# NIST Image Stitching Plugin User Guide

Version 1.1

## 1 Getting Started

This section describes how to install the *NIST Image Stitching* plugin. Section 1.1 describes how to get Fiji and what versions work best with the plugin. Section 1.2 describes how to add the plugin into Fiji.

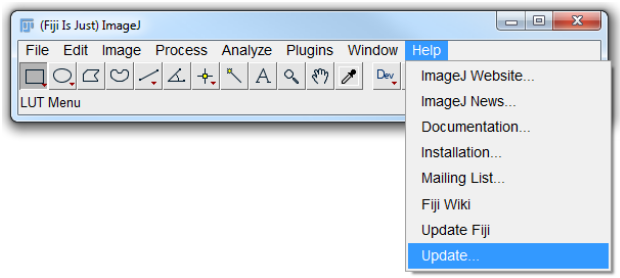
### 1.1 Installing Fiji

To get Fiji go to `http://fiji.sc/Downloads`. If possible, please use the 64-bit version of Fiji. CUDA and FFTW, which get the best accuracy, are only supported in the 64-bit version. The 32-bit version supports only native Java, which is not as accurate. See sections 4 and 5 for more details about FFTW and CUDA.

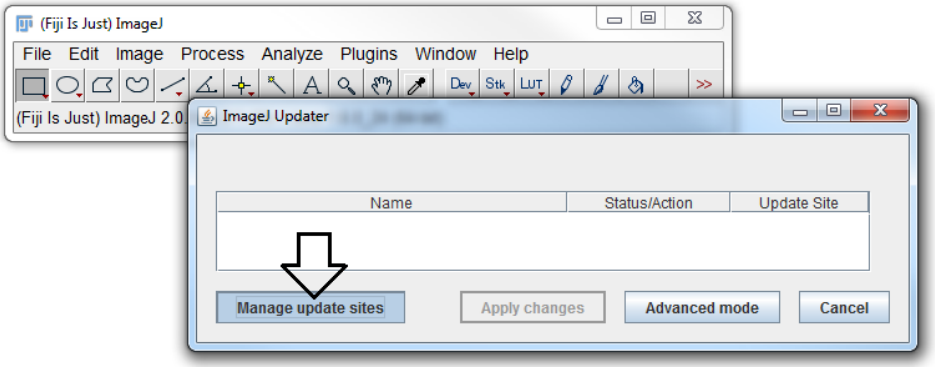
### 1.2 NIST Image Stitching Plugin Installation

Use the following steps to add NIST's site to the list of your Fiji update sites to add the plugin into Fiji.

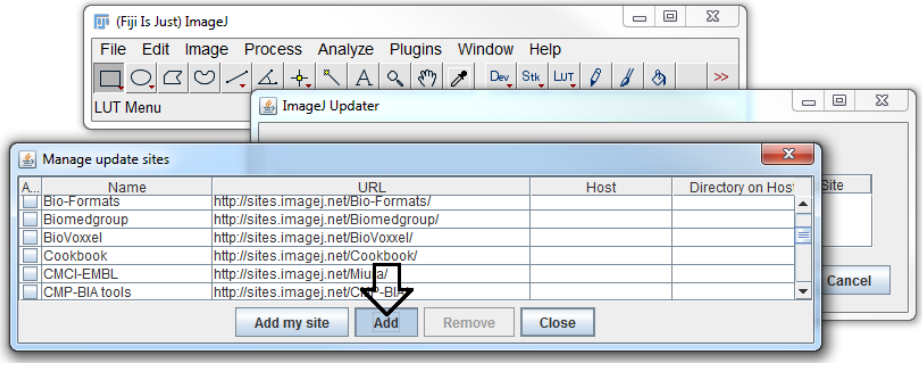
1. Open Fiji
2. Menu item: *Help* → *Update...* (If any updates are available, apply changes first and restart Fiji: Redo steps 1 and 2)



3. Click: Manage update sites

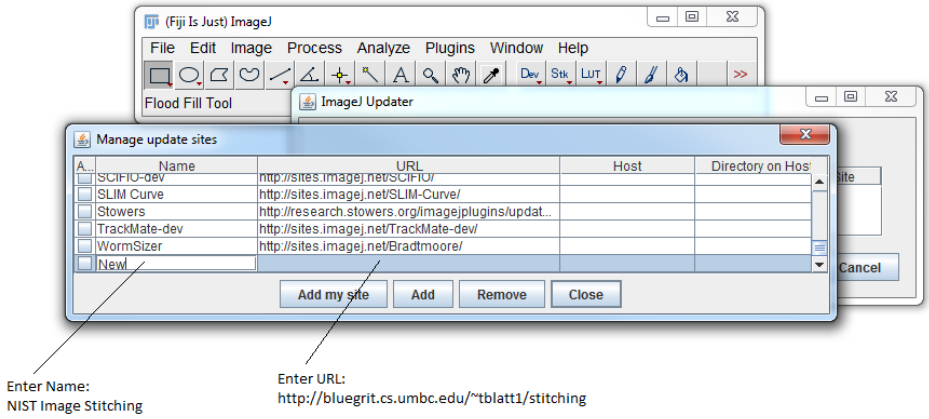



4. Click: Add (This will create a new row)




5. Select name column in new row and type: *NIST Image Stitching*

6. Select URL column in new row and type: `http://bluegrit.cs.umbc.edu/~tblatt1/stitching/`




7. Press Enter and *Apply changes* to install

8. Relaunch Fiji to complete installation

## 1.3 Launching the Plugin

This section assumes you have already installed the plugin using the steps described in section 1.2.

To launch the *NIST Image Stitching* plugin:

