esac
European Space Astronomy Centre (ESAC)
European Space Agency (ESA)
Camino Bajo del Castillo s/n
Urb. Villafranca del Castillo
28692 Villanueva de la Canada - Madrid
SPAIN

SMART-1/AMIE IMAGE MOSAICS OF THE MOON

Prepared by **Björn Grieger and José Fonseca**
Date of Issue **2021/06/14**

Contents

1 Introduction

1.1 Executive Summary

This data set contains a collection of image mosaics that cover (with some tiny gaps) the whole Moon. The mosaics where assembled from images taken by the AMIE instrument aboard the SMART-1 S/C.

1.2 Extended introduction

The SMART-1 S/C was launched on 27 September 2003 end reached its lunar baseline science orbit on 13 March 2005 for a nominal science period of six months and one year extension. During these 18 months, the AMIE camera aboard the spacecraft acquired about 32 000 images that were used to produce this collection of mosaics — a complete atlas of the Moon from SMART-1/AMIE images.

SMART-1 operated in an eccentric polar orbit with the perilune close to the South pole at a minimum distance of 400 km and an apolune distance of about 6400 km. The area south off 87°S and various spots in the southern hemisphere are covered by the AMIE camera with a resolution better than 50 meters per pixel, the complete Southern hemisphere with a resolution better than 100 meters per pixel, and the complete Northern hemisphere with a resolution better than 250 meters per pixel.

However, during the Earth Escape Phase, the radiation significantly impacted the AMIE sensor, invalidating the laboratory dark and flat fields. The altered sensor behaviour was compensated by a new calibration procedure based on in-flight images.

Each AMIE image frame of 1024×1024 pixels is divided into areas covered by four different filters, designed for multi-spectral analysis. We calibrated the different areas to compensate for the the largely different sensitivity, so that they can be used as full-frame grey scale images. The resulting 1024×1024 pixel images were geographically referenced and mosaiced considering illumination angle and image quality in order to produce lunar surface maps with varying resolution, lower towards the North and higher towards the South. The final maps achieved a coverage of approximately 96 % of the Lunar surface.

1.2.1 List of figures

1.2.2 List of tables

1.3 AURORA-UNINOVA_MOON_AMIE_V1.0 introduction

The purpose of this PUG is to enable the user to exploit the data products contained in this data set. All products are GeoTIFF files, either 3000×3000 or 6000×6000 pixels in size, with 8 bit depth.

The contents and format of the data products are described in detail in section 4.

For information about this dataset, contact

Björn Grieger (PhD)
Aurora Technology B. V. for the European Space Agency (ESA)
European Space Astronomy Centre (ESAC)
Camino Bajo del Castillo s/n
28692 Villanueva de la Caada
Madrid
Spain
Email: Bjoern.Grieger@esa.int
Phone +34 91 81 31 107

or

José Manuel Fonseca (PhD)
UNINOVA — CA3

Campus FCT-UNL
2829-516 Caparica
Portugal
Email: jmf@uninova.pt
URL: www.ca3-uninova.org
Phone: +351 212948380

1.4 Abbreviations and Acronyms

AMIE — Advanced Moon micro-Imager Experiment

BMP — bitmap image file format

ESA — European Space Agency

GeoTIFF — Geographic Tagged Image File Format

GIS — Geographic Information System

GSF — Guest Storage Facility of the PSA

GUI — Graphical User Interface

PSA — Planetary Science Archive of ESA

PUG — Product User Guide

R — implementation of the S (Statistical) programming language

S/C — Spacecraft

SMART — Small Missions for Advanced Research and Technology

SPICE — Spacecraft, Planet, Instrument, Camera, Events — an Observation Geometry System for Space Science Missions

1.5 Reference and Applicable Documents

S. Besse, C. Vallat, M. Barthelemy, D. Coia, M. Costa, G. De Marchi, D. Fraga, E. Grotheer, D. Heather, T. Lim, S. Martinez, C. Arviset, I. Barbarisi, R. Docasal, A. Macfarlane, C. Rios, J. Saiz, and F. Vallejo. ESA's Planetary Science Archive: Preserve and present reliable scientific data sets. *Planet. Space Sci.*, 150:131–140, January 2018. doi: 10.1016/j.pss.2017.07.013.

2 Scientific Objectives

2.1 Design, realization, and publication

When the calibration of the AMIE images had been completed in 2008 and it was realized that almost the complete Moon is covered with good quality, the idea to create an Atlas of the Moon from AMIE images came up. Originally, a printed paper "coffee table book" was envisaged. A design for the map layout was developed and efforts to assemble the AMIE images into map mosaics were started at ESA. The software that was developed in Java had a GUI and should enable the user to interactively combine images into a mosaic. However, it turned out that the software was much too slow, which rendered it unusable.

In 2012, a collaboration with UNINOVA, Portugal, was started. UNINOVA went for a purely programmatic, non-interactive approach, and before the end of the year, a complete set of nice looking maps had been created. At that point, the main drivers at ESA had to move on to other obligations, and, unfortunately, the project idled out. An attempt to revive it was undertaken in 2014, but it was not successful.

When ESA introduced its GSF in 2019, the idea came up to finally publish the AMIE Moon mosaics there, also triggered by the nearing of the 50th Apollo 11 anniversary. However, it took until 2021 before this was seriously pursued. When the mosaics created 2012 were closely inspected, it turned out that they did not exactly match the original map layout design. This had not yet been noticed before the project idled out in 2012.

It would have been quite an effort to resurrect and correct the original software from 2012 to recreate the mocaics exactly to specification. Instead, the applied map projections were reversely engineered, see section 3.3, and GeoTIFF files were created which correctly reflect the actual projections. This could be done without resampling the data, so the original accuracy is preserved.

By providing GeoTIFF files, the projection actually used becomes completely transparent to the user. They can reproject the data to any desired projection with a GIS system.

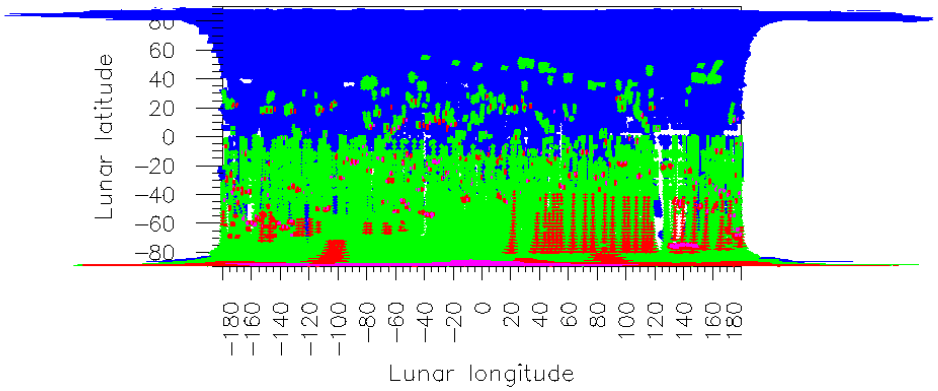


Figure 1: Global coverage and resolution of AMIE images (full frame). For the color coding see Fig. 2.

2.2 Resolution

The AMIE images have typically better resolution in the southern hemisphere, with particularly high resolution close to the south pole, see Figs. 1, 2. The map layout was designed to follow this pattern, so the maps have a varying scale, with moderate resolution in the northern hemisphere and higher resolution in the southern hemisphere, increasing towards the South pole. However, in the available version, the South polar area itself was only mapped with half the envisaged resolution.

2.3 Map projections

The map layout was designed with the following requirements in mind:

- The maps should be approximately square.
- It should be easy to put several maps together to create a larger mosaic. This is nice in order to look at a larger area or to see more context for a feature close to a map edge. Also, it allows to create an "AMIE basemap" from the individual maps without much effort.
- The distortion should be minimal, the maps shoud be conformal (i. e., round craters should still be round).
- The maps should be cut in a way which yields a varying resolution as described in section 2.2.

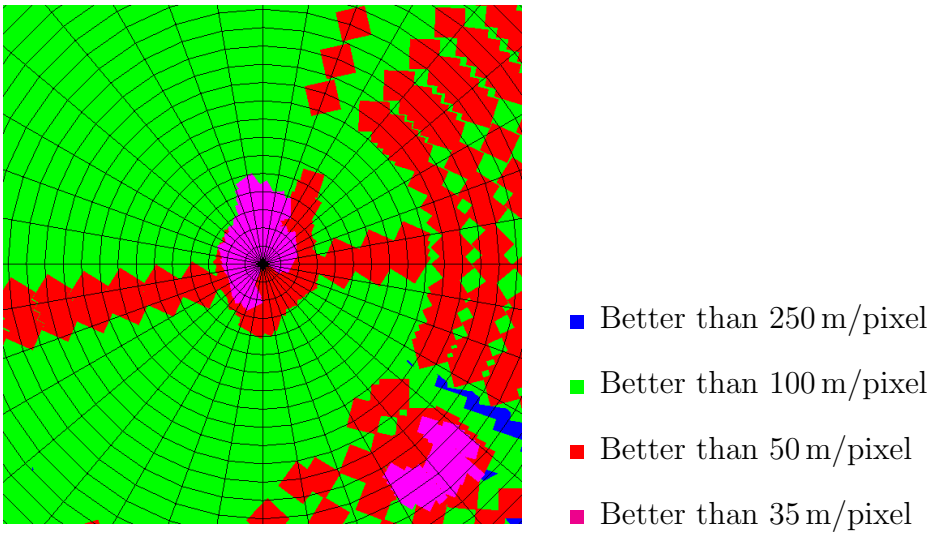


Figure 2: Resolution of AMIE images near the south pole. Grid spacing is 10° in longitude and 1° in latitude.

It is impossible to meet all conditions with a single type of projection for the whole globe. Therefore, two different projection types were envisaged:

- Mercator projection for lower latitudes.
- Polar stereographic projection for the polar areas.

Because of historical reasons outlined in section 2.1, the maps supposed to be in Mercator projection are not exactly that. They are slightly stretched in vertical direction to make them exactly square, which destroys strict conformality. The maps supposed to be in polar stereographic projection are in fact azimuthal equidistant, which is not conformal at all. However, because the actually applied projections are correctly reflected by the georeferencing information in the GeoTIFF files, the maps are correctly displayed in any GIS sytem and can easily be reprojected to any desired projection.

