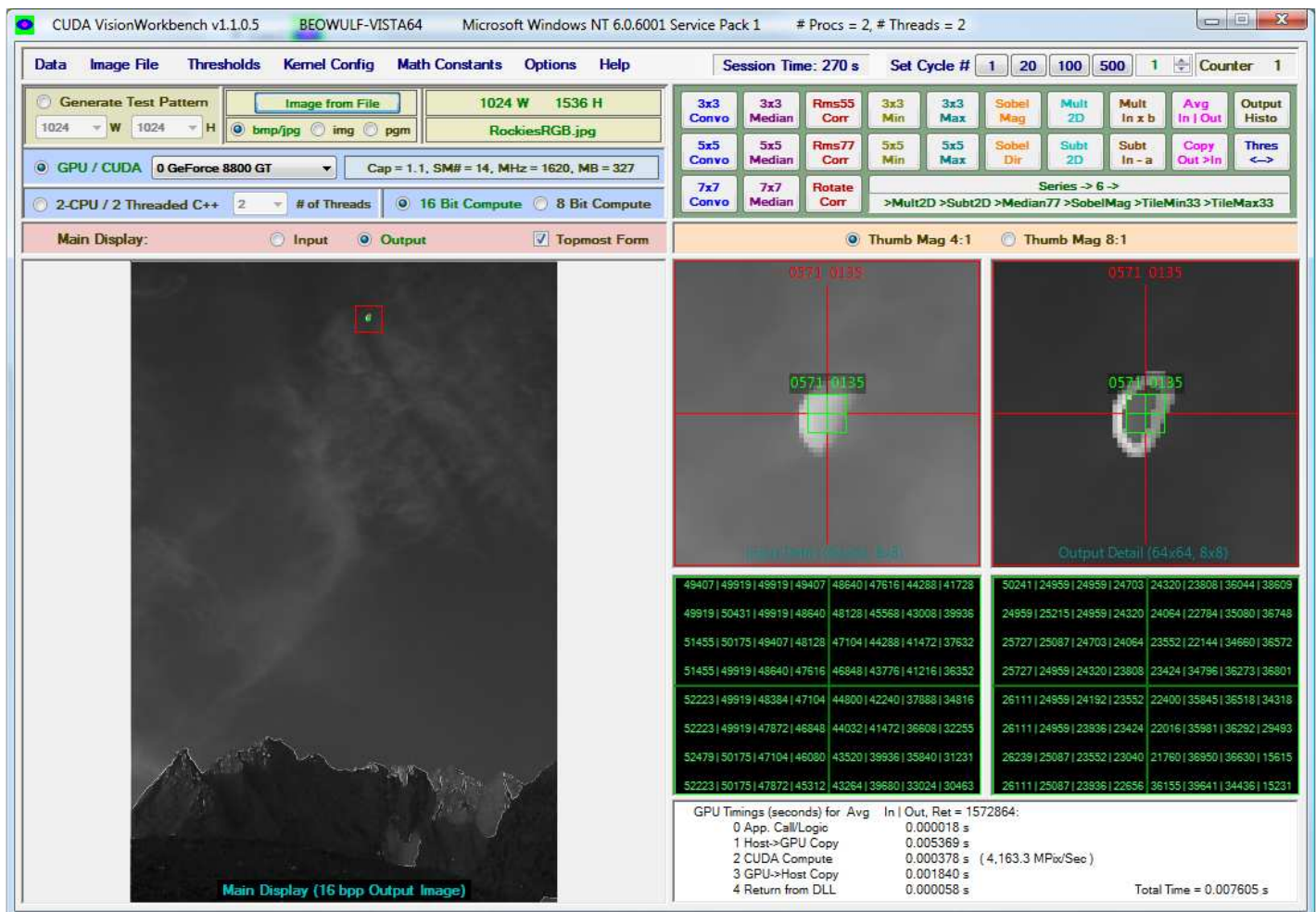


CUDA VisionWorkbench v 1.1.0.5

2/22/2009

Overview

CUDA VisionWorkbench (CVWB) is an application for 32 or 64 bit versions of Windows XP or Vista, used primarily for demonstration, benchmarking and development of vision primitives implemented in CUDA. CVWB presents an interactive UI for calling various image processing operations on 8 bit or 16 bit grayscale images of a variety of sizes. The UI gives the user options to load images from files or generate test patterns, select enumerated GPU's or the CPU for processing, select # of threads for host processing, select processing bit depth, and load, re-configure and save structuring element matrices for convolution and correlation operators. Processing operations can be triggered individually or in a repeating loop, and a representative multi-operator sequence can also be initiated. Results are presented visually for interactive comparison, and key metrics are automatically recorded to pre-formatted files for analysis.



Uses for the CVWB Application and DLL

- 1) Benchmarking and testing vision primitive functions implemented in CUDA for a GPU and C++ for a host CPU. This may be useful, for example, for comparison of different GPU and host CPU hardware configurations, iterative development and evaluation of functions and informal regression testing.
- 2) Demonstrate optimized but reasonably transparent CUDA code in the context of a whole application.
- 3) Provide a basic set of CUDA and C++ vision processing primitives (with plans to continue adding more).
- 4) Demonstrate a way to obtain a large "pipeline benefit" for multiple processing operations, with highly modular CUDA code. Important and effective for masking host-CPU transfer time for processing pipelines.
- 5) Show a way to use managed, safe .Net code for application programming, while still obtaining large acceleration from the GPU via CUDA code contained in an unmanaged DLL.