1. Open Fiji
2. Menu item: *Plugins* → *Stitching* → *NIST Image Stitching*

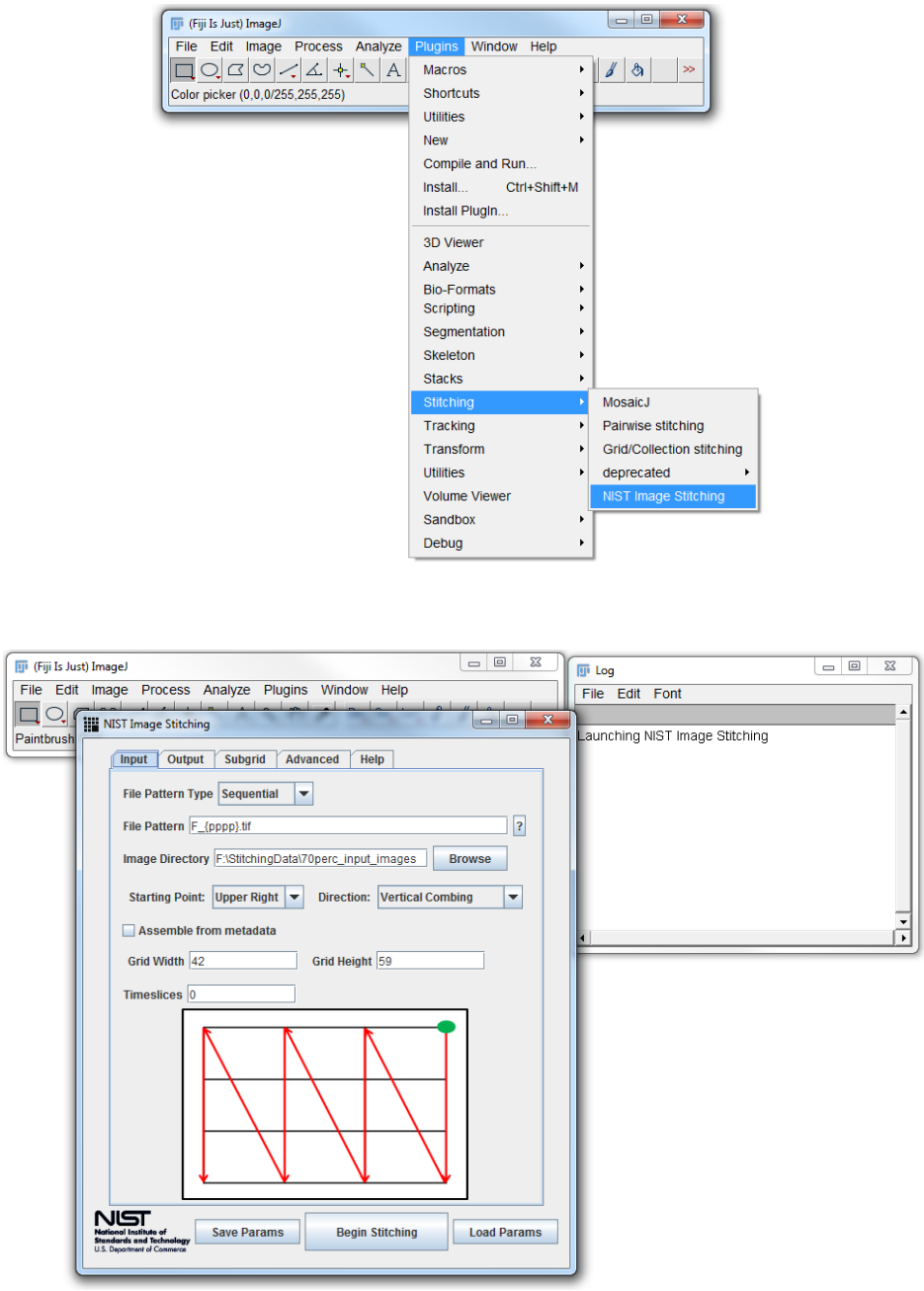



Figure 1: Main GUI Window

# 2 NIST Image Stitching Graphical User Interface

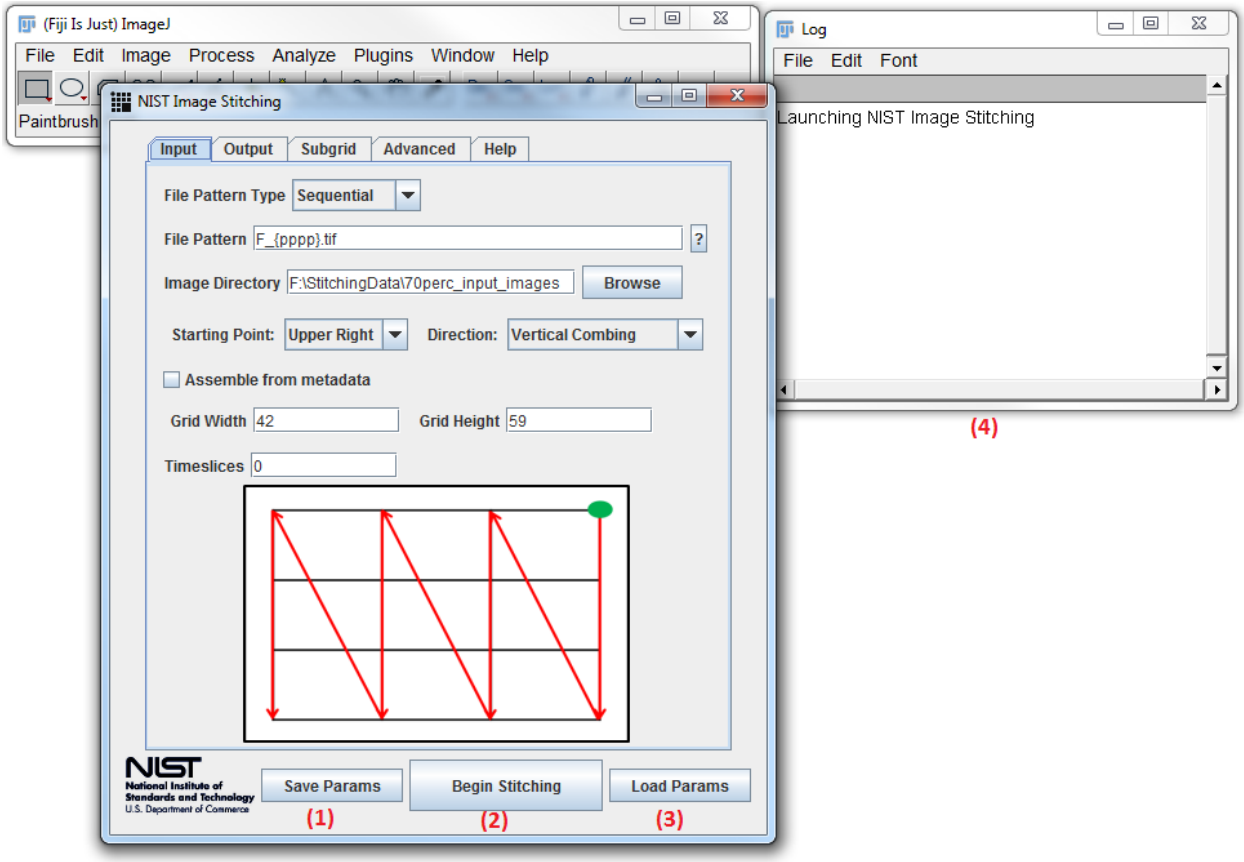



Figure 2: Main GUI Window Parameters

**(1) Save Params** - Saves the current parameters to a file. When clicked, a save dialog will appear.

**(2) Begin Stitching** - Launches the stitching execution. Prior to launching all parameters are checked. Any parameter that fails will appear in red. Also error messages will appear in the log window (see **(4)**)

**(3) Load Params** - Loads parameters from a file. When clicked a load dialog will appear. Users can load the file that was saved from **(1)**, or you can load the statistics file that is generated after every execution. The statistics file is saved in the output directory as described in section 2.2.

