Using the Software for Creating the Lunar Albedo

Oleg Alexandrov

June 11, 2012

1 Inputs and outputs

The albedo generation algorithm takes the following inputs:

1. A set of DRG images (Digital Raster Graphic images of the Moon in the GeoTIFFf format)
2. A set of DEM images (Digital Elevation Models in the GeoTIFFf format)
3. The Sun and spacecraft position at the moment each DRG image was taken.

The albedo algorithm uses the Sun/spacecraft position and DEM data to correct for the illumination conditions in the DRG images, thus computing Moon's true albedo. The output will be a set of non-overlapping DRG tiles containing the albedo.

In addition to creating albedo images, the software can also be used for generating image mosaics (see the parameter REFLECTANCE_TYPE below). In that case, the DEM images and Sun/spacecraft positions are not required.

2 The settings file

The parameters controlling the albedo generation algorithm are specified in a settings file (a sample file is shown in the appendix). It has the following entries:

DRG_DIR	The path to the directory of DRG images.
DEM_DIR	The path to the directory of DEM images.
SUN_POSITION_FILE	The Sun position file.
SPACECRAFT_POSITION_FILE	The spacecraft position file.
NUM_PROCESSES	How many simultaneous processes to start on a given computing node. This number is best set to the number of available cores on the node.

TILE_SIZE	Tile size, in degrees (usually 4 by 4 degrees, or 1 by 1). The output of the algorithm will be a set of image tiles of this size.
SIMULATION_BOX	The region in which to compute the albedo, in the format lon_min : lon_max : lat_min : lat_max. If this parameter is not provided, the region will be assumed to be the entire Moon.
REFLECTANCE_TYPE	The reflectance type. Values: 0 – the reflectance is not be used, the result of albedo generation is just an image mosaic, 1 – the Lambertian reflectance model is used, 2 – the Lunar-Lambertian reflectance model is used.
SHADOW_THRESH	Shadow threshold, with values between 0 to 255 (the input DRG images have values between 0 to 255). Its use depends on the variable SHADOW_REMOVAL_TYPE.
SHADOW_REMOVAL_TYPE	Determines how the shadow pixels will be removed. Values: 0 – do not remove shadow pixels, 1 – remove pixels with values below SHADOW_THRESH, this is the default. Advanced values: 2 – remove pixels for which $I/(R*T) < t$ [1], 3 – analogous to option 2, but use the Lunar-Lambertian reflectance model instead of the Lambertian model.
TR_CONST	A constant which determines how bright the albeo will be, a larger value will make the albedo appear darker.
PHASE_COEFF_A1	The coefficient A_1 in the expression $e^{-A_1\alpha} + A_2$ (here α is the phase angle) which multiplies the limb darkening factor.
PHASE_COEFF_A2	The coefficient A_2 in the expression $e^{-A_1\alpha} + A_2$ above.

[1] I is the image value, R is the Lambertian reflectance, T is the image exposure, and t is the shadow threshold.

MAX_NUM_ITER	Maximum number of albedo update iterations. If it equals 0, the image exposure, phase coefficients, and the albedo itself will only be initialized, and no subsequent iterations will take place to improve the initial values.
NO_DEM_DATA_VAL	The value used in the DEM images to denote that no data is available at the current pixel. This number is normally stored within the DEM images; only if it absent there will the value specified here be used.
USE_WEIGHTS	If to use weights to seamlessly blend the albedo values obtained for individual images (1 means true, and 0 means false).
USE_NORMALIZED_COST_FUN	If to enforce that the sum of weights over all images add up to 1 at each pixel. If true, all pixels will contribute in equal amount to the cost function. This flag makes a difference only if multiple albedo iterations are performed (MAX_NUM_ITER > 0). Turning this on incurs a significant performance hit.
POST_SCALE_ALBEDO	If after the last iteration to multiply the albedo by $1 + A_2$, the value of the phase function when the phase angle α is 0.
COMPUTE_ERRORS	If at the end of the algorithm to compute the albedo error map, showing at each pixel what the error was in computing the albedo at that pixel.