2.4 Detailed map layout

2.4.1 The lower latitudes in Mercator projection

The map layout for the lower latitudes is shown in Fig. 3. There are 68

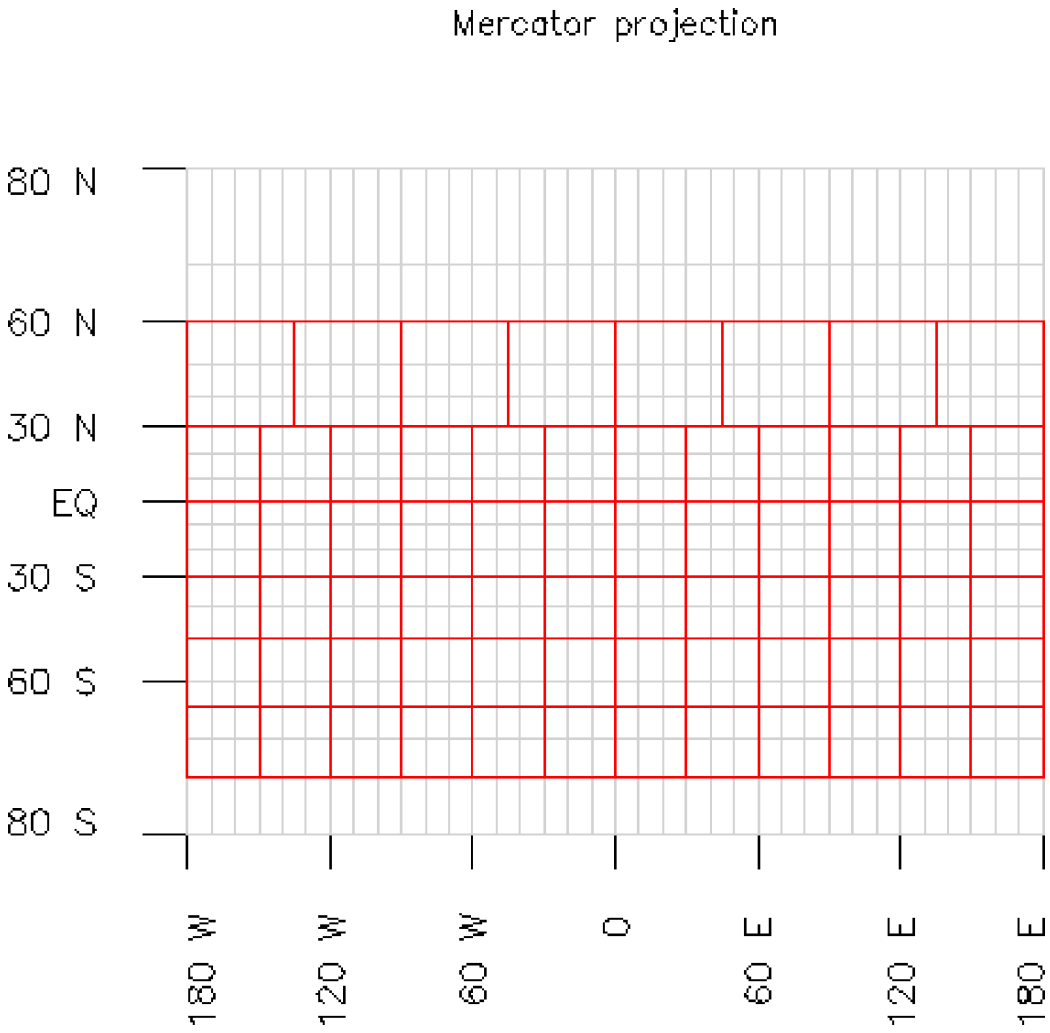


Figure 3: Map layout for the lower latitudes. Gray lines provide a latitude-longitude grid with 10° spacing, red lines outline the map edges. Mercator projection is used for the maps in the area between 75°S and 60°N.

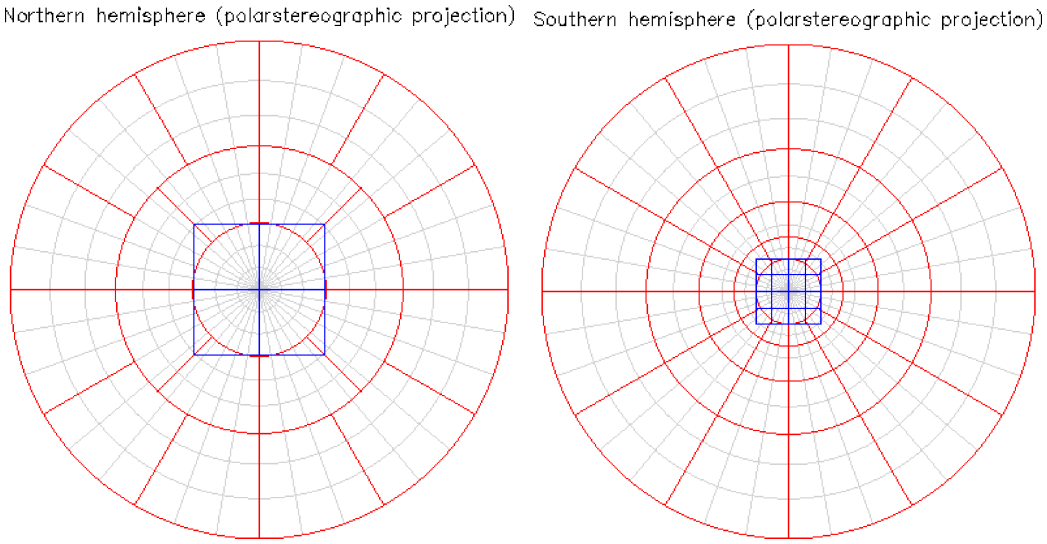


Figure 4: Map layout of the northern hemisphere (left) and the southern hemisphere (right), shown here in polar stereographic projection. Gray lines provide a latitude-longitude grid with 10° spacing, red lines outline the map edges of the Mercator maps also shown in Fig. 3, and blue lines outline the map edges of the polar stereographic maps. The polar maps in this dataset are in azimuthal equidistant projection. This projection is used for the area north of 60°N and south of 75°S.

maps in Mercator projection. South of 30°N, the width in longitude is constantly 30°. This implies an increase of resolution toward higher southern latitudes. The height of the maps in latitude decreases from 30° close to the equator to 10° at higher southern latitudes. This makes all maps approximately square and reflects the increasing resolution. North of 30°N, the maps are 45° wide in longitude and 30° high in latitude. This implies about the same resolution as for the maps close to the equator.

2.4.2 The north polar area in polar stereographic projection

The map layout of the northern hemisphere is shown in Fig. 4, left. Originally, four maps were envisaged for the North polar area, each with a resolution of 3000×3000 pixels, where the North pole is the center of the projection for all four images. In this dataset, the four originally envisaged maps are combined into one single map with a resolution of 6000×6000 pixels, so the planned resolution is kept.

Latitude	Projection	Maps	Height		Resolution
60°N–90°N	Polar	4	30°	900 km	300 m/dot
30°S–60°N	Mercator	32	30°	900 km	300 m/dot
50°S–30°S	Mercator	12	20°	600 km	200 m/dot
65°S–50°S	Mercator	12	15°	450 km	150 m/dot
75°S–65°S	Mercator	12	10°	300 km	100 m/dot
90°S–75°S	Polar	16	7.5°	250 km	85 m/dot

Table 1: List of all maps of the atlas. The height of the maps is given in degrees (along a great circle) and in km. The resolution has been estimated for the originally planned printed book assuming a printing resolution of 300 dots/inch and a page width of 10″, i. e., 3000 dots across a page.

2.4.3 The south polar area in polar stereographic projection

The map layout of the southern hemisphere is shown in Fig. 4, right. Originally sixteen maps were envisaged for the south polar area, each with a resolution of 3000 × 3000 pixels, where the North pole is the center of the projection for all images. Like for the North polar area, the South polar maps are combined into one single map. However, while the original map layout envisage in total 12 000 × 12 000 pixel for the South polar area, the singe map has only 6000 × 6000 pixels, so half the designed resolution.

2.4.4 Overview of the set of maps

The planed maps are listed in Table 1. The number of originally envisaged maps was 88, but because of the combination of all polar maps into one for each hemisphere, we have 70 maps in the dataset. The varying map resolution is mostly close to the resolution of the respective available AMIE images, cf. Figs. 1, 2, however, for the South polar area, it is only half of the planned resolution.

2.5 Acknowledgements

The map mosaics contained in this dataset are based on the datasets of calibrated AMIE images from the Lunar Phase,

`S1-L-X-AMIE-3-RDR-LP-V1.1`,
`https://doi.org/10.5270/esa-qkf4g50`,

and the Extended Phase,

> S1-L-X-AMIE-3-RDR-EP-V1.1,
> https://doi.org/10.5270/esa-555g8qo,

of the SMART-1 mission. The version V1.1 is the latest one to date. The mosaics were in fact created from version V1.0, however, V1.1 only differs in that additional browse images have been added. The SMART-1 data are available at the Planetary Science Archive (Besse et al., 2018).

If you use data from this dataset, please acknowledge it as follows:

> European Space Agency, 2021,
> AURORA-UNINOVA_MOON_AMIE_V1.0,
> https://doi.org/10.*/esa-*

3 Data Product Generation

<In this section more details will be provided on how the source data for these data products was gathered. Then, the specific scientific and/or engineering techniques that were applied to arrive at the final form of the data products should be explained in more detail. Giving explanations of each step is needed for the end users. Any calibration methods and parameters, and the processing level of the data will be discussed here as well. Comparisons to other results, if applicable, should also be included in this section. Other topics to discuss can include instrument modes, assumptions made in the analysis, etc.>

<Finally, there shall be some discussion of how this dataset was reviewed and validated. For most external datasets (i.e. not those provided as part of regular deliveries by a mission's Principal Investigator teams) this can be covered by referencing and including the details of a published, peer-reviewed article which corresponds to these data products being provided to the PSA. In case no such article exists, a review would have to be conducted by a third party, which would need to be discussed in detail with the PSA prior to acceptance of the dataset.>

3.1 Image geometry computation

See ~/MOON/SMART_1/AMIE/ALL4DX/src/geometry.f.

Geometry information, i. e., the location on the Moon in latitude and longitude, was computed for each pixel of a low resolution version of each full frame image. While the full frame image has 1024 × 1024 pixels, the low resolution version has 128 × 128 pixels, and the geometry was computed for the 129 × 129 grid corner points. Computations were performed using SPICE and the SMART-1 SPICE kernels. All SMART-1 kernels can be found in the 'GEOMETRY' directory of any AMIE dataset in the PSA.

Each of the 1024 × 1024 pixels of a full frame image was projected onto the Moon by interpolating longitude and latitude from the low resolution version.

3.2 Map mosaicing

3.3 GeoTIFF map generation

3.3.1 LaTeX, R, and knitr

The software to generate the GeoTIFF files from the BMP files created in 2012 is embedded in this document. The code is written in the R programming language and the knitr package executes the code and inserts the results into the LaTeX document. This works pretty much like a Jupyter notebook, but provides the full power of LaTeX. Instructions to modify and re-create this document are provided in section B.

3.3.2 Packages

We will need these two packages, so we load them:

```
library( 'raster' )


## Loading required package:  sp

library( 'rgdal' )


## rgdal:  version:  1.5-23, (SVN revision 1121)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime:  GDAL 2.2.3, released 2017/11/20
## Path to GDAL shared files:  /usr/share/gdal/2.2
## GDAL binary built with GEOS: TRUE
```

```
## Loaded PROJ runtime:  Rel.  4.9.3, 15 August 2016, [PJ_VERSION: 493]
## Path to PROJ shared files:  (autodetected)
## Linking to sp version:1.4-5
```

3.3.3 Maps in Mercator projection

The motivation for the map layout has been discussed in section 2. We have 68 maps in Mercator projection, numbered from 5 to 72. Here we set the North (n), South (s), West (w), and East (e) boundaries for each map:

```
n = vector( "numeric", 72 )
s = vector( "numeric", 72 )
w = vector( "numeric", 72 )
e = vector( "numeric", 72 )

w[5:12] = seq.int( -180, 135, 45 )
e[5:12] = seq.int( -135, 180, 45 )
n[5:12] = 60
n[13:24] = s[5:12] = 30

n[25:36] = s[13:24] =   0
n[37:48] = s[25:36] = -30
n[49:60] = s[37:48] = -50
n[61:72] = s[49:60] = -65
s[61:72] = -75

w[13:72] = seq.int( -180, 150, 30 )
e[13:72] = seq.int( -150, 180, 30 )
```

We also set the Moon radius and the conversion factor from degree longitude to kilometers:

```
r = 1737.4
kpd = 2 * pi * r / 360
```

We create a new raster for the map with georeferencing information:

```
geomap = raster( ncol = 3000, nrow = 3000 )
```

Now we loop over all Mercator maps:

```
for( i in 5:72 ) {

#  Set file names
    if ( i == 5 ) {
        infile = paste( '../dat/V6P/Map ', i, '.bmp', sep = '' )
        outfile = paste( '../out/Map_', i, '_geo.tif', sep = '' )
    } else {
        infile = paste( '../dat/V5/MapV5_', i, '.bmp', sep = '' )
        outfile = paste( '../out/MapV5_', i, '_geo.tif', sep = '' )
    }

#  Read input file
    map = raster( infile )

#  Apply Mercator projection
    xmin( geomap ) = w[i] * kpd
    xmax( geomap ) = e[i] * kpd
    ymin( geomap ) = r * log( tan( pi / 4 + s[i] * pi / 360 ) )
    ymax( geomap ) = r * log( tan( pi / 4 + n[i] * pi / 360 ) )
    crs( geomap ) = paste( '+proj=merc',
                           paste( '+R=', r * 1000, sep = '' ),
                           '+units=km',
                           sep = ' '
                         )

#  Copy values
    values( geomap ) = values( map )

#  Write output file
    writeRaster( geomap, outfile, 'GTiff', datatype='INT1U',
                 overwrite = TRUE )
}
```