CVWB Prerequisites

1) CVWB Operating Prerequisites

- a) x86 CPU based system running 32 bit or 64 bit versions of Microsoft Windows XP or Vista Operating System with all available service packs, patches and updates.
- b) .Net Framework 2.0 with all available service packs, patches and updates
- c) Minimum recommended display/monitor will have horizontal resolution greater than 1086 and vertical resolution greater than 758.
- d) Windows system DPI setting should be set to standard /default 96 DPI or 120 DPI. To check or set the Windows system DPI:
 - i) For Windows XP, Right-Click on the **Desktop** and choose **Properties** from the pop-up menu. Select the **Settings tab in the** presented **Display Settings** dialog, then click the **Advanced** button. At the presented dialog, select the **General** tab and select **Normal size (96 DPI)** or **Larger Size (120 DPI)** or from the **DPI setting** drop down combo-box and then click **OK**. A restart may be needed if this is a change from the previous DPI setting.
 - ii) For Windows Vista, Right-Click on the **Desktop** and choose **Personalize** from the pop-up menu. Select **Adjust Font Size (DPI)** from the **Tasks** list at the left side of the presented dialog. At the **DPI Scaling** dialog, select the **Default scale (96 DPI)** or **Larger Scale (120 DPI)** radio button and **Apply**. A restart may be needed if this is a change from the previous DPI setting.
- e) Compute-capable NVIDIA GPU on system: See http://www.nvidia.com/object/cuda_learn_products.html
- f) CUDA 2.1 compatible Driver and CUDA 2.1 Toolkit
 - i) Laptop/Notebook Drivers for GeForce and Quadro NVS GPUs
 - (1) http://www.nvidia.com/object/notebook_drivers.html
 - ii) For Windows XP, 32 bit edition
 - (1) Desktop PC Driver:
http://www.nvidia.com/object/thankyou.html?url=/compute/cuda/2_1/drivers/181.20_geforce_winxp_32bit_english_whql.exe
 - (2) Toolkit:
http://www.nvidia.com/object/thankyou.html?url=/compute/cuda/2_1/toolkit/CudaSetup-2.1-win32.exe
 - iii) For Windows XP, 64 bit edition
 - (1) Desktop PC Driver:
http://www.nvidia.com/object/thankyou.html?url=/compute/cuda/2_1/drivers/181.20_geforce_winxp_64bit_english_whql.exe
 - (2) Toolkit:
http://www.nvidia.com/object/thankyou.html?url=/compute/cuda/2_1/toolkit/CudaSetup-2.1-win64.exe
 - iv) For Windows Vista, 32 bit edition
 - (1) Desktop PC Driver:
http://www.nvidia.com/object/thankyou.html?url=/compute/cuda/2_1/drivers/181.20_geforce_winvista_32bit_english_whql.exe
 - (2) Toolkit:
http://www.nvidia.com/object/thankyou.html?url=/compute/cuda/2_1/toolkit/CudaSetup-2.1-win32.exe
 - v) For Windows Vista, 64 bit edition
 - (1) Desktop PC Driver:
http://www.nvidia.com/object/thankyou.html?url=/compute/cuda/2_1/drivers/181.20_geforce_winvista_64bit_english_whql.exe
 - (2) Toolkit:
http://www.nvidia.com/object/thankyou.html?url=/compute/cuda/2_1/toolkit/CudaSetup-2.1-win64.exe

2) Additional CVWB Prerequisites for Development

- a) Visual Studio 2005 Professional/Team Edition with SP1 and security updates installed.
- b) Note that the CUDA Toolkit and SDK installations should create the following definitions in your system Environment Variables (assuming you accepted the default installation parameters):
 - i) CUDA_BIN_PATH should be defined as: **c:\CUDA\bin**
 - ii) CUDA_INC_PATH should be defined as: **c:\CUDA\include**
 - iii) CUDA_LIB_PATH should be defined as: **c:\CUDA\lib**
 - iv) Check these by Left-Clicking on **Start**, Right-Clicking on **My Computer** (XP) or **Computer** (Vista) and Left-Clicking on **Properties**. In the dialog box presented, Left-Click on the **Advanced** tab (XP) or **Advanced System Settings** item (Vista) and Left-click on the **Environment Variables** button. Review the list presented in the **Environment Variables** dialog presented. If the above entries are not listed, add them using the **New** button and typing in the Names above (e.g. “CUDA_BIN_PATH”) and the Paths above (e.g. “c:\CUDA\bin”)

CVWB Installation

1) To install directly from the Web:

- a) Use one of the following links (x86 for 32 bit Windows, x64 for 64 bit Windows), navigate by browser to the installer page matching your OS platform:

x86 for 32 bit Windows: http://openvidia.sourceforge.net/VisionWorkbench_x86/CUDAVisionWorkbenchInstaller_x86.htm

x64 for 64 bit Windows: http://openvidia.sourceforge.net/VisionWorkbench_x64/CUDAVisionWorkbenchInstaller_x64.htm

- b) Click on the “Install” button and follow the prompts to install CVWB.

