# DUGR GUI Evaluation Steps

This document describes the steps that are performed for the calculation of the DUGR value.

It distinguishes between the two algorithms (with & without projective rectification) which are implemented in the GUI.

## Approach without projective rectification:

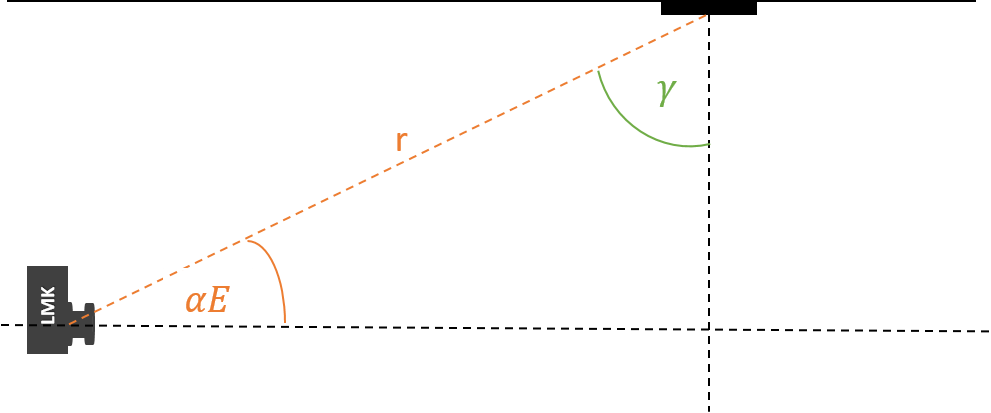
1. **Load Image**

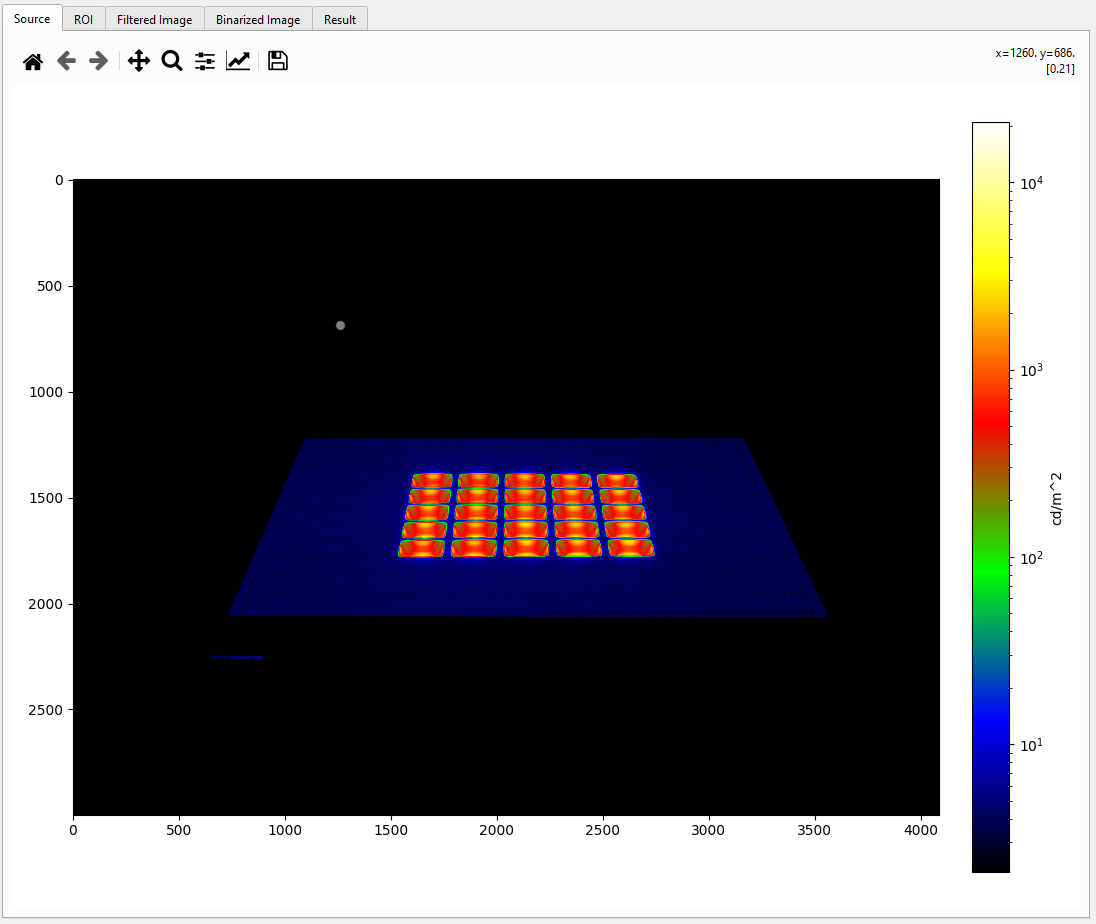
The first thing to do is to load a luminance image into the GUI.  
Currently there are two image formats, which are supported by the software.