We choose one of the resultant files for inspection:

```
xfile = '../out/MapV5_37_geo.tif'
```

This is a summary of the georeferencing information in the file:

```
GDALinfo( xfile )
```

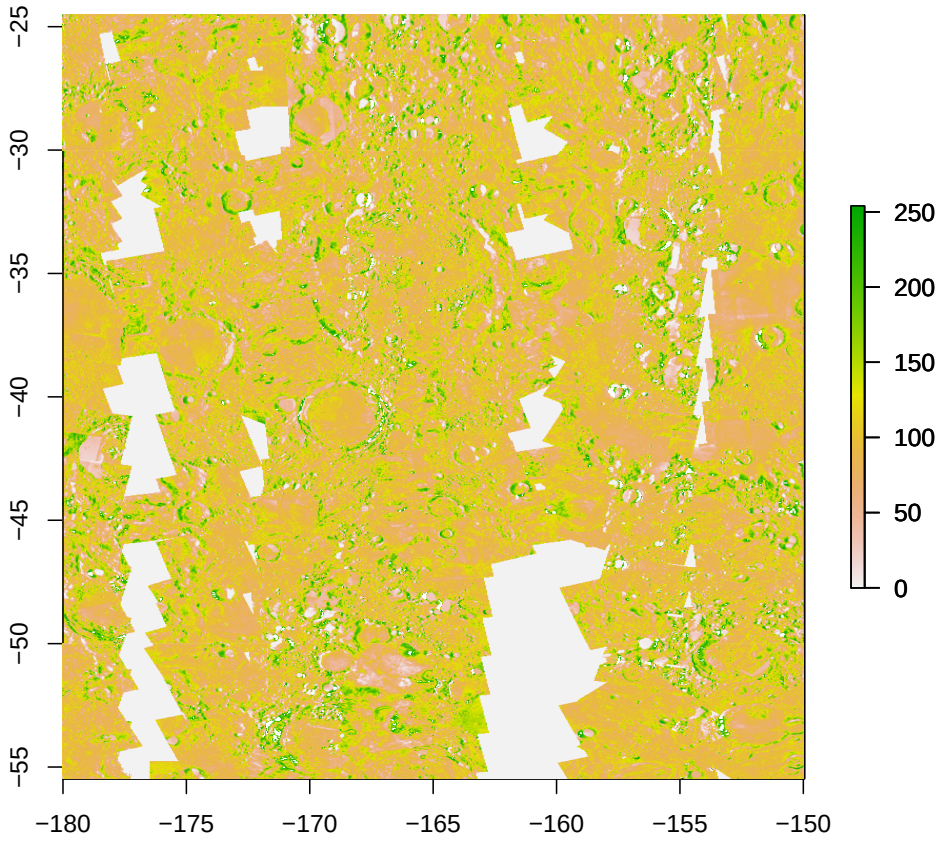
```
## rows        3000
## columns     3000
## bands       1
## lower left origin.x        -5458.203
## lower left origin.y        -1755.961
## res.x       0.3032335
## res.y       0.2671988
## ysign       -1
## oblique.x   0
## oblique.y   0
## driver      GTiff
## projection  +proj=merc +lon_0=0 +k=1 +x_0=0 +y_0=0 +a=1737400 +b=1737400 +units=km
## +no_defs
## file        ../out/MapV5_37_geo.tif
## apparent band summary:
##   GDType hasNoDataValue NoDataValue blockSize1 blockSize2
## 1   Byte           TRUE         255          2       3000
## apparent band statistics:
##   Bmin Bmax    Bmean      Bsd
## 1    0  255 90.43446 47.31612
## Metadata:
## AREA_OR_POINT=Area
```

We read the raster from the file and print it:

```
xmap = raster( xfile )
xmap
```

```
## class      : RasterLayer
## dimensions : 3000, 3000, 9e+06  (nrow, ncol, ncell)
## resolution : 0.3032335, 0.2671988  (x, y)
## extent     : -5458.203, -4548.503, -1755.961, -954.3645  (xmin, xmax, ymin, yma
## crs        : +proj=merc +lon_0=0 +k=1 +x_0=0 +y_0=0 +a=1737400 +b=1737400 +unit
## source     : MapV5_37_geo.tif
## names      : MapV5_37_geo
## values     : 0, 255  (min, max)
```

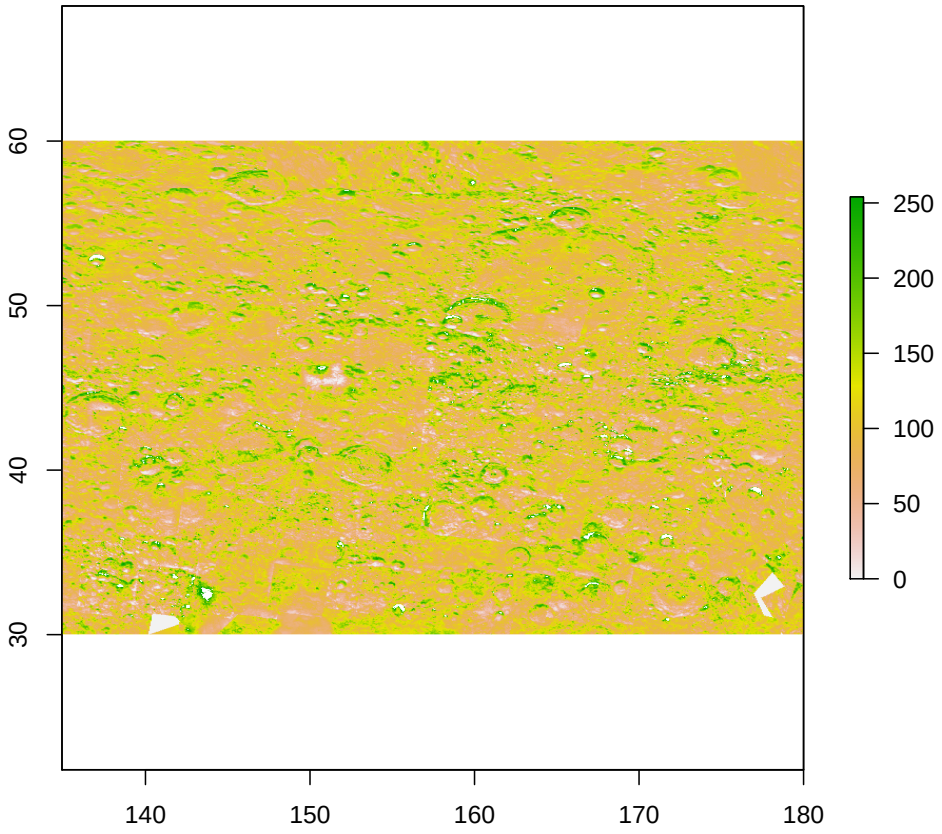
```
xmapeqc = projectRaster( xmap,
                         crs='+proj=longlat +lat_0=0 +lon_0=0 +R=1737400',
                         method='bilinear' )
xmapeqc

## class      : RasterLayer
## dimensions : 3008, 3005, 9039040  (nrow, ncol, ncell)
## resolution : 0.01, 0.00668  (x, y)
## extent     : -180, -149.95, -50.04668, -29.95324  (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +lat_0=0 +lon_0=0 +R=1737400
## source     : memory
## names      : MapV5_37_geo
## values     : -7.041126, 254  (min, max)

plot( xmapeqc, asp=T )

xfile = '../out/MapV5_49_geo.tif'
xmap = raster( xfile )
xmapeqc = projectRaster( xmap,
                         crs='+proj=longlat +lat_0=0 +lon_0=0 +R=1737400',
                         method='bilinear' )
plot( xmapeqc, add=T )

xfile = '../out/MapV5_25_geo.tif'
xmap = raster( xfile )
xmapeqc = projectRaster( xmap,
                         crs='+proj=longlat +lat_0=0 +lon_0=0 +R=1737400',
                         method='bilinear' )
plot( xmapeqc, add=T )
```



We have also added above and below maps. Fit fine. The area plotted is determined by the first raster, so the above and below rasters are cropped.

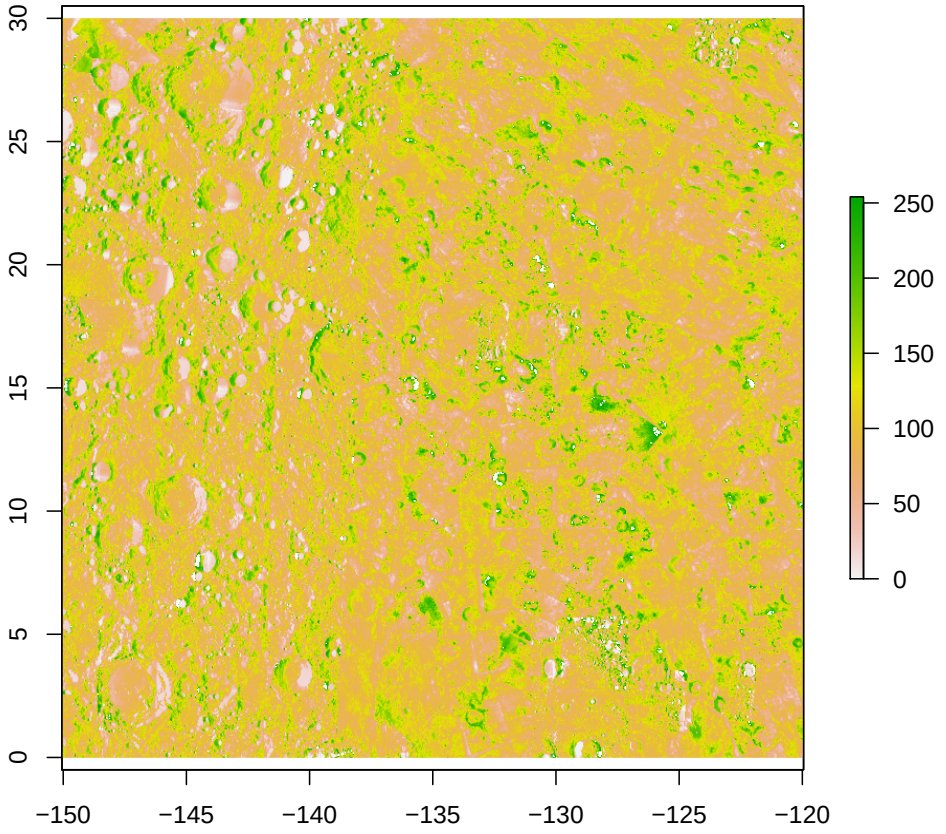
We plot a few other maps to check the boundaries:

```
xfile = '../out/MapV5_12_geo.tif'
xmap = raster( xfile )
xmapeqc = projectRaster( xmap,
                         crs='+proj=longlat +lat_0=0 +lon_0=0 +R=1737400',
                         method='bilinear' )
plot( xmapeqc, asp=T )
```