2) To install CVWB from a downloaded source code package:

- a) Use one of the following links, navigate by browser to the installer page matching your OS platform:

x86 for 32 bit Windows: http://sourceforge.net/project/showfiles.php?group_id=98913&package_id=280176

x64 for 64 bit Windows: http://sourceforge.net/project/showfiles.php?group_id=98913&package_id=291497

- b) Download VisionWorkbench_x.x.x.x_x32.zip or VisionWorkbench_x.x.x.x_x64.zip (where x.x.x.x) is the code package version.

- c) Unzip the downloaded .zip file into:

Windows XP: **C:\Documents and Settings\<UserName>\My Documents\Visual Studio 2005\Projects**

Windows Vista: **C:\Users\<UserName>\Documents\Visual Studio 2005\Projects**

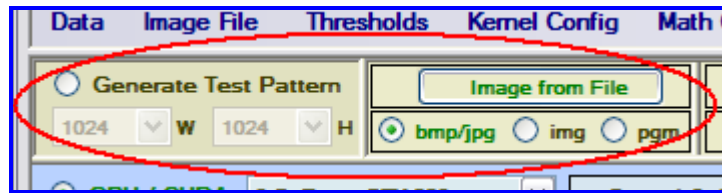
- d) Navigate by file explorer to the **\Publish** subdirectory in the unzipped destination folder.

- e) Double click on **CUDAVisionWorkbenchInstaller_x86.htm** (for 32 bit Windows) or **CUDAVisionWorkbenchInstaller_x64.htm** (for 64 bit Windows), and follow the prompts to install CVWB.

Note: To develop with the downloaded, unzipped code package in Visual Studio, remember to Right-Click on the unzipped destination folder in your file explorer, select Properties, clear the “Read Only” attributes check box and confirm when prompted to apply the change to all files and sub folders.

Basic CVWB Operating Instructions

- 1) Choose and configure the image source from the mustard colored panel at the top left of the main CVWB window:

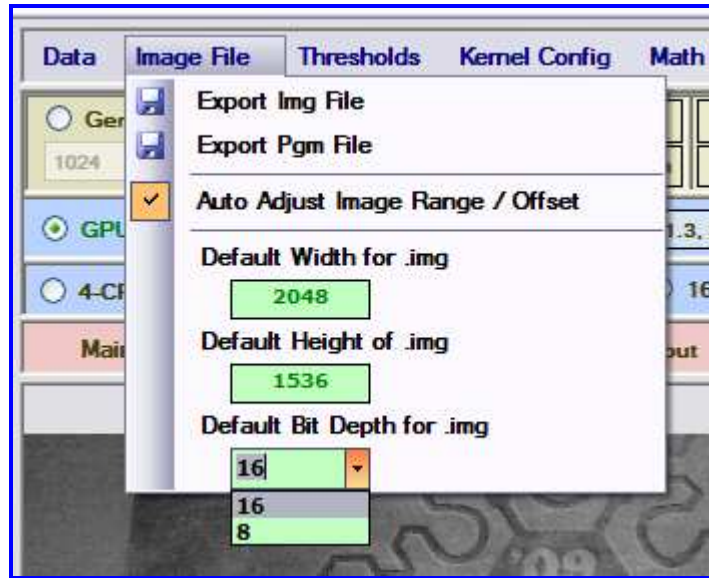


- a) To use the automatically generated test image, select the **Generate Test Pattern** radio button and set the desired width and height or change these using the **W** and **H** combo boxes underneath the **Generate Test Pattern** radio button in the top left panel. These width and height settings are “sticky” from session to session.
- b) To load (or reload) an image from a file, click the **Image from File** radio button and, when prompted, browse to and select a file of one of the supported types:
- 8 bit monochrome or 24 bit RGB color bitmap (bmp) file, or a jpg file
 - 8 or 16 bit .img file,
 - 8 or 16 bit pgm file

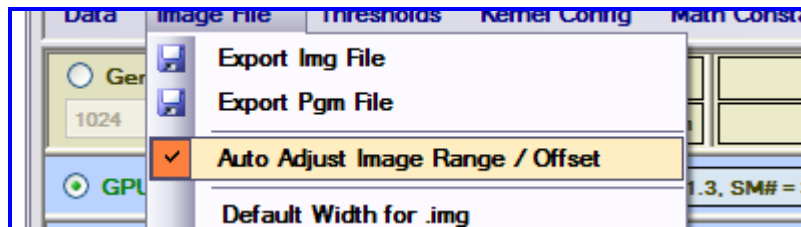
Note: If you downloaded the CVWB source file package, some images are provided in the **Images** directory for your use, or you can supply your own.

