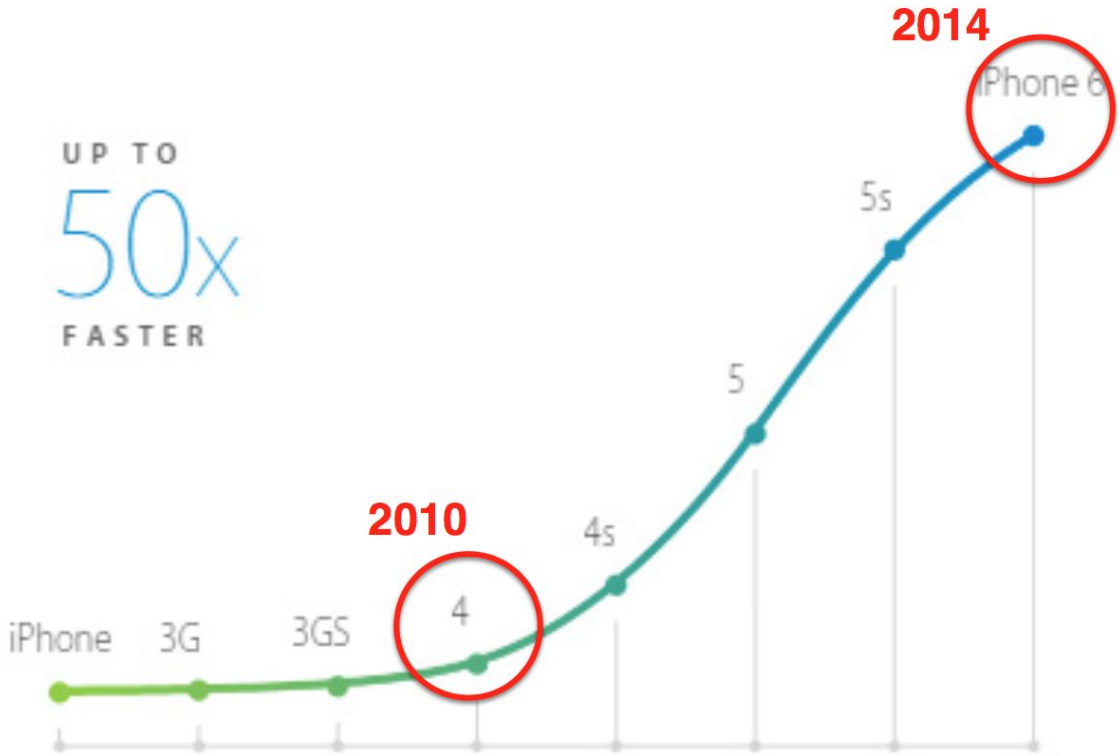
- Each cycle, CPU takes data from registers, does an operation, and puts the result back
- Load/store operations (memory ↔ registers) also take one cycle
- CPU can do different operations each cycle output of one operation can be input to next



CPU clock is stuck!!!!

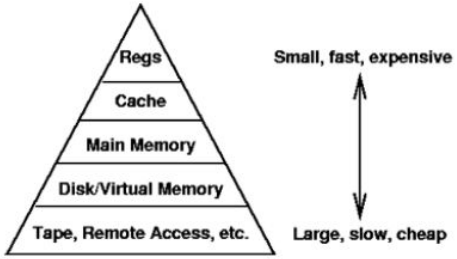
- CPU clock stuck at about 3GHz since 2006 due to high power consumption (up to 130W per chip)
- chip circuitry still doubling every 18-24 months
- ⇒ more on-chip memory and MMU (memory management units)
- ⇒ specialised hardware (e.g. multimedia, encryption)
  ⇒ multi-core (multiple CPU's on one chip)
- peak performance of chip still doubling every 18-24 months

