**Analyzing FLIR Images to retrieve accurate pixel level temperature**

Written by Saleem Ullah, Dar A. Roberts, and Samuel W. Fall

**Summary:**

This Manual describes the general procedure for retrieving accurate temperature from FLIR images at a pixel level using the FLIR GUI Project\_v1.0 or FLIR Image Batch Processing GUI batch processing application in Matlab. Currently, this gui only works in Matlab\_R2015b due to the tree classification structures that are not supported in later version. The FLIR GUI Project\_v1.0 is a constellation of functions (written in MATLAB) that have been integrated into a graphical user interface (gui), which allows batch processing analysis of multiple images. The main two files that form the FLIR GUI application are the FLIR\_Proj\_gui.m and the FLIR\_Proj\_gui.fig files. Using these two files and the other support files in the FLIR GUI folder, the application allows the user to calculate temperatures from raw FLIR images. FLIR is capable of acquiring and saving images in both TIR and visible mode simultaneously (Thermal Fusion Mode). The procedure described starts with an input FLIR image/s (e.g. IR\_1599.dat) and outputs images, figures, and comma separated value files (.csv) for each image.

The user only needs to load the images, load a classification tree, select the destination folder for the output data and then click run. However, following is a high level description of the underlying functions used by the FLIR Image Batch Processing GUI application. Some of the original function files listed below can be found in the FLIR GUI folder, while others have been consolidated into the FLIR\_Proj\_gui.m file at the time the gui was created.

The first function (**irFileOpen**) reads the image files in, then populates a structure that contains the header information. The next function (A Matlab Script called **change\_output.m**), then subsets the original fused image into a 160x120 pixel subset centered on the image. This is the best co-registered portion of the scene. Finally, to calculate a black body image (assumed emissivity of 1.0), **integer2temp.m** is used. To calculate pixel-based emissivity and temperature we use an adaptation of the original code written to classify CCD images from the Nikon Coolpix, with “tree” adapted to the FLIR visible camera. This function is called **gui\_image\_intro\_batch\_flir.m** (this file was consolidated and no longer in present as a stand alone support file). This function creates an emissivity classified jpeg.

The next step is to generate a series of exitance images, including one with an assumed emissivity of 0.95 that includes a downwelling irradiance correction, the other using a pixel-specific estimate (also corrected for downwelling irradiance). For the latter, an emissivity image is created from the tree classified product assigning **0.98** to green vegetation, **0.94** to NPV and **1.0** to shade. Exitance images are then converted to temperature images by inverting Stefan-Boltzmann’s equation (M=). This latter process is all completed in T**emp\_Exit\_Correction.**

The following is the sequence of functions.

**Integ2Temp**

**chang\_output**

**irFileOpen**

**Temp\_Exit\_Correction**

**Emiss\_image**

**gui\_image\_intro\_batch\_flir**

**Save\_images**

* **Temp at BB**
* **Exitance at BB**
* **Temp (Emissivity of 0.95 and correcting for DWR)**
* **Exitance (Emissivity of 0.95 and correcting for DWR)**
* **Corrected Temp for both Emissivity and DWR,**
* **Exitance corrected using pixel based emissivity and DWR**

**Before you start**

It is probably best to start by compiling a list of the files you would like to process in a folder. You should also create an “output” folder in that folder to use as a destination for the output data files. The images you will be working with should have been collected using Thermal Fusion mode on the FLIR camera. If a pair of images was collected using a different mode, this program will not work. You will also need an estimate of downwelling longwave (LWD or DWR) collected by the met tower closest in time to the time the image was collected. Thus, at a minimum you will need:

Location (Transect, Quadrat), Time, Image Name, LWD (Again Note: LWD is the same as DWR mentioned later).

The FLIR GUI will output a table filled with the calculated values out for each image. You will want to assemble a spreadsheet like the one in Appendix B (at the end of this document) using the calculated spreadsheet output values and any additional values needed by your analysis. I suggest compiling all additional information prior to starting image analysis.

**Image Analysis Procedure:**

Step 1. Launch Matlab 2015b and select the “FLIR\_GUI\_2016” folder as your current folder.

Step 2. Run “FLIR\_Proj\_gui” in the command line.

Step 3. Click the “Import Images” button and select the images you wish to analyze.

Step 4. Click the “Load Classification Tree” button and select the “tree.mat” file in the current folder. This will load the binary classification tree for classifying NPV, GV, and Shade emissivity areas in each image. This classification tree has been trained to work best on grass and does not work reliably on images of other object types. If you have been instructed to use a different classification tree, load that tree instead.