**(4) Log Window** - The log window is a text area that contains useful information. Any errors will be appended to the log window.

## 2.1 Input Parameters

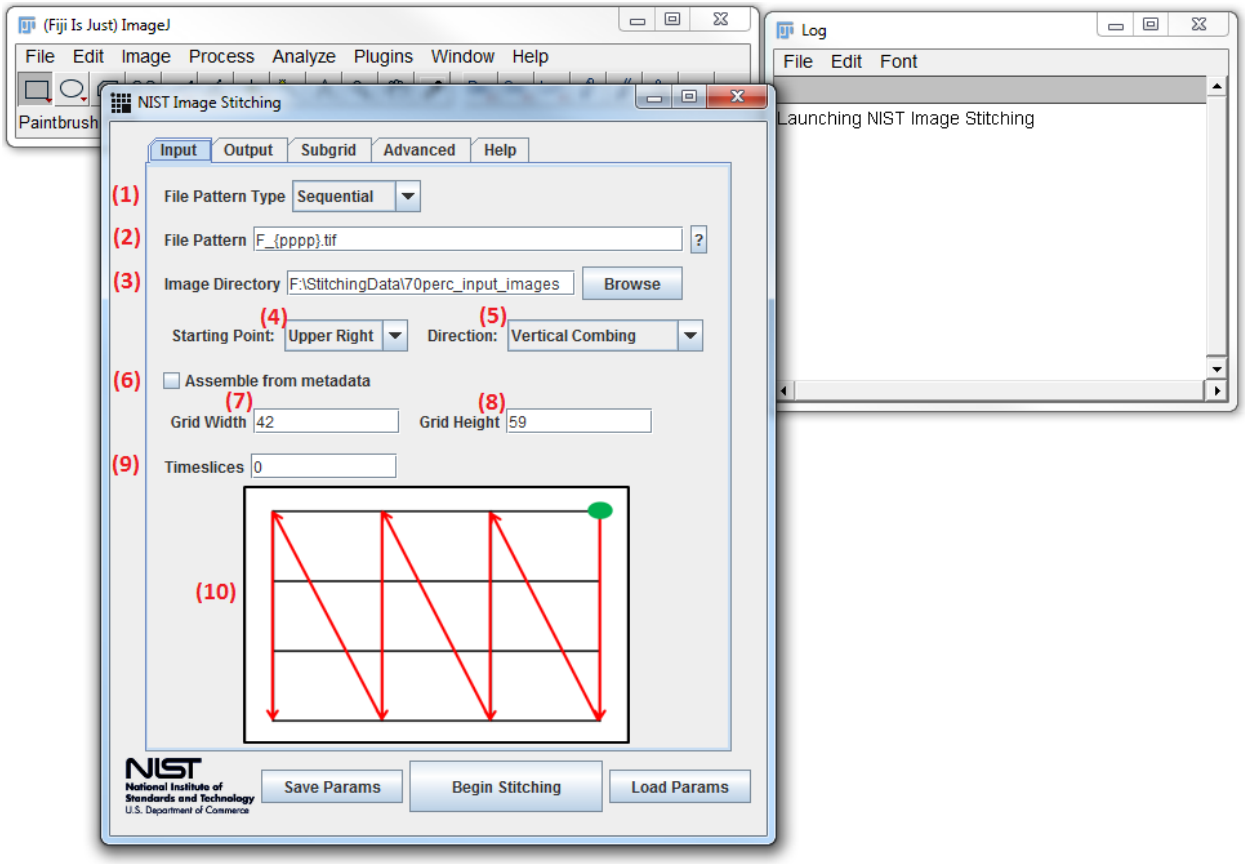



Figure 3: Input Parameters

**(1) File Pattern Type** - Used to specify the type of file pattern.
Possible options: **Sequential** and **Row-Column**
See **(2)** for details on the file pattern types and examples.

**(2) File pattern** - Used to identify specific image files within a directory.

If **Sequential** is selected from **(1)**, then the file pattern is modified to expect sequential style numbering.
Example 1: Img_pos001_time001.tif = Img_pos{ppp}_time{ttt}.tif
Example 2: Img_pos0001_c01.tif = Img_pos00{pp}_c01.tif
**{ppp}** - Special text that represents position numbering between 0 and 999. Increase the number of p's to represent larger numbers.
**{ttt}** - Special text that represents timeslice numbering between 0 and 999. **{ttt}** can be used for z stacks as well as timeslices. Each value of **{ttt}** will be stitched independently. Increase the number of t's to represent larger numbers. Specifying time slices is optional.

If **Row-Column** is selected from **(1)**, then the file pattern is modified to expect row-column style numbering.
Example 1: Img_row01_col01_time01.tif = Img_row{rr}_col{cc}_time{tt}.tif
Example 2: Img_r1_c1_channel01.tif = Img_r{r}_c{c}_channel01.tif
**{rr}** - Special text that represents row numbering between 0 and 99. Increase the number of r's to represent larger numbers
**{cc}** - Special text that represents column numbering between 0 and 99. Increase the number of c's to

represent larger numbers.
**{ttt}** - Special text that represents timeslice numbering between 0 and 999. **{ttt}** can be used for z stacks as well as timeslices. Each value of **{ttt}** will be stitched independently. Increase the number of t's to represent larger numbers. Specifying time slices is optional.

**(3) Image Directory** - The directory where the images are located. Select browse to open a directory browser.

**(4) Starting Point** - The origin microscope stage positioning. Possible options: **Upper Left**, **Upper Right**, **Lower Left**, and **Lower Right**. Modifying this field updates the image shown in **(10)**.

**(5) Direction** - The direction the microscope stage first moves when acquiring image tiles. Possible options: **Vertical Combing**, **Vertical Continuous**, **Horizontal Combing**, and **Horizontal Continous**. Modifying this field updates the image shown in **(10)**.

**(6) Assemble from metadata** - Generates the image mosaic using metadata from a previous run.
**Important**: *You must specify the meta data directory from the previous run in the Output tab. See section 2.2 for more details.*

**(7) Grid Width** - The number of images in a row. (The number of columns, the width of the image grid)

**(8) Grid Height** - The number of images in a column. (The number of rows, the height of the image grid)

**(9) Timeslices** - The number of timeslices to stitch. Leave this field blank to stitch all timeslices. To stitch timeslices you must add the special format **{ttt}** to the file pattern shown in **(2)**. The field supports a comma separated list and/or a range using a '-'.
Example: 1-25,35,45 = stitches timeslices 1 through 25, 35, and 45. Also can be associated with z-stacks.

**(10) Stage Movement Image** - This image provides a graphical representation of the starting point and direction associated with the microscope stage movement.

