

Distorsion

Plug-In



Copyright

Copyright © NeuroCheck GmbH
All rights reserved.
Version 6.2.1
Neckarstraße 76-1, 71686 Remseck, Germany

Phone: +49 (0) 7146 - 89 56-0
Fax: +49 (0) 7146 - 89 56-29
E-Mail: info@neurocheck.com
Web: www.neurocheck.com

Table of Contents

NeuroCheck Distorsion Plug-In Help - PI_Distortion.NET.dll	3
General Information	3
Introduction	3
Installation	4
Check Functions	5
Grid-based Calibration	5
Introduction	5
How to Use	6
Parameter Dialog	7
Visualization	8
Grid-based Transformation	9
Introduction	9
How to Use	10
Parameter Dialog	11
Perspective Transformation	12
Introduction	12
How to Use	13
Parameter Dialog	15
Support Contact	16
Support Services	16

Introduction

About NeuroCheck plug-in DLLs in general

A plug-in DLL is a .NET assembly that serves to enhance NeuroCheck with user-defined image processing functionality. The NeuroCheck Plug-In Interface offers the opportunity to integrate user-defined check functions for image processing and data handling. A Plug-In can contain an arbitrary number of self-developed check functions.

These check functions have full access to the NeuroCheck runtime data objects such as Images, ROI Lists or Measurement Lists. The Plug-In check function can be added to a check as well as the built-in standard check functions of NeuroCheck.

Please note that for integration of a plug-in check function into your check routine, a Premium license is required. The completed check routine then can be run with any NeuroCheck license (except the Demo Version).

Installation

Installation

Copy the following files from the zip archive to the plug-in directory within the desired NeuroCheck project (e.g. 'C:\Users\Public\Documents\NeuroCheck\6.2\Default\Software Extensions\PlugIns').

- All files inside the `Binaries` directory
- All *.chm files inside the `Documentation` directory

Loading a Plug-In

In order to use a Plug-In the Plug-In assembly must be loaded in NeuroCheck. The management of Plug-Ins takes place within the Software Settings dialog. The Software Settings dialog can be found in the System menu of NeuroCheck.

Please note that it is impossible to load or unload a Plug-In as long as a check routine is opened that contains the Plug-In check functions. If the currently opened check routine contains Plug-In check functions then close the check routine first.

Within the Software Settings dialog please select the node Plug-Ins and the sub-node Plug-In in the tree to the left. The loaded Plug-In assemblies are shown in the List of Plug-Ins. Press the Add button to open a file selection dialog in order to select a further Plug-In assembly.

Inserting a Plug-In check function to a check routine

A Plug-In check function is inserted using the Check Function Select dialog. All check functions of loaded Plug-Ins are listed in the Plug-In category of the Check Function Select dialog. Within the Plug-In category the check functions are ordered in sub-categories where each sub-category represents the check functions of one Plug-In.

Besides the category the user will hardly notice any difference between the usage of Plug-In check functions and built-in check functions.

Grid-based Calibration: Introduction

Function overview

This function calculates the distortion in the current image in a calibration run. The target position of each grid point in image is calculated and saved to an xml file. This file will be used by the check function "Grid-based Transformation" to compensate the distortion. In the parameter dialog you can define the size of your calibration grid.

See "[How to Use](#)" for additional information.

Input data

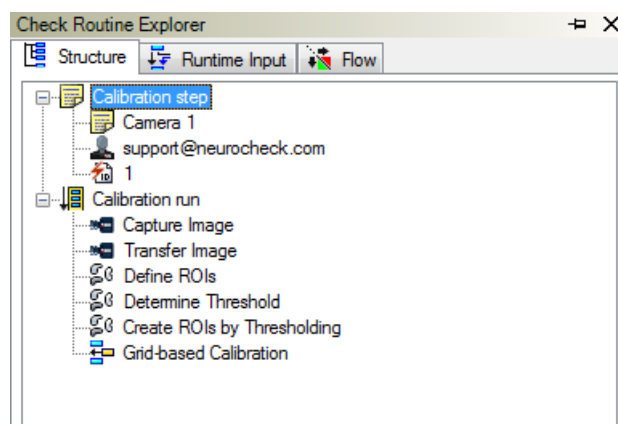
This check function requires an image and a list of ROIs as input data objects. This check function determines distortion by using a special calibration target. The distortion is saved into a xml file which will be used by the check function "Grid-based transformation".

Output file

The calculated distortion vectors are written into a xml file. This file is used by "Grid-based transformation" to transform the image and is usually done in another check routine.