Step 5. Click the “Choose Save Folder” button. Navigate to the folder you have created to hold the data output file that will be generated. A good place to put this output folder is in the same folder as your original images.

Step 6. Click the “Run Batch” button. This will generate the data output files.

You can now view the processed images by selecting an image from the batch queue list and clicking “Open Selected.” You can close all the images by clicking “Close All.”

For each image a csv file (comma separated file) has been created containing all of the calculated values for that image. In addition, a “Combined\_Output…” csv file has also been created, which lists all calculated values for each image as separate rows.

Lastly, to run a new batch of images, click the “Clear Images” button and then repeat steps 3 through 6.

If you run into a problem such as accidently clearing the value of the DWR field, you can revert to the default values by clicking the “Quit” button in the upper left corner of the app window, and then starting again from Step 2. Note: You should always “Quit” the application, not just close the FLIR Image Batch Processing GUI window. Closing the app window manually will not clear all the variables in memory, and can cause errors to occur when you run the FLIR GUI application again.

**Appendix A:**

The following is a detailed example of the original Matlab workflow for a single image. This is included only to help you understand the details of each of the original processing functions. This is only for educational purposes:

We will work through the analysis of a single image. This is IR\_1599, acquired on February 20, 2015 at 11:20 AM at Coal Oil Point. All analysis is being performed in the 2015\_02\_20 subdirectory. I have also placed all necessary Matlab code and the FLIR specific tree there.

**Detail step by step procedure**

As an example, we will work with an image acquired in Thermal Fusion mode IR\_1599.jpg. This image was acquired on February 20, 2015 at 11:20 AM by Santa Barbara City College (1-3W). Based on the IDEAS website, CG3up was 379.8Wm-2

**1) Strip apart the Visible and TIR images using the function “irFileOpen.m”**

irFileOpen.m function read the header and data from FLIR images and produce a structure file **[fileInfo]** containing TIR data, VISIBLE data and camera setup parameters. For detail read the comments of the **irFileOpen.m** function.

[fileInfo] = irFileOpen(analysisDir,name,fType,savedat)

Example Input using IR\_1599.jpg

[fileInfo]=irFileOpen(pwd, 'IR\_1599.jpg', 'jpg', 'false');

Note, the use of a semi-colon at the end suppresses output to the screen, which can be useful.

**2) Run the script “change\_output.m”.**

This script is changes the output content of previous function (irFileOpen.m). These variable act as an input for the next function **integer2temp**. The key function of this script is to extract the overlapping portion of TIR images (instead of the whole TIR image) and convert this portion to temperature. Key variables used include a width of 160 and height of 120, meaning that only the center subset 160 pixels wide and 120 pixels tall is saved. Make sure to use [FileInfo] in step one, because that is the structure expected by **change\_output.m.**

**3) Run function “integer2temp.m”.**

The output of **irFileOpen.m** acts as an input for **integer2temp.m**. The script **integer2temp.m** reads and converts the raw integer data to temperature. Remember the resultant temperature image at this stage is uncorrected for reflected downwelling, since we are assuming the surface is a blackbody (i.e., R =1-1 =0). The correction for emissivity and down welling radiance (dwr) will be implemented in the upcoming steps. For detail read the comments associated with **integer2temp.m**.

To successfully execute **integer2temp.m**, provide the input variable as

temp\_sub = integer2temp(fileInfo,'false');

**4) Save Images using Imwrite and csvwrite**

This set of commands writes the overlapping Visible and TIR Image (TIR image assuming Black body) to the working directory which is then used in further analysis. Make sure to change the name of the images that are saved into the directory. The visible image already exists in the structure and as the name ***fileInfo.overlayDefaultVisible***. All we have to do is save it.

An example for IR\_1599.jpg is shown below:

Save visible image:

imwrite(fileInfo.overlayDefaultVisible,'vis\_1599.jpg');% Change the name of the Image

Save uncorrected temperature image as a csv file:

csvwrite('Uncor\_temp\_1599.dat',temp\_sub);

**5) Run function gui\_image\_intro\_batch\_flir.m**

This function classifies visible images into three classes (NPV, Shade and GV). Type **gui\_image\_intro\_batch\_flir** in the command window and press enter. An Image Analysis GUI will pop up. Upload the Visible Image which is saved in the previous step. Then Click “Show Classification” or click the “Batch classify” button. Note, for this function to work properly, you also need **gui\_image\_intro\_batch2.m**

For detail see the comments of the functions or the manual of the function.

