1. HRMET: The Basics

1.1. Legalese

HRMET was created by Samuel C. Zipper at the University of Wisconsin-Madison in 2012/2013. Any publications using HRMET should cite:

Zipper, Samuel C., and Steven P. Loheide II. “Using Evapotranspiration to Assess Drought Sensitivity on a Subfield Scale with HRMET, a High Resolution Surface Energy Balance Model.” *Agricultural and Forest Meteorology* In press (2014). doi:10.1016/j.agrformet.2014.06.009.

While we expect HRMET to work well in many environments, users of HRMET should ensure that they properly validate the model and assess its accuracy for their particular needs, especially if the study region is markedly different than that of a Wisconsin cornfield (for example, the Amazonian rainforest, or the Sahara Desert).

The creators of HRMET are not responsible for anything bad or good that may or may not happen to you as a result of using HRMET.

1.2. Background Info

HRMET (High Resolution Mapping of EvapoTranspiration) is a surface energy balance that relies on established biophysical relationships (many from Campbell & Norman’s *An Introduction to Environmental Biophysics*) and an iterative approach to solve the surface energy balance. Net radiation (R) is partitioned into a ground heat flux (G), sensible heat flux (H), and latent heat flux (λET). HRMET was designed to map evapotranspiration (ET) rates at high spatial resolution over agricultural fields in south-central Wisconsin. HRMET does not require a ‘hot’ and ‘cold’ pixel to partition energy into sensible and latent heat fluxes.

HRMET is written in MATLAB. For a detailed description of the model, please see the publication mentioned in Section 1.1.

Sam Zipper’s e-mail address is samuelczipper@gmail.com – contact him with any questions or if you find errors in the code.

2. Running HRMET at a point

HRMET is a point-based model. The core script that carries out the surface energy balance calculations in HRMET is titled `HRMET_shared.m`. If you open `HRMET_shared.m`, the top of the file shows the necessary inputs:

```
[ET_mmHr] = HRMET_shared(datetime, longitude, latitude, Tair, SWin, u, ea,
pa, LAI, h, T, albSoil, albVeg, emissSoil, emissVeg)
```

Each of these variables is described in the code:

```
% Necessary inputs are:
% (1) Site information:
%      -datetime = Date & time in Matlab datenum format
%      -longitude = Longitude, decimal degrees
%      -latitude = Latitude, decimal degrees
%      -albSoil = Albedo of soil, 0-1
%      -albVeg = Albedo of vegetation, 0-1
%      -emissSoil = Emissivity of soil, 0-1
%      -emissVeg = Emissivity of vegetation, 0-1
% (2) Meteorological data:
%      -Tair = Air temperature, degrees celsius
%      -SWin = Incoming shortwave radiation, W/m2
%      -u = Wind speed, m/s
%      -ea = Air vapor pressure, kPa (can be calculated from Tair and
%            relative humidity)
%      -pa = Atmospheric pressure, kPa
% (3) Canopy structure data:
%      -LAI = Leaf Area Index, m2/m2
%      -h = Canopy height, m
% (4) Canopy surface temperature:
%      -T = Canopy surface temperature, degrees celsius
```

Later on in the script, we will discuss how to run HRMET over a grid and vary the inputs to estimate uncertainty, as described in Zipper & Loheide (2014).

To run HRMET for the first time, run it as you would any other function in Matlab:

```
>> [ET_mmHr] = HRMET_shared(735080.45833, 89.38420, 43.2966, 27.25, 731, ...
               1.85, 2.2834, 97.883, 4.04, 2.54, 25.62, 0.105, 0.2, 0.945, 0.94)

ET_mmHr =

    0.7653
```

As you can see, we provide HRMET with the necessary inputs for a point at an instant in time, and it spits out an instantaneous ET rate in [mm hr^{-1}].

2.1. Things you might have to change in HRMET_shared.m

`HRMET_shared.m` is *mostly* ready to go. However, there are a couple variables I have set as "constant" in there that you may need to change:

- `daylightSavings = 1`

  - This variable defines whether it is daylight savings time or not. I have it constantly set to 1, because all my measurements are during the growing season. However, if you are making measurements in the winter, set this to 0.
- `Zair = 3.66`
  - Zair is the height of your air temperature measurement station [m]. Change it to whatever the appropriate value is for your study site.
- `Zu = 9.144`
  - Zu is the site of your wind measurement station [m]. Change it to whatever the appropriate value is for your study site.