## 2.2 Output Parameters

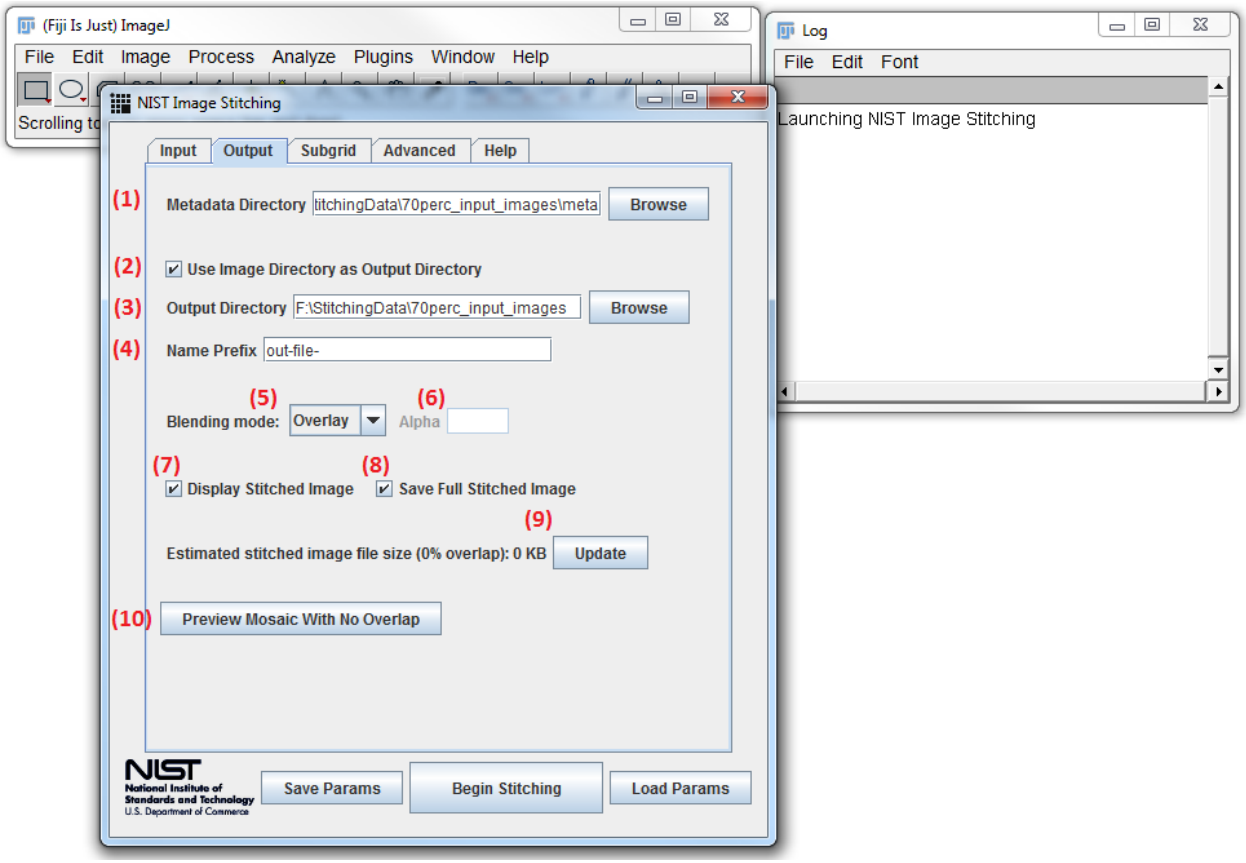



Figure 4: Output Parameters

**(1) Metadata Directory** - The directory where metadata is written.
Metadata generates the following files:

- **global-positions-<#>.txt** - the position data for each tile. This file is used when assembling from metadata.
- **relative-positions-<#>.txt** - the relative displacements between neighboring images.
- **relative-positions-no-optimization-< # >.txt** - the relative displacements between neighboring images without any optimization. Used for debugging purposes.

<#> - represents the timeslice number.

**(2) Use Image Directory as Output Directory** - Checkbox to indicate to use the image directory as the output directory. Checking this will modify **(3)**.

**(3) Output Directory** - Specify the output directory where the full stitched images and stitching statistics are saved. Stitching statistics provide relevant debugging information and can also be used with the **Load Params** button.
Files are saved in the format:

- $<name_prefix>$full-image-<#>.tiff
- $<name_prefix>$statistics.txt.

<# > - represents the timeslice number.
<*name_prefix*> - the prefix given to file names found in **(4)**.

**(4) Name Prefix** - The prefix name given to output files. Files that will be overwritten throws a warning. To remove the warning change the **Name Prefix** to a unique value. Value must contain valid file name characters.

**(5) Blending mode** - Selects the blending mode to be used for composing the full stitched image.
Blending types:

- **Overlay** - Choose only one pixel from overlapping pixels based on highest accuracy.
- **Linear** - Smoothly alters the intensity of the overlapping area between images. Smoothness is determined by Alpha (more details in **(6)**).
- **Average** - Computes the average intensity for each image.

**Warning**: *RGB images may cause out of memory issues with linear and average blending for large images. When composing large RGB images, it is encouraged to use overlay blending.*

**(6) Alpha** - Only used with linear blend. The alpha value controls how much weight is given to pixels near the edge of the image in the overlap region. An alpha value of 0 results in an average blend. The higher the alpha value the lower the weight of the pixels near the edge of the image. Valid values are $\geq 0$; however practically alpha should not exceed 5.

**(7) Display Stitched Image** - Blends and displays the full stitched image into Fiji.

**(8) Save Full Stitched Image** - Blends and saves the full stitched image to disk. The file name is in the format: **(Name Prefix)**full-image-<#>.tiff
<# > - represents the timeslice number.

**(9) Update** - Button to update the estimated stitched image file size assuming 0% overlap. This button will attempt to parse all input fields and subgrid fields.

**(10) Preview Mosaic With No Overlap** - Assembles the input and subgrid parameters to preview the full stitched image assuming no overlap. This method is often used to analyze a grid quickly and to test input parameters.
**Warning**: *A large grid of images may cause out of memory issues.*