- c) If the **bmp/jpg** radio button option is selected, the **Image from File** radio button will prompt with filters for bmp and jpg files.
- d) If the **img** radio button option is selected, the **Image from File** radio button will prompt with filters for img files.
- When the **.img** option is enabled and img files are opened, the function called by the **Image from File** radio button will import a raw image from a binary data file using a metadata ASCII text file containing the width, height and bit depth information for the raw binary image file.
 - The data format for the binary .img file type is: sequential byte values (in LittleEndian order for multi-byte values). Only bit depths of 16 and 8 (2 byte and 1 byte) are currently supported for this .img raw binary storage file type. No header or metadata is contained in the binary file.
 - Image width, height and bit depth are specified externally to the binary data file within a simple metadata file having the same name but a **.dat** extension instead of **.img** extension. The format for the metadata file is ASCII text:
 - <ushort Width value> followed by “W” and CR/LF on the 1st line
 - <ushort Height value> followed by “H” and CR/LF on the 2nd line
 - <ushort Bitdepth value> followed by “B” and CR/LF on the 3d line
 - EOF
 - Any information after the 3d line will be ignored.
 - If the metadata file for the selected raw binary .img file is found and the data is successfully read, the image is imported according to the parameters in the metadata file, and the **Image File** menu parameters **Default Width for .img**, **Default Height for .img** and **Default Bit Depth for .img** are then updated to match the current file.
 - If no metadata file is found in the same directory as the binary raw image file or if there is an error reading the metadata file, the CVWB will use the defaults shown in the **Image File** menu parameters **Default Width for .img**, **Default Height for .img** and **Default Bit Depth for .img**.

- xi) Note that the defaults shown in the **Image File** menu parameters **Default Width for .img**, **Default Height for .img** and **Default Bit Depth for .img**. may be overridden and stored (type in the values and store them by hitting enter). Access to these menu parameters is enabled by the **img** radio button selection.

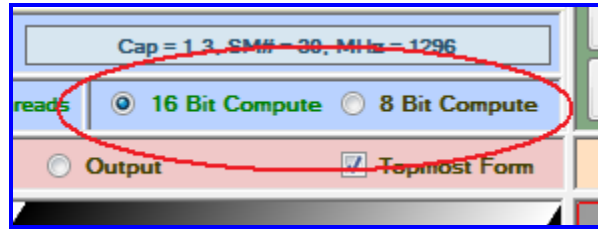


- xii) The **img** radio button and the associated **ImageFile** menu default values are sticky from session to session.
- e) If the **pgm** radio button option is selected, the **Image from File** radio button will prompt with filters for pgm files. 16 bit and 8 bit monochrome pgm files are currently supported. Image data within the file follows the header lines and is stored in Big-Endian order for multi-byte types.
- f) Export of img and pgm and images is available from the **ImageFile** menu as shown. The exported image will be the current image in the Main Display, and the format will be in the current processing bit depth (8 or 16 bit).
- 2) Decide if you want image auto-ranging enabled, and set the **Auto Adjust Image Range/Offset** checkbox option in the **Image File** menu accordingly.

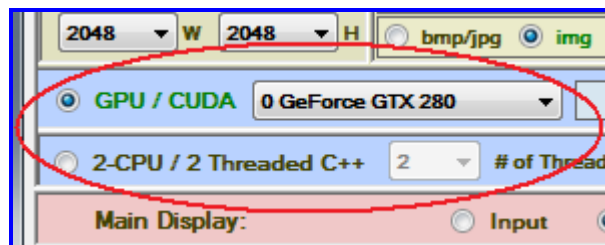


- a) When this option is enabled, image data is automatically offset and scaled to full dynamic range for the currently selected processing bit depth during import from a file or when changing processing bit depth.
- b) The routine is a linear one... the minimum value in the parent image is subtracted from all values, and the resulting image is multiplied by the ratio of (the GV range available at the current bit depth)/(GV Range in the Parent Image). This has the effect of greatly increasing contrast in some images whose raw data only occupies a fraction of the available dynamic range for the current processing bit depth.

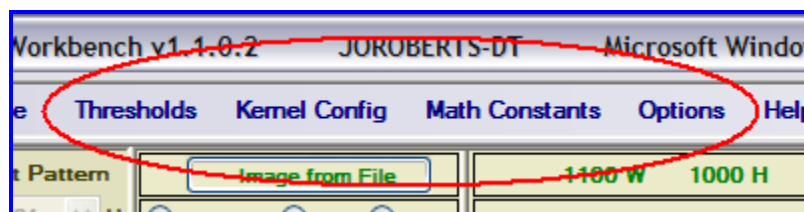
- 3) Accept the loaded processing bit depth or select 8 or 16 bit processing using the **16 Bit Compute** or **8 Bit Compute** radio buttons in the sky blue colored panel near the top middle of the main CVWB window.



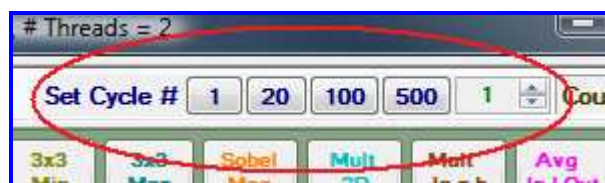
- a) Whether an image source is a file or a generated test pattern, and whether a file image source is 8, 16 or 24 bit, the date imported will be converted for processing according to this selection.
 b) This processing bit depth setting is “sticky” from session to session.
- 4) Select desired processing system using the **GPU/CUDA** or **n-CPU/n-threaded C++** radio buttons in the sky blue colored panel near the top left of the main CVWB window.