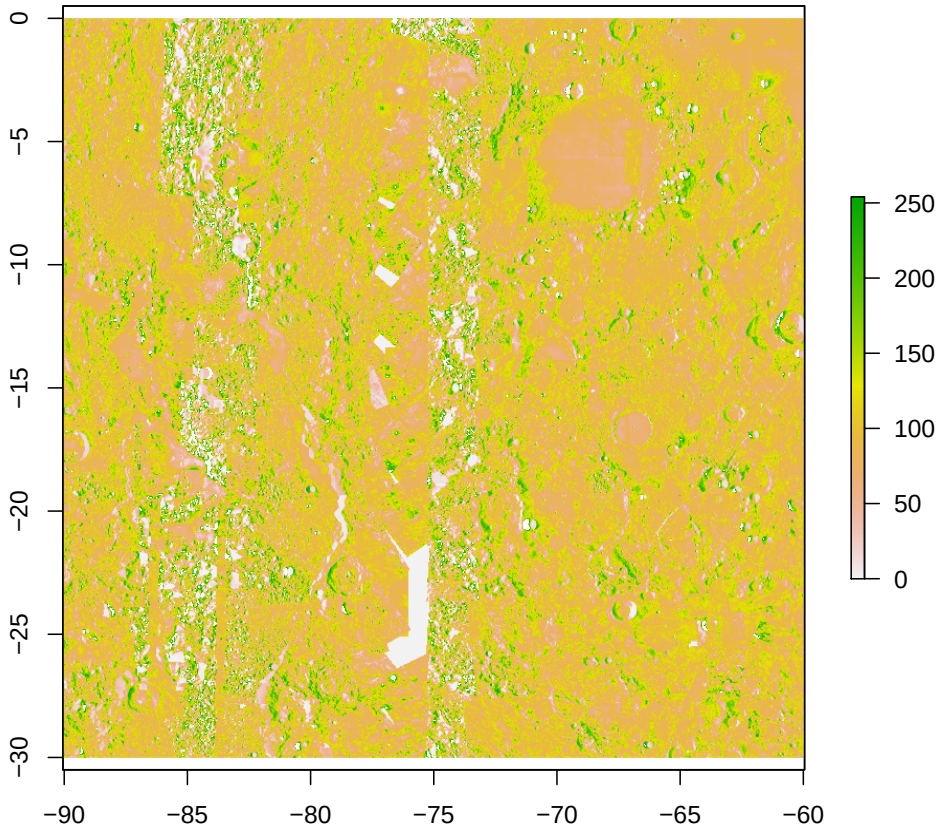



```
xmapeqc12 = xmapeqc
```

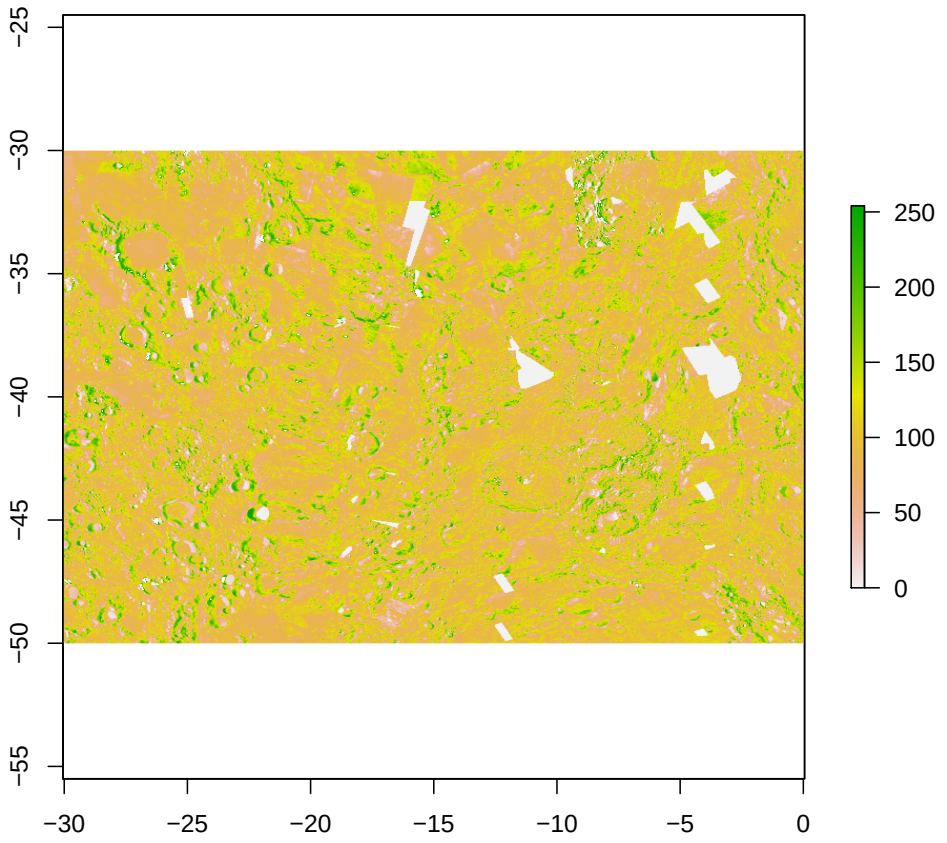
```
xfile = '../out/MapV5_14_geo.tif'
xmap = raster( xfile )
xmapeqc = projectRaster( xmap,
                         crs='+proj=longlat +lat_0=0 +lon_0=0 +R=1737400',
                         method='bilinear' )
plot( xmapeqc, asp=T )
```



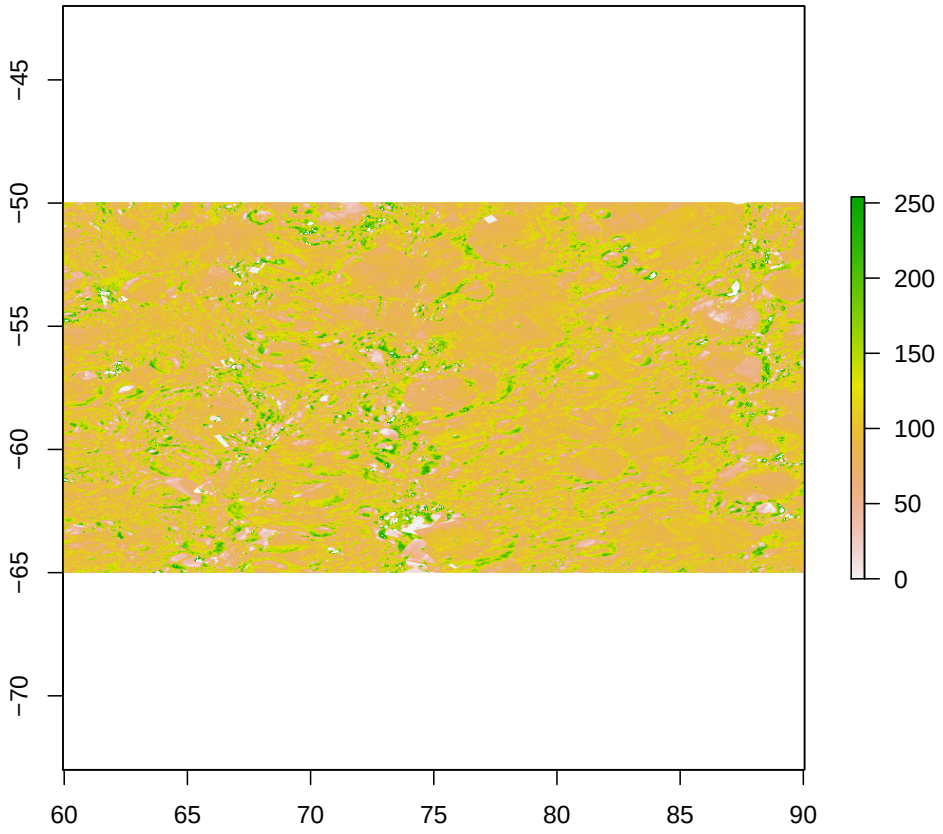

```
xfile = '../out/MapV5_28_geo.tif'
xmap = raster( xfile )
xmapeqc = projectRaster( xmap,
                         crs='+proj=longlat +lat_0=0 +lon_0=0 +R=1737400',
                         method='bilinear' )
plot( xmapeqc, asp=T )
```



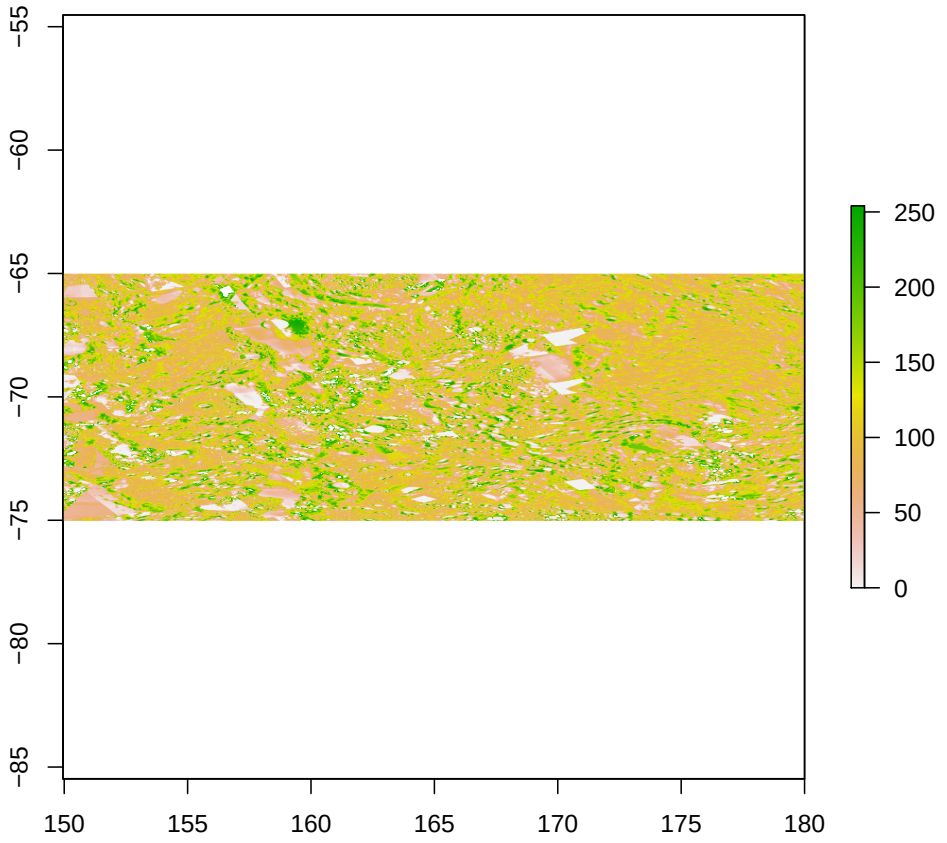

```
xfile = '../out/MapV5_42_geo.tif'
xmap = raster( xfile )
xmapeqc = projectRaster( xmap,
                         crs='+proj=longlat +lat_0=0 +lon_0=0 +R=1737400',
                         method='bilinear' )
plot( xmapeqc, asp=T )
```




```
xfile = '../out/MapV5_57_geo.tif'
xmap = raster( xfile )
xmapeqc = projectRaster( xmap,
                         crs='+proj=longlat +lat_0=0 +lon_0=0 +R=1737400',
                         method='bilinear' )
plot( xmapeqc, asp=T )
```




```
xfile = '../out/MapV5_72_geo.tif'
xmap = raster( xfile )
xmapeqc = projectRaster( xmap,
                         crs='+proj=longlat +lat_0=0 +lon_0=0 +R=1737400',
                         method='bilinear' )
plot( xmapeqc, asp=T )
```




```
xmapeqc72 = xmapeqc
```

3.3.4 North polar area

The north polar maps where supposed to be in polaratereographic projection, but reversed engineering revealed that they are in fact in azimuthal equidistant projection. Moreover, they are mirrored around the diagonal from top left to bottom right.