Check routine sample

▣ [Screenshot of Check Routine Sample](#)



Result view

The result view shows the input image with the calibration table, the created ROIs and transformation vectors. These vectors visualize in which direction and length the calibration points will be moved by the check function "Grid-based transformation".

Properties

▣ Check function group Plug-In.

123 The check function has a [Parameter Dialog](#).

The check function has own result [Visualizations](#).

Grid-based Calibration and Transformation: How to Use

The functions in this plug-in can be used for image transformation to compensate lens distortion.

There are two steps required to apply such a correction. The first step is to measure the dimension and the type of the distortion. This will be done once in a calibration phase "Grid-based Calibration". The second step is to apply the transformation using check function "Grid-based Transformation" to avoid problems caused by distortion in a run time phase.

In any case it is advisable to avoid lens distortion. It is usually not possible to completely compensate image distortion by image processing.

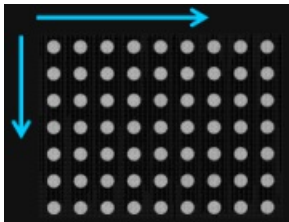
Step 1. Calibration phase (Grid-based Calibration)

The calibration has to be done once for a fix installation. If the camera objective or the position has been changed the calibration step has to be repeated.

In this calibration phase an image is taken from a calibration table. This table can be composed of black circles on a white background (or vice versa).

The geometrical requirement is an orthogonal order and a constant spacing (in x and y direction) of the grid because the points in the distorted image will be mapped in a regular grid by the following transformation. The points of the calibration table must be arranged parallel to the camera orientation.

[Screenshot of Grid Target](#)



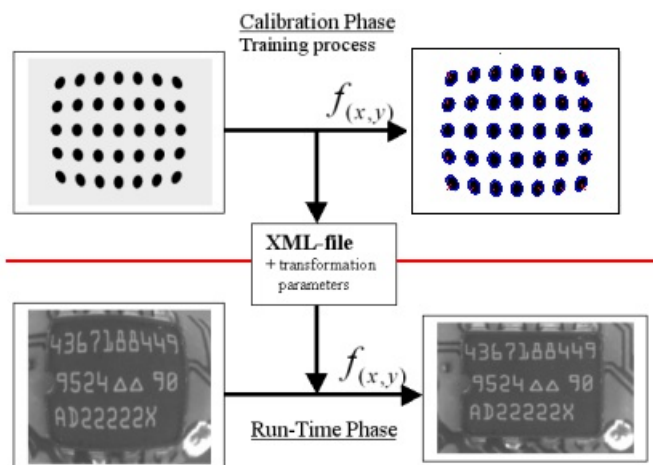
After capturing the image the points of the calibration table have to be created by threshold. The check function creates an xml file which will be used by "Grid-based Transformation".

Step 2. Run time phase (Grid-based Transformation)

In the run time phase the created xml file from "Grid-based Calibration" will be used to calculate the correction algorithm and applies the image transformation.

Diagram

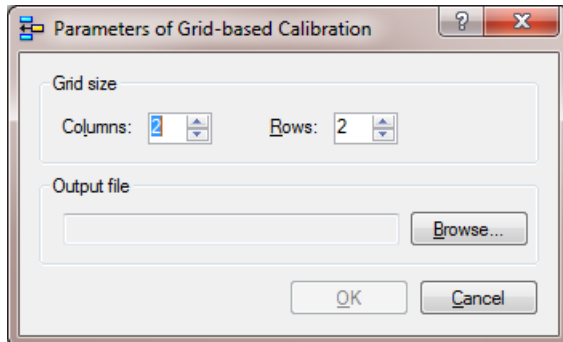
[Diagram Overview](#)



Grid-based Calibration: Parameter Dialog

This plug-in check function has a **Parameter** dialog.

☑ [Screenshot of Parameter Dialog](#)



The **Parameter** dialog contains the following elements:

Element	Description
Grid Size	Select here the size of your calibration grid. Be sure that the input ROIs fit the grid size you choose here.
Columns	Number of columns of your calibration grid.
Rows	Number of rows of your calibration grid.
Output file	Xml output file name.

Grid-based Calibration: Visualization

This section describes the result visualizations the check function "Grid-based Calibration" provides.

Element	Description
Image and Vectors	Displays the input boundary ROIs of the check function. Each ROI got a vector which represents the distortion calculated by this check function. The visualization can be helpful to verify the distortion.

Grid-based Transformation: Introduction

Function overview

This check function creates an image without distortion which is determined by the calibration step in the check function "Grid-based Calibration".

See "[How to Use](#)" for additional information.

