# DUGR GUI

This document describes how the DUGR GUI, currently under development, could be used to determine the extended UGR value according to CIE 232:2019.

The GUI is still evolving, neither the process nor the frontend is final. Bugs and errors may occur.

1. Choose whether the perspective distortion should be corrected or not

From v\_0.4 onwards 2 algorithms are implemented to calculate the DUGR value.

One that performs perspective correction and one that works on distorted images.  
You can use the checkbox to swap between both algorithms.

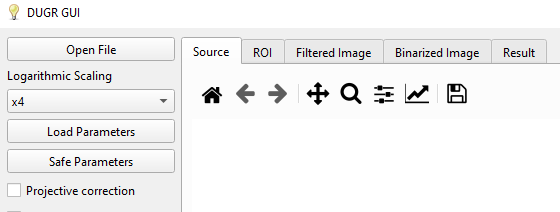


Figure 1: Checkbox for projective correction and "open file" button

1. Load a source file from the luminaire which is under evaluation

By using the “Open File” Button you will be able to select an image from the windows explorer.  
Currently supported image file formats are \*.pf files and pixel information stored in \*.txt files.

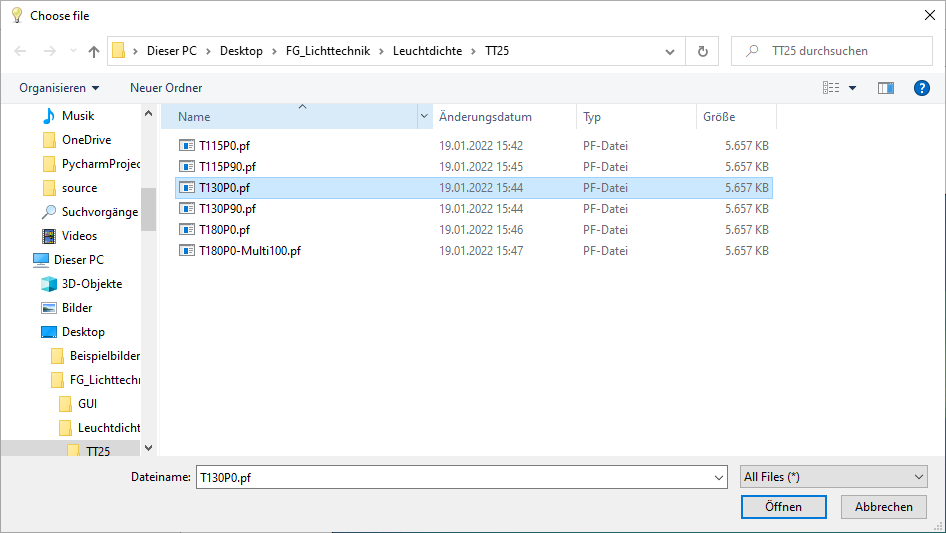


Figure 2: Loading a source file (\*.pf Image)

After loading an image the GUI Interface should look somewhat equal to figure 3.

(Depending on which algorithm is used the sidebar and tabs are different)

Ein Bild, das Text, Screenshot, Display, Software enthält.

Automatisch generierte Beschreibung

Figure 3: GUI Interface Source Tab projective distorted algorithm

Ein Bild, das Text, Screenshot, Display, Software enthält.

Automatisch generierte Beschreibung

Figure 4: GUI Interface Source Tab projective corrected algorithm

As of version 1.0, a colormap has been implemented as in the LabSoft software from TechnoTeam.

In addition, it is possible from version 1.0 to display different logarithmic scalings of the image to possibly ensure a better contrast.

Ein Bild, das Text, Screenshot, Display, Software enthält.

Automatisch generierte Beschreibung

Figure 5: Log Scaling: x3

Ein Bild, das Text, Screenshot, Display, Software enthält.

Automatisch generierte Beschreibung

Figure 6: Log Scaling: x5

1. Projective distorted algorithm

The sidebar of the projective distorted algorithm has 8 parameters which have to be filled before executing the calculation. **The Luminous Intensity is optional** and can be used to calculate a second result if given.

* Ein Bild, das Text, Zahl, Quittung, Screenshot enthält.

