3DEQUALIZER 4

sdv_image_warp

Command line based warp tool

Science-D-Visions, 2020-04-22

Abstract. sdv_image_warp is a command line tool shipped with 3DE4 for applying or removing Lens distortion from single images. It is either applied directly to the footage, or it may be used to create ST maps compatible with Nuke's STMap nodes. This document shows how to use the tool in various situations. In addition, the results of automated tests are presented, which are intended to ensure that the tool remains free of bugs in the course of further development.

Contents

1 *Introduction*
2 *The tool*
2.1 Installation
2.1.1 *Windows*
2.1.2 *Linux, OSX*
2.2 Usage / Tests
2.2.1 *Test 1: Original footage without overscan*
2.2.1.1 Remove distortion by parameters
2.2.1.2 Create an ST-map by parameters
2.2.1.3 Remove distortion by an ST-map
2.2.1.4 Apply distortion by parameters
2.2.1.5 Create an ST-map by parameters
2.2.1.6 Apply distortion by an ST-map
2.2.1.7 Cropping
2.2.2 *Test 2: Original footage with overscan*
2.2.2.1 Remove distortion by parameters
2.2.2.2 Remove distortion by an ST-map
2.2.2.3 Apply distortion by parameters
2.2.2.4 Apply distortion by an ST-map
2.2.2.5 Verifying results with Nuke
2.2.3 *Test 3: Display window larger than data window*
2.2.3.1 Solution 1
2.2.3.2 Solution 2
2.2.3.3 Solution 3
2.2.4 *Test 4: Various lens distortion models with non-default values*
2.2.4.1 LD 3DE4 Anamorphic - Rescaled, Degree 4
2.2.4.2 LD 3DE4 Anamorphic - Standard, Degree 4
2.2.4.3 LD 3DE4 Anamorphic - Degree 6
2.2.4.4 Radial - Fisheye, Degree 8
2.2.4.5 Classic LD Model
2.3 Image processing scheme

1 *Introduction*

sdv_image_warp is a command line tool for applying or removing lens distortion. It operates on single frames. The idea is to provide a tool you can use for implementing scripts for your own lens distortion workflow. The tool can

- Apply or remove lens distortion from an image.
- Create an ST-map for given camera and lens distortion parameters.
- Apply an ST-map to an image.

This document contains various examples for using the command-line warp tool sdv_image_warp. Internally, we use it to verify that the tool behaves as expected, by means of automatic tests (*confidence tests*).

Version of **sdv_image_warp**	Released	Changes
1.3	2020	Minor changes and bugfixes
1.2	2019-11-28	Confidence test series 4.
1.0	2019-06-06	Confidence test series 1, 2, 3.

2 *The tool*

2.1 Installation

sdv_image_warp is part of the 3DE4 distribution. It does not require access to 3DE4's license server, and you can move it to any other place in your file system, or run it on a different machine. sdv_image_warp consists of at least two components: a python script named **sdv_image_warp.py** and a binary executable named **sdv_image_comp[.exe]**. On windows, additional runtime libraries are required. There is no tool for installation, you simply copy the files to the place you like. It also possible to run the tool directly in 3DE4's installation directory. The script and the binary are located in

```
$INSTALL_DIR
├── sys_data
│   └── py_scripts
│       └── sdv_image_warp.py
└── bin
    └── sdv_image_comp[.exe]
```

The binary is invoked by the script, and it is not meant to be started directly.

2.1.1 *Windows*

In case you prefer a different location for the tool, a complete installation on Windows will look like this:

```
P:\ath\to\mydir
├── libgcc_s_seh-1.dll
├── libstdc++-6.dll
├── libwinpthread-1.dll
├── sdv_image_comp.exe
└── sdv_image_warp.py
```

The DLLs are part of the 3DE4 installation. You can simply copy them to the target directory.

2.1.2 *Linux, OSX*

On unix-like systems a complete installation will look like this:

```
/path/to/mydir
├── sdv_image_comp
└── sdv_image_warp.py
```

2.2 Usage / Tests

The following sections result from automated testing of sdv_image_warp. These automated tests are done prior to each release of 3DE4 on platforms Linux and Windows/Cygwin.

sdv_image_warp can perform the following tasks:

- Remove / apply lens distortion to an input image by means of a lens distortion model driven by camera and model parameters.
- Create an ST-map which may serve as a lookup-table for map-based warp compositing (like e.g. the ST-Map node in Nuke). This ST-map represents a distortion or undistortion mapping for a given lens distortion model driven by camera and model parameters.
- Apply a distort- or undistort-mapping to an input image based on an ST-map.

Let us consider a few examples. We start with a checkerboard image which represents our original footage. Usually, the footage already contains distortion. Our checkerboard does not show any distortion, yet we shall use it as proxy for distorted footage. This makes it easier to understand on an abstract level, what is going on. Our lens distortion model in tests 1, 2 and 3 will be

`3DE4 Radial - Standard, Degree_4`

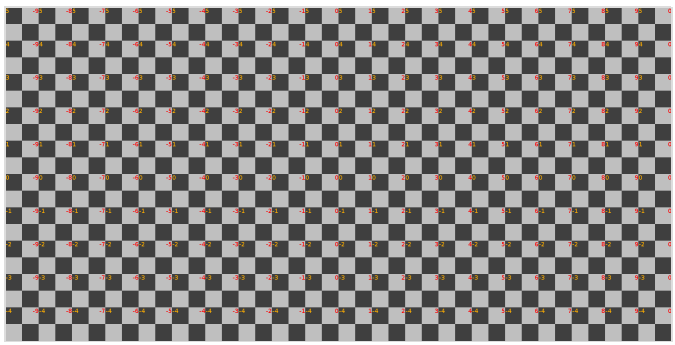
and we shall use the following parameters:

Param	Value
filmback	2.0 1.0
field_of_view	0.0 0.0 1.0 1.0 (default)
lens center_offset	0.3 0.1
parameters	0.1 0 0 0 0 0 0 0

Test 4 is dedicated to the other lens distortion models. All parameters, including lens center offset are set to non-default values, in order to ensure that all parameters are passed to and handled correctly by the underlying compositing engine (i.e. to verify that the implementation is correct).