CPU performance



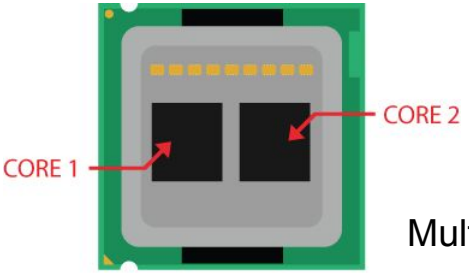
Architecture Considerations

- Memory hierarchy



- Vector instructions

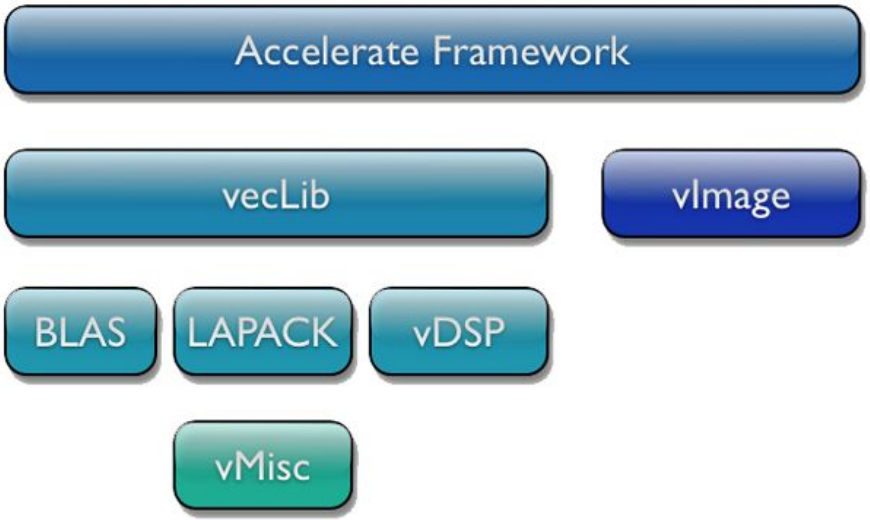
- Multiple threads



Multi-core CPU

Accelerate Framework (iOS, OS X)

- Fast with low energy usage
- Works on both OS X and iOS
- Image processing
- Digital signal processing
- **Linear algebra (LAPACK, BLAS)**



BLAS

- Basic Linear Algebra
  - Level 1 (70s) $\mathbf{y} = \alpha\mathbf{x} + \mathbf{y}$
  - Level 2 (mid 80s) $\mathbf{y} = \alpha\mathbf{A}\mathbf{x} + \beta\mathbf{y}$
  - Level 3 (late 80s) $\mathbf{C} = \alpha\mathbf{A}\mathbf{B} + \beta\mathbf{C}$

- BLAS was originally used to implement the linear algebra subroutine library (LINPACK)

The Path to LAPACK

- EISPACK and LINPACK (early 70s)
  - Libraries for linear algebra algorithms
  - Jack Dongarra, Jim Bunch, Cleve Moler, Gilbert Stewart
  - LINPACK still the name of the benchmark for the TOP500 (Wiki) list of most powerful supercomputers
- Problem
  - Implementation vector-based = low operational intensity
    (e.g., MMM as double loop over scalar products of vectors)
  - Low performance on computers with deep memory hierarchy (in the 80s)
- Solution: LAPACK
  - Reimplement the algorithms “block-based,” i.e., with locality
  - Developed late 1980s, early 1990s
  - Jim Demmel, Jack Dongarra et al.

Availability of LAPACK

- LAPACK available on nearly all platforms.
- Numerous implementations,
  - Intel MKL (Windows, Linux, OS X)
  - AMD ACML
  - OpenBLAS (Windows, Linux, Android, OS X)
  - Apple Accelerate (OS X, iOS)

MATLAB

- Invented in the late 70s by Cleve Moler
- Commercialized (MathWorks) in 84
- Motivation: Make LINPACK, EISPACK easy to use
- Matlab uses LAPACK and other libraries but can only call it if you operate with matrices and vectors and do not write your own loops
    - A*B (calls MMM routine)
    - A\b (calls linear system solver)

Problem with MATLAB

- Proprietary command line interpreted package.
- Extremely large (current desktop version is several Gigabytes).
- Designed more for prototyping, on high-end desktops.
- Not very useful for embedded (mobile) development.