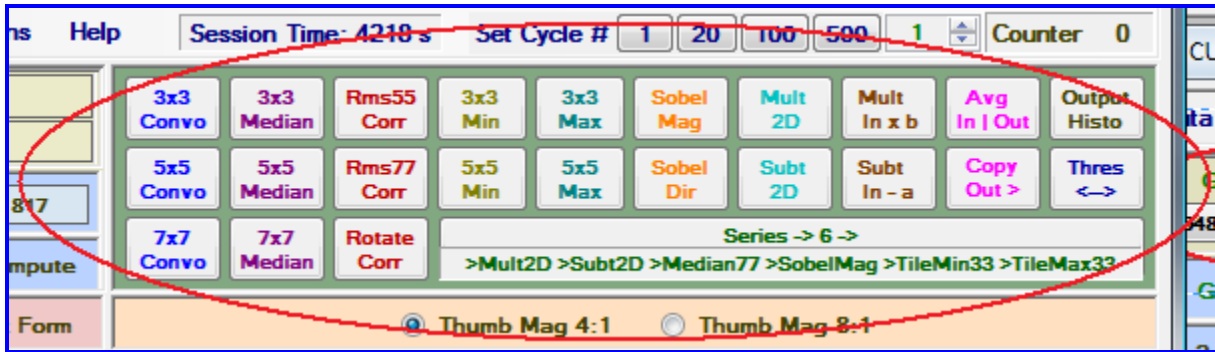
- a) If using GPU processing, accept the present GPU or select a new one from those enumerated in the combo box adjacent the **GPU/CUDA** radio button. Changes in selected GPU presently require you a quick restart of CVWB (you’ll be prompted to accept or cancel this). This setting is “sticky” from session to session.
 b) Accept the # of host processing threads for Host CPU / C++ processing or select desired # using the combo box adjacent the **n-CPU/n-threaded C++** radio button. This value will initially default to the # of processors enumerated on the system, but user selection will override this. This setting is “sticky” from session to session.
- 5) If running Thresholds, Correlations, Convolutions, Subtraction, Multiplication or Histograms, check and set the self-explanatory menu items in the **Thresholds, Kernel Config, Math Constants and Options** menus.



- 6) Accept the processing cycle count shown in green at the **Set Cycle #** controls at the right end of the menu strip at the top, or set the # of cycles desired. This setting is “sticky” from session to session.



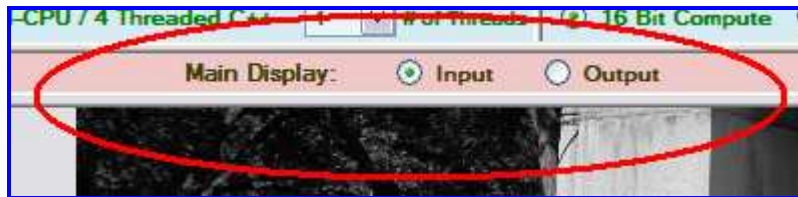
- 7) Click one of the colorful processing operator buttons from the olive colored panel at the top right of the main CVWB window.



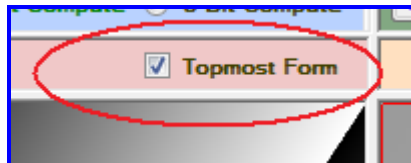
Note: Note that some functions (histogram, Right-Click->Save Bitmap, Export Img File and Export Pgm file) are context sensitive to the Input/Output state of the main image.

Interacting with Images in CVWB

- 1) The main image pane can be toggled between the input and preprocessed output image using the **Input** and **Output** radio buttons in the rose colored **Main Display** pane above the main image display.

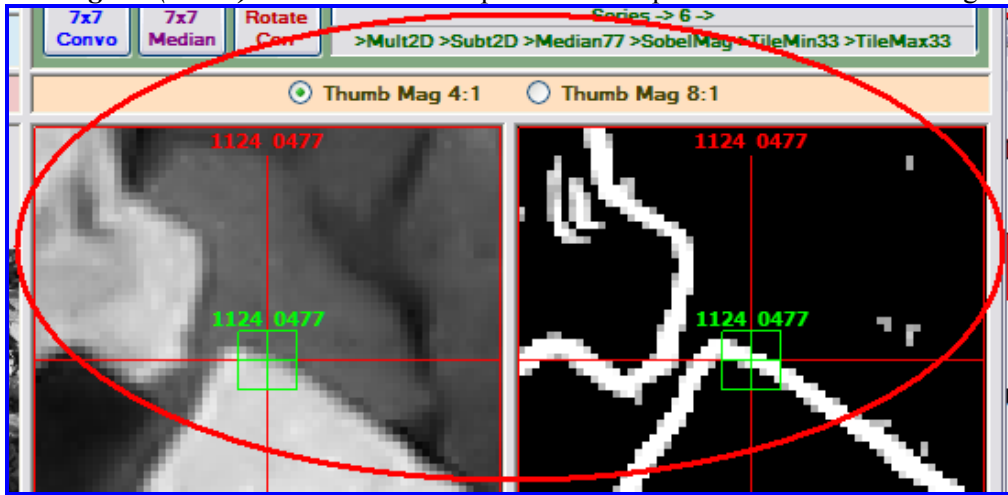


- 2) By default, the main CVWB form is set to be the topmost form on the Windows desktop. To disable this, clear the **Topmost Form** checkbox at the right side the rose colored **Main Display** pane.