3. Running HRMET over a field or grid

3.1. A Simple Example

HRMET calculates the energy budget in one dimension, meaning that there is no interaction between neighboring points. Therefore, running HRMET over any spatial domain is simply a matter retrieving the appropriate input data for each grid cell and plugging it into the `HRMET_shared` function.

To go through a simple example, view the corresponding files:

-`HRMET_Example_CreateInputData.m` makes some input data that can be used to run HRMET

-`HRMET_Example_RunGrid.m` runs HRMET over a grid of input data

-`HRMET_Example_RunGrid_withUncertainty.m` runs HRMET over the same grid, this time demonstrating how to incorporate uncertainty in the estimates of your input parameters.

After running this example, you should be able to substitute in your data and run HRMET on your own.

3.2. An Extremely Detailed Example

In this section, I will go through a detailed example for running HRMET using my typical workflow, mostly as an instruction manual to myself – keep in mind that this is definitely not the only, and probably not the best, way to do things!! The specific images and Matlab files referenced here are not provided.

3.2.1. Collect your data

In this example, I will use the following sources of data:

Variable	Data	Units	Source	Spatial Character
datetime	Date & time of image collection	Matlab datenum	Field notes	Uniform
longitude	Longitude of site	Decimal degrees	ArcGIS	Uniform
latitude	Latitude of site	Decimal degrees	ArcGIS	Uniform
Tair	Air temperature	°C	Arlington AWON	Uniform
SWin	Incoming shortwave radiation	W m^{-2}	Arlington AWON	Uniform
u	Wind speed	m s^{-1}	Arlington AWON	Uniform
ea	Air vapor pressure	kPa	Arlington AWON	Uniform
pa	Amospheric pressure	kPa	Pressure transducer	Uniform
LAI	Leaf Area Index	m^2 m^{-2}	Regressed from VI	Variable
h	Canopy height	m	Regressed from VI	Variable
T	Canopy surface temperature	°C	FLIR imagery	Variable

albSoil	Soil albedo	-	Literature	Uniform
albVeg	Vegetation albedo	-	Literature	Uniform
emissSoil	Soil emissivity	-	Literature	Uniform
emissVeg	Vegetation emissivity	-	Literature	Uniform

3.2.2. Process spatial thermal data

Canopy surface temperature information collected using the Loheide lab's FLIR camera needs several steps of pre-processing before it is ready to be used.

3.2.2.1. Atmospherically correct thermal data

In ThermaCAM Researcher Pro 2.7, open your SEQ file and navigate to the image you want to output. In the 'Image' menu, open 'Image Settings' and go to the 'Object Parameters' tab. Change the values to correspond with the details of your image collection date. After changing these values, hit 'OK'. You should see the temperature scale change. Then, export the data as a .csv file ('Image' menu → 'Save as' → Select file type 'CSV' → 'Save'). Repeat this for all the thermal images you want to export.

3.2.2.2. Convert thermal data to TIFs

Using the Matlab script `MatrixToArcASCII.m`, convert your .csv file into an ASCII. Then, open ArcMap. In the ArcToolbox menu, select 'Conversion Tools'→'To Raster'→'ASCII to Raster'. The input file should be the ASCII you just created. The output raster can be named whatever you want, as long as it ends in '.tif'. Make sure you change the output data type to 'FLOAT', otherwise you will lose all data after the decimal. Repeat this for all your images.

3.2.2.3. Georeference TIFs

Georeference the .tif file you just created. For a basemap, I typically use National Agricultural Imaging Program (NAIP) imagery, which is at high resolution and is released every few years, depending on your county of interest. Repeat this for all your images.

3.2.2.4. Mosaic TIFs and clip study area

At this point, you should have several .tif files with coverage over your whole field. It may look something like this:

The red outline is our field of interest. You can see there are three thermal images covering the field, which look different due to different scales. We want to mosaic these into a single image, and clip it to get rid of data we're not interested in.