## 2.3 Subgrid Parameters

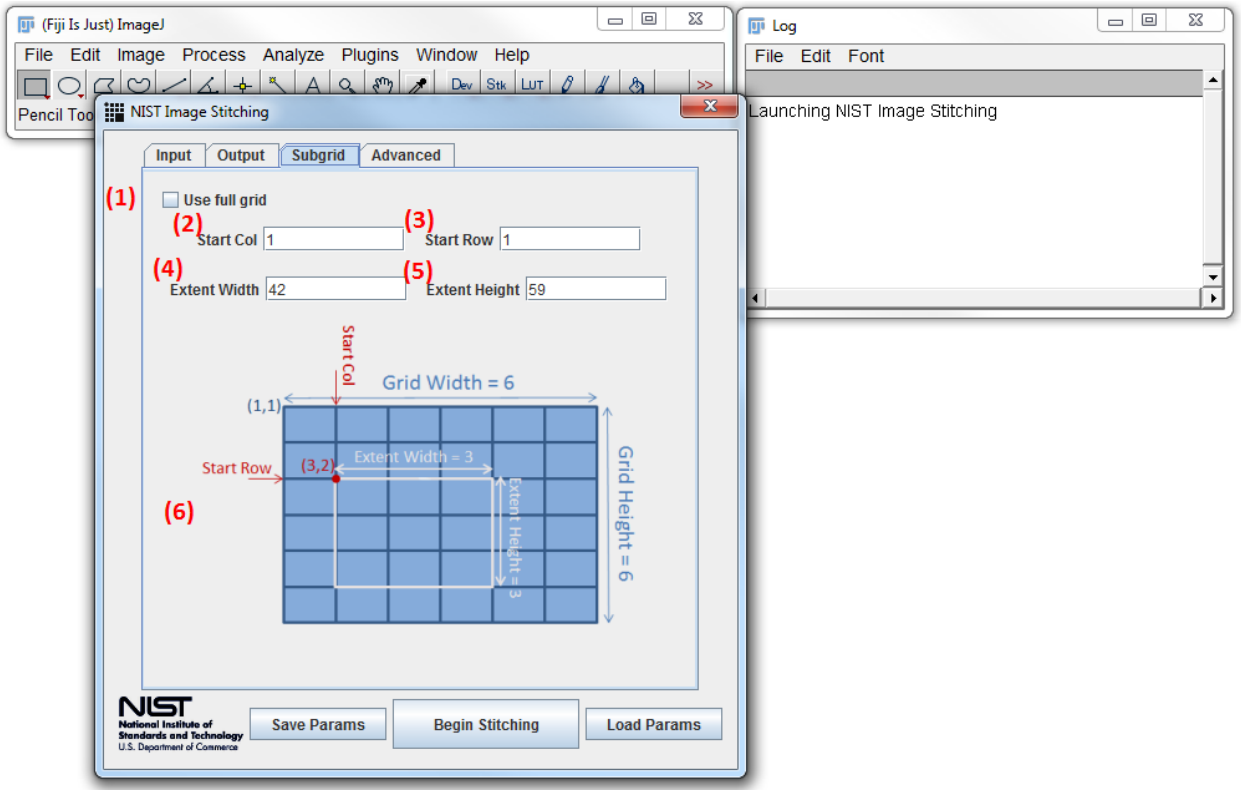



Figure 5: Subgrid Parameters

**(1) Use full grid** - Updates the subgrid to use the full grid of images. Checking this option will update **(2)**, **(3)**, **(4)**, and **(5)**

**(2) Start Col** - Specifies the start column for the subgrid. Refer to **(6)** for a visual representation.

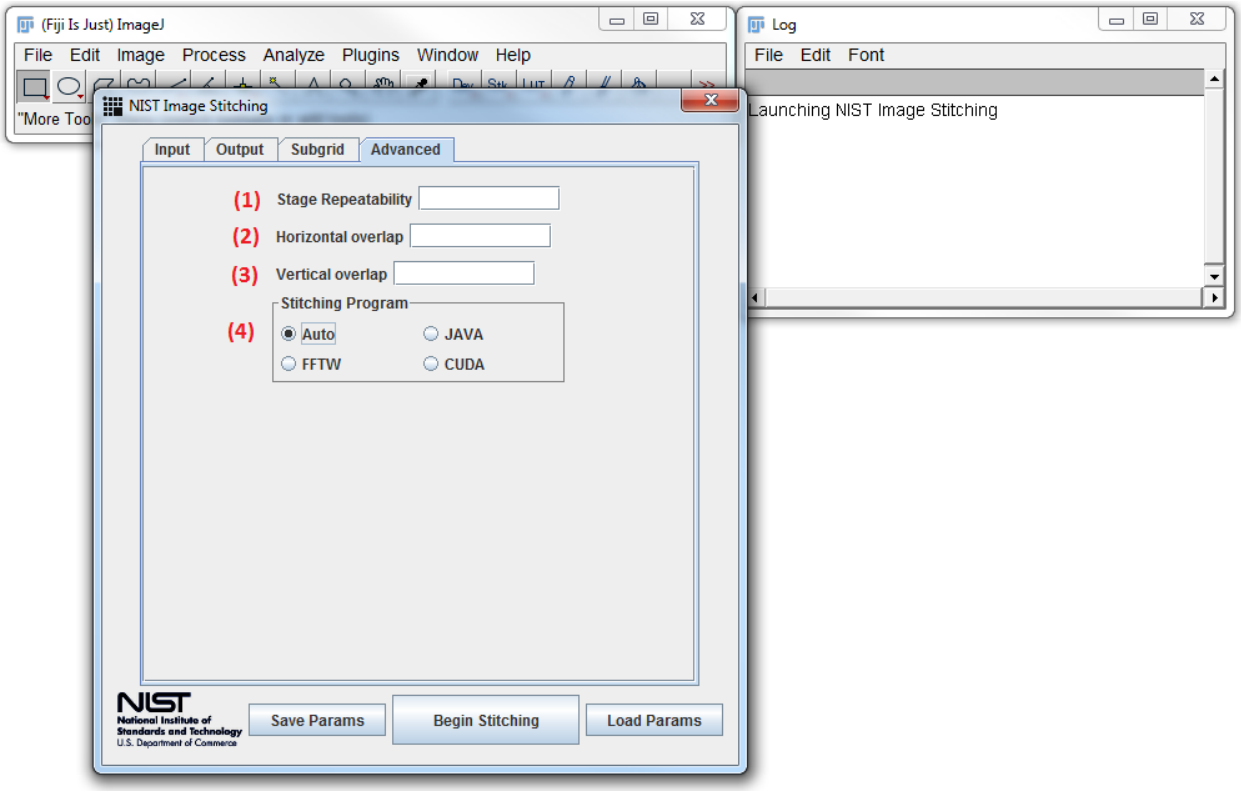
**(3) Start Row** - Specifies the start row for the subgrid. Refer to **(6)** for a visual representation.

**(4)Extent Width** - Specifies the width (number of columns) for the subgrid. Refer to **(6)** for a visual representation.

**(5) Extent Height** - Specifies the height (number of rows) for the subgrid. Refer to **(6)** for a visual representation.

**(6) Subgrid Image** - Provides an example of subgrid parameters

## 2.4 Advanced Parameters (Optional)




**(1) Stage Repeatability** - During optimization, uses the user-specified repeatability of the stage. Leave this field blank to use the default. This value represents the uncertainty that the microscope stage has related to the mechanics that move the stage. Setting this value may increase the search space that is used to find the correct translation between neighboring images.

**(2) Horizontal overlap** - During optimization, uses the user-specified horizontal overlap for computing the repeatability of the stage. Leave this field blank to use the default. Modifying this field will aid in correcting translations that have low correlation in the horizontal direction. By default we compute the horizontal overlap based on the translations of the highest correlated tiles.