Input data

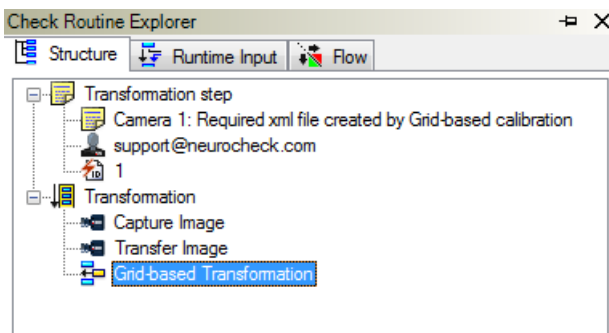
This check function requires an image as input data object. Specially it requires an xml file created by check function "Grid-based Calibration" which determines distortion by using a special calibration target. The distortion is saved into an xml file which will be used in this check function.

Output data

The transformed image.


Check routine sample

[Screenshot of Check Routine Sample](#)



Properties

 Check function group Plug-In.

 The check function has a [Parameter Dialog](#).

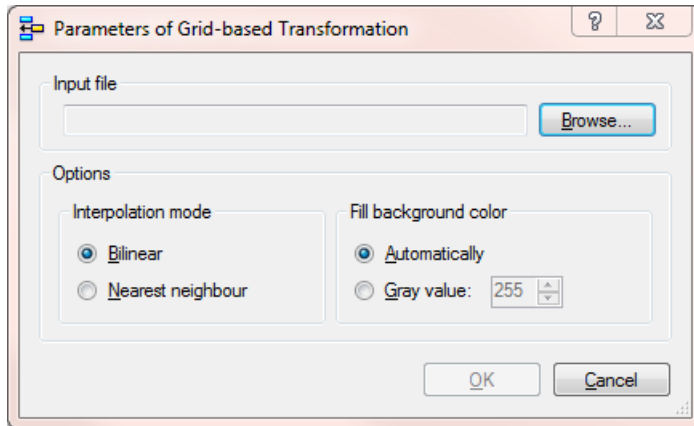
Grid-based Calibration and Transformation: How to Use

Please read "[How to Use](#)" of Grid-based Calibration and Transformation.

Grid-based Transformation: Parameter Dialog

This plug-in check function has a **Parameter** dialog.

☑ [Screenshot of Parameter Dialog](#)



The **Parameter** dialog contains the following elements:

Element	Description
Input file	Xml input file name with the distortion which will be used to transform the image. This file was created previously by the check function "Grid-based Calibration".
Bilinear	Good quality resampling: The information from the next four neighbor pixels from the calculated coordinate is taken for the pixel in the new image. This method is a little bit slower than nearest neighbour but the transformed image does not have such square-edged borders. The image is smoothed.
Nearest neighbour	The information of the pixel is calculated by choosing its nearest neighbour. This method is faster than bilinear but result image got less quality.
Automatically	Every pixel outside the input image will be automatically set to its nearest valid pixel in the input image.
Gray level	Every pixel outside the input image will be set to the specified gray value.

Perspective Transformation: Introduction

Function overview

This check function calculates a projective transformation based on user defined regions and transforms the image.

See "[How to Use](#)" for additional information.

Input data

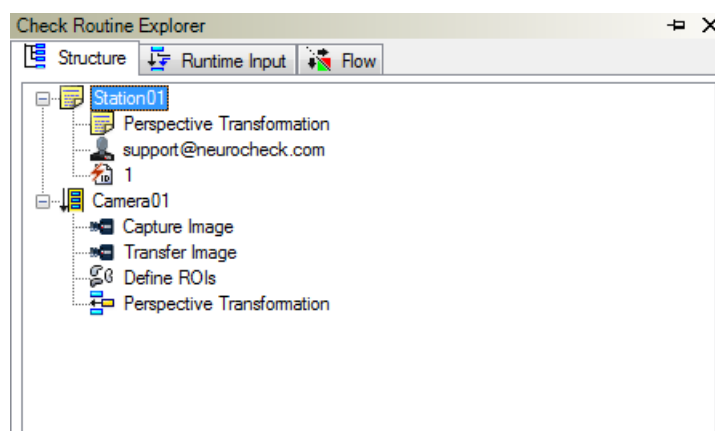
This check function requires an image and a collection of ROIs as input data objects. The number of ROIs has to be exactly four.

Output data

Image which shows the perspective transformation of the defined region.


Check routine sample

☑ [Screenshot of Check Routine Sample](#)



Properties

 Check function group Plug-In.