2.2.1 *Test 1: Original footage without overscan*

Our test image in this first test series is a PNG image with size 2000×1000 pixel:

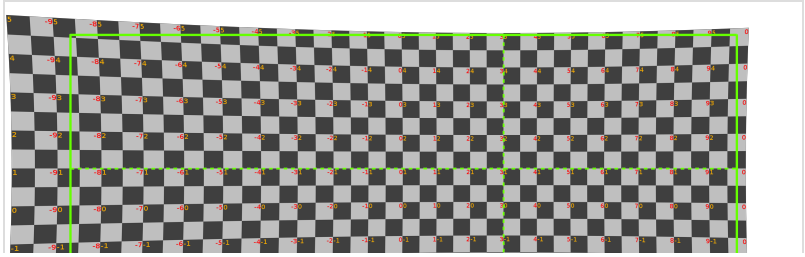


2.2.1.1 Remove distortion by parameters

In order to remove distortion we would invoke the tool like this:

```
$SDV_IMAGE_WARP          -model LD_3DE4_Radial_Standard_Degree_4\
                         -direction remove\
                         -output_mode image\
                         -interpolation cubic\
                         -filmback 2.0 1.0\
                         -field_of_view 0.0 0.0 1.0 1.0\
                         -lens_center_offset 0.3 0.1\
                         -overscan 200 100\
                         -parameters .1 0 0 0 0 0 0 0\
                         -original_size 2000 1000\
                         -infile $MYDIR/grid_2000x1000.png\
                         -outfile $MYDIR/grid_remove_2400x1200.png
```

We have added a few helper lines by hand. The result is:



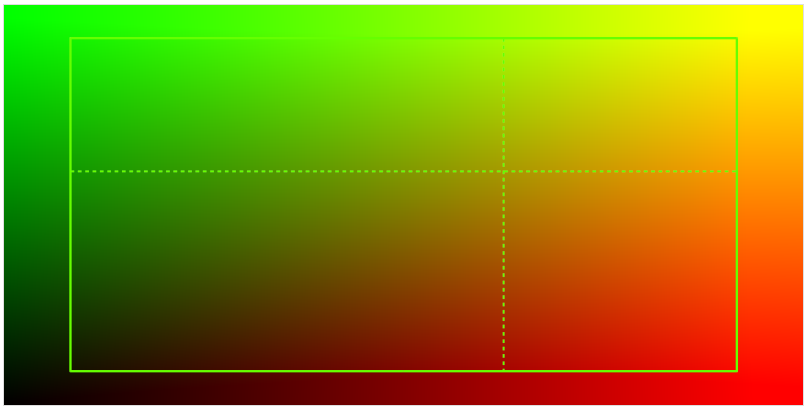
The solid rectangle marks the boundaries of the original footage. The dotted lines intersect at the lens center offset 0.3,0.1, and as you see, this point is the fixed point of the distortion mapping. The overscan 200×100 on either side is a little bit small, which you can see at the lower left corner of the image.

2.2.1.2 Create an ST-map by parameters

We should be able to reproduce this result by createing and applying an ST-map, so let's try this next. The main difference in contrast to applying the distortion function to the footage is, that here we only specify the size of the footage, by means of option `-original_size`. Cubic interpolation is not required here, so we can omit option `-interpolation`. Instead of a PNG image, we must create an OpenEXR image with `float`-valued channels.

```
$SDV_IMAGE_WARP          -model LD_3DE4_Radial_Standard_Degree_4\
                         -direction remove\
                         -output_mode stmap\
                         -field_of_view 0.0 0.0 1.0 1.0\
                         -filmback 2.0 1.0\
                         -lens_center_offset 0.3 0.1\
                         -overscan 200 100\
                         -parameters .1 0 0 0 0 0 0 0\
                         -original_size 2000 1000\
                         -outfile $MYDIR/stmap_remove_2400x1200.exr
```

Here, we have added helper lines as well. The result is a typical ramp image with overscan:

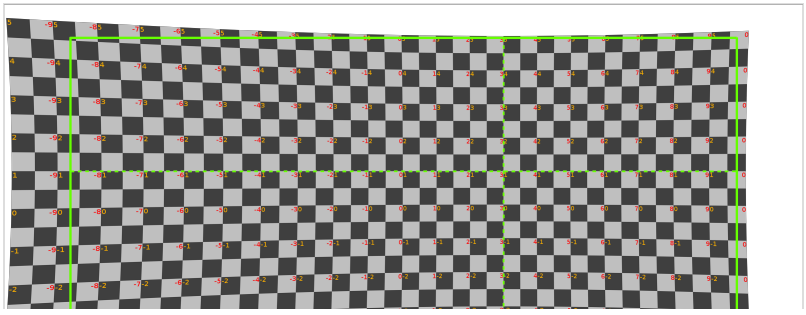


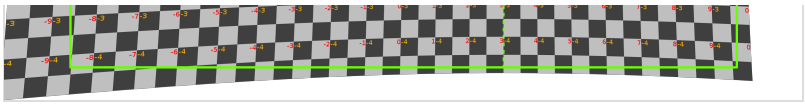
2.2.1.3 Remove distortion by an ST-map

We shall apply now the ST-map from the previous section to our checkboard image. Please note that an option `warp_by_stmap` is passed to the tool. Also, we add an overscan margin, since our original footage does not have one. There are no camera or model parameters, since the entire mapping is encoded in the ST-map image passed at `-stmapfile`.

```
$SDV_IMAGE_WARP          -command warp_by_stmap\
                         -overscan 200 100\
                         -infile $MYDIR/grid_2000x1000.png\
                         -stmapfile $MYDIR/stmap_remove_2400x1200.exr\
                         -outfile $MYDIR/grid_remove_by_stmap_2400x1200.png
```

The result (including helper lines) is:



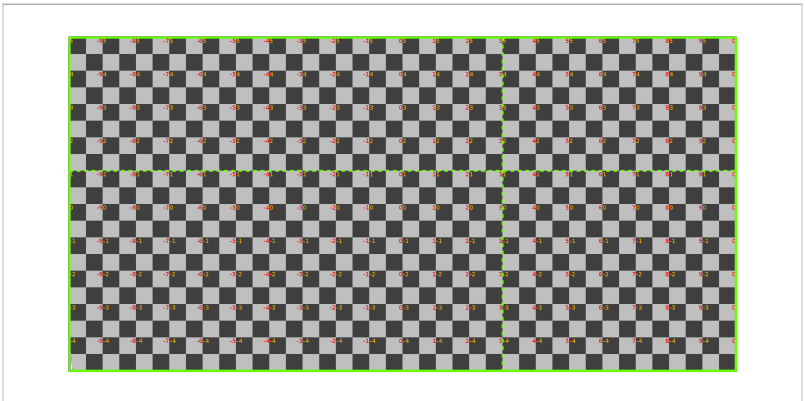


which is the same as in section 2.2.1.1, which demonstrated the equivalence of the two methods in sdv_image_warp.

2.2.1.4 Apply distortion by parameters

```
$SDV_IMAGE_WARP          -model LD_3DE4_Radial_Standard_Degree_4\
                         -direction apply\
                         -output_mode image\
                         -interpolation cubic\
                         -filmback 2.0 1.0\
                         -field_of_view 0.0 0.0 1.0 1.0\
                         -lens_center_offset 0.3 0.1\
                         -parameters .1 0 0 0 0 0 0 0\
                         -original_size 2000 1000\
                         -infile $MYDIR/grid_remove_2400x1200.png\
                         -outfile $MYDIR/grid_apply_2400x1200.png
```

Result:

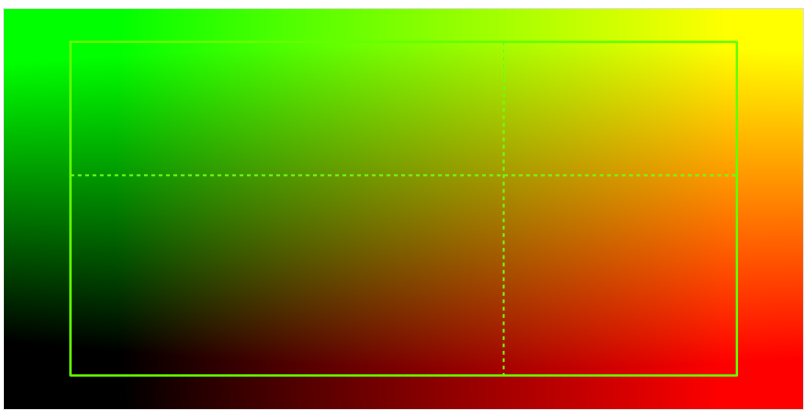


2.2.1.5 Create an ST-map by parameters

The ST-map for applying lens distortion is created almost the same way as for removing lens distortion, except for a different value passed at option `-direction`:

```
$SDV_IMAGE_WARP          -model LD_3DE4_Radial_Standard_Degree_4\
                         -direction apply\
                         -output_mode stmap\
                         -field_of_view 0.0 0.0 1.0 1.0\
                         -filmback 2.0 1.0\
                         -lens_center_offset 0.3 0.1\
                         -overscan 200 100\
                         -parameters .1 0 0 0 0 0 0 0\
                         -original_size 2000 1000\
                         -outfile $MYDIR/stmap_apply_2400x1200.exr
```

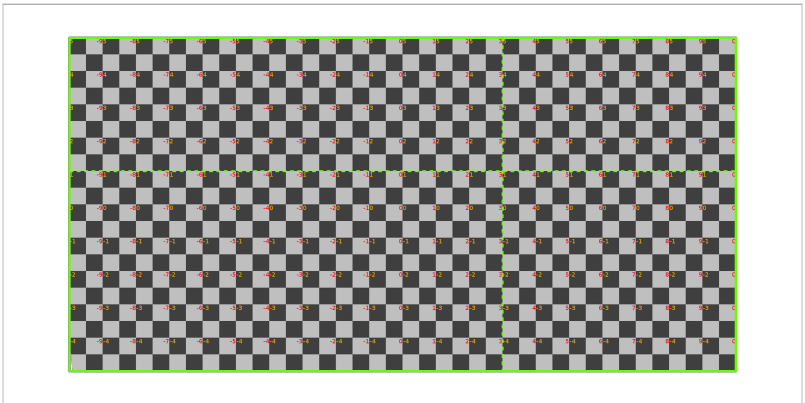
The result is again a ramp image:



2.2.1.6 Apply distortion by an ST-map

```
$SDV_IMAGE_WARP          -command warp_by_stmap\
                         -overscan 0 0\
                         -infile $MYDIR/grid_remove_by_stmap_2400x1200.png\
                         -stmapfile $MYDIR/stmap_apply_2400x1200.exr\
                         -outfile $MYDIR/grid_apply_by_stmap_2400x1200.png
```

Result:

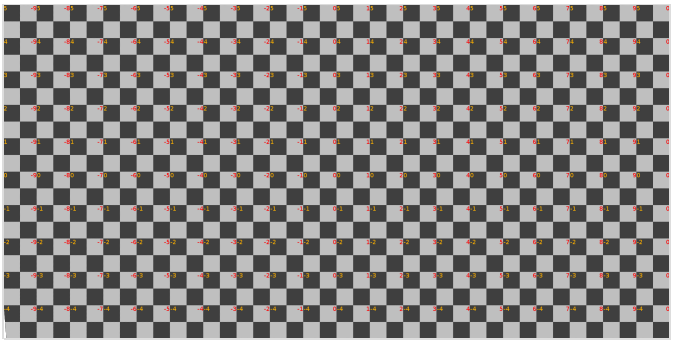


2.2.1.7 Cropping

In the previous tests when distortion is re-applied, all images still had the size from the overscan, which is not always what you want. For this reason there is an option `-crop` which reverses the effect of `-overscan`, before the image is written to the filesystem.

```
                         -crop 200 100\
```

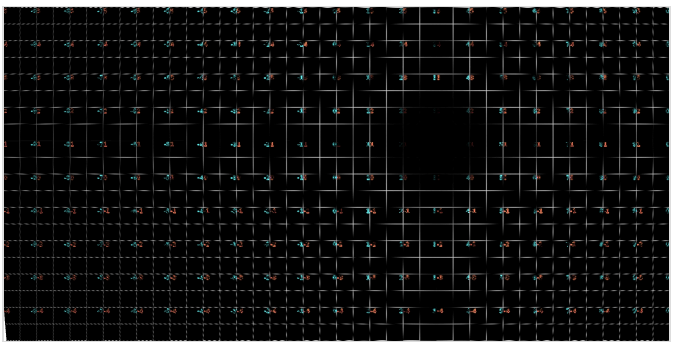
If you add this to set of options, the result is



Finally, let us compare this image to the original footage, so that we get an idea which artefacts will occur. We use gimp for this test. Both images are loaded:

```
linux> gimp $MYDIR/grid_2000x1000.png $MYDIR/grid_apply_by_stmap_crop_2000x1000.final.png &
```

Then we select one of the images, copy it and paste it onto the other image as new layer, set layer mode to *difference* and merge the layers. The result is pretty black, but now we amplify the dark regions **by a factor of 8** using the tool *Colors → Curves* and get the following result:



Clearly, the hard transitions between the fields of the checkerboard and the pixel-style numbers are most sensible to reconstruction filter artefacts. The area around the lens center is less affected, since there is almost no distortion. Artefacts of this type will occur in each compositing system, but

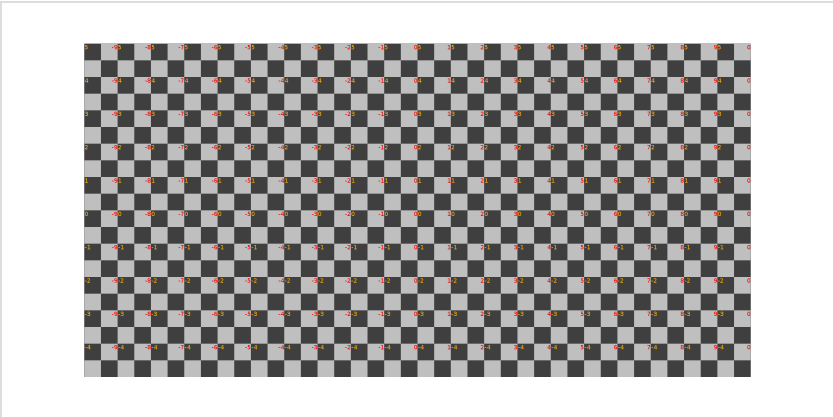
depending on the filter being used they might be lower or look different.

