Image Processing: Stop Developing Code From Scratch

Walt Shands
Four Steps to a Parallel Application

**DESIGN - Intel® Parallel Advisor Lite**
Gain insight on where parallelism will most benefit existing source code - usually begins with a “hotspot”

**CODE, DEBUG - Intel® Parallel Composer**
Develop effective applications with a C/C++ compiler and comprehensive threaded libraries and API’s, and a parallel debugger

**VERIFY - Intel® Parallel Inspector**
Help ensure application reliability with proactive parallel memory and threading error checking

**TUNE - Intel® Parallel Amplifier**
Enhance applications with an intuitive performance analyzer and tuner

www.intel.com/go/parallel
Intel® Parallel Composer

Find where to start parallelizing

Introduce threads, compile, and debug with Intel® Parallel Composer

Find threading and memory errors with Intel® Parallel Inspector

Tune for optimal concurrency usage with Intel® Parallel Amplifier

CODE & DEBUG PHASE

Add parallelism to Windows* applications more quickly, with better scaling!

• Highly optimizing C/C++ compiler with built-in parallelism features:
  ➢ OpenMP 3.0*
  ➢ Intel® Threading Building Blocks
  ➢ Intel® Integrated Performance Primitives
  ➢ Parallel Debugger Extension

Increase productivity in implementing parallelism

www.intel.com/go/parallel
Intel® Integrated Performance Primitives (Intel® IPP) — Overview and Benefits

**Application Source Code**

Intel IPP Usage Code Samples
- Sample video/audio/speech codecs
- Image processing and JPEG
- Signal processing
- Data compression
- .NET and Java integration

**Intel IPP Library C/C++ API**
- Cryptography
- Image processing
- Image color conversion
- JPEG / JPEG2000
- Computer Vision
- Video coding
- Audio coding
- Data Compression
- Data Integrity
- Signal processing
- Matrix mathematics
- Vector mathematics
- String processing
- Speech coding
- Speech recognition

**Intel IPP Processor-Optimized Binaries**
- Intel® Core™ i7 Processors
- Intel® Atom™ Processors
- Intel® Core™ 2 Duo and Core™ Extreme Processors
- Intel® Core™ Duo and Core™ Solo Processors
- Intel® Pentium® D Dual-Core Processors
- Intel® Xeon® 64-bit Dual-Core Processors
- Intel® Pentium® M and Pentium® 4 Processors
- Intel® Itanium® 64-bit Processor Family
- Intel® Xeon® DP and MP Processors

**Free Code Samples**

**Cross-platform API**

**Processor-Optimized Implementation**

**Rapid Application Development**

**Compatibility and Code Re-Use**

**Outstanding Performance**

Intel Parallel Studio

is now available.

Get free eval software: intel.com/software/products/eval
Industry Leading Performance

- Efficient parallelism on multicore platforms
- Instruction set-level optimizations
- Intel-compatible platforms

Enhances Developer Productivity

- Optimized and future-scaling multimedia, signal, and data processing routines
- Comprehensive high level APIs and code samples for ease of implementation
- Utilize one performance library for 32 & 64 bit Windows*, Linux* and Mac OS

Future Proof

- Optimized for current multicore and future manycore processors
- Ensures that applications benefit seamlessly from the latest architecture enhancements
IPP image resize operations can run up to 3x faster than compiled C++ code on multicore systems.
Image processing: Intel IPP is a premier library of image processing algorithms, and includes a rich selection of functions operating on images and regions within images.

