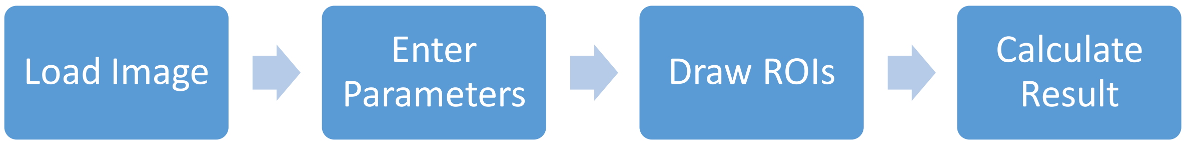
DUGR GUI Evaluation Steps

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)

2. **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.

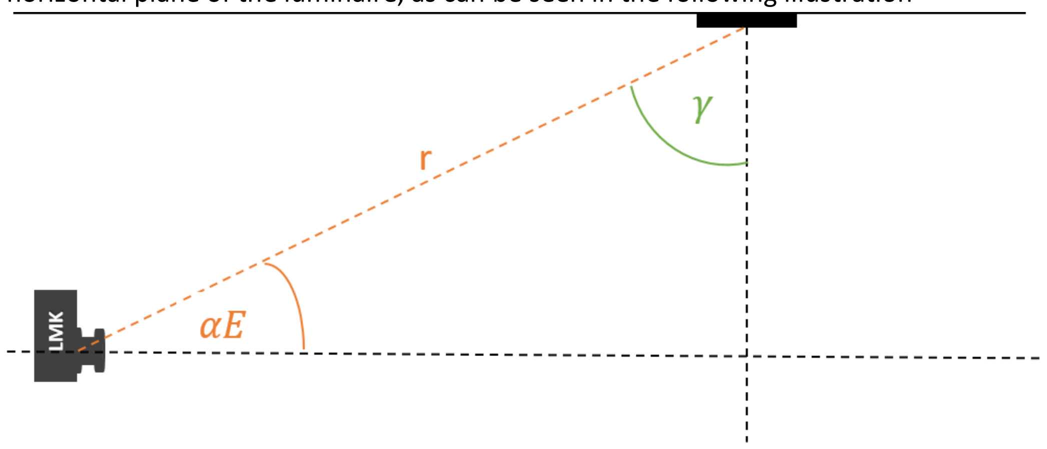
- Luminance Threshold:
  The luminance threshold defines the threshold used to define the binarization level and thus the pixels associated with the illuminated area.
  It defaults to the value of $500\,\frac{cd}{m^2}$

- 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 αE:
  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