2.2.2 *Test 2: Original footage with overscan*

For this test series we use an OpenEXR image with the following properties:

- Data window: `-100,-175:2399,1074` - size `2500×1250`
- Display window: `150,-50:2149,949` - size `2000×1000`

The display window is placed symmetrically on the data window, but both are shifted for testing purposes. Let us assume, the data window contains overscan data. In practice, it is not clear *a priori* how data and display window are used in order to handle overscan. The data window may be located at (0,0) or somewhere else. The aim of this test series is to find out, if sdv_image_warp behaves naturally also in the general case. This is (the browser compatible version of) our original OpenEXR footage:

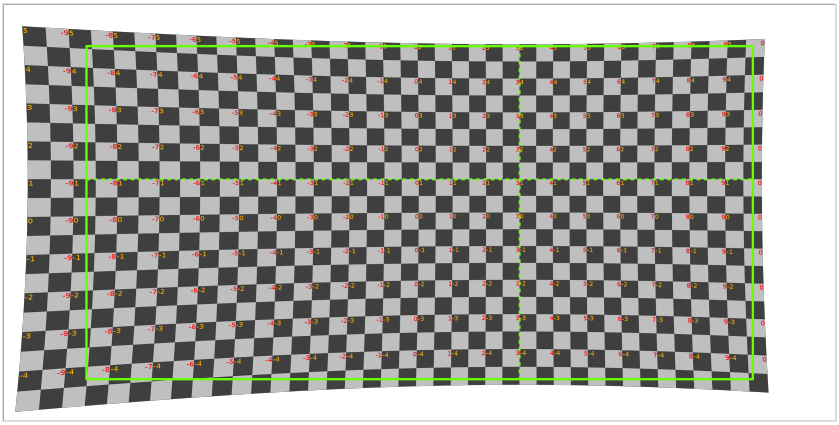


2.2.2.1 Remove distortion by parameters

We invoke sdv_image_warp without additional overscan, because the image already has 250×125 pixels on either side.

```
$SDV_IMAGE_WARP          -model LD_3DE4_Radial_Standard_Degree_4\
                         -direction remove\
                         -output_mode image\
                         -interpolation cubic\
                         -filmback 2.0 1.0\
                         -field_of_view 0.0 0.0 1.0 1.0\
                         -lens_center_offset 0.3 0.1\
                         -parameters .1 0 0 0 0 0 0 0\
                         -original_size 2000 1000\
                         -infile $MYDIR/grid_test_2_2500x1250.exr\
                         -outfile $MYDIR/grid_test_2_remove_2500x1250.exr
```

The result is:



You can easily check that is is equivalent to the result from 2.2.1.1, except for the enlarged data area.

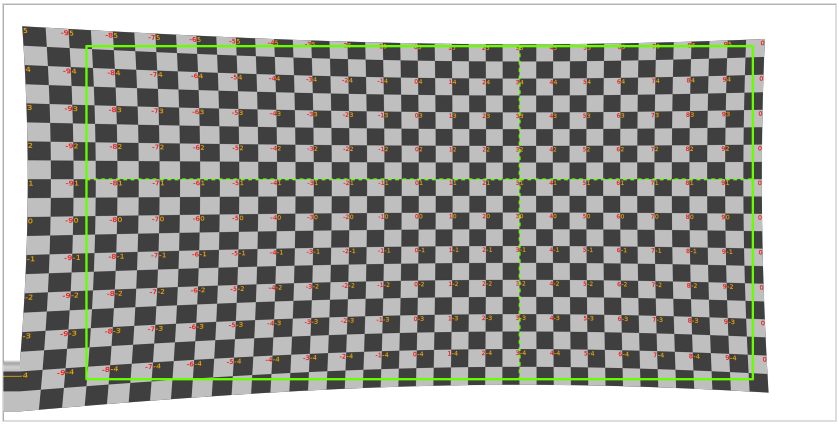
2.2.2.2 Remove distortion by an ST-map

In this section we shall use the ST-map from section 2.2.1.3 for removing lens distortion. We expect to get the same result as in section 2.2.2.1 since both methods are supposed to be equivalent. Note that our ST-map has a size of 2400×1200, which is a bit too small and causes the little glitch in

the lower left corner. We invoke sdv_image_warp without overscan:

```
$SDV_IMAGE_WARP          -command warp_by_stmap\
                         -infile $MYDIR/grid_test_2_2500x1250.exr\
                         -stmapfile $MYDIR/stmap_remove_2400x1200.exr\
                         -outfile $MYDIR/grid_test_2_remove_by_stmap_2500x1250.exr
```

The result is:

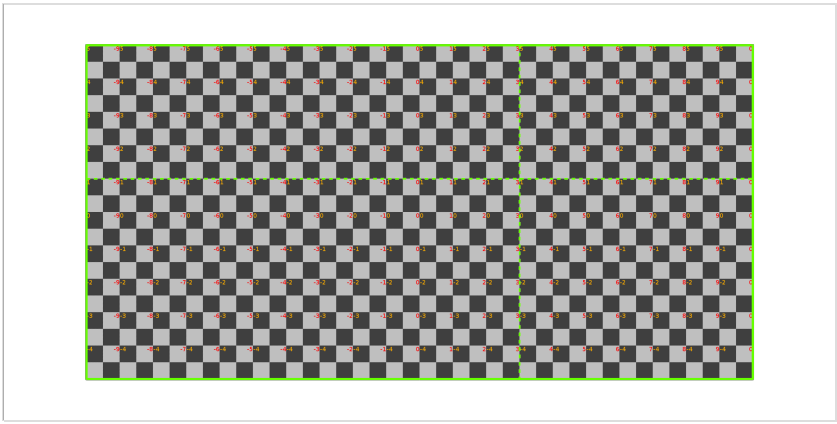


Here again you see the effect of the undersized ST-map in the lower left corner. Apart from that, the result is the same as in section 2.2.2.1.

2.2.2.3 Apply distortion by parameters

```
$SDV_IMAGE_WARP          -model LD_3DE4_Radial_Standard_Degree_4\
                         -direction apply\
                         -output_mode image\
                         -interpolation cubic\
                         -filmback 2.0 1.0\
                         -field_of_view 0.0 0.0 1.0 1.0\
                         -lens_center_offset 0.3 0.1\
                         -parameters .1 0 0 0 0 0 0 0\
                         -original_size 2000 1000\
                         -infile $MYDIR/grid_test_2_remove_2500x1250.exr\
                         -outfile $MYDIR/grid_test_2_apply_2500x1250.exr
```

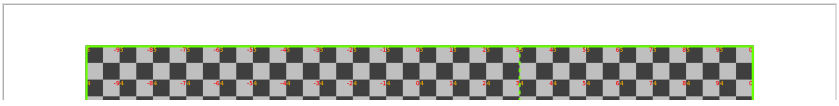
The result is:



2.2.2.4 Apply distortion by an ST-map

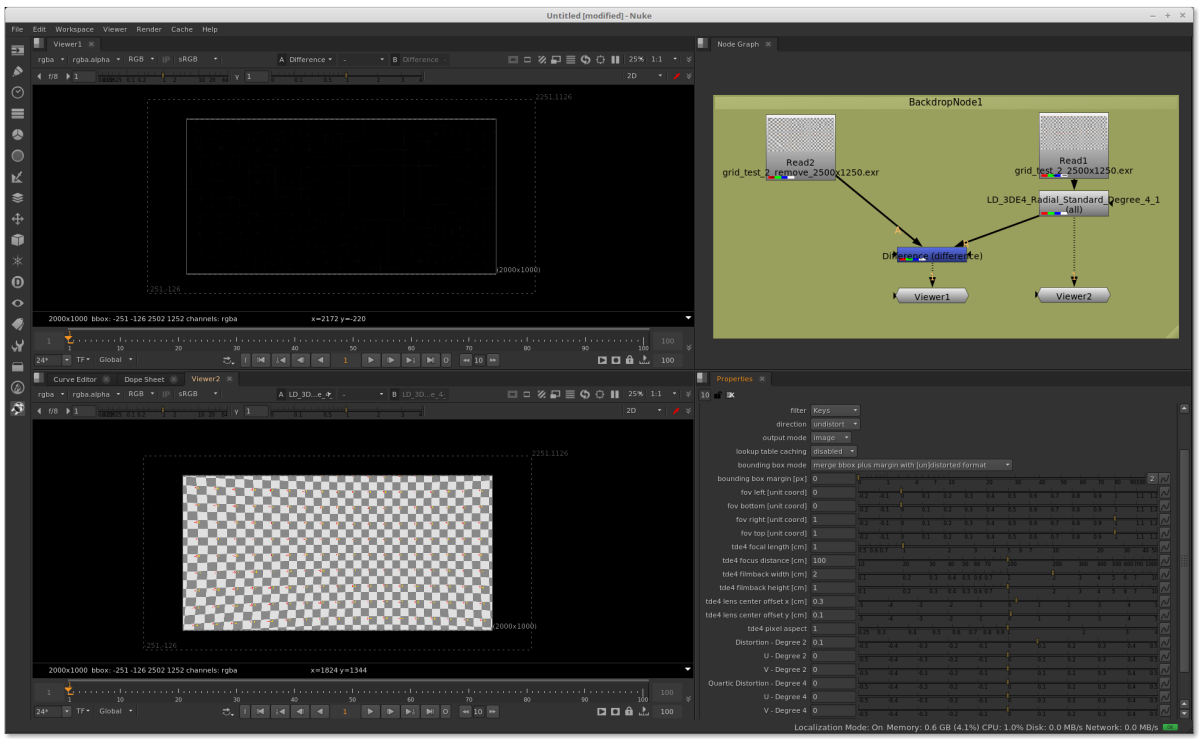
```
$SDV_IMAGE_WARP          -command warp_by_stmap\
                         -infile $MYDIR/grid_test_2_remove_by_stmap_2500x1250.exr\
                         -stmapfile $MYDIR/stmap_apply_2400x1200.exr\
                         -outfile $MYDIR/grid_test_2_apply_by_stmap_2500x1250.exr
```

The result is:



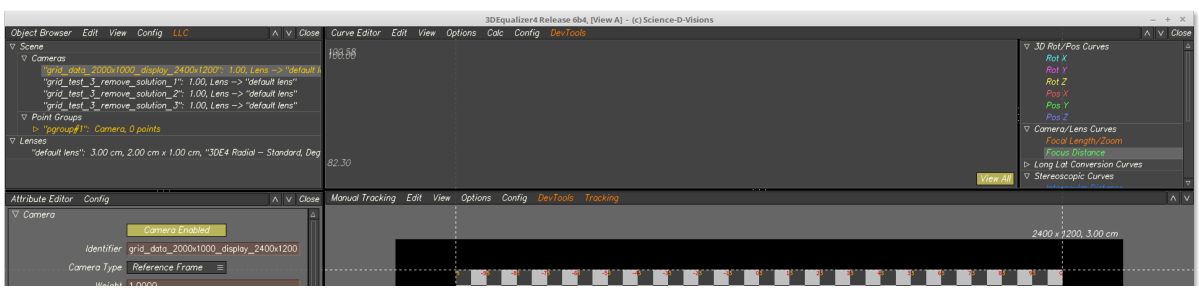
2.2.2.5 Verifying results with Nuke

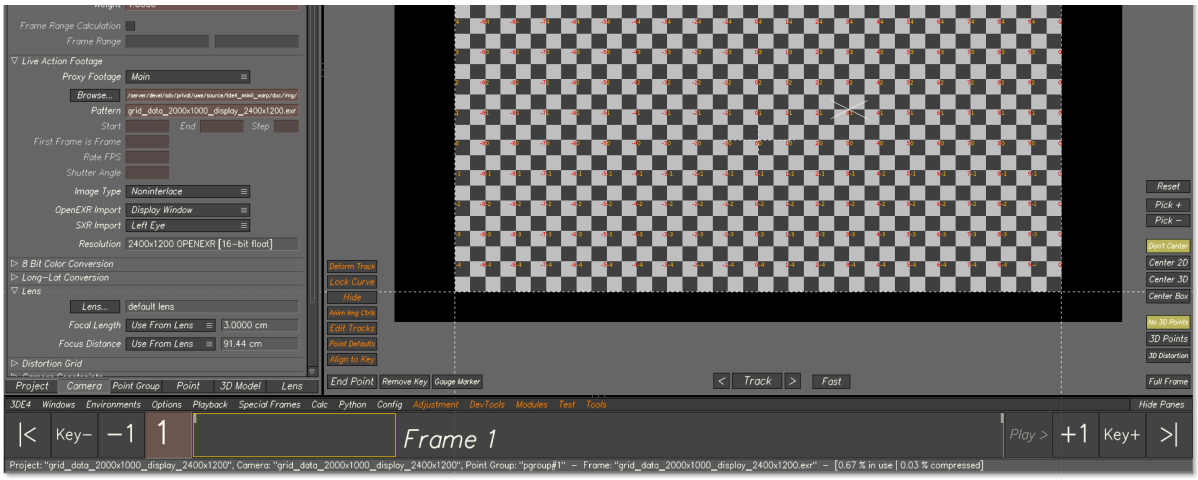
Of course, the results should not depend on the tool used, so let us have a look at Nuke. The following screen shot tells us that results from sdv_image_warp and Nuke are equivalent. Note that in Nuke filter mode *Keys* is the mode which corresponds to our mode *interpol_cubic*. On the left hand side in the backdrop node you see a read node which loads the image generated by sdv_image_warp. On the right hand side the original image is loaded and then undistorted using one of the plugins from the LDPK. The absolute difference image in the upper viewer is black, which shows that both methods are equivalent.



2.2.3 *Test 3: Display window larger than data window*

In the following we will investigate how sdv_image_warp responds to the following situation: Assume, the original footage given bei OpenEXR-images does not have any overscan data. Yet, the display window is enlarged and is supposed to mark the overscan area after removing lens distortion. In our example we consider our common grid image with a data window of 2000×1000, but the display window has size 2400×1200 and is aligned symmetrically around the data window. In 3DE4 the OpenEXR-image is loaded with property *OpenEXR Import* set to *Display Window* so that we can see the full domain of the image. The field of view is set correctly to the area given by the data window, i.e. to the domain of the original footage:





We should be able to remove lens distortion from this footage and get the same numeric results as in the tests before. There are several **equivalent** solutions for this task.