<table>
<thead>
<tr>
<th>Transforms:</th>
<th>Image Statistics:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavelet</td>
<td>Sum, Integral, Tilted Integral</td>
</tr>
<tr>
<td>Fourier (FFT/DFT, real/complex)</td>
<td>Mean, Min, Max, Histogram, StDev</td>
</tr>
<tr>
<td>Windowing (Hamming, Bartlett)</td>
<td>Image Moments</td>
</tr>
<tr>
<td>Discrete Cosine (DCT)</td>
<td>Image Norms (L1, L2, infinity)</td>
</tr>
<tr>
<td></td>
<td>Image Quality Index calculation</td>
</tr>
<tr>
<td></td>
<td>Proximity Measures (Cross-correlations, Square Distance)</td>
</tr>
<tr>
<td>Filtering Functions:</td>
<td>Threshold/Compare Operations</td>
</tr>
<tr>
<td>General linear filtering</td>
<td></td>
</tr>
<tr>
<td>Convolution/Deconvolution (LR and FFT)</td>
<td></td>
</tr>
<tr>
<td>Box, Min, Max, Median</td>
<td></td>
</tr>
<tr>
<td>Wiener Filters</td>
<td></td>
</tr>
<tr>
<td>Fixed Filters (Prewitt, Sobel, Laplace, Gauss, Scharr, Roberts)</td>
<td></td>
</tr>
<tr>
<td>Sharpening/Hi-pass/Low-pass Filters</td>
<td></td>
</tr>
<tr>
<td>Geometric Transforms:</td>
<td></td>
</tr>
<tr>
<td>Resize, Mirror, Rotate, Shear</td>
<td></td>
</tr>
<tr>
<td>Affine transforms</td>
<td></td>
</tr>
<tr>
<td>Perspective transformations</td>
<td></td>
</tr>
<tr>
<td>Bilinear warping</td>
<td></td>
</tr>
<tr>
<td>Coordinate remapping</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image Arithmetic/Logic Operations:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha composition</td>
<td>Arithmetic operations</td>
</tr>
<tr>
<td>(add/sub/mul/div/sqrt/sqr/ln/exp/abs)</td>
<td>Logical operations (AND, OR, XOR, Shift, NOT)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image Data Exchange/Initialization:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy/Set/Transpose</td>
<td></td>
</tr>
<tr>
<td>Channel swapping</td>
<td></td>
</tr>
<tr>
<td>Jaehne/Ramp/Zigzag initialization</td>
<td></td>
</tr>
<tr>
<td>Memory allocation for multiple image types</td>
<td></td>
</tr>
</tbody>
</table>
Intel® Parallel Inspector

Find where to start parallelizing

Introduce threads, compile, and debug with Intel® Parallel Composer

Find threading and memory errors with Intel® Parallel Inspector

Tune for optimal concurrency usage with Intel® Parallel Amplifier

CODE CHECKING PHASE

Ensure parallel application reliability!

- Memory Error Detection
- Threaded Error Detection

Help ensure application reliability with proactive parallel memory and threading error checking

www.intel.com/go/parallel
Intel® Parallel Inspector

Memory Error Detection
- Memory Leaks, i.e., allocation errors
- Memory Corruption, i.e., crashes, etc.
- Uninitialized Memory Accesses
- Dangling pointers
- Buffer Overflows, Stack Overflows
- Depth of error checking can be configured

Thread Error Detection
- Data race errors
- Deadlocks
- Finds latent (or likely to occur) errors and maps them to the source-code line, call stack, and memory reference
- Displays useful warnings for effective diagnosis, highlighting potentially severe errors

Works on standard debug builds
Intel® Parallel Amplifier

Find where to start parallelizing

Introduce threads, compile, and debug with Intel® Parallel Composer

Find threading and memory errors with Intel® Parallel Inspector

Tune for optimal concurrency usage with Intel® Parallel Amplifier

TUNING PHASE

Tune for multi-core performance scalability!

- Hotspot Analysis
  Where is my program spending time running?

- Concurrency Analysis
  Where is my program not concurrent?

- Lock/Wait Analysis
  Where is my program waiting on Sync or I/O?

www.intel.com/go/parallel
Hotspot Analysis
- Where is my program spending time running?
  And how it got there
  Identify Hotspot functions

Concurrency Analysis
- Where is my program not concurrent?
  Useful when moving serial code to parallel
  Identify processor cores utilization

Lock/Wait Analysis
- Where is my program waiting on Sync or I/O?
  Identify locking problems that slow threaded software
  Identify objects limiting parallelism

Source View
- Shows data so it correlates with your code

Statistical Call Tree
- Helps catch call path most impacting performance

Compare Multiple Runs
- Quickly see the impact of your changes
Sobel Edge Detection

Apply vertical edge filter to each pixel to find vertical edges

Apply horizontal edge filter to each pixel to find horizontal edges

Find vertical and horizontal edges
Intel IPP vs. C on single processor
- 200% faster (average over all domains)
- Optimized C performance normalized to 1

Multicore performance scaling example: H.264 decode

System configuration: Intel® Xeon® 4 Processor, 2.8GHz, 2GB using Windows® XP

Source: Intel Corporation. Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, refer to www.intel.com/performance/resources/benchmark_limitations.htm.
Threading In Application

1 of 2 cores

application thread

application thread 1

call Intel IPP

call Intel IPP

application thread continues

2 of 2 cores

application thread 2

call Intel IPP

call Intel IPP

application thread continues

does threading

thread-safe functions

Intel Parallel Studio is now available.
Get free eval software:
intel.com/software/products/eval
Intel® IPP Functions and Code Samples: 
Image Processing/Coding