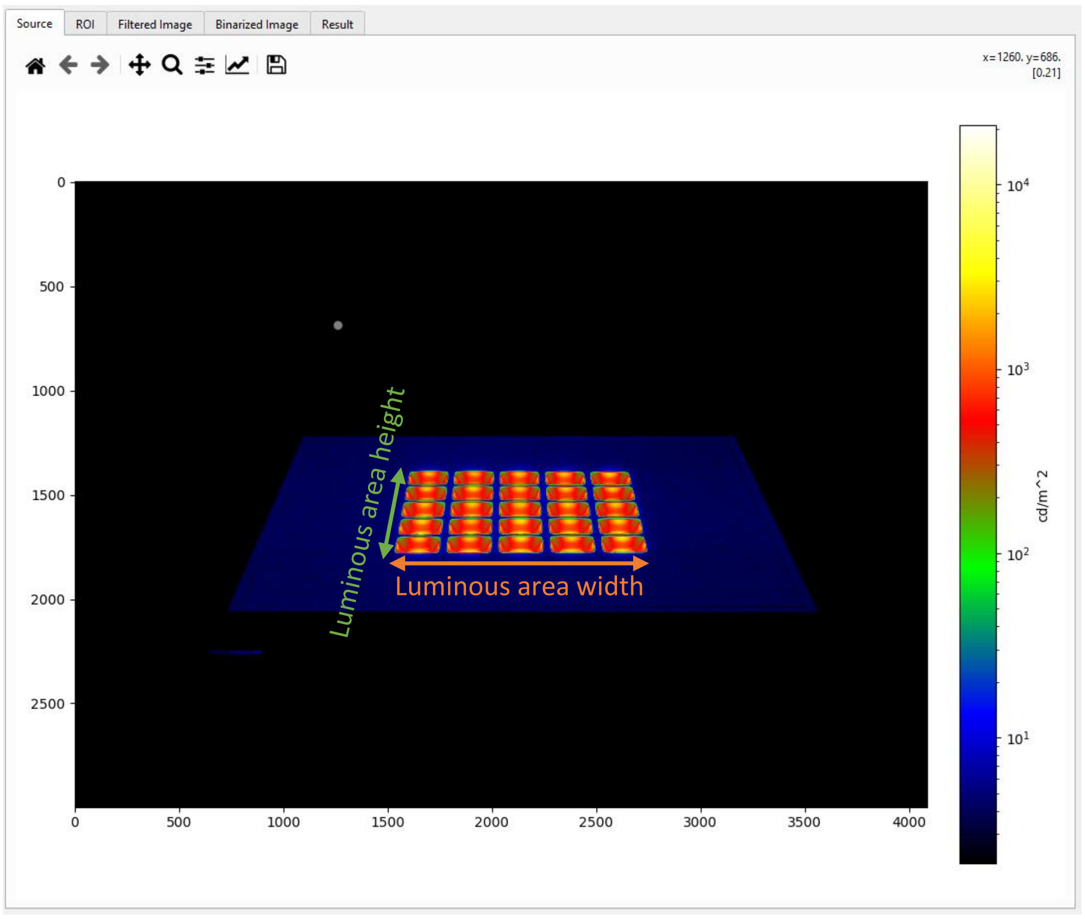
→ $\alpha E = 90° - \gamma$

- 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.

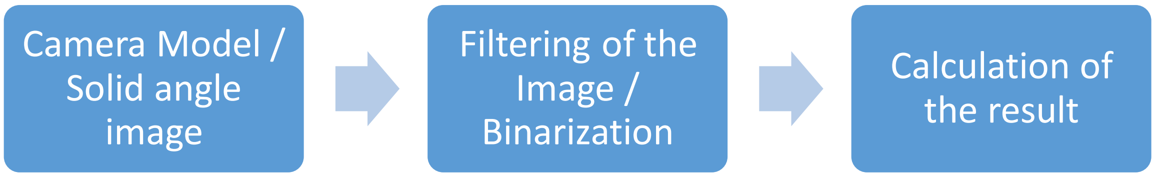


3. Draw ROIs

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).

4. 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.


Camera Model / Solid angle image
Filtering of the Image / Binarization
Calculation of the result

4.1. 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 θ and ϕ 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 (Ω) 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]])
```

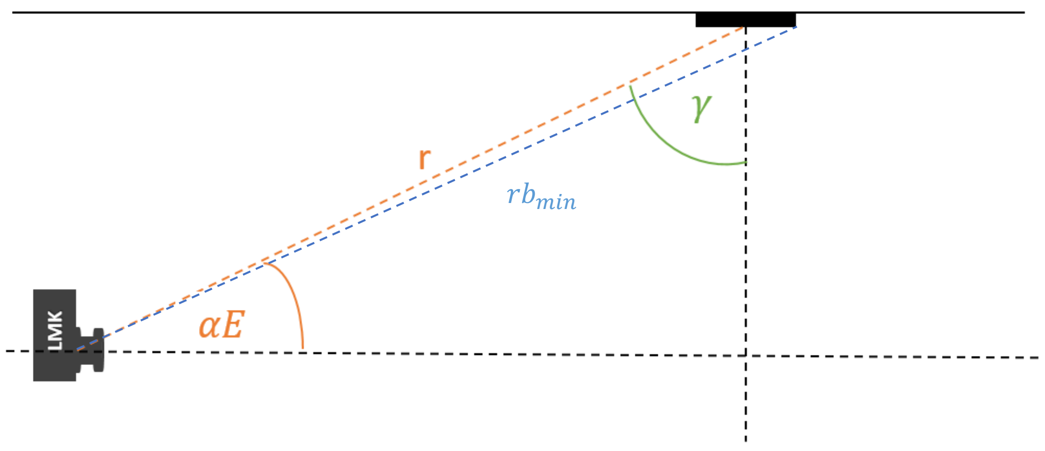
```
#  Steps for euclidian distance calculation:
#  1. Calculate one horizontal and one vertical filtered image
theta_filtered_h = (filter2D(theta, -1, kernel_h)) / 2
theta_filtered_v = (filter2D(theta, -1, kernel_v)) / 2

#  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.

4.2. Filtering of the image / binarization

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

$$rb_{min}, ro_{min}, ro$$



$$rb_{min} = \sqrt{r^2 + \left(\frac{luminous\ area\ height}{2}\right)^2 + r \cdot luminous\ area\ height \cdot \cos(\alpha E)}$$

$$ro_{min} = \frac{\arctan\left(\frac{d}{rb_{min}}\right)}{10}$$

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.

$$FWHM = \frac{ro_{min}}{ro}$$

$$\sigma = \frac{FWHM}{2{,}3584}$$

$$Filter\ width = 2 \cdot \lceil 3 \cdot \sigma \rceil + 1$$

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.