Create a new raster for the map with georeferencing inforamtion (redefining the previously used raster for the Mercator maps):

```
geomap = raster( ncol = 6000, nrow = 6000 )
```

Read input file:

```
map = raster( '../dat/V6P/NorthPole.bmp' )
```

Set map boundaries (assuming azimuthal equidistant projection down to 60°N):

```
#xmin( geomap ) = -30 * kpd
#xmax( geomap ) =  30 * kpd
#ymin( geomap ) = -30 * kpd
#ymax( geomap ) =  30 * kpd
#d = 2 * r * ( sin( 30 * pi / 180 ) / ( 1 + cos( 30 * pi / 180 ) ) )
d = r *  30 * pi / 180
xmin( geomap ) = -d
xmax( geomap ) =  d
ymin( geomap ) = -d
ymax( geomap ) =  d
```

Apply azimuthal equidistant projection:

```
crs( geomap ) = paste( '+proj=aeqd', # stere',
                       '+lat_0=90',
                       paste( '+R=', r * 1000, sep = '' ),
                       '+units=km',
                       sep = ' ' )
```

Copy values from original map transposing rows and columns:

```
values( geomap ) = values( t( map ) )
```

Write output file:

```
writeRaster( geomap, '../out/Map_1-4_geo.tif', 'GTiff',
             datatype='INT1U', overwrite = TRUE )
```

Inspect the output file:

```
GDALinfo( '../out/Map_1-4_geo.tif' )
```

```
## rows        6000
## columns     6000
## bands       1
## lower left origin.x        -909.7005
## lower left origin.y        -909.7005
## res.x       0.3032335
## res.y       0.3032335
## ysign       -1
## oblique.x   0
## oblique.y   0
## driver      GTiff
## projection  +proj=aeqd +lat_0=90 +lon_0=0 +x_0=0 +y_0=0 +a=1737400 +b=1737400
## +units=km +no_defs
## file        ../out/Map_1-4_geo.tif
## apparent band summary:
##   GDType hasNoDataValue NoDataValue blockSize1 blockSize2
## 1   Byte           TRUE         255          1       6000
## apparent band statistics:
##   Bmin Bmax    Bmean     Bsd
## 1    0  255 82.84658 34.0178
## Metadata:
## AREA_OR_POINT=Area
```

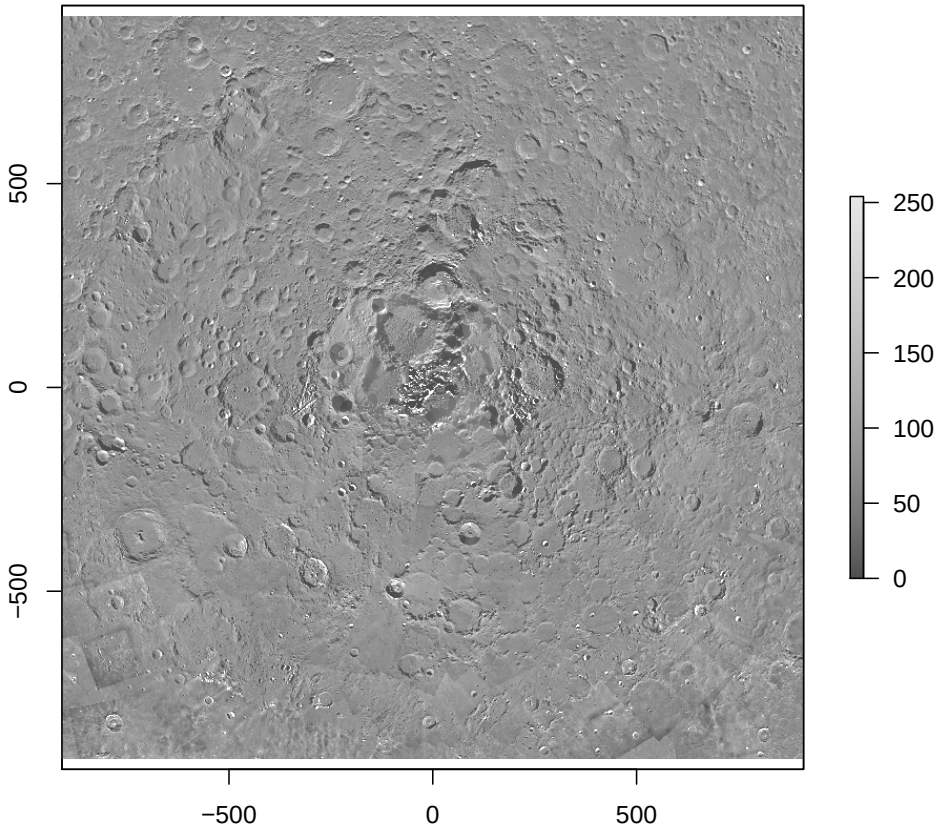
Re-read output file:

```
xmap = raster( '../out/Map_1-4_geo.tif' )
xmap
```

```
## class      : RasterLayer
## dimensions : 6000, 6000, 3.6e+07  (nrow, ncol, ncell)
## resolution : 0.3032335, 0.3032335  (x, y)
## extent     : -909.7005, 909.7005, -909.7005, 909.7005  (xmin, xmax, ymin, ymax)
## crs        : +proj=aeqd +lat_0=90 +lon_0=0 +x_0=0 +y_0=0 +a=1737400 +b=1737400 +units=km +no_defs
## source     : Map_1-4_geo.tif
## names      : Map_1.4_geo
## values     : 0, 255  (min, max)
```

Plot it in original polarstereographic projection:

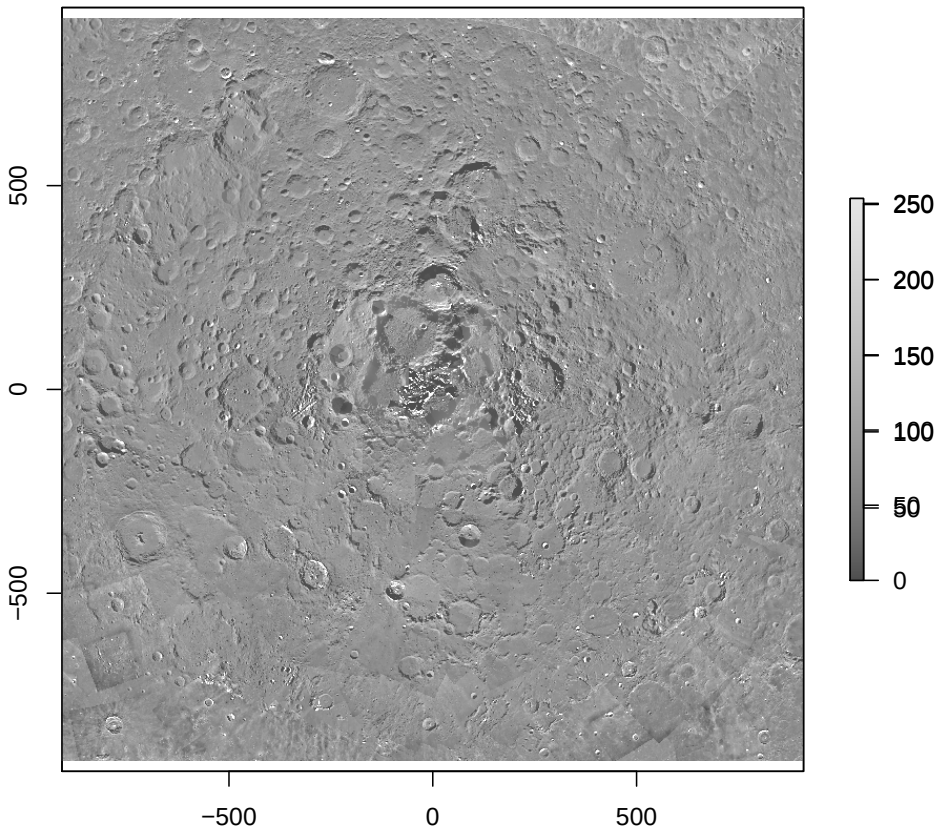
```
plot( xmap, col=gray.colors(256) )
plot( xmap, col=gray.colors(256) )
```



And with added reprojected Mercator map:

```
xmap12 = projectRaster(
    xmapeqc12,
    xmap,
    method='bilinear' )
```

```
plot( xmap, col=gray.colors(256) )
plot( xmap12, add=T, col=gray.colors(256) )
```

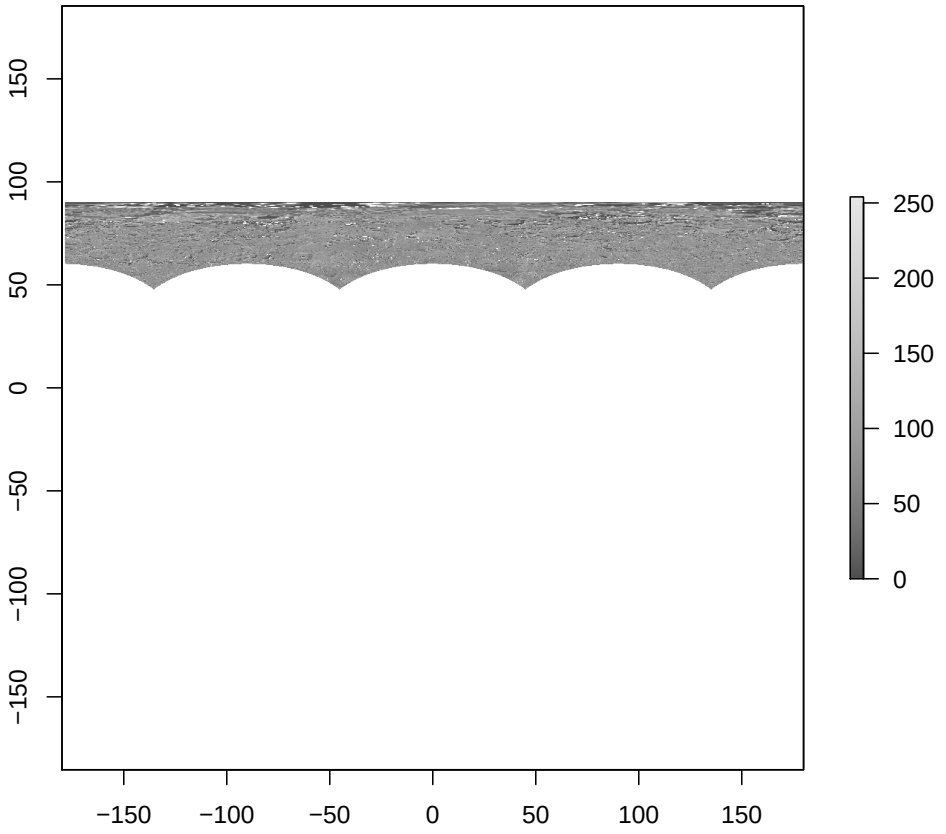


Reproject to equidistant cylindrical projection:

```
xmapeqc = projectRaster(
    xmap,
    projectExtent(
        raster( ncol = 6000, nrow = 3000 ),
        crs='+proj=longlat +lat_0=0 +lon_0=0 +R=1737400'
        ),
    method='bilinear' )
```

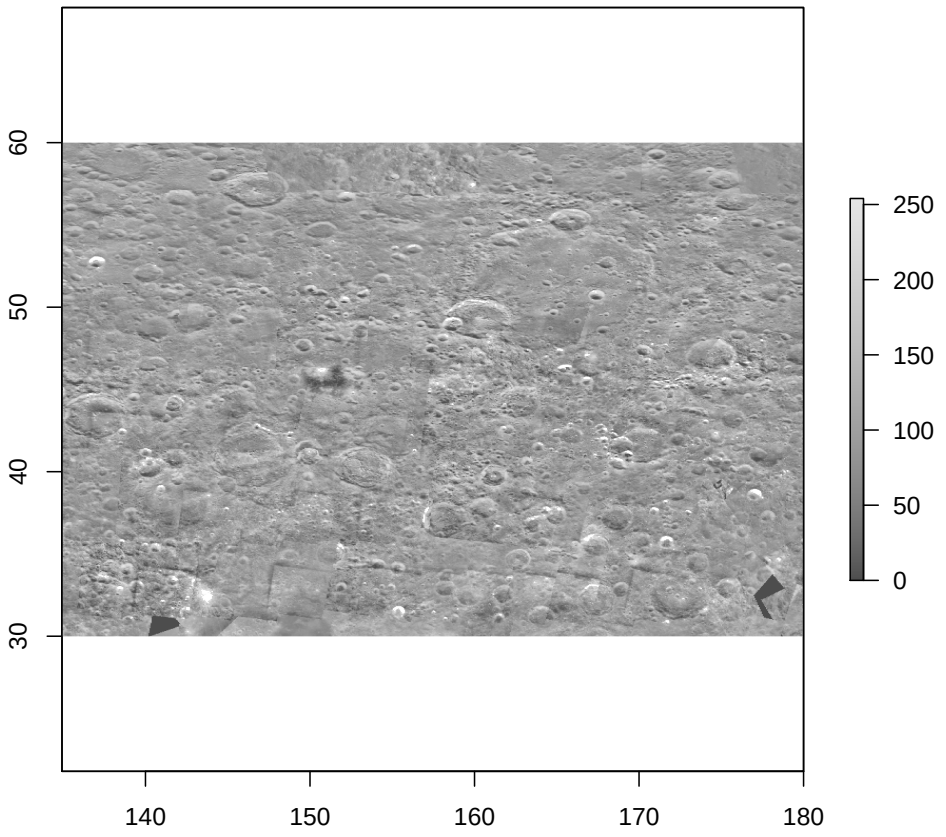
Plot it in equidistant cylindrical projection:

```
plot( xmapeqc, col=gray.colors(256) )
```