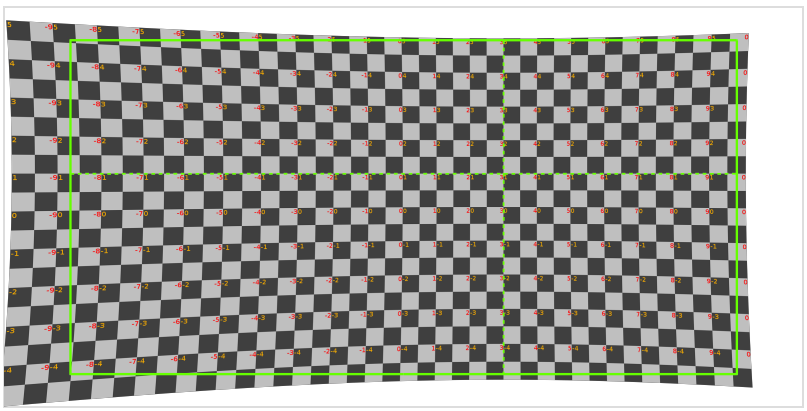
2.2.3.1 Solution 1

For the first solution we do the following:

- We interpret the image by means of the *display window*, that is `-window display`.
- We do not add overscan, since it is already defined by the display window: `-overscan 0 0`.
- We pass the information about the size of the original footage: `-original_size 2000 1000`.
- We pass the default field of view 0 0 1 1.

```
$SDV_IMAGE_WARP         -model LD_3DE4_Radial_Standard_Degree_4\
                        -window display\
                        -overscan 0 0\
                        -original_size 2000 1000\
                        -field_of_view 0 0 1 1\
                        -direction remove\
                        -output_mode image\
                        -interpolation cubic\
                        -filmback 2.0 1.0\
                        -lens_center_offset 0.3 0.1\
                        -parameters .1 0 0 0 0 0 0 0\
                        -infile $MYDIR/grid_data_2000x1000_display_2400x1200.exr\
                        -outfile $MYDIR/grid_test_3_remove_solution_1.exr
```

The drawback of this method is that the field of view we pass here does not coincide with the values given in 3DE4. Using this method in a more complicated situation with non-trivial field of view in 3DE4 would demand to calculate a combined, complicated field of view and then pass this combined result to sdv_image_warp, which is definitely not what we want. Yet, it makes sense to use this method in consistency checks.



2.2.3.2 Solution 2

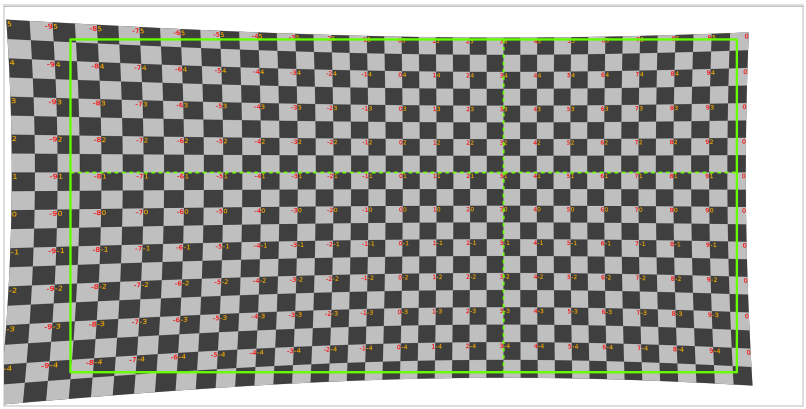
The second solution is not so different from the first one.

- We interpret the image by means of the *data window*, that is `-window data` (the default).
- We add some overscan: `-overscan 200 100`

- As in the first solution we pass the information about the size of the original footage: `-original_size 2000 1000`.
- As in the first solution we pass the default field of view 0 0 1 1.

```
$SDV_IMAGE_WARP          -model LD_3DE4_Radial_Standard_Degree_4\
                         -window data\
                         -overscan 200 100\
                         -original_size 2000 1000\
                         -field_of_view 0 0 1 1\
                         -direction remove\
                         -output_mode image\
                         -interpolation cubic\
                         -filmback 2.0 1.0\
                         -lens_center_offset 0.3 0.1\
                         -parameters .1 0 0 0 0 0 0 0\
                         -infile $MYDIR/grid_data_2000x1000_display_2400x1200.exr\
                         -outfile $MYDIR/grid_test_3_remove_solution_2.exr
```

The drawback of this method is the same as in the first solution; the field of view we pass here does not coincide with the values given in 3DE4. The result coincides with the result from the first method.



2.2.3.3 Solution 3

For the third solution we do the following:

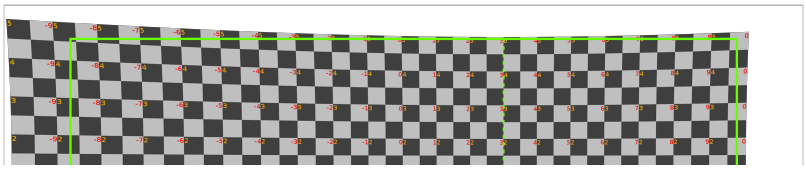
- We interpret the image by means of the *display window*
- We do not add overscan.
- We set the original size to 2400×1200. Albeit this is not really the size of the original footage, it does at least describe the footage imported in 3DE4 due to the setting of *OpenEXR Import*.
- We pass the same field of view as we used in 3DE4: 0.08333333 0.08333333 0.91666667 0.91666667.

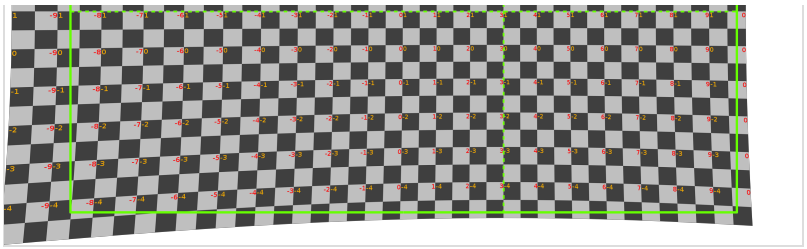
```
$SDV_IMAGE_WARP          -model LD_3DE4_Radial_Standard_Degree_4\
                         -window display\
                         -overscan 0 0\
                         -original_size 2400 1200\
                         -field_of_view 0.08333333 0.08333333 0.91666667 0.91666667\
                         -direction remove\
                         -output_mode image\
                         -interpolation cubic\
                         -filmback 2.0 1.0\
                         -lens_center_offset 0.3 0.1\
                         -parameters .1 0 0 0 0 0 0 0\
                         -infile $MYDIR/grid_data_2000x1000_display_2400x1200.exr\
                         -outfile $MYDIR/grid_test_3_remove_solution_3.exr
```

Equivalently, you can pass options

```
                         -window data\
                         -overscan 200 100\
```

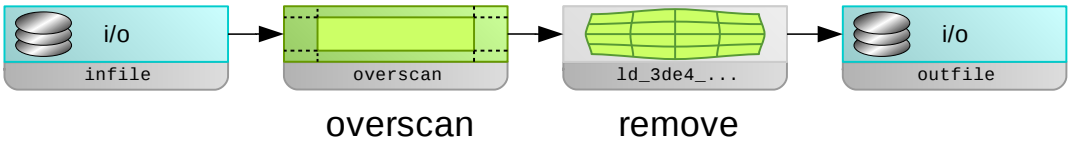
The options passed here are close to the settings in 3DE4, so this seems to be at least more intuitive than the first and second method. The result is