* The TechnoTeam luminance image format: „picture float“ (\*.pf)
* An Ascii image format that stores the luminance image information in a text file (\*.txt)

1. **Enter Parameters**

The parameters are needed in order to execute the calculation of the DUGR value.  
Each of them will be described briefly in this section.

* Luminance Threshold:  
  The luminance threshold defines the threshold used to define the binarization level and thus the pixels associated with the illuminated area.  
  It defaults to the value of
* Focal Length:  
  This parameter represents the focal length of the optical system which is used for the measurement.  
  It is needed to generate a solid angle image (representing the solid angle covered by each pixel).
* Pixel Size:  
  Describes the size of each pixel on the optical systems sensor (also known as pixel pitch).  
  Also needed for the generation of the solid angle image.  
  (It is assumed that each pixel is square)
* “Eye Resolution” d:  
  Describes the minimum feature diameter.   
  According to CIE232:2019, this was set at 12mm to reflect the worst case.
* Measurement angle :  
  This describes the angle between the optical measurement system (observer) and the horizontal plane of the luminaire, as can be seen in the following illustration  
    
  🡪
* Viewing Distance:  
  The viewing distance is equal to the r in the illustration above. Ideally it would be the distance between the entrance pupil of the optical measurement system and the center of the luminaire.
* Luminous Intensity I:  
  **This parameter is optional**. If a luminous intensity is entered, two DUGR (k^2, A\_new) values are calculated. The use of the luminous intensity for this calculation is questionable, since the values under the relevant angles are often faulty because of dynamic issues. Furthermore the luminous intensity is a quantity which is measured in the far field. The measurement of the DUGR value is typically associated with the near field.  
  If the luminous intensity is left at 0, only the calculation based on the luminance image is considered
* Luminous area width:  
  This parameter describes the physical width of the luminous area.  
  The width always corresponds to the x dimension of the luminance image.
* Luminous area height:  
  This parameter describes the physical height of the luminous area.  
  The height always corresponds to the y dimension of the luminance image.



Luminous area width

Luminous area height

1. **Draw ROIs**

In order to calculate the DUGR value correctly, it is necessary to define the regions, which are part of the luminous area according to the manufacturer.  
It is possible to use the zoom function (magnifying glass symbol) to set the points more precisely.  
If needed it´s also possible to select multiple ROIS.  
There is also an option “Filter only ROIs”, which can be used to filter only the defined regions instead of the whole image (Can safe computational time).

1. **Calculate Result**

Since the main goal of this document is, to explain the calculation and to make it comprehensible, the calculation is divided into further substeps.

* 1. **Camera Model / Solid angle image**

In order to avoid to have a huge number of parameters to enter before starting the calculation, the solid angle (image) calculation is based on a rather simple pinhole camera model.

At first the and angle images are calculated based on the following function:

def cart2theta\_phi(focal\_length, pixel\_size, image, opt\_x=None, opt\_y=None):  
 *"""  
 Function to retrieve theta and phi angles from cartesian coordinates and optical axis  
  
 opt\_x and opt\_y define the coordinates of the optical axis in the image, default value is in the middle of the image  
  
 Args:  
 focal\_length: Focal length of the camera  
 pixel\_size: Pixel size of the camera sensor  
 image: Input image  
 opt\_x: x-Coordinates of the optical axis  
 opt\_y: y -Coordinates of the optical axis  
  
 Returns:  
 theta: Numpy array representing the theta angle of each pixel  
 phi: Numpy array representing the phi angle of each pixel  
 """* if not opt\_x:  
 opt\_x = image.shape[1] // 2  
 if not opt\_y:  
 opt\_y = image.shape[0] // 2  
  
 theta = np.zeros(image.shape)  
 phi = np.zeros(image.shape)  
 for n in range(image.shape[0]):  
 for m in range(image.shape[1]):  
 theta[n][m] = np.degrees(atan2(sqrt(((opt\_x-m)\*pixel\_size)\*\*2 + ((opt\_y-n)\*pixel\_size)\*\*2), focal\_length))  
 phi[n][m] = atan2(radians(opt\_y-n), radians(opt\_x-m))  
  