  Automatisch generierte Beschreibung**Luminance Threshold:** Threshold value to determine which pixels are assigned to the effective area and the effective luminance
* **Focal Length:** The focal length of the optical system (the lens). Together with pixel size this value is used to calculate the solid angles of each pixel.  
  Therefore, the parameter should be the distance between the entrance pupil and the sensor center. However, using the focal length of the lens seems to be an acceptable approximation for the time being.
* **Pixel Size:** This parameter is the actual size of the pixels on the sensor (Pixel pitch). As said before this is used to calculate the pixel solid angles.
* **Eye resolution:** Minimum feature diameter based on the resolution of the human eye. This parameter defaults to 12mm (As proposed in   
  CIE 232:2019
* **Measurement angle:** Measuring angle with respect to the horizontal
* **Viewing distance:** Distance between the measuring device and the center of the luminaire. (In an ideal scenario this distance would also be measured from the entrance pupil)
* **Luminous Intensity:** This parameter is optional and can be used to generate a second result if you have determined the luminous intensity under the measuring angle. First measurements with this parameter have shown that luminous intensities under the relevant measuring angles are mostly error-prone due to the magnitude.
* **Luminous area width & height:** Indication of the extent of the luminous area. (This can be obtained from a \*.ldt file). Please note that height and width are the extensions based on the image (height is corresponding to the y axis and width is corresponding to the x axis)!

From v\_0.6 onwards it´s possible to load the parameters from a predefined \*.json file. (This can be useful if you have several measurements with the same camera).

Afterwards, the button „Calculate DUGR“ can be used to start the calculation. The calculation steps of this algorithm are explained below.

If you want to load the 9 parameters from a preset \*.json file, the file has to look like the following:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Figure 7: \*.json file for parameter import

1. Calculation steps oft he projective distorted algorithm

In order to execute the DUGR calculation with the projective distorted algorithm, the first step hasto be drawing a region containing the luminaire onto the image.  
For a more precise drawing you can use the zoom function (magnifying glass icon).

Ein Bild, das Text, Screenshot, Display, Software enthält.

Automatisch generierte Beschreibung

Figure 8: Drawing the ROI onto the plot (projective distorted algorithm)

If the Region is drawn onto the image you can use the button safe ROI.

Ein Bild, das Screenshot, Text, Display, Farbigkeit enthält.

Automatisch generierte Beschreibung

Figure 9: ROI tab after saving the ROI drawn before

From v\_0.7 onwards you can decide to either filter and binarize the whole image or only a window around the drawn ROI. The window is rectangular and half the filter width bigger than the biggest extension of the ROI.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Figure 10: Filter only ROI checkbox

After saving the ROI and loading or filling out the parameters you can use the button “Calculate DUGR”.

In the tabs “Filtered Image” and “Binarized Image” you can check the result of the filtering and binarization operation after the calculation is completed.

Ein Bild, das Elektronik, Screenshot, Display, Text enthält.

Automatisch generierte Beschreibung

Figure 11: Filtered Image Tab

Ein Bild, das Screenshot, Text, Display, Software enthält.

Automatisch generierte Beschreibung

Figure 12: Binarized Image Tab

1. Result tab

After the Calculation has finished, the result tab should look similar to this:

Ein Bild, das Text, Zahl, Schrift, Dokument enthält.

Automatisch generierte Beschreibung

Figure 13: Result Tab

Please note that the parameters indexed with I are the ones calculated if luminous intensity is given.  
If it´s not all those values default to 0.

**Calculation of the parameters:**

By using the effective luminance the effective solid angle, aswell as the mean luminance of the whole luminaire and the solid angle of the whole luminaire, the parameter can be calculated as follows:

Where the effective luminance is the mean value of all of the pixels above the luminance threshold and the mean value of all of the pixels which are part of the luminaire.

Values calculated from the luminous intensity:

The DUGR value can then be calculated using the following formula:

The calculation of the “solid angle image”, which is used for the calculation of effective and total solid angle of the luminaire is based on the procedure used by the labsoft for standard UGR estimation.

The results can be exported to a \*.pdf file. If the „Show all evaluation“ checkbox is selected all of the generated images during calculation are exported aswell.

1. Perspective correction algorithm

For the perspective correction algorithm the GUI currently uses a transformation matrix calculated based on 4 points and the OpenCV function: *getPerspectiveTransform*

And mapped with the help oft he OpenCV function: *warpPerspective*

*(*[OpenCV: Geometric Image Transformations](https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html#ga20f62aa3235d869c9956436c870893ae))

Currently the transformation is mapped to the width and height of the luminaire. For now, the image is downscaled to achieve a resolution of 1 mm/px. (Simplifies the process and shortens calculation time).

So in order to do the projective transformation the width and the height of the luminaire hast to be entered and the four boundary points of the luminaire have to drawn onto the previously imported source image (By clicking into the plot).

The status bar shows know if the required resolution is met by calculation the mm/px resolution based on the input.

By pressing the „Projective Transformation“ button the rectification is executed.