3 Sample settings file

```
# Files/directories
DRG_DIR                    DIM_input_2560mpp
DEM_DIR                    DEM_tiles_sub64
SUN_POSITION_FILE          meta/sunpos.txt
SPACECRAFT_POSITION_FILE   meta/spacecraftpos.txt

# Constants
NUM_PROCESSES              8
TILE_SIZE                  4.0
SIMULATION_BOX             -180:180:-180:180
```

```
REFLECTANCE_TYPE         2
SHADOW_THRESH            40
SHADOW_REMOVAL_TYPE      1
TR_CONST                 1.6
PHASE_COEFF_A1           1.4
PHASE_COEFF_A2           0.5
MAX_NUM_ITER             0
NO_DEM_DATA_VAL          -32767

# Actions
USE_WEIGHTS              1
USE_NORMALIZED_COST_FUN  0
COMPUTE_ERRORS           0
```

4 Requirements for the input

As mentioned earlier, the inputs to the albedo generation software are a set of DRG images, a set of DEM images, and the Sun and spacecraft position for each DRG image. Here we specify the exact requirements for the data as expected by the software.

1. The DRG images must be in a directory and stored in the GeoTIFF format. It is assumed that the intensity values for each image pixel are between 0 and 255 (uint8). The first 11 characters of each image in that directory must be unique, and will be the means by which we will later look up the Sun and spacecraft position for that image. For example, given the image:

   DRG_input_sub64/AS15-M-0074_0075-DRG.tif

   the lookup key is the 11-character string: AS15-M-0074.

2. The DEM images must be in a directory and stored in the GeoTIFF format. They may or may not overlap. If the DEM images overlap, the values from all images at a given pixel will be averaged. It is expected that the DEM images are either in int16 or float format. If the images lack the No Data Value, it must be explicitly set in the settings file using the variable NO_DEM_DATA_VAL.

3. The Sun positions for the DRG images must be provided as a list in a file, with one line in the file for each DRG image. The same must hold for the spacecraft position. Each line in these lists needs to be in the format:

   lookup_key x y z

   where lookup_key is the 11-character string that identifies the DRG image, and x, y, and z are the coordinates of the Sun (spacecraft) in the Cartesian coordinate system whose origin is the center of the Moon. The values must be in kilometers. An example entry for the Sun position is

   AS15-M-0074 -1345328.6130903 151849061.51791 689530.60102555

4. The directories containing the DRG and DEM images must be user-writeable, as the software will create in those directories a list of the GeoTIFF images contained within them, together with the coordinates of the corners of each image. These lists are created only once and speed up subsequent computations.

5. The amount of memory used by the algortihm is proportional to the tile size. Generating the albedo at high resolution, such as 10 meters/pixel requires using small tiles, such as 1 by 1 degree. Otherwise larger tile sizes can be used, such as 4 by 4 degrees.

5 Orthoprojecting ISIS cubes

The albedo algorithm assumes that the input images are in GeoTIFF format, and that the sun and spacecraft positions are in text files. This section describes how to obtain this data from ISIS cubes.

The images can be obtained by orthoprojecting the ISIS cubes onto the terrain data (a set of DEM tiles) using the script named *orthoproject_cube.sh* in the directory PhotometryTK/src/tools. The script assumes that the following packages are installed: Ames Stereo Pipeline, ISIS libraries, and the GDAL tools. The paths to them can be set in the script. It can be called as:

```
orthoproject_cube.sh input.cub input_isis_adjust DEM_dir mpp num_proc output.tif
```

The arguments passed to it are, respectively, the cube file to orthoproject, the ISIS adjust file storing the adjusted camera position, the directory containing the DEM tiles to orthoproject onto, the desired output resolution in meters per pixel, the number of processors to use, and the name of the output image.