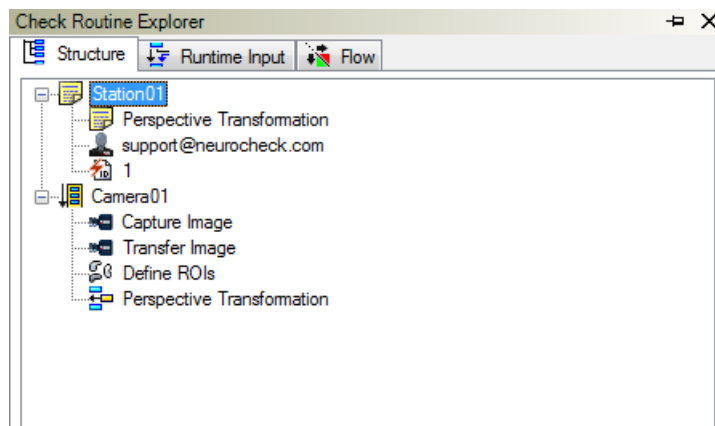
 The check function has a [Parameter Dialog](#).

Perspective Transformation: How to Use

This plug-in function can be used to suppress perspective rectification. It calculates a homography matrix based on user defined regions and transforms the input image. The user defined regions have to represent a convex quadrangle which will be transformed into a rectangle of a user defined size.

Check routine sample

[Screenshot of Check Routine Example](#)

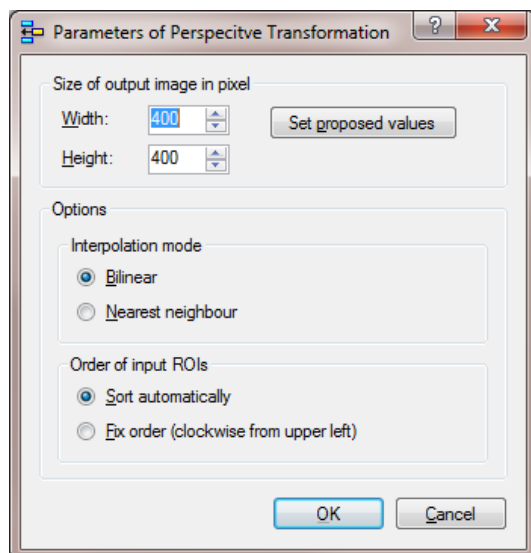


Step 1. ROIs definition

The function requires exactly four ROIs. The edges between these ROIs have to form an arbitrary convex quadrangle. The input order (index of each ROI) which represent a corner of the rectangle will be mapped to a quadrangle corner.

Step 2. Parameter dialog configuration

[Screenshot of Parameter Dialog](#)



Size of output image in pixel

The width and height are the size of the output image which is a quadrangle. It is important to know that the configuration of the size implies **a priori knowledge** of the user. The size highly influence the utility of the result. Generally speaking a wrong size can deform the image instead of rectifying it. The Button "Set default values" will set default width and height parameter which wont resize the rectangle. Be aware that these values don't have to be the best choice but they can help you to get a size for a transformation without resizing the output image.

Options - Interpolation mode

The default interpolation mode is "Bilinear". Its quality is better than "Nearest neighbour". Indeed the speed of "Bilinear" is worse than nearest neighbour.

Options - Order of input ROIs

The default order mode is "Sort automatically". This mode will set the ROI with the smallest Euclidean distance from the top left of the image to the top left corner of the result quadrangle. In the example Diagram this would be the ROI with index 3. It continues clockwise from this ROI and builds a rectangle mapping the corners clockwise to the result quadrangle. In our example it would end with ROI of index 1 at the bottom left corner mapping it to the bottom left corner of the result quadrangle. Generally speaking in this mode the input order of the ROIs won't influence the result. This mode is recommended if the order of the ROIs is not fixed which can happen if the ROIs are determined by the check function "Template Matching".