Ein Bild, das Text, Screenshot, Display, Software enthält.

Automatisch generierte Beschreibung

Figure 14: Parameters and Polygon for the projective Transformation

The result of the projective transformation can then be analyzed in the „Rectified Image“ Tab.

Ein Bild, das Screenshot, Text, Farbigkeit, Software enthält.

Automatisch generierte Beschreibung

Figure 15: Image after projective transformation (Rectified Image Tab)

1. Draw ROIs

*It is now possible to decide between two types of ROIs Rectangular and Circular. After selecting the required type the ROI can be drawn by clicking and dragging in the rectified image*

In our example, the luminaire has two light emitting surfaces, so in the next step we draw two rectangle ROIs by clicking and dragging in the rectified image. After placing a rectangular ROI it can be safed by pressing the button „Safe ROI“.

Ein Bild, das Screenshot, Text, Farbigkeit, Software enthält.

Automatisch generierte Beschreibung

Figure 16: Drawing and saving a rectangular ROI

Each saved ROI can be viewed in the ROI tab. In addition, it is possible to delete either the last ROI or all ROIs via the respective buttons.

Ein Bild, das Screenshot, Text, Farbigkeit, Software enthält.

Automatisch generierte Beschreibung

Figure 17: Viewing the ROIs in the ROI Tab

1. Filter the image.

To account for human visual acuity, the image is filtered using a Gaussian filter. Since the image is scaled to a resolution of 1mm/px the default value for the full width at half maximum (FWHM) is 12mm, as suggested in CIE 232:2019. But it can be changed in the textbox.

Once the ROIs are determined and a full width at half maximum is defined, filtering can be performed via the filter image button.

Standard deviation () and filter width are calculated according to formulas 12, 13 and 14 from CIE 232:2019.

To ensure that no luminance information is lost during filtering, a projective transformed image of the luminaire is generated with a frame of half the filter width.

This is done by calculating factors for the luminaire dimensions based on the input for luminaire width and height and the previously drawn polygon.  
With the help of these factors new polygon points are calculated and the resulting polygon is projectively transformed.

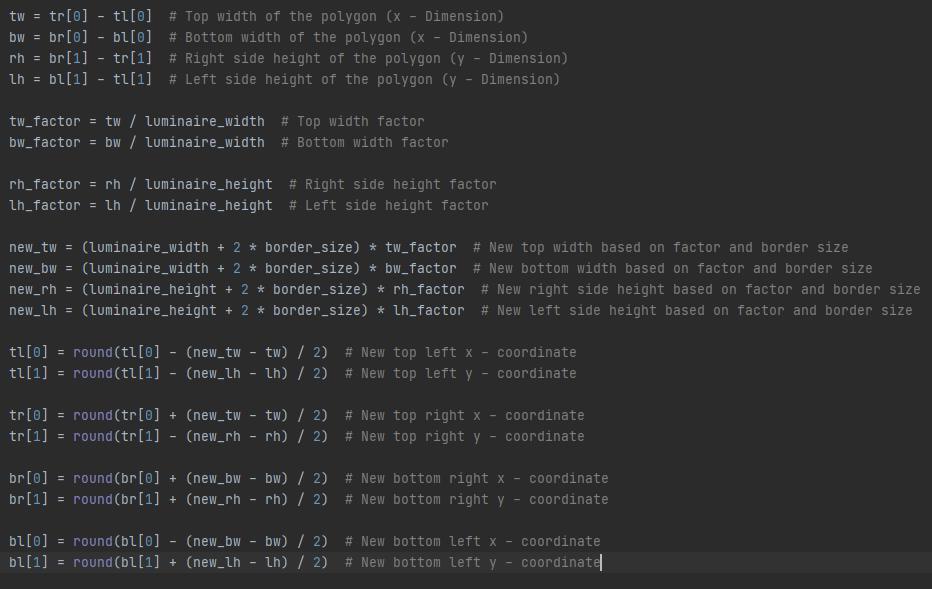


Figure 18: Code Snippet for the calculation of the projective transformed image with borders

The result is a rectified image of the luminaire with the calculated border size and can be analyzed in the Filtered Image Tab.  
The previously selected ROIs are marked by red rectangles in the resulting filtered image.

Ein Bild, das Screenshot, Text, Farbigkeit, Software enthält.

Automatisch generierte Beschreibung

Figure 19: Gauss filtered image (Filtered Image Tab)

1. Thresholding the ROIs

In the next step, all pixels that are below the set threshold, in the previously defined ROIs, are set equal to 0. The default value for the threshold is as suggested in CIE 232:2019.