 return theta, phi

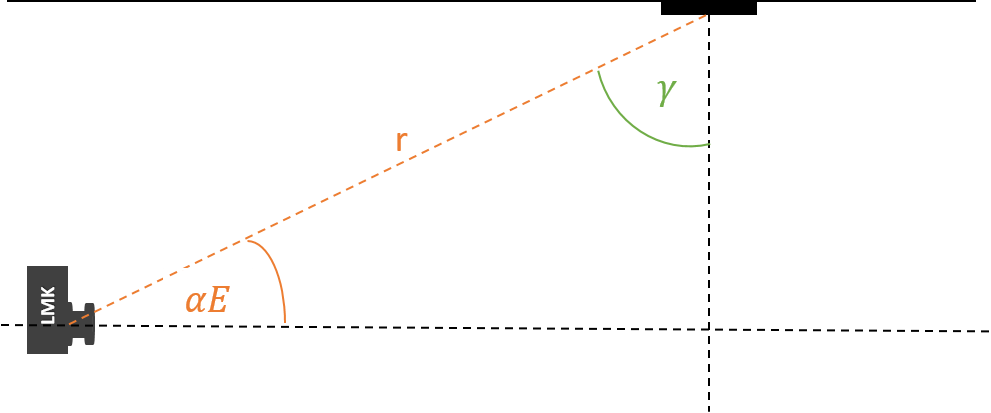
After that the following steps are executed to calculate the solid angle () image.

# Convert Theta to radians  
theta\_rad = np.radians(theta)  
  
# Calculate the sin theta image  
sin\_theta\_rad = np.sin(theta\_rad)  
  
# Define Filter kernels for horizontal and vertical filtering of an image  
kernel\_h = np.array([[0, 0, 0],  
 [-1, 0, 1],  
 [0, 0, 0]])  
  
kernel\_v = np.array([[0, -1, 0],  
 [0, 0, 0],  
 [0, 1, 0]])  
  
# Steps for euclidian distance calculation:  
# 1. Calculate one horizontal and one vertical filtered image  
theta\_filtered\_h = (filter2D(theta, -1, kernel\_h)) / 2  
theta\_filtered\_v = (filter2D(theta, -1, kernel\_v)) / 2  
  
# 2. Square the filterd Images  
pow\_theta\_filtered\_h = theta\_filtered\_h \*\* 2  
pow\_theta\_filtered\_v = theta\_filtered\_v \*\* 2  
  
# 3. Sum of the squared images  
theta\_add = pow\_theta\_filtered\_h + pow\_theta\_filtered\_v  
  
# 4. Square root of the sum  
theta\_diff = np.sqrt(theta\_add)  
  
# Convert the euclidian distance image to radians  
theta\_diff\_arc = np.radians(theta\_diff)  
  
# Calculate the cartesian distance to the image optical axis  
cart\_dist\_img = img2cart\_dist\_img(image=src\_image)  
  
# Divide the euclidian theta distance in radians by the cartesian distance  
# Ignore warnings for 0 division because we set our NAN element to 0 manually  
with np.errstate(divide='ignore', invalid='ignore'):  
 theta\_diff\_arc\_by\_cart\_dist = np.nan\_to\_num(theta\_diff\_arc / cart\_dist\_img)  
  
# Calculate the omega image (Solid angle image)  
omega = theta\_diff\_arc\_by\_cart\_dist \* sin\_theta\_rad

This approach of the solid angle calculation is based on a macro which is implemented in the Measurement Software LabSoft from TechnoTeam and the results have been crosschecked.

* 1. **Filtering of the image / binarization**

The first step in order to calculate the filter parameters is to evaluate the parameters:



Because the resolution is not constant over the whole image, the resolution at 5 degrees was taken as an approximation.

r\_5deg = (tan(radians(5.0)) \* focal\_length)/pixel\_size  
r\_o = 5.0 / r\_5deg

The filter parameters can then be calculated as follows.

If the option “Filter only ROI” is selected the defined ROI is extended by half of the filter size to make sure extension of the luminous area due to blurred pixels is taken into account.

If the option is not selected, the whole image is filtered.

After that the binarization is executed based on the luminance threshold parameter defined earlier.

* 1. **Calculation of the result**
* **Effective solid angle :** The pixel solid angles of the pixels above the threshold are summed up
* **Effective Luminance :** The luminance values above the threshold are summed up and the mean value is estimated
* **Mean Luminance of the luminous area** : The mean luminance of the whole luminous area is evaluated
* **Luminous area solid angle** : The solid angle of the whole luminous area is estimated
* **Correction factor :**