<table>
<thead>
<tr>
<th>Domain</th>
<th>Functions</th>
<th>Samples</th>
</tr>
</thead>
</table>
| **Image Processing** | * Geometry transformations, such as resize/rotate  
* Linear and non-linear filtering operation on an image for edge detection, blurring, noise removal and etc for filter effect.  
* Linear transforms for 2D FFTs, DFTs, DCT.  
* image statistics and analysis | * Tiled Image Processing / 2D Wavelet Transform /C++ Image Processing Classes/Image Processing functions Demo |
| **Computer Vision** | * Background differencing, Feature Detection (Corner Detection, Canny Edge detection), Distance Transforms, Image Gradients, Flood fill, Motion analysis and Object Tracking, Pyramids, Pattern recognition, Camera Calibration | * Face Detection |
| **Color Models** | * Convert image/video color space formats: RGB, HSV, YUV, YCbCr  
* Up/Down sampling  
* Brightness and contrast adjustments | |  
| **JPEG Coding** | * High-level JPEG and JPEG2000 compression and decompression functions  
* JPEG/JPEG2000 support functions: DCT, Wavelet transforms, color conversion, downsampling | * Integration with the Intel® JPEG Library (IJL) / Integration with the Independent JPEG Group (IJG) library /JPEG2000 encoder/decoder /JPEG viewer |
| **Realistic Rendering** | * Acceleration Structures, Ray-Scene Intersection and Ray Tracing  
* Surface properties, shader support, tone mapping | * Ray Tracing |
## Intel® IPP Functions and Code Samples: Video/Audio/Speech

<table>
<thead>
<tr>
<th>Domain</th>
<th>Functions</th>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Video Coding</strong></td>
<td>* VC-1, H.264, MPEG-2, MPEG-4, H.261, H.263 and DV codec support functions</td>
<td>* Simple Media Player/ Video Encoder / h.264/DV decoding/video transcoder /Reverberation Demo/ Virtual Multi-Channel Audio Player/</td>
</tr>
<tr>
<td><strong>Audio Coding</strong></td>
<td>* Echo cancellation and audio transcoding, BlockFiltering, Spectral Data prequantization.</td>
<td>* Audio Codec Console application</td>
</tr>
<tr>
<td><strong>Speech Recognition</strong></td>
<td>* Feature Processing, Model Evaluation/Estimation/Adaptation, Vector Quantization, Polyphase Resampling, Advanced Aurora, Ephraim-Malah Noise Supression, AEC, Voice Detection</td>
<td>* Aurora, Advanced Aurora, Audio Processing, Gaussian Mixture, Speech Processing</td>
</tr>
</tbody>
</table>
| **Signal Processing** | * Transforms: DCT, DFT, MDCT, Wavelet (both Haar and user-defined filter banks), Hilbert  
* Convolution, Cross-Correlation, Auto-Correlation, Conjugate  
* Filtering: IIR/FIR/Median filtering, Single/Multi-Rate FIR LMS filters  
* Other: Windowing, Jaehne/Tone/Triangle signal generation, Thresholding | * Signal Processing Function Demo |
**Intel® IPP Functions and Code Samples:**

Data Processing/Compression

<table>
<thead>
<tr>
<th>Domain</th>
<th>Functions</th>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Compression</td>
<td>* Entropy-coding compression: Huffman, VLC</td>
<td>* zlib, bzip2, gzip-compatible /General data compression examples</td>
</tr>
<tr>
<td></td>
<td>* Dictionary-based compression: LZSS, LZ77</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* Burrows-Wheeler Transform (BWT), MoveToFront (MTF), Run-Length-Encoding (RLE), Generalized Interval Transformation (GIT)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* Compatible feature support for zlib and bzip2</td>
<td></td>
</tr>
<tr>
<td>Cryptography</td>
<td>* Big-Number Arithmetic / Rijndael, DES, TDES, SHA1, MD5, RSA, DSA, Montgomery, prime number generation and pseudo-random number generation (PRNG) functions</td>
<td>* Intel IPP crypto usage in Open SSL*</td>
</tr>
<tr>
<td>String Processing</td>
<td>* Compare, Insert, change case, Trim, Find, Regexp, Hash</td>
<td>* &quot;ippgrep&quot; – regular expression matching</td>
</tr>
<tr>
<td>Vector Math</td>
<td>* Logical, Shift, Conversion, Power, Root, Exponential, Logarithmic, Trigonometric, Hyperbolic, Erf, Erfc</td>
<td></td>
</tr>
<tr>
<td>Matrix Math</td>
<td>* Addition, Multiplication, Decomposition, Eigenvalues, Cross-product, transposition</td>
<td></td>
</tr>
<tr>
<td>Common Functions</td>
<td>* CPUTypes, Thread number control, Memory Allocation</td>
<td>* Linkages/Different language support</td>
</tr>
</tbody>
</table>
Deferred Mode Image Processing
Image Processing Task using a Sequence of Basic Operations