Ein Bild, das Text, Screenshot, Software, Farbigkeit enthält.

Automatisch generierte Beschreibung

Figure 20: Binarized / Thresholded ROIs (Binarized Image Tab)

1. Calculating the results

After binarizing / thresholding the ROIs the „Calculate DUGR“ button can be pressed to calculate the results.

The results can be seen in a tabular form in the Result tab.

From v\_0.4 onwards it´s possible to export the protocol as \*.pdf file. The file contains the evaluation images and the result table.  
It the export can be done via the button or the shortcut CTRL+S

Ein Bild, das Text, Zahl, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Figure 21: Result table in the Result tab

## Further Notes:

Shortcuts:

|  |  |
| --- | --- |
| Open File | CTRL + O |
| Load Parameters | CTRL + P |
| Reset Polygon Selection (Polygon drawing) & Reset Rectangle Selection (ROI drawing) | Escape |
| Save protocol \*pdf file | CTRL+S |

# Changelog

## v\_0.1 --> v\_0.2

New Functions:

- Check of the required resolution (1.2[mm/px]) based on input dimensions and the image

- Possibility to draw circular ROIs on the rectified image to calculate DUGR of luminaires which are not Rectangular

(--> Polygonal ROIs are going to be available in one of the next versions )

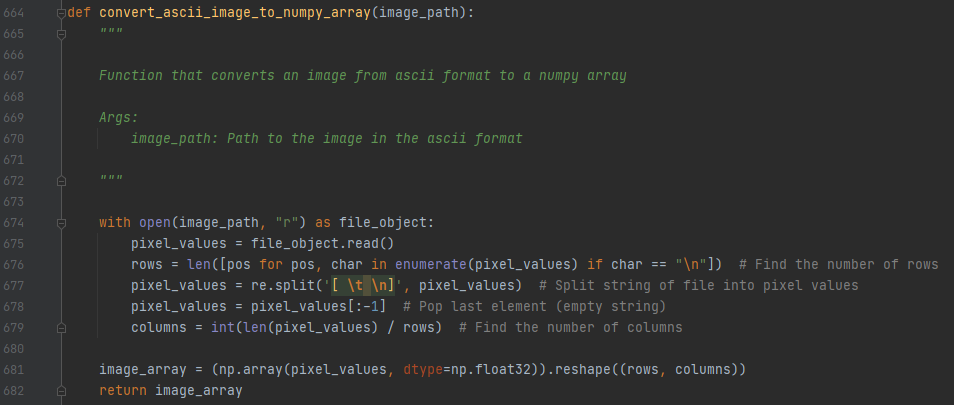
## v\_0.2 --> v\_0.3

New Functions:

* Possibility to load an image from a \*.txt file

The number of rows is estimated by counting the number of line breaks

The number of columns is then estimated by dividing the number of total values with the number of rows



An example image as \*.txt file has been added to the dist folder.

## v\_0.3 --> v\_0.4

New Functions:

- The GUI has now 2 Different procedures implemented that can be changed via checkbox

(Calculating DUGR on projective distorted and on projective corrected image)

- The Matplotlib frontend has been polished and the layout tightened.  
the matplotlib toolbar was added to all of the figures which also allows to view pixel values by mouse hovering aswell as zooming etc.

- Polygon and ROI selection can be undone by pressing escape now

- The Result table is now also a matplotlib figure -> allows pdf export and looks better.

- A possibility to save the measurement protocol to a \*.pdf file has been implemented

- A possibility to look at every evaluation image of the projective distorted algorithm via checkbox has been added (They can be exported to \*.pdf aswell)

## v\_0.4 --> v\_0.5

The calculation of the DUGR value on projective distorted images has been corrected.

In order to execute the calculation, the ROI of the whole luminaire has to be drawn onto the the source image.  
NOTE: At this point it only works to select trapezoid ROIs. (Rectangular and Circular ROIs produce bugs that crash the software. Therefore they will be postponed to the next version)

A possibility to load a predefined parameter set from a \*.json file for the algorithm without projective correction has been implemented.