**(3) Vertical overlap** - During optimization, uses the user-specified vertical overlap for computing the repeatability of the stage. Leave this field blank to use the default. Modifying this field will aid in correcting translations that have low correlation in the vertical direction. By default we compute the vertical overlap based on the translations of the highest correlated tiles.

**(4) Stitching Program** - Select which type of program to execute. Refer to section 2.5 for more details.

## 2.5 Advanced Parameters - Stitching Program

Select which type of execution to run. Possible options:

- Auto - Automatically determines the stitching program to run. First checks for FFTW, if FFTW fails, then uses the JAVA version.
- JAVA - Executes using the native Java-only execution. This version uses single precision and is not as accurate as FFTW and CUDA.
- FFTW - Executes using native library bindings to FFTW. For a guide to install FFTW refer to section 4.
- CUDA - Executes using native library bindings to CUDA. For a guide to install CUDA refer to section 5.

### 2.5.1 JAVA Execution Parameters

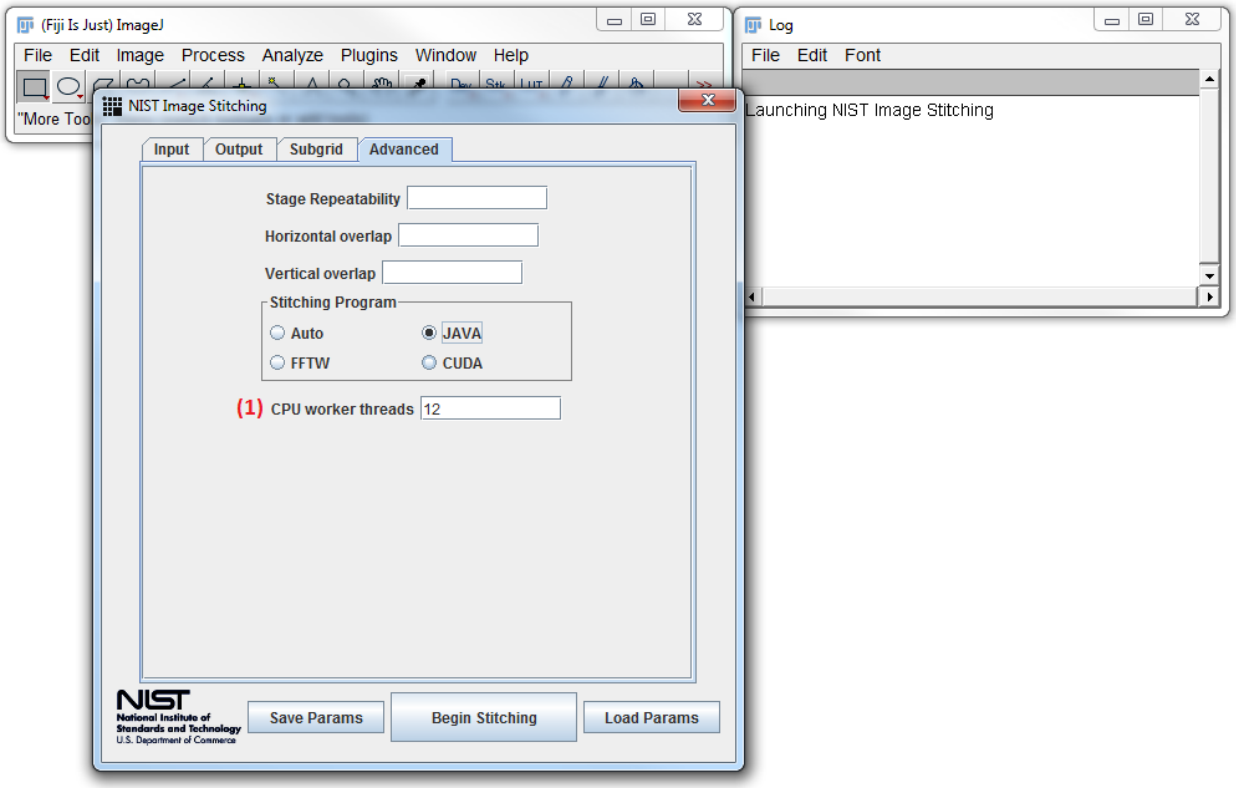



Figure 6: Stitching program Java

(1) **CPU worker threads** - Specifies the number CPU threads to execute the stitching in parallel. The Java version uses 32-bit precision, so the results are not as accurate as FFTW and CUDA.
**Warning:** *Executing with a large number of threads can cause a system to become slow/unresponsive. Specify fewer threads if you need your system to be more responsive.*

### 2.5.2 FFTW Execution Parameters

FFTW (Fastest Fourier Transform in the West `http://fftw.org`) is a C implementation out of MIT that computes the discrete Fourier transform. It uses non-Java native libraries and therefore requires libraries to be loaded at run-time. Currently our plugin only supports Windows 64-bit machines out of the box; however, installation instructions for Linux and Mac OS can be found in section 4.

NOTE: FFTW must generate a FFTW plan for a given image dimension. The plan is used to compute the Fourier transform for images. It can be saved, so it will only need to be generated once for a particular image dimension. The plan file can be loaded for subsequent runs to eliminate the time to generate the plan.