- Load all the rasters you want to mosaic together into ArcMap.
- Create a new file geodatabase to work in.
    - Open ArcCatalog tab
    - Navigate to your folder of interest
    - Right click → New folder
    - Right click on New Folder → New File Geodatabase
    - Give it a name (e.g. mosaics.gdb)
- Create a new Raster Dataset that will be your mosaic
    - Right click on your geodatabase in Catalog → New → Raster Dataset (do NOT choose Mosaic Dataset).
    - Enter a dataset name (no extension)
    - Change the Pixel Type to correspond with whatever your input data will be (for FLIR imagery, should be 32_BIT_FLOAT).
    - Leave everything else unchanged, hit OK.
    - Arc will think for a while, then the new Raster Dataset should appear in your geodatabase.
- Add data to the Raster Dataset
    - Right click on the Raster Dataset in ArcCatalog → Load → Load Data
    - Select the rasters you want to mosaic
        - The order in which you have these listed matters. If your output mosaic looks bad, try going back and changing the order here.
        - If you try multiple combinations and none look good (weird triangles), do your blending in two or more phases: first blend together two of your rasters, then do another mosaic option to add in the third.
    - IMPORTANT: Mosaic Operator = BLEND
        - This will smoothly blend the output data in areas where you input rasters are overlapping
    - Enter a NODATA value if you have one (often NODATA=-9999); everything else should be as default
    - Hit OK, Arc Will think for a while and then your raster should appear!
        - Your mosaic may be all gray – if it is, right click on the raster layer in ArcMap Table of Contents → Properties → Symbology → Change stretch type to Standard Deviations (say 'yes' when it asks if you want statistics).
- Clip the raster to your study extent
    - ArcToolbox → Data Management Tools → Raster → Raster Processing → Clip
    - Select the mosaicked raster you just created as your Input Raster
    - Input another shapefile containing the outline of your study site as the Output Extent
    - Check the 'Use Input Features for Clipping Geometry' box
    - Tell it where to save the Output Raster Dataset, this time with a '.tif' extension

Your new, blended, clipped raster should look something like this:

3.2.2.5. Get data from ArcMap into Matlab

There are several ways to get your data from ArcMap into Matlab. However, the way I've found most consistent is creating a grid of points over the field in Matlab, loading that into Arc, extracting the surface temperature at each point, exporting that data as a text file, and then loading it back into Matlab. As you might guess based on the number of steps, this is inefficient from both a computational and labor perspective.

- In Matlab: use the `meshgrid` function to create a grid over your model domain at whatever spacing you desire in the coordinate system & projection that your spatial data is georeferenced in (in this case, I am doing 1 m spacing in UTM Zone 16N projection based on NAD83). Use `reshape` to convert that to a list with two columns, one for 'x' and one for 'y'. Save it as a .csv file with a column header.
- In ArcMap, import the list of XY points.
    - File → Add Data → Add XY Data → Choose your saved .csv file → Tell it which column is your x and y data → Hit OK
- These points will be loaded in as Events, and it may give you a warning message; you'll want to convert it to a shapefile.
    - Right click on the Events layer → Data → Export Data → Tell it where to save the output feature class and select shapefile as the file type.
- Now you have a grid of points covering your entire field. At each one of these points, you want to extract the surface temperature from the underlying clipped mosaic.
    - ArcToolbox → Spatial Analyst → Extraction → Extract Values to Points
    - Put in your new point shapefile as 'Input point features', your clipped surface temperature raster as 'Input raster', and choose a name for your Output point features.
    - Hit 'OK'. Depending on how many points you have, this could take a while.
    - Once the operation is complete, your new shapefile will automatically be added to the table of contents. If you open the Attribute Table, there will be five columns: FID, Shape, x, y, and RASTERVALU (which is the surface temperature data you just extracted).
- Export your new shapefile containing surface temperature as text.

  - Right click on layer in Table of Contents → Open Attribute Table → Click the little arrow next to the white square in the upper left corner → Export → Click the little yellow folder to tell it where to save the output table, and make sure you save as a '.txt' file → Click 'OK'.
- In Matlab: import your text file and use `reshape` to switch it from a list back to the same dimensions as your grid.

3.2.3. Process spatial canopy data

Spatial canopy structure inputs (LAI and h) can be derived from remotely sensed vegetation indices. Using the Loheide lab's Tetracam MCA system, there are several steps that need to be gone through.