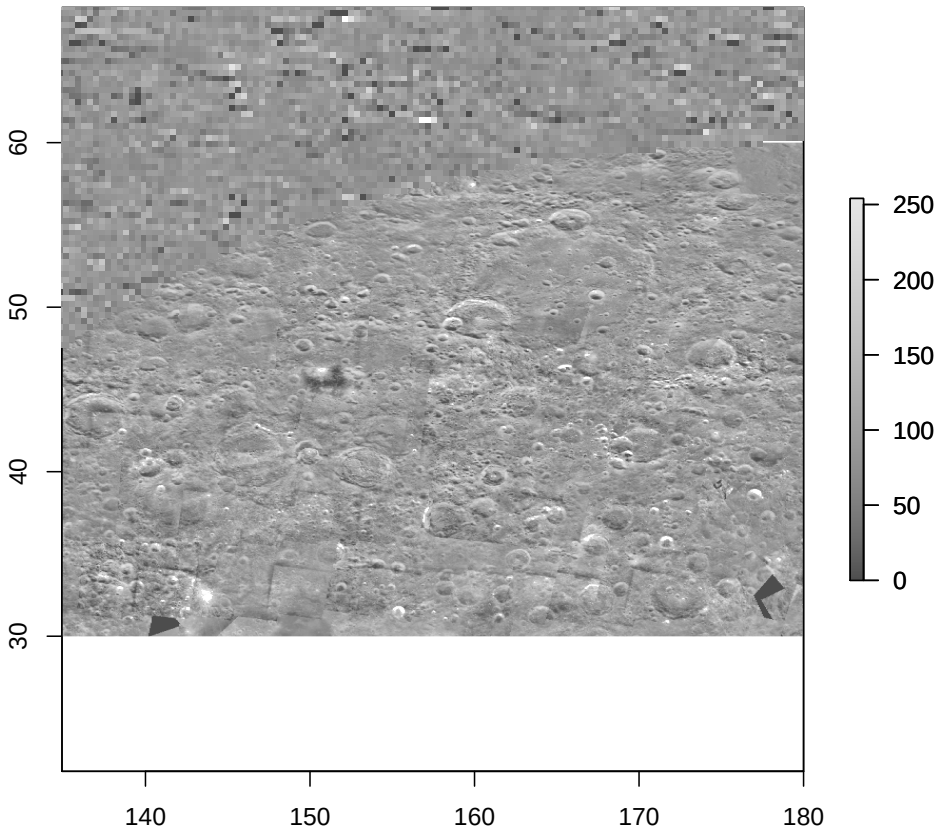
Plot one of the northmost Mercator maps:

```
plot( xmapeqc12, asp=T, col=gray.colors(256) )
plot( xmapeqc12, asp=T, col=gray.colors(256) )
```



And then with the former polarstereographic map added:

```
plot( xmapeqc12, asp=T, col=gray.colors(256) )
plot( xmapeqc, add=T, col=gray.colors(256) )
```



3.3.5 South polar area

The south polar maps where also supposed to be in polaratereographic projection. I assume that — like the north polar maps — they are in fact in azimuthal equidistant projection but not mirrored around the diagonal from top left to bottom right.

(Re-)Create a raster for the map with georeferencing inforamtion:

```
geomap = raster( ncol = 6000, nrow = 6000 )
```

Read input file:

```
map = raster( '../dat/V6P/SouthPole.bmp' )
```

Set map boundaries (assuming azimuthal equidistant projection down to 60°N):

```
d = r * 15 * pi / 180
xmin( geomap ) = -d
xmax( geomap ) =  d
ymin( geomap ) = -d
ymax( geomap ) =  d
```

Apply azimuthal equidistant projection:

```
crs( geomap ) = paste( '+proj=aeqd',
                       '+lat_0=-90',
                       paste( '+R=', r * 1000, sep = '' ),
                       '+units=km',
                       sep = ' ' )
```

Copy values from original map transposing rows and columns:

```
values( geomap ) = values( t( flip( map, direction='x' ) ) )
```

Write output file:

```
writeRaster( geomap, '../out/Map_73-88_geo.tif', 'GTiff',
             datatype='INT1U', overwrite = TRUE )
```

Inspect the output file:

```
GDALinfo( '../out/Map_73-88_geo.tif' )
```

```
## rows        6000
## columns     6000
## bands       1
## lower left origin.x        -454.8503
## lower left origin.y        -454.8503
## res.x       0.1516168
## res.y       0.1516168
## ysign       -1
## oblique.x   0
## oblique.y   0
## driver      GTiff
## projection  +proj=aeqd +lat_0=-90 +lon_0=0 +x_0=0 +y_0=0 +a=1737400 +b=1737400
## +units=km +no_defs
## file        ../out/Map_73-88_geo.tif
## apparent band summary:
##   GDType hasNoDataValue NoDataValue blockSize1 blockSize2
## 1   Byte           TRUE         255          1       6000
## apparent band statistics:
##   Bmin Bmax    Bmean      Bsd
## 1    0  255 78.01629 60.28187
## Metadata:
## AREA_OR_POINT=Area
```

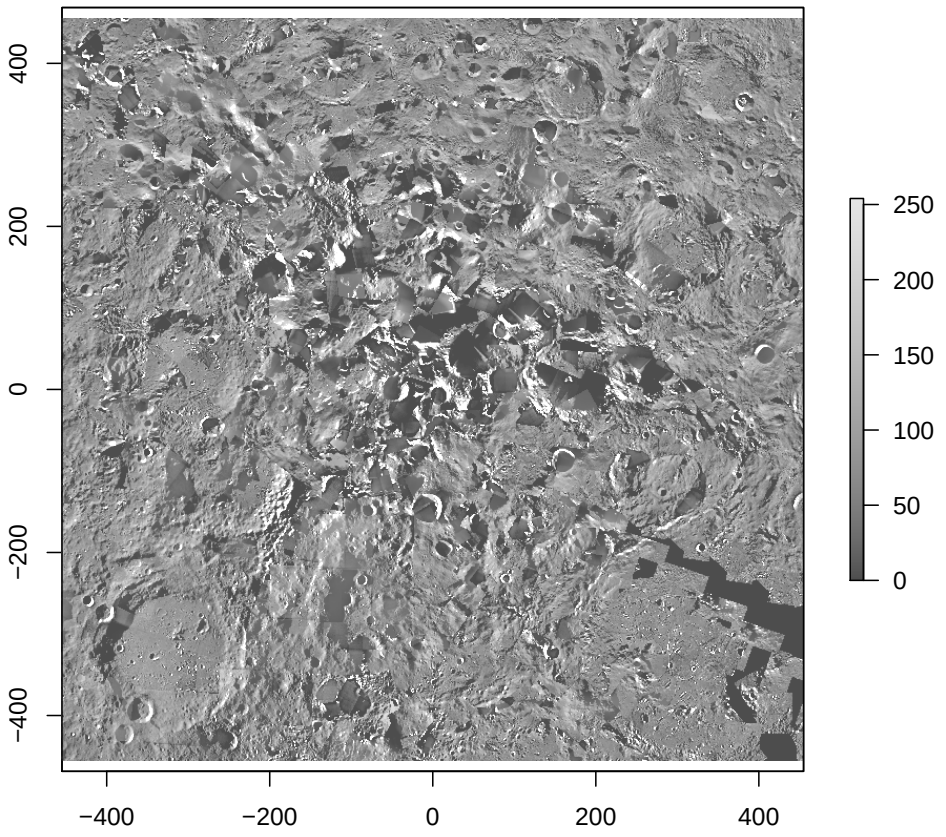
Re-read output file:

```
xmap = raster( '../out/Map_73-88_geo.tif' )
xmap
```

```
## class      : RasterLayer
## dimensions : 6000, 6000, 3.6e+07  (nrow, ncol, ncell)
## resolution : 0.1516168, 0.1516168  (x, y)
## extent     : -454.8503, 454.8503, -454.8503, 454.8503  (xmin, xmax, ymin, ymax)
## crs        : +proj=aeqd +lat_0=-90 +lon_0=0 +x_0=0 +y_0=0 +a=1737400 +b=1737400 +units=km +no_defs
## source     : Map_73-88_geo.tif
## names      : Map_73.88_geo
## values     : 0, 255  (min, max)
```

Plot it in original polarstereographic projection:

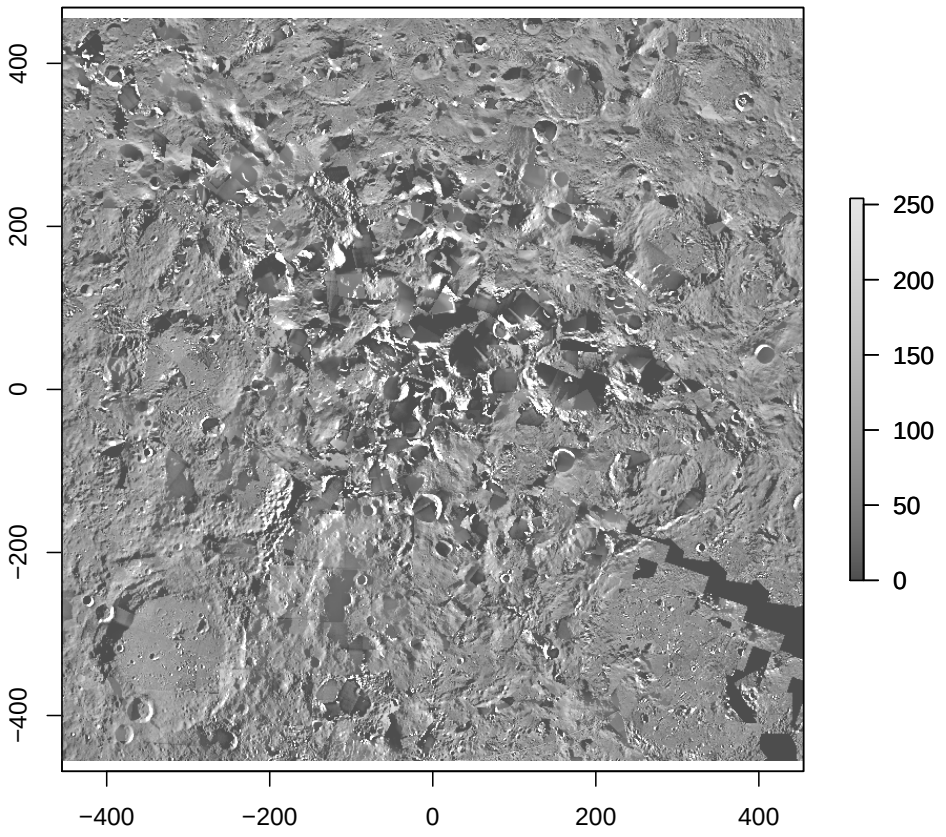
```
plot( xmap, col=gray.colors(256) )
plot( xmap, col=gray.colors(256) )
```



And with added reprojected Mercator map:

```
xmap72 = projectRaster(
    xmapeqc72,
    xmap,
    method='bilinear' )
```

```
plot( xmap, col=gray.colors(256) )
plot( xmap72, add=T, col=gray.colors(256) )
```



4 Archive Format and Content

<Here, please indicate to the users which archiving standard was used, e.g. PDS3 or PDS4, though it is possible your dataset may not use either of those 2 formats. The directory and file structures used in the dataset must be explained. For example, the DATA directory (in the case of PDS3) can include several levels of subdirectories, for separate instrument channels or to divide the data by different observation periods. The method which was used to subdivide the data should be explained here. All folders or collections are to be described in this section.>

<The structure on contents of the individual data products shall be explained here as well. For example, if you provide the data in binary format, an explanation of how to decode the data into human-legible form should be provided. Please keep in mind that this willbe read by new scientists many decades from now who are not familiar with your data yet, thus provide as much detail as you can.>

<Additionally, instructions to the users on how to open the products is very important. For example, it is useful to reference the tools that are best to use for these products (e.g., ArcGIS for shape files, etc...), and any specific information on how to open the product. The user should be able to open the product easily following your guidelines.>

I use '`dat`' as the root directory for the dataset und put there symbolic links to all included data. So I can access it independently of the final name of the dataset. The final name is a symbolic link in '`out`' pointing to '`dat`'. This is all done from the '`Makefile`'.