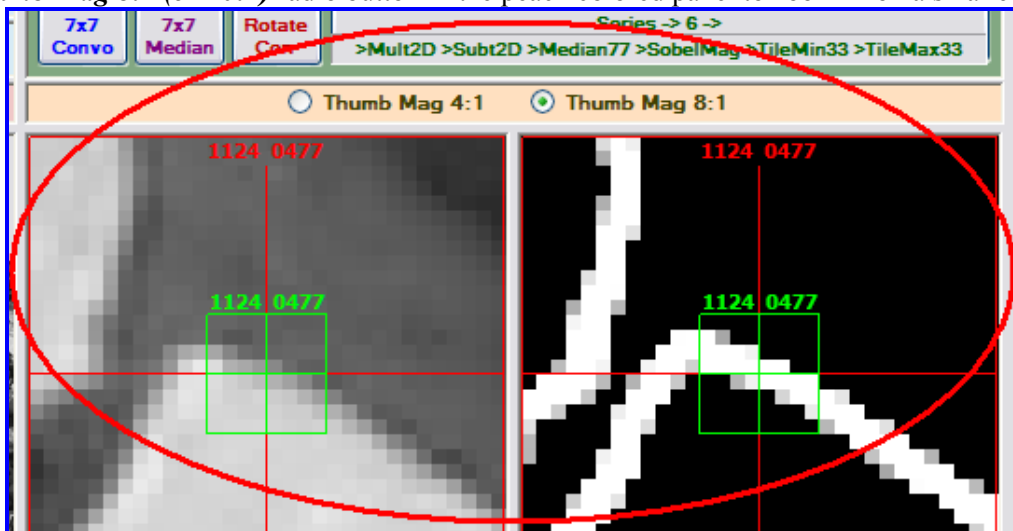


- 3) Details of the region around the cursor in the main image pane are presented in the thumbnail views to the right.
- 4) Mouse Left-click on the main picture box engages/disengages Thumbnail Scroll Mode to move cursor location and associated thumb details. Mouse Left-Click again on Main Bitmap to disengage Thumbnail Scroll Mode and freeze the Thumbnail Locations
- 5) Arrow keys nudge the cursor location in the main picture box and thumbnail details by discrete 1-pixel increments.
 - a) If you have a Keyboard with a Number Pad, turn off numlock to nudge thumbnails incrementally using Number Pad:
 - i) **Keys: Up arrow, Down arrow, Right arrow and Left arrow**
Move: N S E and W
 - ii) **Keys: Page Up, Page Down, End and Home**
Move: NE SE SW and NW
 - b) Otherwise, if you have an extended keyboard with basic arrow keys, use the Up-Down-Right-Left Arrow keys to nudge Thumbnail locations N-S-E-W respectively.
- 6) The magnification (ratio of Display Pixels to underlying input/output image data pixels) of the thumbnail views can be changed using the radio buttons above the thumbnail views.
 - a) The available low and high magnification options will be 4:1 and 8:1 if the Windows system DPI is set to 96 DPI and will be 5:1 and 10:1 respectively if the Windows system DPI is set to 120 DPI.
 - b) The underlying input/output image data depicted in the thumbnails will always be a 64 x 64 pixel block for low mag or a 32 x 32 pixel block for high mag.

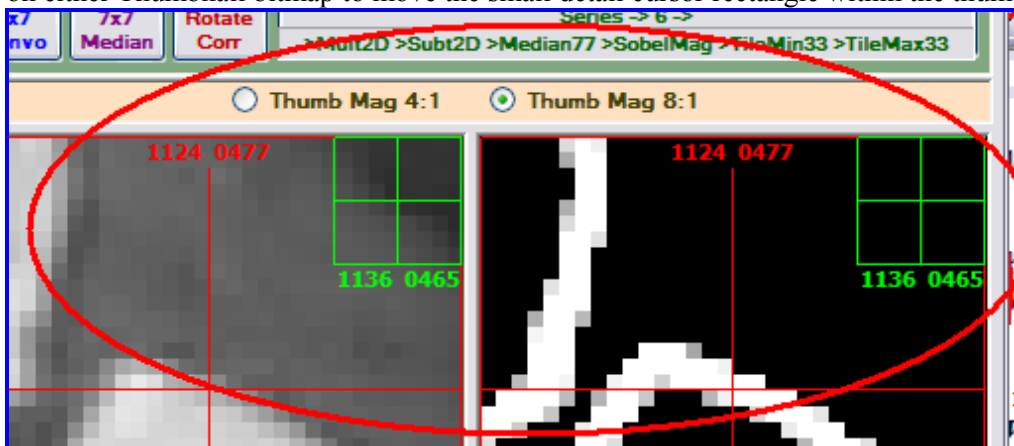
7) Click the *Thumb Mag 4:1 (or 5:1)* radio button in the peach colored panel to zoom out for a larger field of view