Bugfixes:

* Fixed a bug where the polygon Selector didn´t reset properly
* Fixed a bug where the result table was overwritten with multiple entries instead of updated correctly

## v\_0.5 --> v\_0.6

Improved the projective distorted algorithm

## v\_0.6 --> v\_0.7

- Option to only filter inside a rectangular window around the specified luminous area for the algorithm without projective correction

- Performance in terms of execution time improved

--> Execution time of calculation with filtering of a whole image with high resolution (4087x1996px) down to approx 1 minute  
(Was around 6-7 Minutes before for images of such high resolution)

Bugfixes:

* Fixed a bug where a trapezoid region with a bigger width at the top than the bottom side would crash the GUI
* Fixed a bug where the ellipsoid region selector was not selectable for the algorithm with projective correction
* Fixed a bug where loading a new source image did not reset the old axis and resulted in stretching the image

## v\_0.7 --> v\_1.0

- The colormap for luminance images has been changed to one similar to the one which is used by the LabSoft Software from TechnoTeam

- It is now possible to change between logarithmic scalings for the colormap.

--> This allows contrast enhancement in order to draw borders onto the image.

- A lot of Bugfixes regarding different procedures of the program.  
Deviation from the sequence often led to crashes of the software before.  
This behavior is now intercepted and a hint is given in the status bar.

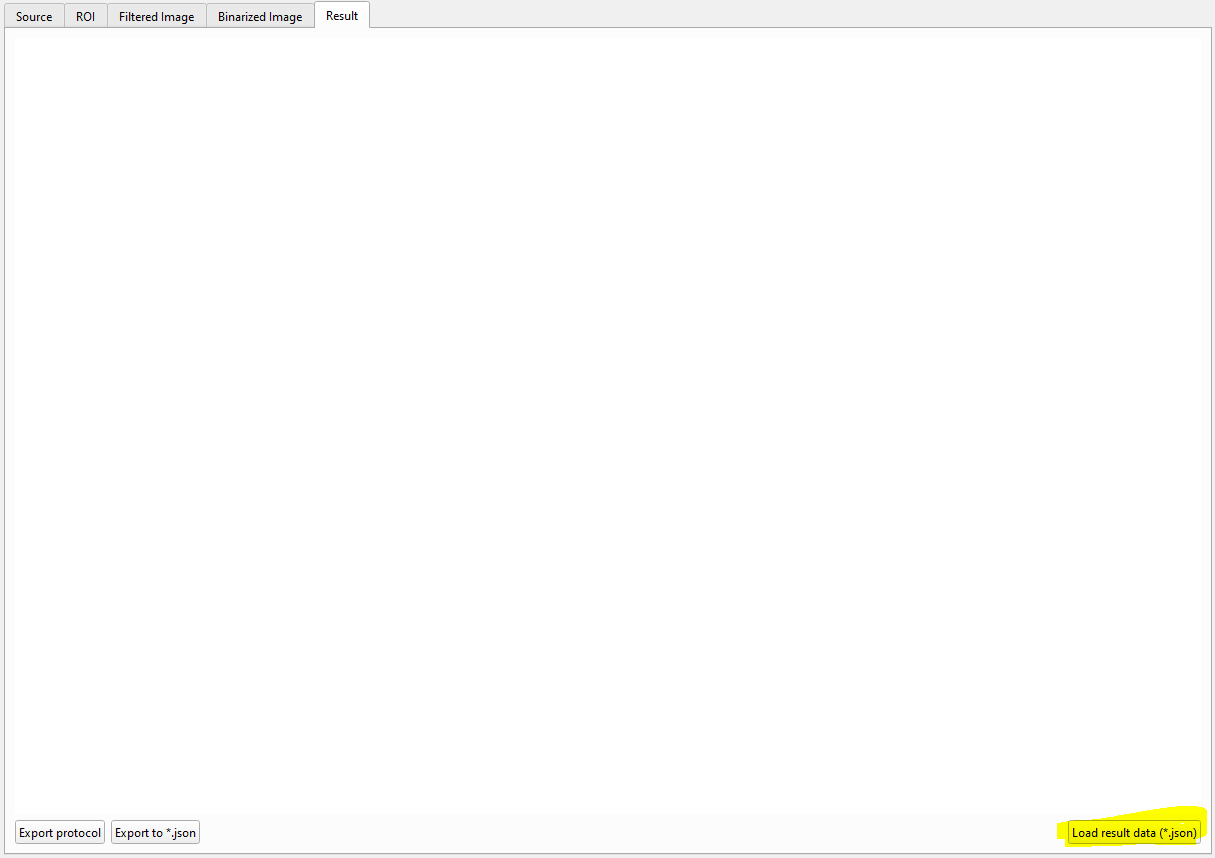
## v\_1.0 --> v\_1.2.2

- Changed the Python script architecture

-> Script splitted into seperate Python Files to allow easier debugging

- Ability to safe Parameters from the GUI to a \*.json File

- Ability to load Result \*.json Files into the Result Tab



- Small Bugfixes and stability changes

- Created an Installer for the GUI