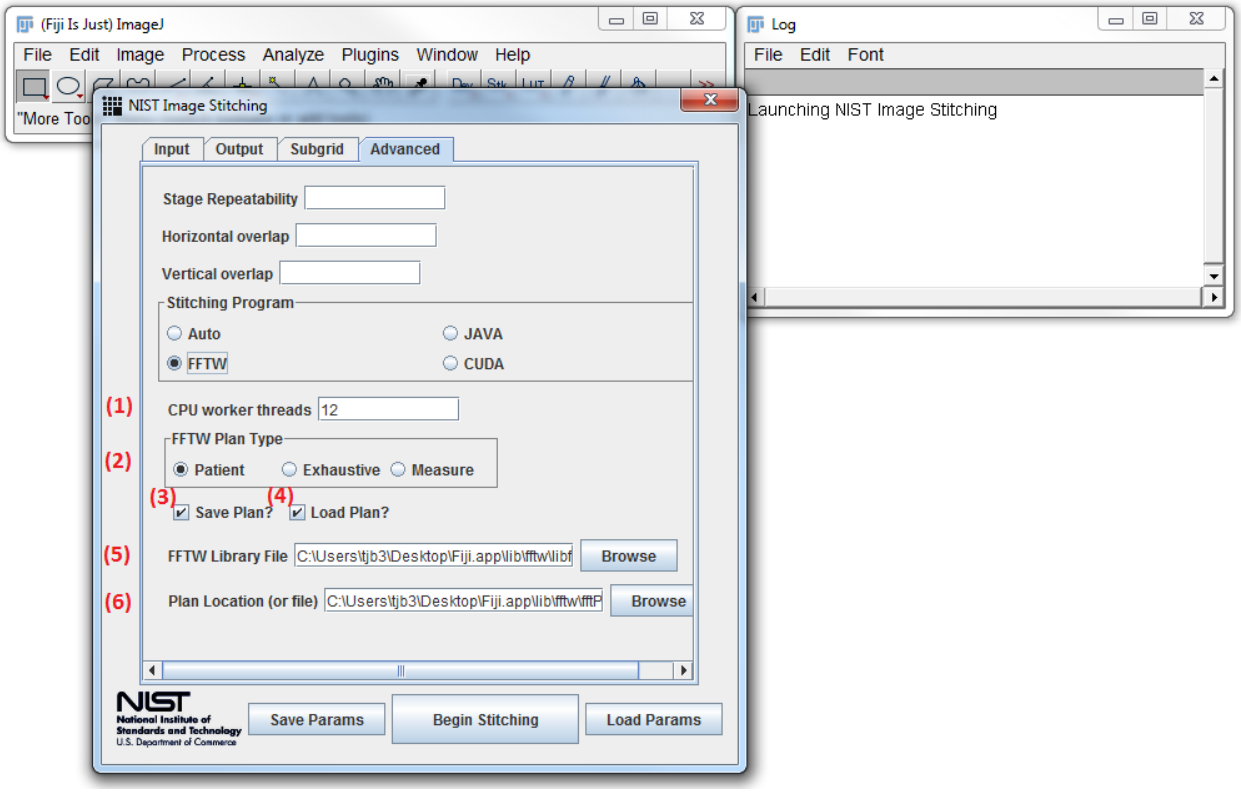



Figure 7: Stitching program FFTW

**(1) CPU worker threads** - Specifies the number CPU threads to execute the stitching in parallel.
**Warning:** *Executing with a large number of threads can cause a system to become slow/unresponsive. Specify fewer threads if you need your system to be more responsive.*

**(2) FFTW Plan Type** - Specifies the type of FFTW plan to generate.
Plan types:

- Patient - Used to get excellent performance, but at the cost of long plan creation time.
- Exhaustive - Used to get best performance, but at the cost of very long plan creation time. In our experiments, we found patient and exhaustive plans to yield the same performance for our data sets.
- Measure - Used to get a basic plan thus yielding the lowest performance, but has very short plan creation time.

**(3) Save Plan?** - When enabled, saves the FFTW plan to file. The plan is saved in the Plan location path, please refer to **(6)** for more details.

**(4) Load Plan?** - When enabled, attempts to loads the FFTW plan to file. If the plan is unable to be found, then a plan is created. The plan is loaded from the Plan location path, please refer to **(6)** for more details.

**(5) FFTW Library File** - Specifies the FFTW library file. This file is packaged with the plugin for Windows 64-bit. To generate the FFTW library file please refer to section 4.

**(6) Plan Location (or file)** - Specifies the FFTW plan location or file. This location/file is used when loading the plan. It is also used as the path for saving plans

### 2.5.3 CUDA Execution Parameters

CUDA is the Compute Unified Device Architecture and is a language developed by NVIDIA. It is used to execute code on NVIDIA GPUs to obtain high performance. In order to use this library you need to have an NVIDIA GPU installed on your system and have the CUDA toolkit version 5.5 installed. For details on how to install CUDA refer to section 5.

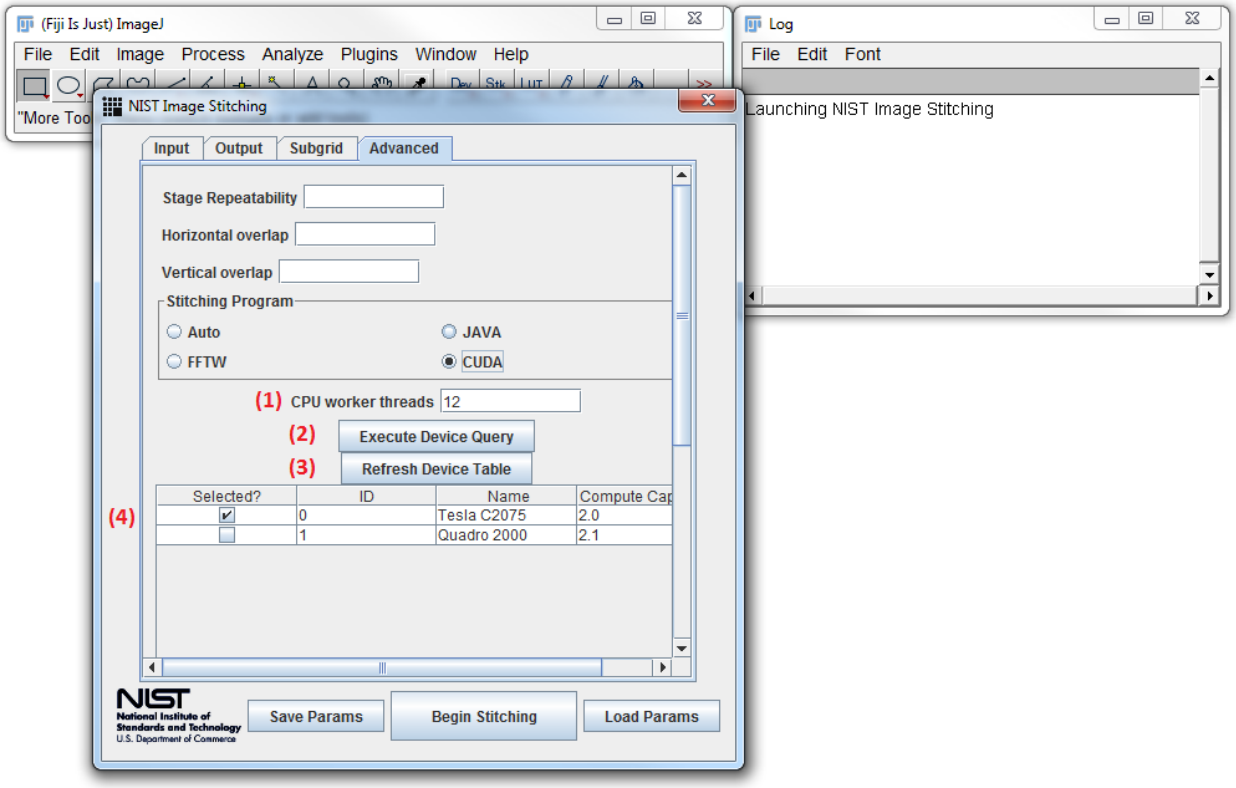



Figure 8: Stitching program CUDA

**(1) CPU worker threads** - Specifies the number CPU threads to execute the stitching in parallel.
**Warning:** *Executing with a large number of threads can cause a system to become slow/unresponsive. Specify fewer threads if you need your system to be more responsive.*

**(2) Execute Device Query** - Queries the machine and prints statistics about all GPUs found.

**(3) Refresh Device Table** - Forces the table **(4)** to query all GPUs again and re-add them to the table.

**(4) GPU Table** - Displays all available GPUs found on the machine. To use one or more GPUs, click the check box in the **Selected?** column for the GPU(s) that you wish to use.
*NOTE: GPUs have limited memories, it is recommended to use GPUs with at least 1 GB of graphics memory. Larger data sets might require more than 1 GB of graphics memory.*

# 3 Stitching Status Frame

After launching the stitching application. A *Stitching Status Frame* is created that describes the status and progress of the application.