4.3. Calculation of the result

- **Effective solid angle $\boldsymbol{\omega_{eff}}$:** The pixel solid angles of the pixels above the threshold are summed up
- **Effective Luminance $\boldsymbol{L_{eff}}$:** The luminance values above the threshold are summed up and the mean value is estimated
- **Mean Luminance of the luminous area $\boldsymbol{L_s}$**: The mean luminance of the whole luminous area is evaluated
- **Luminous area solid angle $\boldsymbol{\omega_s}$**: The solid angle of the whole luminous area is estimated
- **Correction factor $\boldsymbol{k^2}$:**

$$\boldsymbol{k^2 = \frac{L_{eff}^2 \cdot \omega_{eff}}{L_s^2 \cdot \omega_s}}$$

- $\boldsymbol{DUGR}$:

$$\boldsymbol{DUGR = 8 \cdot log_{10}(k^2)}$$

- $\boldsymbol{A_{eff}}$:

$$\boldsymbol{A_{eff} = \omega_{eff} \cdot r^2}$$

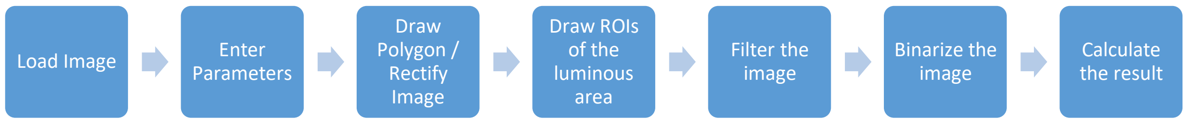
- $\boldsymbol{Ap_{new}}$:

$$\boldsymbol{Ap_{new} = \frac{Ap}{k^2}}$$

- $\boldsymbol{A_{new}}$:

$$\boldsymbol{A_{new} = \frac{Ap_{new}}{cos(90° - \alpha E)}}$$

Approach with 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)

2. 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 $500\,\frac{cd}{m^2}$

- Gauss filter FWHM:
  Describes the full width at half maximum in pixels of the gaussian filter kernel.

- Rectification width:
  The width (x – dimension of the image) which is used for the rectification. Please note that if the luminaire has multiple separatable luminous areas, it might be useful to use something like the housing for the rectification.
  In order to evaluate reliable results the dimensions should be measured and drawn onto the image precisely.

- Rectification height:
  The height (y – dimension of the image) which is used for the rectification.

If you want to enter the size of the luminous are directly instead of evaluating it from the rectified image and the parameters for the rectification width and height, the checkbox “Use Luminous Area Parameters” with the two parameters “luminous area width” and “luminous area height” can be used.

3. Draw Polygon / Rectify the image

In order to rectify the image under perspective, 4 polygon points of the edges have to be selected. This can be done by clicking on the image under the source tab. To set the points more precisely, the zoom function (magnifying glass icon) can be used.

If the 4 points are selected, the image rectification can be executed by pressing the projective Transformation button.

The rectification is based on the opencv functions:

getPerspectiveTransform():
https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html#ga8c1ae0e3589a9d77fffc962c49b22043

➔ Used to calculate the transformation matrix

warpPerspective():
https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html#gaf73673a7e8e18ec6963e3774e6a94b87

➔ Used to apply the transformation matrix
➔ The parameters entered for rectification width and height are used to scale to the new image size (-> 1mm/px)

4. Draw ROIs of the luminous area

The next step is to select the ROIs of the luminous areas. This can be done by selecting either rectangular or circular from the "ROI shape" dropdown menu and then drawing the ROIs onto the image in the rectified image tab. By using the button "safe ROI" the previously drawn ROI is added to the ROI tab.

5. Filter the image

After the selection of the ROIs the gaussian image filter can be applied by pressing the button filter image. Because the image is rescaled to a resolution of 1mm/px the following parameters are used for the calculation:

$$FWHM = 12[px]$$

$$\sigma = \frac{FWHM}{2{,}3548}$$

$$filter\ width = 2 \cdot \lceil 3 \cdot \sigma \rceil + 1$$

Each of the ROIs is extended by half of the filter width to ensure that blurred pixels due to the filtering are taken into account.

6. Binarize the image

By pressing the button "Binarize ROI(s) the threshold defined earlier is used to binarize the filtered ROIs

7. Calculate the result

If all the steps from above were executed, the calculation of the result can be done by pressing the button calculate DUGR.

- From the ROIs the area A and the mean luminance L_s is evaluated.

- From the binarized ROIs the effective area A_{eff} and the effective luminance L_{eff} is evaluated by using the luminance threshold

With the help of these parameters, the following can be calculated:

$$k^2 = \frac{L_{eff}^2 \cdot A_{eff}}{L_s^2 \cdot A}$$

$$A_{new} = \frac{A}{k^2}$$

$$DUGR = 8 \cdot \log_{10}(k^2)$$