Problems with OpenCV

- OpenCV improves greatly upon this issue.
  - Completely free and written in C++.
  - Has an OK matrix library, relatively easy to interpret.
  - Much light(er) weight (in size) than MATLAB.
- However, has problems.
  - Still relatively big - opencv2.framework is 23Mb compressed!!!
  - Not as fast as it should/could be.
- Alternate light-weight math libraries can help for iOS applications
  - Eigen (support for ARM NEON intrinsics)
  - Armadillo (uses LAPACK, MATLAB syntax)

Side Note: How Big Should an iOS App Be?

- Customers and clients care about app size...
  - Average size of App is around 23 Mb, and for games is now 60Mb (see link).
  - Apple has a maximum cellular download limit of 100MB.
  - Size of current opencv2.framework is 78.7 Mb - uncompressed!
- Important consideration in the design of a computer vision app is its size.

- Accelerate Framework comes “built-in” to all iOS devices.
  NOTHING TO DOWNLOAD!!

Demo : OpenCV vs LAPACK (MKL)

- https://github.com/CSE6000/Fall2016/tree/master/codes/OpenCV/OpenCV_vs_MKL

Mathematical Software

It is best to use high-quality software as much as possible,
for several reasons:

- It will take less time to figure out how to use the software than to write your own version. (Assuming it's well documented!)
- Good general software has been extensively tested on a wide variety of problems.
- Often general software is much more sophisticated that what you might write yourself, for example it may provide error estimates automatically, or it may be optimized to run fast.

LAPACK

Linear **A**lgebra **PACK**age www.netlib.org/lapack/

standard software library for numerical linear algebra

Written in Fortran

- solving systems of linear equations and linear least squares, eigenvalue problems, and singular value decomposition
- matrix factorizations such as LU, QR, Cholesky and Schur decomposition

LAPACK - Naming scheme

Many routines for linear algebra. Typical name: XYYZZZ

- X is precision
- YY is type of matrix, e.g. GE (general), BD (bidiagonal),
- ZZZ is type of operation, e.g. SV (solve system),
- EV (eigenvalues, vectors), SVD (singular values, vectors)

Examples:

- DGESV can be used to solve a general n × n linear system in double precision.
- DGTSV can be used to solve a general n × n tridiagonal linear system in double precision.

LAPACK on Windows

Basically,

It is for unix/linux. ➜ LAPACK for Windows

It is written in Fortran ➜ The LAPACKE C Interface to LAPACK

You can try it, if you want to.

But, we'll use Intel MKL and MS Visual Studio. It will save your time.

BLAS

Subroutine names start with:

- S: single precision
- D: double precision
- C: single precision complex
- Z: double precision complex

Examples:

- SAXPY: single precision replacement of y by αx + y.
- DDOT: dot product of two vectors
- DGEMV: matrix-vector multiply, general matrices
- DGEMM: matrix-matrix multiply, general matrices
- DSYMM: matrix-matrix multiply, symmetric matrices

Installing Intel MKL

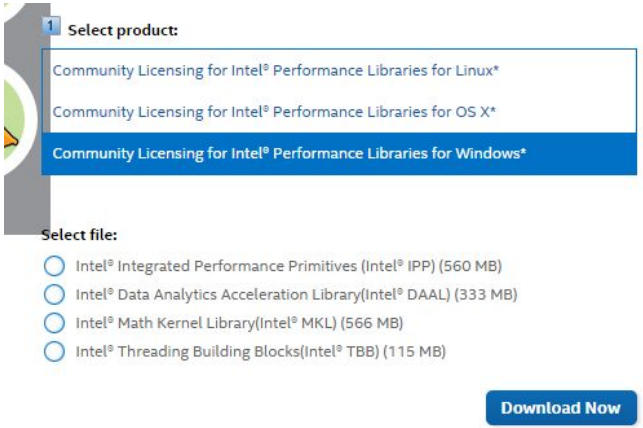
Community Licensing for Intel Performance Libraries

Register and download.

https://software.intel.com/sites/campaigns/nest/

Intel(R) Math Kernel Library 11.3 Update 2 for Windows (w_mkl_11.3.2.180.exe)

But, We have Intel® Parallel Studio XE which includes MKL.