8) Click the *Thumb Mag 8:1 (or 10:1)* radio button in the peach colored panel to zoom in on a smaller field of view



9) Double-click on either Thumbnail bitmap to move the small detail cursor rectangle within the thumbnail



10) Right-Click on Main or Thumbnail Bitmaps to bring up a menu that allows you to

- a) Reset the small detail cursor rectangle to thumbnail center.
- b) Save the thumbnail to a .bmp File, or

11) Right-Click on Thumbnail Text Boxes, Timings Box or Histogram Text Box to Save their contents to .RTF Files

Reviewing Data

- 1) Numerical values representing the pixel gray levels in the small thumbnail cursor region are presented in the green text boxes beneath each of the thumb views.
- 2) Timing info is presented in the lower right pane after operation execution.
- 3) Right-clicking on **any of the main CVWB window panes** allows saving the pane contents (image, text) with a context sensitive popup menu.
- 4) To export the current Main Display image (whether an input image or a processed output image) to the binary **.img** file (along with a companion metadata **.dat** file), use the **Export Img File** option in the **Image File** menu. The exported image data will come from the data underlying the current Main Image display, and will have bit depth matching the current selection of **16 Bit Compute** or **8 Bit Compute** radio buttons.
- 5) To export the current Main Display image (whether an input image or a processed output image) in the **.pgm** format (with self contained header data), use the **Export Pgm File** option in the **Image File** menu. The exported image data will come from the data underlying the current Main Image display, and will have bit depth matching the current selection of **16 Bit Compute** or **8 Bit Compute** radio buttons.
- 6) Use the **Data** Menu to open the Data file (.csv), the Trace Log file (.txt) or the Error Log (.log), or change the Default Data Directory or Data file tag
- 7) The default directory for the performance Data File, Error Log and Trace Log files is the “My Documents” or “Documents” folder for the current user.
 - a) The absolute path for this varies by Windows installation and version, but it is typically:
Windows XP: **C:\Documents and Settings\<UserName>\My Documents\Visual Studio 2005\Projects**
Windows Vista: **C:\Users\<UserName>\ Documents\Visual Studio 2005\Projects**
 - b) View these files using the **Data** menu or your program of choice.
 - c) Change the persistent directory for these files as desired also using the **Data** menu.
- 8) The performance Data File is a comma-delimited ASCII text file with a comma-delimited header whose fields are descriptive of data to be written in corresponding portions of each subsequent data record (row). The current data file name is **VWBData_XXXXXXXX.csv**, where **XXXX** is user configurable. The current file name is shown by the **Data File** item in the **File** menu. The suffix of this file name (**_XXXXXX**) may be adjusted using the **Set User Datafile Tag** in the **File** menu to is the current day (all evaluated at G.M.T.). If this file doesn't exist, it will be created. If it does exist, it will be appended to. A data record (row) is written to this file each time a processing operation is executed. If **Cycle Count** is set higher than 1, a data record will be written to this file at the end of each execution cycle. Each data row/record contains extensive configuration/traceability information to document the test, along with key timing information. A new performance data file will be created for sessions occurring on new days at G.M.T.—this helps manage data and prevent the data the file from getting very large if extensive testing is done.
- 9) The **Error Log** file is an ASCII text file named **VWBError.log** that records handled errors.
 - a) If this file doesn't exist, it will be created. If it does exist, it will be appended to as needed.
 - b) The error log file can be opened directly from the **Error Log** item in the **File** menu. It can also be accessed external to CVWB with any text editor
- 10) The **Trace Log** file is an ASCII text file named **VWBTrace.txt** that records explicitly coded function call entries, exits and other status messages, along with precision timing information.
 - a) The formatting gives a sense of the program structure and flow, while the timing is useful for optimization and trouble shooting.
 - b) By default, this logging is enabled. But it can be disabled by clearing the associated checkbox in the **Data** menu.
 - c) If this file doesn't exist, it will be created. If it does exist, it will be cleared at the beginning of each CVWB session and appended to throughout the session as needed.
 - d) The Trace Log file can be opened directly from the **Trace Log** item in the **Data** menu. It can also be accessed external to CVWB with any text editor

- e) Note: If you plan a session with a very large number of tests (perhaps 10's of thousands of cycles or more in a single session), this file might get unwieldy in size, so you might want to disable this option.
- 11) The **Kernel Config** menu brings up dialog boxes to configure, load or save matrices associated with convolution or correlation operations: ~ self explanatory. Note that changes won't be available for processing until the "Commit" button within those dialog boxes is clicked.
- 12) Press Alt key to highlight Menu Hotkeys
- 13) The session timer in the Title Bar is simply an elapsed time indicator to show how long since the current CVWB session began. Double-Click on the timer to reset it to zero and restart it.