4.1 Data provided in image format

We have 606 downlinked images. The numbering is not consecutive because one of the two downlink channels did not work and about half of the images were lost. The images are provided in different processing levels. The orientation of lower level images followed the original pixel numbering from the CCD. This is called "a orientation". These images are mirrored, so you have to flip them left/right. Higher level images are rotated by 180° with respect to the lower level images. This is called "e orientation". These images need to be flipped up/down.

Details on the data format are provided below the list of images.

These are the available images:

a images: "raw" (but processed onboard, a orientation). The decompression results in pixel values in the range 0–255. These have been multiplied by 128 and then rounded to the nearest integer.

c images: calibrated but not smoothed (a orientation). Besides other processing detailed in section **??**, these images have been "inversely square rooted" to restore the range from 8 bit to 12 bit. Thus, the value range is 0–4095. These values have been multiplied by eight and rounded to the nearest integer.

e images: further processed to remove compression artifacts which implies some smoothing (e orientation), same value range as c images There are additional new versions of some images, with 1000 added to their image number.

g images: absolutely calibrated and distortion corrected (e orientation) There are two version, one in the V1.3 dataset and one in '`CHUCK_NEW`'. They differ, Erich's note on g image processing may explain that. I take the new version. In his note on g images, EK writes that the new scaling yielded pixel values above 32767, and these have to be interpreted as unsigned two-byte integers.

m images: "haze subtracted" (e orientation) Have spurious white patches; maybe they go below zero? There are only 588.

j images: m images devided by the atmospheric transmission (e orientation) There are only 587.

Then we have calibration data provided in image format individually for each image:

d images: modeled dark current (a orientation) Only the first 136. Afterwards, the dark current is supposed to be zero.

f images: improved flat fields (a orientation)

k images: atmospheric transmission (e orientation) Only 587.

All these images reside in respective directories named '`?_images`'.

Some calibration data is the same for all images from the same imager, so there are only three different images for the three imagers SLI, MRI, and HRI (or maybe a few more). All these images are in a directory named '`calibration`'.

Onboard flat fields: normalized to 10000 average (a orientation), files '`flatha`' (HRI), '`flatma`' (MRI), and '`flatsa`' (SLI) In '`DISR/FLATs`'.

Maps of distortion corrected images: (e orientation) files '`g-001`' (SLI), '`g-002`' (MRI), and '`g-003`' (HRI), '`g-004`', '`g-005`', '`g-006`'

All images are provided in Portable Gray Map (PGM) binary format. The binary data is preceded by an ASCII header. The header is 19 characters long and contains the magic number for PGM binary (`P5`), the number of columns, the number of rows, and the maximum pixel value. Header fields are separated by blanks. For PGM binary, the standard requires exactly one white space character between the header and the binary data. For almost all images herein, a Line Feed is used to separate header and data. However, the a images have a Data Link Escape character instead, which may not be interpreted as white space. Therefore, some image viewers — like the Linux Image viewer — cannot read these images. The file editor emacs can. When writing your own code to read the images, just skip the first 20 bytes for reading the binary data. This should work for all images herein.

The actual binary data is in two byte big endian format. PGM readers expect the data row by row from left to right, starting with the top row. The higher processed images (labeled with e orientation above) were intended to be in the real world orientation, but accidentally they were written starting with the bottom row. Therefore, if you use a standard PGM reader, you have to flip the images up/down. The lower processed images (labeled with a orientation above) follow the original pixel numbering of the CCD and are rotated upside down relative to the higher processed images, therefore you have to flip them left/right (and *not* up/down).

4.2 Data provided in ASCII table format

For each image, there is a "square routing" table for the onboard conversion from 12 bit to 8 bit.

There is one table providing the estimated compression parameters for each image.

File timeh.dd lists the image number of the HRI image and a transmitted value which is related to the pixel-independent dark value that is not included in the dark images.
The pixel-independent dark data number is 0.9 + 0.25 * value in table.
The image number of MRI and SLI images of the same exposure is 2 and 1 less than that of the HRI image.

4.3 Downlinked compressed data

A DESCRIPTION in the label files for the XDR files echoes header entries from the XDR file. These provide some information on the original ESOC files:

```
engineer:          Chuck See
ESOC_File:         C:\df3\15Jan05\ESOC_Files\o524sd__.1h_
```

5 Known Issues

<Here is the place to report any known issues, if applicable. This information is important to provide to the user to avoid mis-usage of your products.>

6 Software

<If there are any commonly available software applications that would be useful in working with this dataset, they shall be listed here, including links for users to download said software. Additionally, if you included your own custom code as part of the dataset delivery, details on how to use this code should be explained in this section.>

A Check of projections

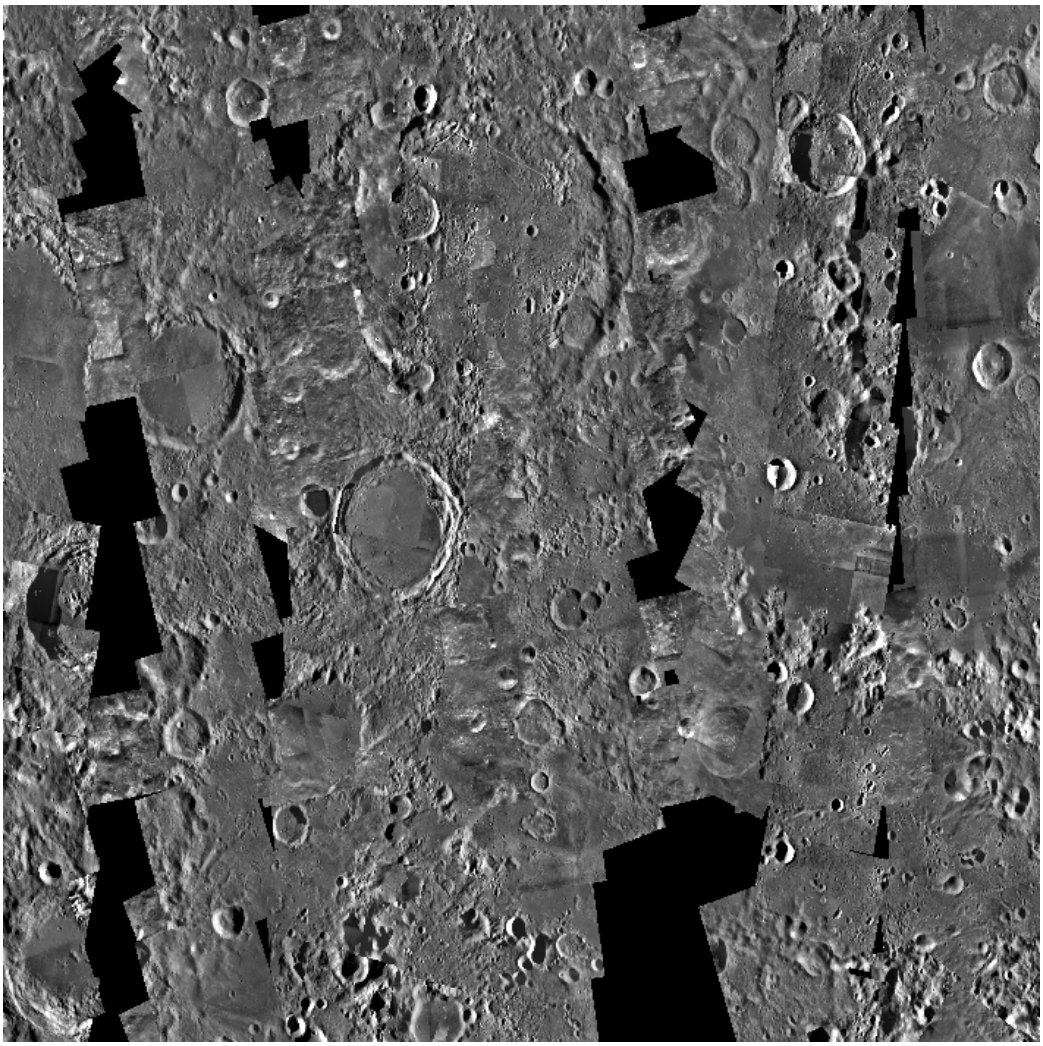
A.1 Mercator projection

The Mercator maps are all exactly square, 3000 × 3000 pixels. By design, they should be only approximately square. I was first thinking that some area was added to make them square, but boundaries seem to match.

Maps numbered 37–48 represent the latitude band with the largest portrait deviation from square.

Let's load and plot one:

```
img = raster( '../dat/V5/MapV5_37.bmp' )
plot( img )
```

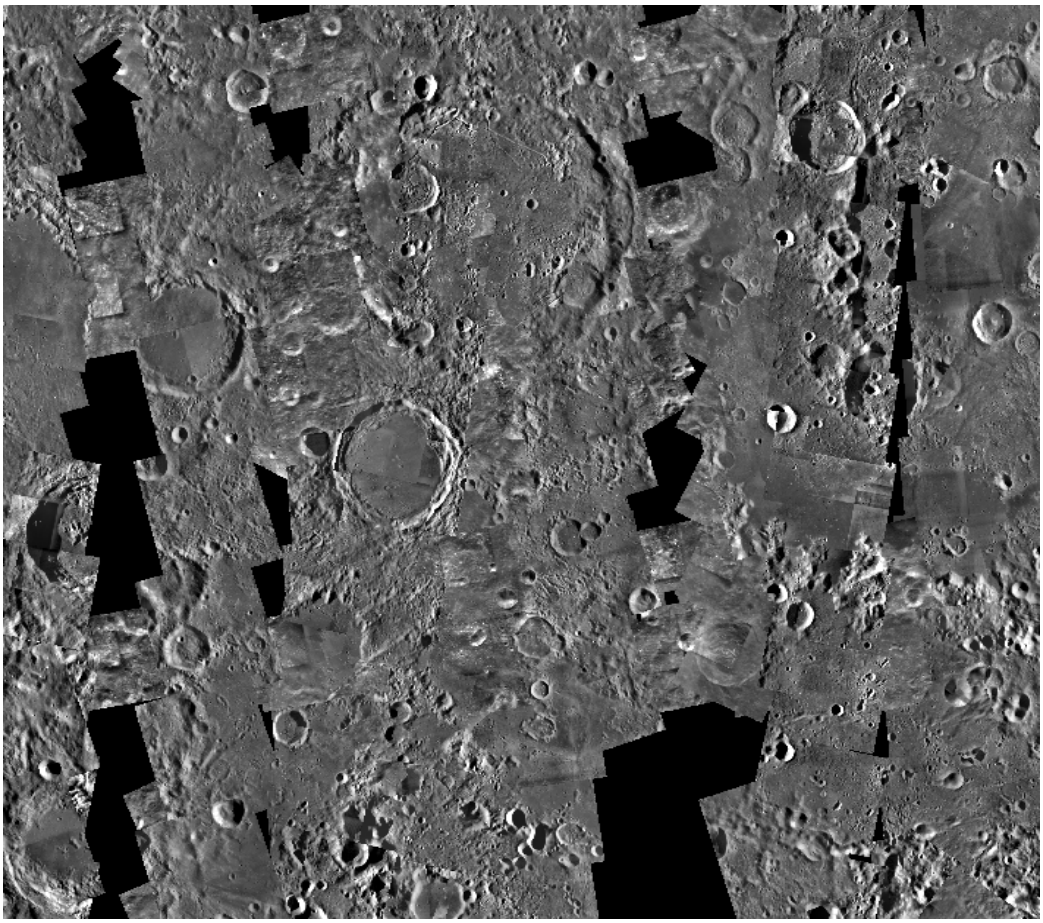


This looks vertically stretched. Probably, the nominal coordinates are actually honored, the image is just stretched to become square. This is a weird idea for me.