Intel Math Kernel Library

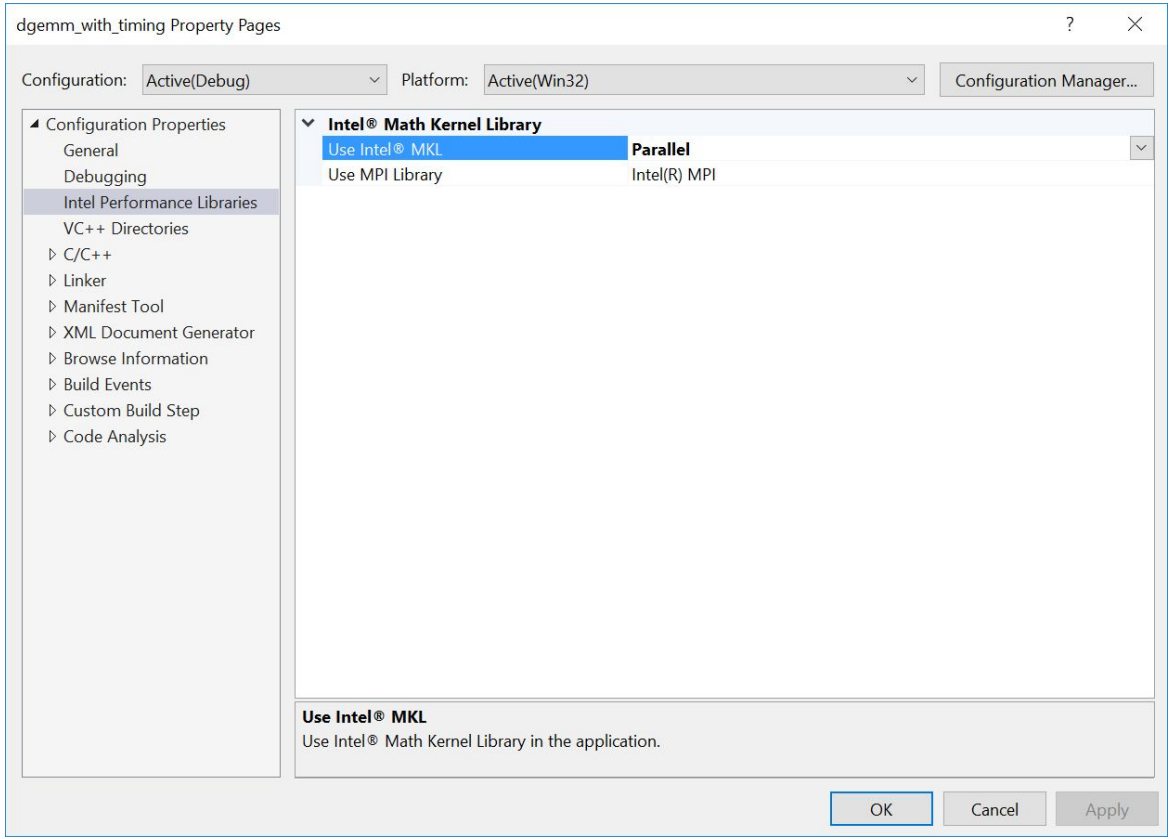
Documentation

Intel® Math Kernel Library Developer Reference - C

http://software.intel.com/en-us/mkl-reference-manual-for-c

Using MKL on MS Visual Studio

Examples
C:\Program Files (x86)\IntelSWTools\samples_2016\en\mkl\
matrix_multiplication_c.zip



C Interface Conventions

CBLAS

the C interface to the Basic Linear Algebra Subprograms (BLAS)

LAPACKE

See the `install_dir/include/mkl_lapacke.h` file for the full list of alternative C LAPACK interfaces.

Demo : matrix multiplication

Demo : dgemm

[Multiplying Matrices Using dgemm]

[cblas_?gemm]

```
C := alpha*op(A)*op(B) + beta*C
```