Basic operations

Source Image ➔ byte-to-float conversion ➔ buffer ➔ Filtering ➔ buffer ➔ Threshold ➔ buffer ➔ float-to-byte conversion ➔ Result Image

Intermediate Buffering
• Memory allocation for intermediate buffering
• Cache thrashing
Image Processing Task, using Deferred Mode Image Processing (DMIP)

Configurable Compound Operation – defined using DMIP

- byte-to-float conversion
- Filtering
- Threshold
- float-to-byte conversion

Inter-operation optimizations
- Image partitioning for cache use maximization
- Pipelining and parallelization of basic operations
- Calculation on L2 cache
• DMIP provides a mechanism to chain multiple image processing operations together
• DMIP performs scan line based processing to better utilize cache behavior
• DMIP shows significant performance benefit for large images
• Provided in Intel® IPP
DMIP Essentials

• Intuitive programming layer based on C++
  - .NET interface also supported

• Performance benefits
  - Utilize existing optimized IPP functions
  - Utilize parallel processing on multi-core platforms
  - Perform scan line based processing for better cache utilization

• Easy to extend with user defined operations
DMIP implementation

Image A(s, Ipp8u, IppC3, roi, sStep)
Image D(d, Ipp8u, IppC1, roi, dStep)
Kernel KH(Horiz, k3x3, ipp8u, ipp16s)
Kernel KV(Vert, k3x3, ipp8u, ipp16s)
Graph G = ColorToGray(*A)
D = To8u(Abs(G*KH) + Abs(G*KV))
<table>
<thead>
<tr>
<th>Live Date</th>
<th>Event</th>
<th>Title</th>
<th>Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/10/2009</td>
<td>Webinar</td>
<td>Go-Parallelism! Ease the Onramp for C/C++ Windows Development</td>
<td>James Reinders</td>
</tr>
<tr>
<td>3/17/2009</td>
<td>Tech. Session</td>
<td>Solve Parallelism with Intel® Parallel Studio</td>
<td>Joe Wolf</td>
</tr>
<tr>
<td>3/24/2009</td>
<td>Webinar</td>
<td>Simplify Parallelism with Intel® Parallel Composer</td>
<td>Joe Wolf</td>
</tr>
<tr>
<td>3/31/2009</td>
<td>Tech. Session</td>
<td>Parallel Implementation Methods with Intel® Parallel Composer</td>
<td>Ganesh Rao</td>
</tr>
<tr>
<td>4/7/2009</td>
<td>Webinar</td>
<td>Debugging Parallel Code for Fast, Reliable Applications</td>
<td>Jay Desouza</td>
</tr>
<tr>
<td>4/14/2009</td>
<td>Tech. Session</td>
<td>Find Errors in Windows C++ Parallel Applications</td>
<td>Gerold Mueller or Robert Mueller</td>
</tr>
<tr>
<td>4/21/2009</td>
<td>Webinar</td>
<td>Easy Ways to Solve Parallel Performance Challenges</td>
<td>Gary Carleton</td>
</tr>
<tr>
<td>5/5/2009</td>
<td>Webinar</td>
<td>The Key to Scaling Applications for Multicore</td>
<td>Paul Petersen/Mark Davis</td>
</tr>
<tr>
<td>5/12/2009</td>
<td>Tech. Session</td>
<td>Identify and Address Threading Opportunities</td>
<td>Caroline Davidson</td>
</tr>
<tr>
<td>5/19/2009</td>
<td>Webinar</td>
<td>Image Processing: Stop Developing Code from Scratch</td>
<td>Walt Shands, Intel Corporation</td>
</tr>
<tr>
<td>5/26/2009</td>
<td>Tech. Session</td>
<td>Simplifying Parallelism Implementation with Intel® Threading Building Blocks</td>
<td>Mike D’Mello</td>
</tr>
<tr>
<td>6/2/2009</td>
<td>Tech. Session</td>
<td>Static Analysis and Intel® C++ Compilers</td>
<td>Dmitry Putinin</td>
</tr>
</tbody>
</table>

www.intel.com/go/parallel
Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel® Compiler User and Reference Guides” under “Compiler Options.” Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101