We apply Mercator projection and plot the image:

```
i = 37
xmin( img ) = w[i] * kpd
xmax( img ) = e[i] * kpd
ymin( img ) = r * log( tan( pi / 4 + s[i] * pi / 360 ) )
ymax( img ) = r * log( tan( pi / 4 + n[i] * pi / 360 ) )
crs( img ) = paste( '+proj=merc',
```

```
                    paste( '+R=', r * 1000, sep = '' ),
                    '+units=km',
                    sep = ' ' )
plot( img )
```



```
img
```

```
## class      : RasterLayer
## dimensions : 3000, 3000, 9e+06  (nrow, ncol, ncell)
## resolution : 0.3032335, 0.2671988  (x, y)
## extent     : -5458.203, -4548.503, -1755.961, -954.3645  (xmin, xmax, ymin, yma
## crs        : +proj=merc +R=1737400 +units=km
## source     : MapV5_37.bmp
## names      : MapV5_37
## values     : 0, 255  (min, max)
```

This looks OK, but there is a subtlety involved here. Assumed the maps were actually not created in Mercator projection, but in equidistant cylindrical projection, stretching them to square would approximately correct this. So that needs to be tested.

We load a map from the highest (Southern) latitude band:

```
    map = raster( '../dat/V5/MapV5_61.bmp' )
```

We create a georeferenced raster under the assumption that the map is in equidistant cylindrical projection:

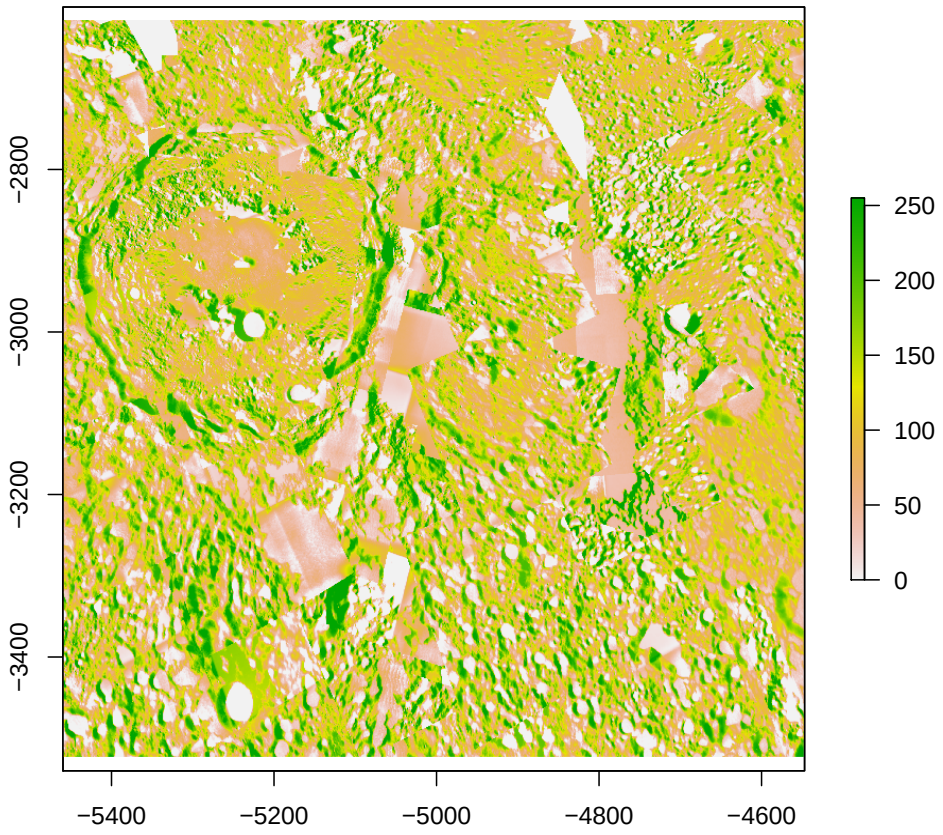
```
i = 61
geomap = raster( ncol = 3000, nrow = 3000 )
xmin( geomap ) = w[i]
xmax( geomap ) = e[i]
ymin( geomap ) = s[i]
ymax( geomap ) = n[i]
crs( geomap ) = paste( '+proj=longlat', #merc',
                       paste( '+R=', r * 1000, sep = '' ),
#                        '+units=km',
                       sep = ' ' )
values( geomap ) = values( map )
```

We remap it to Mercator projection:

```
remap = projectRaster( geomap,
                       crs='+proj=merc +R=1737400 +units=km',
                       method='bilinear' )
```

And we plot it:

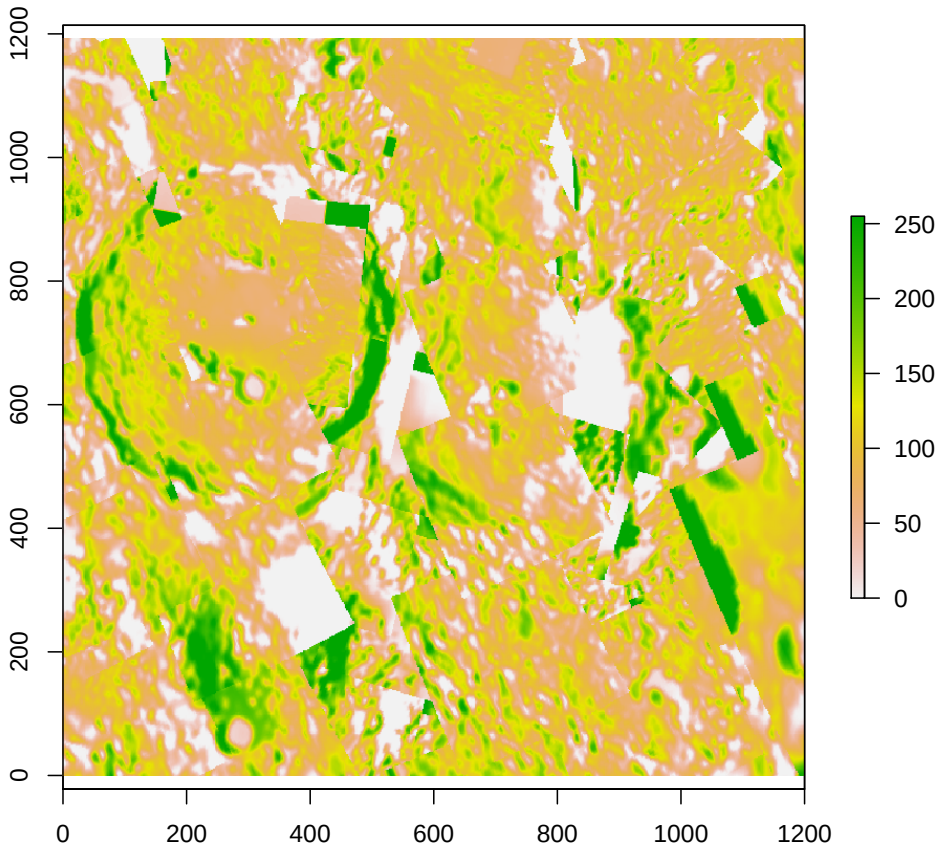
```
plot( remap )
```




```
draftmap = raster( '../../ATLAS/fig/map611.tiff' )
```

We have here also loaded an original draft map, which is guarantied to be in Mercator projection and plot that for comparison.

```
plot( draftmap )
```



These are clearly different.

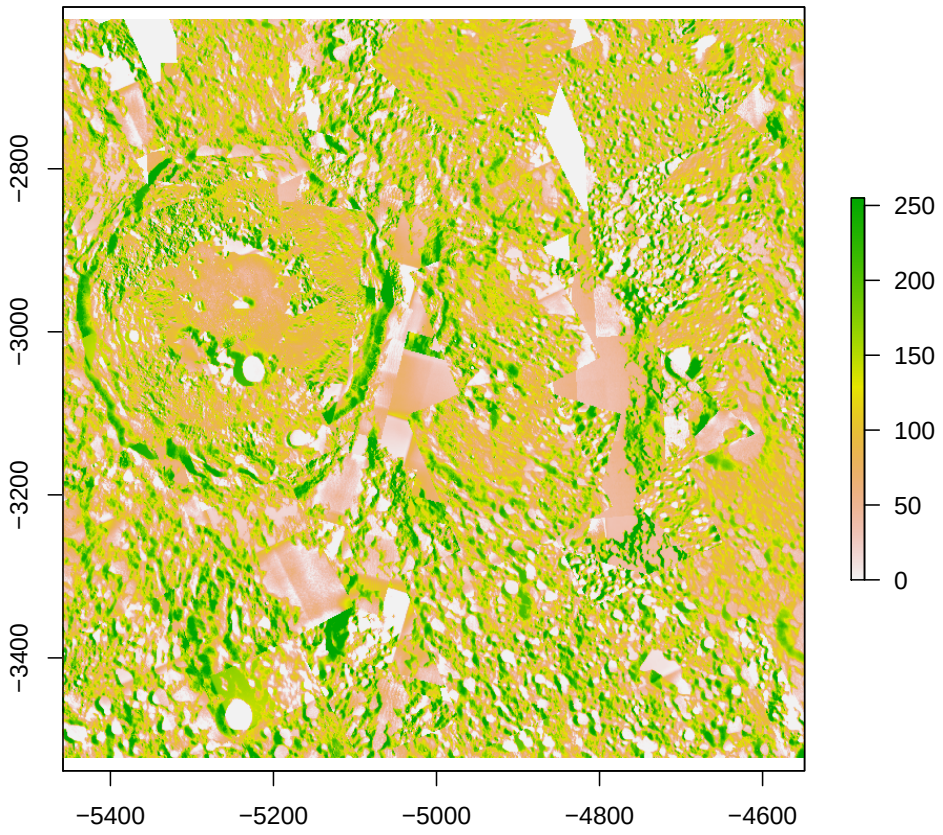
Now we assume that the new map is in Mercator projection:

```
xmin( geomap ) = w[i] * kpd
xmax( geomap ) = e[i] * kpd
ymin( geomap ) = r * log( tan( pi / 4 + s[i] * pi / 360 ) )
ymax( geomap ) = r * log( tan( pi / 4 + n[i] * pi / 360 ) )
crs( geomap ) = paste( '+proj=merc',
```

```
                        paste( '+R=', r * 1000, sep = '' ),
                        '+units=km',
                        sep = ' ' )
```

And we plot it:

```
plot( geomap )
```



This matches prefectly the draft map, so the new maps are indeed in Mercator projection, just stretched to become square.

B Modifying and re-creating this document

TBD

END OF PUG