6 Using the software and the output

1. The albedo reconstruction software can be called by using the script *reconstruct.sh* in the directory PhotometryTK/src/tools. It is used as follows:

   ```
   reconstruct.sh setings.txt labelStr
   ```

   Here, setings.txt is the settings file, and labelStr is a string label.

   Using this script requires that Vision Workbench and PhotometryTK be complied (see the installation notes of these two packages for instructions). The script reconstruct.sh needs to be edited to specify the path to these packages.

2. The output albedo will be in the directory albedo_labelStr/albedo. The output will be a set of image tiles, with each pixel being uint8 (the same as the input DRG images).

3. The exposure time for each image computed during albedo estimation is stored in the directory albedo_labelStr/exposure. Other subdirectories in albedo_labelStr

contain data used for intermediate computations, such as the weights assigned to each image, the averaged DEM in each tile, etc.

7 The algorithm

The albedo is generated in several steps:

0. Initial setup.
1. Compute the weights for each DRG image (refer here to the weights).
2. Create the average DEM for each albedo tile based on the input DEM images.
3. Compute the sum of weights over all images at each pixel, if the variable USE_NORMALIZED_COST_FUN is set to 1.
4. Compute the initial guess for the exposure of each DRG image (refer here to the formula).
5. Compute the initial guess for the albedo in each tile (refer here to the formula).

Next, the algorithm will repeat MAX_NUM_ITER times the following steps:

6. Update the exposure for each DRG image (note: we don't do that anymore as the results are bad).
7. Update the phase coefficients (this is done in two steps, first the components of these coefficients are found for each tile, and then the results from all tiles are combined).
8. Update the albedo in each tile.

Lastly, if the flag COMPUTE_ERRORS is set to 1, the algorithm has another step.

9. Compute the albedo error, with one value at each albedo pixel (refer here to the equation having the formula).

The algorithm can be distributed over a very large number of computing nodes, as long as those nodes share storage. In each of the steps, some data is read from disk, processed, and then saved back to disk, to be used by subsequent steps. Each of the steps, except step 0 (the setup) consists of taking a DRG image or a tile, and performing a calculation. These calculations are independent of each other, and as such, they can take place in parallel over the specified nodes.

8 Advanced usage

The script *reconstruct.sh* computes the albedo by performing a series of calls to the executable *PhotometryTK/build/src/tools/reconstruct*. Advanced users may call this program directly to execute one of the steps listed in the albedo algorithm above.

The *reconstruct* executable takes the following options:

```
-s [ --settings-file ] arg     Settings file
-r [ --results-directory ] arg Results directory
-f [ --images-list ] arg       The list of images
-t [ --tiles-list ] arg        The list of albedo tiles
-i [ --image-file ] arg        Current image or tile
--initial-setup                Initial setup
--save-weights                 Save the weights
--compute-weights-sum          Compute the sum of weights at each pixel
--compute-shadow               Compute the shadow
--init-dem                     Initialize the DEM
--init-exposure                Initialize the exposure times
--init-albedo                  Initialize the albedo
--update-exposure              Update the exposure times
--update-tile-phase-coeffs     Update the phase coefficients per tile
--update-phase-coeffs          Update the phase coefficients by combining the
                               results over all tiles
--update-albedo                Update the albedo
--update-height                Update the height (shape from shading)
--compute-errors               Compute the errors in albedo
--is-last-iter                 Is this the last iteration
-h [ --help ]                  Display this help message
```

For example, to initialize the exposure times for a given image, one may issue the command:

```
reconstruct -s settings.txt -r albedo_run -f albedo_run/imagesList.txt -t
albedo_run/albedoTilesList.txt –init-exposure -i DRG_DIR/AS16-M-2961.tif
```

Each time the *reconstruct* executable is run, it echoes the precise command and options which were used to call it, so these calls can be inferred by looking at the output of *reconstruct.sh*.