3.2.3.1. Coregister and georeference MCA image (copied from 'MCA_PostProcess_Instructions.docx' file, made by Steve Kochaver & Eric Booth)

Initial Organization

1. Separate the raw MCA images (6 .RAW or .DCM files per trigger) into the specific site locations and dates and put into the following folder on the water_sustain file server:
   - data→WSCremoteSensing→MCA→RAW_files

Convert all .RAW/.DCM Images to .tif format

2. Open Tetracam PixelWrench2
3. Open any .RAW/.DCM
   - File -> Open -> (.RAW/.DCM Image)
4. Open up IndexTools window
   - View -> IndexTools
5. In the middle of the window, click the 'Single Tifs' button next to either 'Save DCM sets as:' or 'Save RAW sets as:' depending on the file format you currently need to convert
6. Select the folder location of the .RAW/.DCM files that need to be transformed and click 'OK'
7. Keep the default output folder location (same as input location) and click 'OK'
8. Repeat for all sites and dates

Co-register the 6 .tif image files

9. Copy the 'Create6BandImages' folder onto your hard drive, if it hasn't been already
10. Put all six .tif files from previous step in this folder so they exist in ascending band order (0-5) in the folder. Six and only six .tif files should be in the folder
11. Double check that 'LinkFiles' folder contains all correct rectification files (5 files)
12. Double click on 'WarpFromFile.py' application
13. Let the script run. When prompted enter the name of the final image (see naming convention of previously co-registered files in 'CoReg_TIF_files' folder on water_sustain server
14. Move all .tif images and script outputs from the folder for the next batch
    - Delete the non-Coregistered single TIFs (they still exist in RAW_files folder)
    - Copy the only remaining .tif file (that was just named), delete the newly-created extra files

Georeference the new co-registered .tif image file

15. Open ArcMap
16. Add the following basemap image (2010 NAIP image of Dane County) to ArcMap
    - ortho_1-1_1n_s_wi025_2010_1.sid
17. Copy the co-registered .tif image file from the 'CoReg_TIF_files' folder to the corresponding 'Georef_files' folder and rename 'XXXXX_georef.tif'
18. Add the co-registered .tif image file from the 'Georef_files' folder to ArcMap
19. Georeference the .tif image to the NAIP image using at least 3 ground control points (GCPs)
    - click on 'Add Control Points' button in Georeferencing Toolbar
    - click a point in the .tif image
    - click corresponding location in the basemap image
20. Open the 'View Link Table' (button at far right of Georeferencing Toolbar)
21. Save the GCPs as a text file in the 'GCPs' folder of the corresponding site folder within the 'MCA→Georef_files' folder
22. Once the georeferencing is satisfactory, click 'Update Georeferencing' under the Georeferencing Toolbar dropdown menu

3.2.3.2. Convert MCA images from DN to surface reflectance

By default, MCA stores its output data as DN, or digital numbers, spanning from 0-255. Unfortunately, these are physically meaningless. To calculate vegetation indices (VIs) which are used to estimate LAI & h, we first have to convert the DN values to surface reflectance (SR).

First, you need some spectrometer measurements on the ground. Ideally, this would be done simultaneously to each aerial image acquisition; for practicality purposes, I just did it once at 4 control points (surfaces that aren't expected to change much over time - concrete, road, blacktop, green grass). To do this, take a bunch of spectrometer readings at each of your control points. The spectrometer will give you SR at 1 nm intervals. You then average the spectrometer SR for each of your 6 MCA bands, centered at (450, 570, 620, 650, 670, 860) with a width of 10 nm. This gives you a mean SR for each band at each control point.

Second, go into ArcMap and extract the DN for each band at each control point. You can now plot surface reflectance of the control points against DN for each of the 6 bands, forming a relationship for each image in the form 'SR=m*DN+b'. I generate these relationships in Excel (note to self: this is the spreadsheet 'WIBU_MCAreflectance.xlsx').

Third, you need to apply these relationships over the entire field. This can be done in ArcMap using band math (ArcToolbox → Spatial Analyst → Map Algebra → Raster Calculator), or using Python. I use a Python script in which I give it the 'm' and 'b' values calculated in Excel, and then it automatically calculates SR, clips the output to my field extent, and calculates VIs (note to self: this is the 'MCA_calcSR_clip_calcVIs_singleDate.py' script).

Note that, depending on your image extent, you may have to clip off areas within the field if they contain the margins of your MCA image. For example, in this case we have to clip off the SE corner:

3.2.3.3. Get data from ArcMap into Matlab.

This step is identical to step 3.2.2.5 (Get data from ArcMap into Matlab, Processing thermal data section). You should extract the VI of your choice to your grid over the field.