Brief Description of CVWB Operators

3x3 Convo, 5x5 Convo, 7x7 Convo: 3x3, 5x5 and 7x7 arbitrary, non-separable convolutions. Default version is Gaussian blur, but the structuring matrix is user configurable with dialogs accessed from the Kernel Config menu.

3x3 Median, 5x5 Median, 7x7 Median: 3x3, 5x5 and 7x7 median filters using a selection-sort based median computation, except for CUDA versions, which are using a binary search method.

3x3 Min, 5x5 Min: 3x3 and 5x5 tile minimum operator replacing each pixel with the minimum pixel value in the 5x5 neighborhood. This can be used for image erosion.

3x3 Max, 5x5 Max: 3x3 and 5x5 tile maximum operator replacing each pixel with the maximum pixel value in the 5x5 neighborhood. This can be used for dilation.

Sobel Mag: 3x3 Sobel filter convolution where normalized magnitude is computed with max value clamped to full scale for the current processing bit depth, and the current lower threshold (from the “*Thresholds*” menu) is applied to the result (set the lower threshold to zero if no thresholding effect is desired).

Sobel Dir: 3x3 Sobel filter where gradient direction is computed in degrees, rectified if less than zero and scaled so that the possible range of (0 - 180 degrees) is proportionally scaled to (0 - full scale) for the current processing bit depth

Subt 2D In – sub: Full 2D subtraction of 1 image frame from the input image (such as for variable 2D offset compensation). Results are clamped at zero. The ‘sub’ image is fixed/canned presently and intentionally introduces some structured noise to the source image (~ the opposite of what it would normally be used for, but useful for demo).

Rms55 Corr, Rms77 Corr: 5x5 and 7x7 tile correlation operators that replace each pixel with the RMS of differences between elements of the structuring matrix and corresponding elements of the source image. Default version is a diagonal high-left to low-right transition, but the structuring matrix is user configurable with dialogs accessed from the Kernel Config menu.

Rotate Corr: Rotate Corr runs a sequence of 7x7 correlation operators representing a high/low transition with an axis rotating sequentially over 360 degrees in increments, where the number of angular intervals is equal to the number of cycles currently set. More cycles will result in more intervals and finer angular increments.

Mult 2D In x mul: Full 2D multiplication of 1 image frame times the input image (such as for variable 2D gain compensation). Results are scaled and clamped to full scale for the current bit depth. The ‘mul’ image is fixed/canned presently and intentionally introduces some structured variation to the source image (~ the opposite of what it would normally be used for, but useful for demo).

Avg In | Out: Replaces the output image with the average of the input image and the preexisting output image.

Input/Output Histo: Computes a 256 bin histogram. This operator runs on whichever image is selected in the main image pane (Input or Output). Results are displayed in a dialog, if this is enabled in the *Options* menu. Histograms computed on the GPU are available for GPU’s with arch 1.2 or later (using SMEM and GMEM atomics)... if GPU is of capability less than 1.2, histograms are still computed when GPU processing is selected, but the multithreaded host/C++ implementation is used.

Thresh ←→: This runs a double-sided threshold on the input image with results to the output image. Pixels with values within the threshold range (between high and low thresh values) are preserved. Pixels with values outside this range are set to zero. The threshold range is set in the “*Thresholds*” menu, and the allowable range is context sensitive to the current processing bit depth.

Subt 2D In – a: Subtracts a single constant value from all pixels in the input image (such as for simple offset compensation). Results are clamped at zero. The constant value “a” is set in the “*Math Constants*” menu.

Mult 2D In x b: Multiplies a single constant value times all pixels in the input image (such as for simple optical gain compensation). Results are scaled and clamped to full scale for the current bit depth. The constant value “b” is set in the “*Math Constants*” menu.

Copy Out > In: Copies all pixel values from the output image to the input image. Using this after each operation is a way to manually iterate through a progression of sequential operations representing a processing pipeline.

Series 6: Runs a series of 6 operations in a daisy-chain fashion, using the output image of 1 operation for the input of the next and returning the final result to the GUI “Output Image”. The particular sequence of operations implemented for demonstration purposes presently is:

Input Image > Mult2D > Subt2D > Median77 > SobelMag > TileMin33 > TileMax3 > Output Image.

If a GPU is used for this computation, the whole sequence is performed on the GPU after the input image is copied to the GPU, and no result image is returned to the host until the whole series is complete. The technique for “daisy chaining” operations in a pipeline like this is of particular interest for GPU processing. In the method illustrated in the underlying OpSeries wrapper functions in CudaVisDLL, the relevance of the time required for copying input data from host to GPU and output data from GPU to host is diminished. Between operations in the sequence, intermediate data is kept on the GPU in Global Memory, so transfer time to/from host becomes less relevant for deeper pipelines. By reviewing the CudaVisDLL code for the *OpSeriesGPU16()* and *OpSeriesGPU8()* functions, it can be seen that this is accomplished while still retaining a highly modular and optimized function library structure. Developers may customize the number and order of operations in the sequence using the simple code patterns presented or may develop their own approach.