**6) Run script “Emiss\_image.m”**

Executing **emiss\_image.m** will assign emissivity values to NPV, Shade and GV of the classified images result in previous Step (step 5).The Emissivity Image (emiss) is then used for retrieving accurate temperature at Pixel level. Just type emiss\_image and hit return.

**7) Run script “Temp\_Exit\_Correction.m”**

This Script extracts temperature at three levels.

1) Raw Temperature (considering target as Black Body)

2) Temperature at 0.95 emissivity and applying down welling radiance (dwr) corrections

3) Final Temperature (using Pixel based emissivity and correcting for down welling radiance)

Before running this, you will need one new item (downwelling longwave, **dwr**), the subsetted temperature image ***(temp\_sub***) and the emissivity image (***emiss***)

The function expects the Uncorrected temperature image to be named ***unCorTem***, the emissivity image to be called ***emiss***, and longwave downwelling to be assigned to ***dwr***.

For IR\_1599, we already have ***temp\_sub*** and ***emiss***, so we just need ***dwr***.

Example

unCorTem=temp\_sub;% Raw temperature image.

emiss=emiss;% emissivity image.

The DWR reading can be obtained from IDEAS website of Coal Oil point. The DWR reading close in term of time to the FLIR image would be an ideal one for corrections.

dwr=379.8 % Down welling radiance. It should be change according to the Radiometer reading at time of Image acquisition

After running the script, five figures will pop up. The title of each figure is self-explanatory and shows the level of processing. These figures are temperature and exitance at various processing levels. You can save these if you wish, although the actual data is superior.

You will want to save a number of values from the above analysis. These include emissivity images, exitance images and temperature images at various levels of correction. In addition, you will want to record scene averages, scene maxima and minima.

To save images as CSV files, you would use csvwrite. For IR\_1599.jpg, the outputs would be as follows

**Appendix B:**

**Black body image (Exit\_at\_BB)**

Save image as CSV

csvwrite('BB\_exit\_1599.dat',Exit\_at\_BB);

**Exitance corrected for downwelling assumed emissivity 0.95 (Surf\_exit**)

csvwrite('Surf\_exit\_1599.dat',Surf\_exit);

**Temperature correcting for downwelling (0.95) (Temp\_calcul\_from\_Surf\_exit)**

csvwrite('Surf\_temp\_1599.dat',Temp\_calcul\_from\_Surf\_exit);

**Emissivity image (**emiss)

csvwrite('emiss\_1599.dat',emiss);

**Surface exitance using class emissivity (Surf\_exit\_using\_class\_emiss)**

csvwrite('surf\_exit\_class\_1599.dat', Surf\_exit\_using\_class\_emiss);

**Surface temperature using pixel based emissivity (Surf\_temp\_using\_class\_emiss)**

csvwrite('surf\_temp\_class\_1599.dat', Surf\_temp\_using\_class\_emiss);

You will also want to record various values. This information is stored in the workspace.

Avg\_Exit\_BB (average black body emission)

Exit\_BB (min, max black body exitance)

Surf\_exit (min, max, 0.95 corrected exitance)

Avg\_Surf\_exit (average exitance, 0.95 corrected)

Surf\_exit\_using\_class\_emiss (min, max, pixel based)

Avg\_surf\_exit\_using\_class\_emiss (average, pixel-based)

Scene\_temp\_calcul\_from\_Avg\_exit\_at\_BB (one number)

UnCorTem (Min and max temperature for a black body).

Scene\_temp\_at\_Emiss\_95 (one number, average for 0.95 emissivity)

Temp\_calcul\_from\_Surf\_exit (Min and Max temperature based on 0.95 emissivity)

Scene\_temp\_calcul\_from\_Avg\_surf\_exit\_using\_Scene\_emiss (average using pixel based emissivity)

Surf\_temp\_using\_class\_emiss (Min and Max temperature based on class emissivities)

For IR\_1599 I got the following

Table of values

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Image | Time | LWD | MBB | M0.95 | MClass | TBB | T0.95 | Tclass |
| IR\_1599 | 11:20 | 379 | 394.82 | 375.86 | 385.09 | 288.86 | 289.01 | 288.94 |
| Min |  |  | **386.06** | 367.11 | 365.02 | 287.25 | 287.31 | 287.27 |
| Max |  |  | 402.24 | 383.29 | 402.24 | 290.21 | 290.43 | 290.36 |