3.2.3.4. Transform VI data into LAI and canopy height

Once the data is in Matlab, you can convert from VI to LAI & h, which are the inputs you've been trying to get all along. To do this, you'll first have to identify a relationship between you VI and LAI & h that is appropriate for your field site. For example, Figure 4 in Zipper & Loheide (2014) does this using Landsat data:

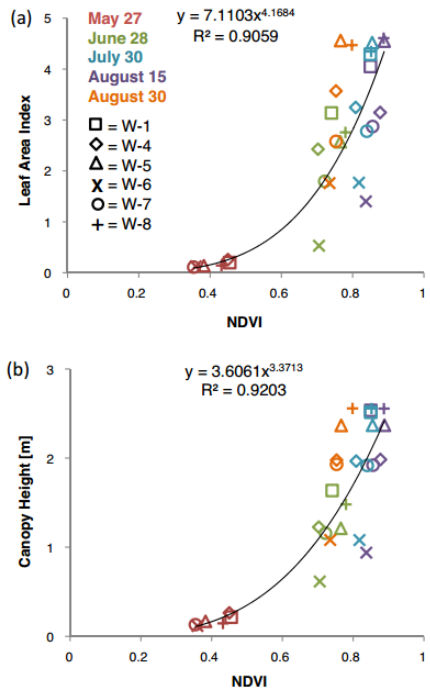

Fig. 4. Regressions of NDVI to (a) LAI and (b) canopy height data from the 2012 growing season ($n = 29$). Any estimates outside the range of observed values ($0\ m \le h \le 3\ m, 0 \le LAI \le 5$) were set to the appropriate limit to avoid overextrapolation. Points are color-coded by date and shape-coded by site, as in Fig. 1.

The relationship will likely be different for different crops, VIs, and study sites. Kang et al. (*in prep*) attempts to develop global relationships between LAI and VIs for various crops using Landsat data and will be a valuable resource. In that case, h could probably be estimated as a function of LAI as well.

3.2.4. Estimating uncertainty

At this point, you should have your canopy surface temperature (Section 3.2.2) and LAI & h (Section 3.3.3) in Matlab, which are the only HRMET inputs that we plan to vary spatially. Now, we are basically ready to run HRMET! It can be run without uncertainty following the example in `HRMET_Example_RunGrid.m`, or with uncertainty following the example in `HRMET_Example_RunGrid_withUncertainty.m`.

To estimate uncertainty following the example, we need to provide a mean and standard deviation for a Gaussian distribution from which we will randomly select input parameters and run HRMET *n* times. This will provide us with *n* output ET results, which can then be used to estimate the actual ET (as the mean of all results) and uncertainty based on the standard deviation, variance, or any other metric. Below, we will briefly go through potential techniques for estimating uncertainty in different input variables.

3.2.4.1. Uncertainty in Spatially Variable Inputs – the moving window technique

A moving window is used to assess uncertainty due to imprecise spatial location, for example due to the georeferencing process. In essence, it calculates the uncertainty at a point as the standard deviation of all the other pixels within a given window surrounding that point. This technique is demonstrated in `HRMET_Example_RunGrid_withUncertainty.m`, and the script `HRMET_Example_MovingWindow.m` can be applied to any gridded input data to calculate mean and standard deviation for each pixel.

3.2.4.2. Uncertainty in Estimated Data – the repeated regression technique

A repeated regression technique is used to estimate uncertainty in our canopy characteristics LAI and h. To do this, you randomly drop a subset of your samples (we dropped ~20%) prior to running the regression between NDVI and LAI or NDVI and h. You can then use the points that were dropped to determine the model fit (RMSE) and apply them to the spatially distributed NDVI data to estimate LAI and h at each pixel. Repeat this process *n* times, randomly dropping a different subset of data each permutation, and then calculate the mean and standard deviation LAI and h at each pixel.

3.2.4.3. Uncertainty in Meteorological Data

While meteorological data was available at our site every 30 minutes, surface temperature and reflectance measurements were collected instantaneously. We used meteorological data from 1 hour before to 1 hour after our flyovers (total data points=5) to estimate uncertainty in inputs. For meteorological variables with a trend over time (e.g. temperature), we fit a linear regression to the variable as a function of time, and estimated uncertainty as the standard deviation of the residuals from the trend. For inputs without a time-dependence (e.g. wind speed), we estimated uncertainty as the standard deviation of all measurements over the entire time interval. As our field site is relatively small (~21 ha), we did not explicitly consider spatial variability in meteorological conditions over the field.