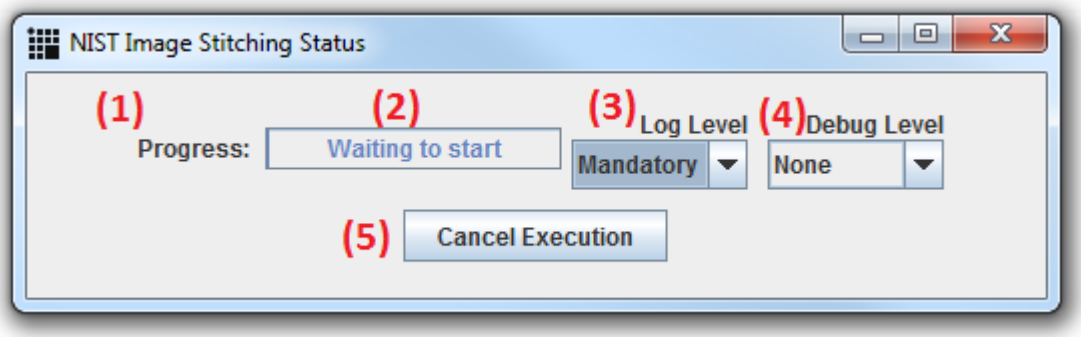



Figure 9: Stitching status frame

**(1) Progress** - Text that describes the progress of the stitching application. If time slices are used, the progress is updated to show which timeslice the execution is currently running.

**(2) Progress Bar** - Displays the progress for various stages in the stitching execution.

**(3) Log Level** - Represents the logging level for the execution. Changing the log level will increase the amount of output shown in the log window.
Types of log levels:

- None - Shows no logging information
- Mandatory - Shows logging information that is important. Example: The current execution stage.
- Helpful - Shows logging information that is useful. Example: relative displacements
- Info - Shows logging information that is informative. Example: What index did we find before computing the cross correlation.
- Verbose - Shows all logging information.

**(4) Debug Level** - Represents the debugging level for the execution. Changing the debug level will increase the amount of output shown in the log window.
Types of debug levels:

- None - Shows no debugging information
- Mandatory - Shows debugging information that is important.
- Helpful - Shows debugging information that is useful.
- Info - Shows debuggingging information that is informative.
- Verbose - Shows all debugging information.

**(5) Cancel Execution** - Cleanly cancels the execution ensuring that resources are freed. Closing the status frame will also cancel the execution.

# 4 Installing FFTW

FFTW (Fastest Fourier Transform in the West, `http://fftw.org`) is a C implementation out of MIT that computes the discrete Fourier transform. The plugin loads the library at run-time and uses the high performance FFTW functions in Java.

Currently only 64-bit Windows is supported out of the box. So users that use 64-bit Windows and the 64-bit version of Fiji will be able to use FFTW straight away.

Due to memory limitations of 32-bit operating systems, we currently do not support using FFTW in 32-bit operating systems. Users with 32-bit operating systems can use the Java Stitching version (See section 2.5.1).

## 4.1 FFTW Installation on Unix (64-bit Mac OSX/Linux)

1. Download fftw-3.3.4.tar.gz from `http://www.fftw.org/download.html`. Currently we have only tested with version 3.3.4 of FFTW, but other versions may work.
2. Extract the source code to some directory (tar xzvf fftw-3.3.4.tar.gz)
3. Navigate into the new directory that was created during extraction.
4. From a terminal execute the following commands:

   ```
   ./configure --prefix=PATH_TO_FIJI.APP --enable-shared
   make
   make install
   ```

   For more details: `http://www.fftw.org/fftw3_doc/Installation-on-Unix.html`
5. After running *make install*, the necessary libraries should now be in '*PATH_FIJI.APP*/lib'. If configure failed, then you may need some additional compilers that are required by FFTW.
6. Load the *NIST Image Stitching* plugin and specify the FFTW library file in the FFTW execution parameters, refer to section 2.5.2 for details. In Linux the library file is named *libfftw3.so* (or similar). In Mac OSX the library file is named *libfftw3.dylib* (or similar).
7. You should now be able to start stitching with FFTW.

# 5 Installing CUDA

CUDA is the Compute Unified Device Architecture and is a language developed by NVIDIA. It is used to execute code on NVIDIA GPUs to obtain high performance. In order to use this library you need to have an NVIDIA GPU installed on your system and have the CUDA toolkit version 5.5.

Due to memory limitations of 32-bit operating systems, we currently do not support using CUDA in 32-bit operating systems. Users with 32-bit operating systems can use the Java Stitching version (See section 2.5.1).

## 5.1 CUDA Toolkit Installation on Windows 64-bit

1. Download CUDA toolkit v5.5 for the appropriate version of Windows 64-bit `https://developer.nvidia.com/cuda-toolkit-55-archive`
2. Launch the executable and complete the installation
3. Restart/Open Fiji

4. CUDA option should be available in the *NIST Image Stitching* plugin.

For more details on how to install the CUDA toolkit for Windows please refer to `http://docs.nvidia.com/cuda/cuda-getting-started-guide-for-microsoft-windows/index.html#axzz34MbYZuma`

### 5.2 CUDA Toolkit Installation on Mac OSX 64-bit

1. Download CUDA toolkit v5.5 for the appropriate version of Mac OSX 64-bit `https://developer.nvidia.com/cuda-toolkit-55-archive`
2. Follow the instructions found here: `http://docs.nvidia.com/cuda/cuda-getting-started-guide-for-mac-os-x/index.html#axzz34MbYZuma`
3. Restart/Open Fiji
4. CUDA option should be available in the *NIST Image Stitching* plugin.

### 5.3 CUDA Toolkit Installation on Linux 64-bit

1. Download CUDA toolkit v5.5 for the appropriate version of Linux 64-bit `https://developer.nvidia.com/cuda-toolkit-55-archive`
2. Follow the instructions found here: `http://docs.nvidia.com/cuda/cuda-getting-started-guide-for-linux/index.html#axzz34MbYZuma`
3. Restart/Open Fiji
4. CUDA option should be available in the *NIST Image Stitching* plugin.

## 6 Final Remarks

Special thanks ... Coming soon

## 7 FAQ

Coming soon

My file format does not match the file pattern syntax, what do I do?

How do I stitch time series when time series are in different folders?