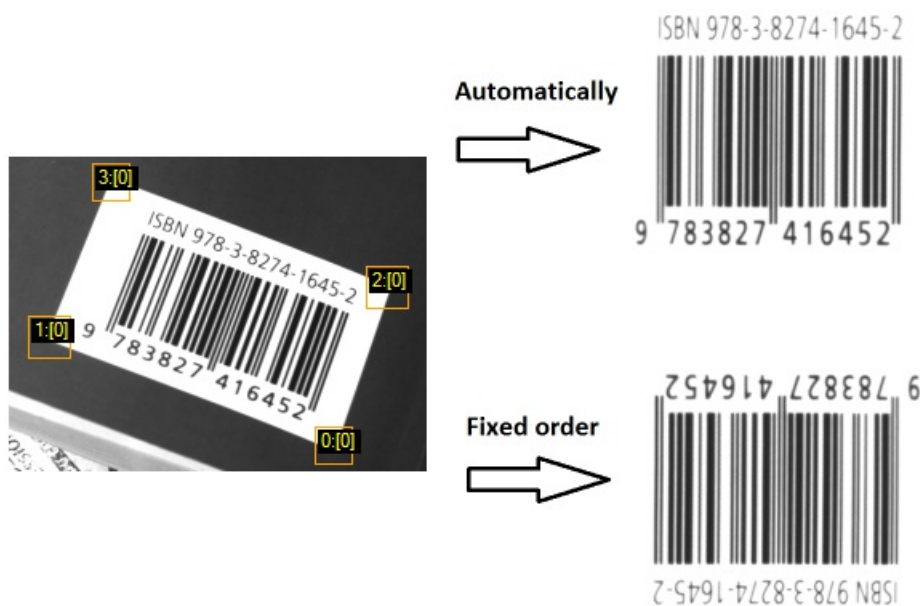
The mode "Fix order (clockwise from upper left)" will statically map the first given ROI to the top left corner of the result quadrangle. Analog it maps the next three ROIs clockwise to the result quadrangle corners. Using this mode it is possible to flip images by changing the input ROIs order. This mode should be used if the input order cannot change.

More details can be found in the [Parameter Dialog](#) section.

Diagram

The screenshot below shows the principle of the order mode "Automatically" and "Fixed order". The order mode "Automatically" maps the given input order of ROIs from (0,1,2,3) to (3,2,0,1) whereas "Fixed order" won't change the order and maps the first input ROI to the top left corner and continues clockwise.

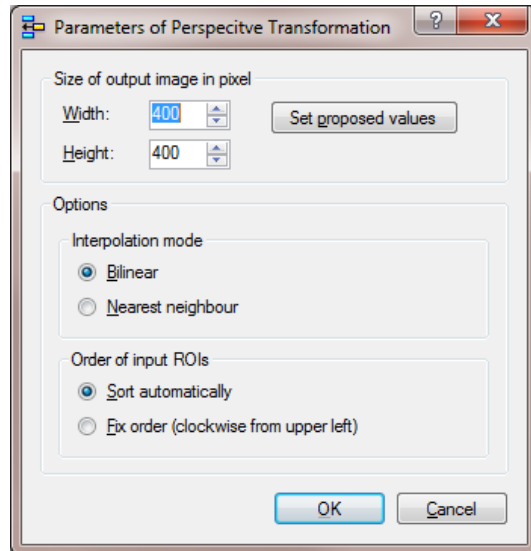
[Diagram-Overview](#)



Perspective Transformation: Parameter Dialog

This plug-in check function has a **Parameter** dialog.

☑ [Screenshot of Parameter Dialog](#)



The **Parameter** dialog contains the following elements:

Element	Description
Width	Defines the width of the result image.
Height	Defines the height of the result image.
Bilinear	Good quality resampling: The information from the next four neighbor pixels from the calculated coordinate is taken for the pixel in the new image. This method is a little bit slower than nearest neighbour but the transformed image does not have such square-edged borders. The image is smoothed.
Nearest neighbour	The information of the pixel is calculated by choosing its nearest neighbour. This method is faster than bilinear but result image got less quality.
Sort automatically	This mode will automatically order the input ROIs. It sets the top left ROI to the top left corner of the result quadrangle. The algorithm determine the top left ROI by the smallest Euclidean distance from the top left point of the input image. Starting from this top left ROI the algorithm build a rectangle clockwise and maps each found corner to the corresponding corner in the quadrangle. It ends on the button left corner which is mapped to the button left corner of the result quadrangle. Generally speaking in this mode the input order of the ROIs won't influence the result. This mode is recommended if the order of the ROIs is not fixed which happens for example if the ROIs are determined by the check function "Template Matching".
Fix order (clockwise from upper left)	The mode "Fix order (clockwise from upper left)" will statically map the first ROI to the top left corner of the result quadrangle. Analog it maps the next three ROIs in the sequence of their input clockwise to the result quadrangle corners.

Support Services

For technical support, please contact your local NeuroCheck partner or NeuroCheck GmbH:

Phone: +49 (0) 7146 - 89 56-40

E-Mail: support@neurocheck.com

Web: www.neurocheck.com

Before contacting us, please provide some important information about your system:

Information about your NeuroCheck installation and your PC setup:

- Use the NeuroCheck Diagnostics tool to check your installation and computer configuration.
- The NeuroCheck Diagnostics is installed in the "Tools" folder within your NeuroCheck installation.

Log file information:

- Logging for NeuroCheck can be activated in **System > Software Settings > Diagnosis > Logging**.