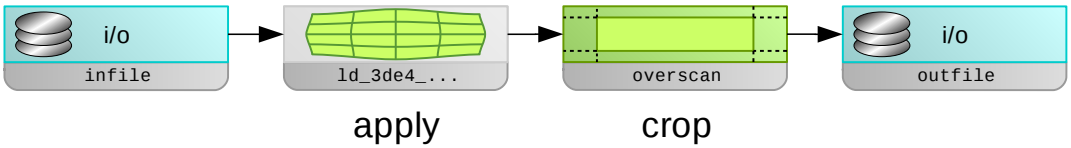


2.2.4 *Test 4: Various lens distortion models with non-default values*

In this section we test the pipeline undistort-redistort for all lens distortion models. All lens distortion parameters as well as lens center and field of view are set to non-default values. We start from the original grid as in the previous sections, then remove distortion and reapply distortion again. The pipeline for removing lens distortion in this test is:

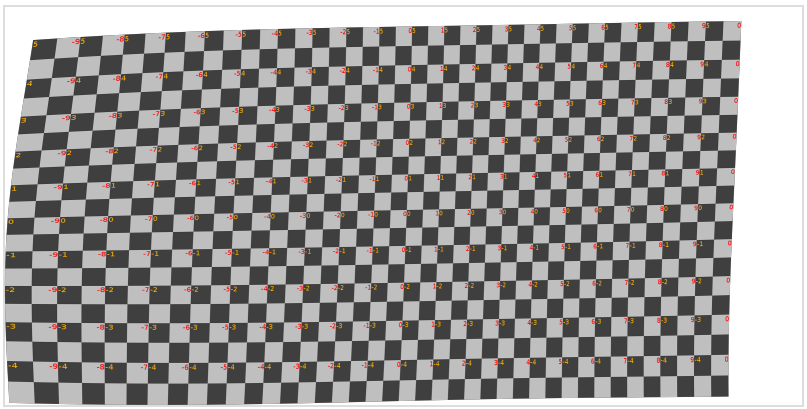


We then take the resulting image, apply lens distortion and crop it back to its original size:



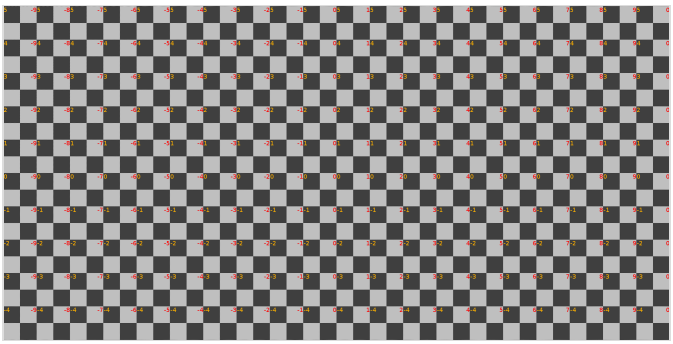
2.2.4.1 LD 3DE4 Anamorphic - Rescaled, Degree 4

```
$SDV_IMAGE_WARP         -model LD_3DE4_Anamorphic_Rescaled_Degree_4\
                        -direction remove\
                        -output_mode image\
                        -interpolation cubic\
                        -filmback 2.0 1.0\
                        -field_of_view 0.05 0.025 .975 .95\
                        -lens_center_offset 0.3 0.1\
                        -overscan 200 100\
                        -parameters .2 .1 .15 -.15 0.12 -.13 .14 -.1 .13 -.12 2.5 0.95 1.065 2.0\
                        -original_size 2000 1000\
                        -infile $MYDIR/grid_2000x1000.png\
                        -outfile $MYDIR/grid_remove_anamorphic_rescaled_degree_4_2400x1200.png
```

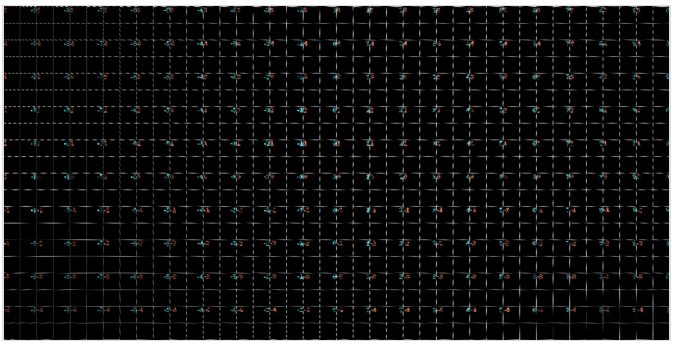


```
$SDV_IMAGE_WARP         -model LD_3DE4_Anamorphic_Rescaled_Degree_4\
                        -direction apply\
                        -output_mode image\
                        -interpolation cubic\
                        -filmback 2.0 1.0\
                        -field_of_view 0.05 0.025 .975 .95\
                        -lens_center_offset 0.3 0.1\
                        -crop 200 100\
                        -parameters .2 .1 .15 -.15 0.12 -.13 .14 -.1 .13 -.12 2.5 0.95 1.065 2.0\
                        -original_size 2000 1000\
                        -infile $MYDIR/grid_remove_anamorphic_rescaled_degree_4_2400x1200.png\
```

```
                        -outfile $MYDIR/grid_reapply_anamorphic_rescaled_degree_4_2000x1000.png
```

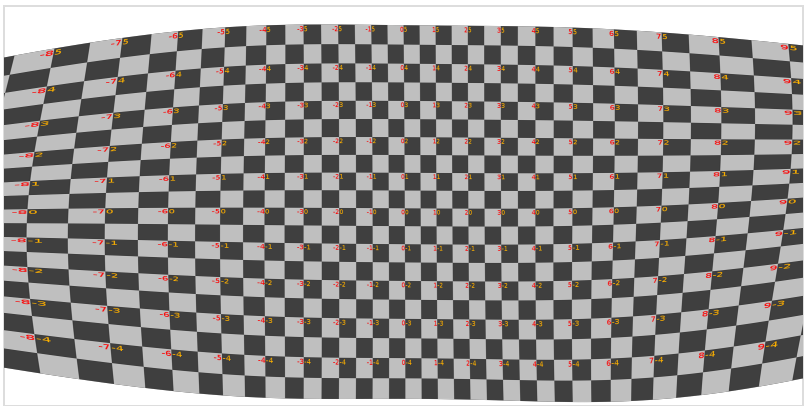


Compute difference image with respect to original image, and amplify:

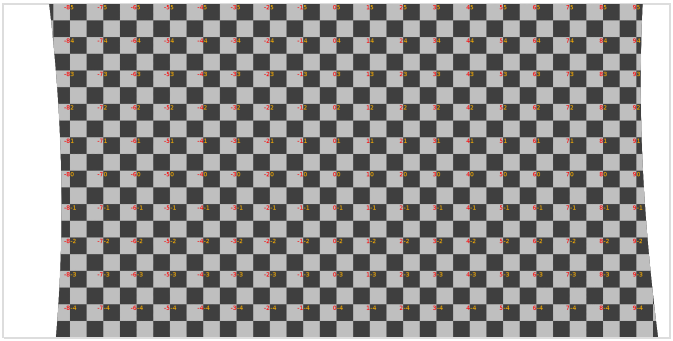


2.2.4.2 LD 3DE4 Anamorphic - Standard, Degree 4

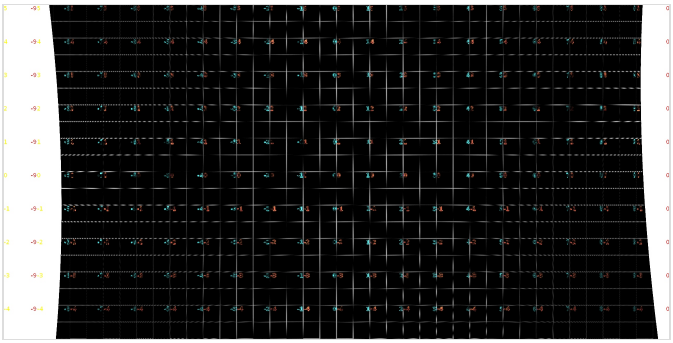
```
$SDV_IMAGE_WARP         -model LD_3DE4_Anamorphic_Standard_Degree_4\
                        -direction remove\
                        -output_mode image\
                        -interpolation cubic\
                        -filmback 2.0 1.0\
                        -field_of_view 0.05 0.025 .975 .95\
                        -lens_center_offset 0.05 0.1\
                        -overscan 200 100\
                        -parameters .2 .1 .15 -.15 0.12 -.13 .14 -.1 .13 -.12 2.5 0.95 1.065\
                        -original_size 2000 1000\
                        -infile $MYDIR/grid_2000x1000.png\
                        -outfile $MYDIR/grid_remove_anamorphic_standard_degree_4_2400x1200.png
```



```
$SDV_IMAGE_WARP         -model LD_3DE4_Anamorphic_Standard_Degree_4\
                        -direction apply\
                        -output_mode image\
                        -interpolation cubic\
                        -filmback 2.0 1.0\
                        -field_of_view 0.05 0.025 .975 .95\
                        -lens_center_offset 0.05 0.1\
                        -crop 200 100\
                        -parameters .2 .1 .15 -.15 0.12 -.13 .14 -.1 .13 -.12 2.5 0.95 1.065\
                        -original_size 2000 1000\
                        -infile $MYDIR/grid_remove_anamorphic_standard_degree_4_2400x1200.png\
                        -outfile $MYDIR/grid_reapply_anamorphic_standard_degree_4_2000x1000.png
```

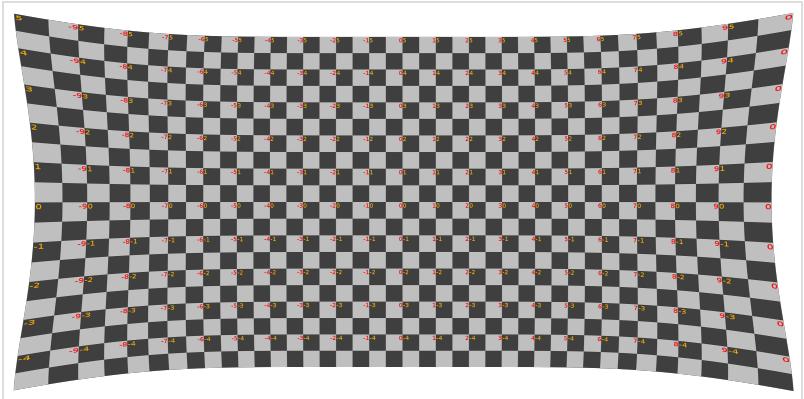


Compute difference image with respect to original image, and amplify:

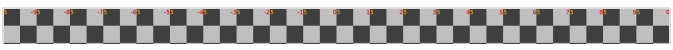


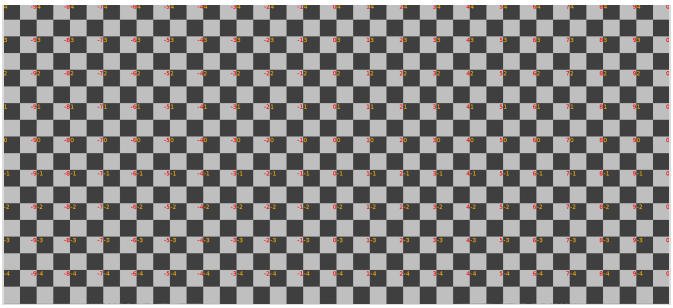
2.2.4.3 LD 3DE4 Anamorphic - Degree 6

```
$SDV_IMAGE_WARP          -model LD_3DE4_Anamorphic_Degree_6\
                         -direction remove\
                         -output_mode image\
                         -interpolation cubic\
                         -filmback 2.0 1.0\
                         -field_of_view 0.05 0.025 .975 .95\
                         -lens_center_offset -0.03 0.02\
                         -overscan 200 100\
                         -parameters -.025 -.02 -.0252 .0253 0.02 .03 .04 .035 .03 .035 .035 .04 .06 .07 .045 .01 -.05 .03\
                         -original_size 2000 1000\
                         -infile $MYDIR/grid_2000x1000.png\
                         -outfile $MYDIR/grid_remove_anamorphic_degree_6_2400x1200.png
```

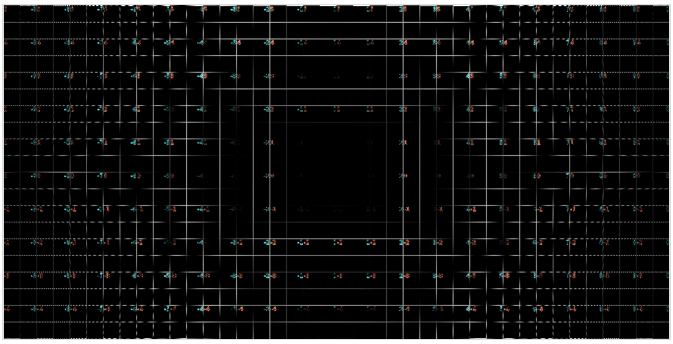


```
$SDV_IMAGE_WARP          -model LD_3DE4_Anamorphic_Degree_6\
                         -direction apply\
                         -output_mode image\
                         -interpolation cubic\
                         -filmback 2.0 1.0\
                         -field_of_view 0.05 0.025 .975 .95\
                         -lens_center_offset -0.03 0.02\
                         -crop 200 100\
                         -parameters -.025 -.02 -.0252 .0253 0.02 .03 .04 .035 .03 .035 .035 .04 .06 .07 .045 .01 -.05 .03\
                         -original_size 2000 1000\
                         -infile $MYDIR/grid_remove_anamorphic_degree_6_2400x1200.png\
                         -outfile $MYDIR/grid_reapply_anamorphic_degree_6_2000x1000.png
```



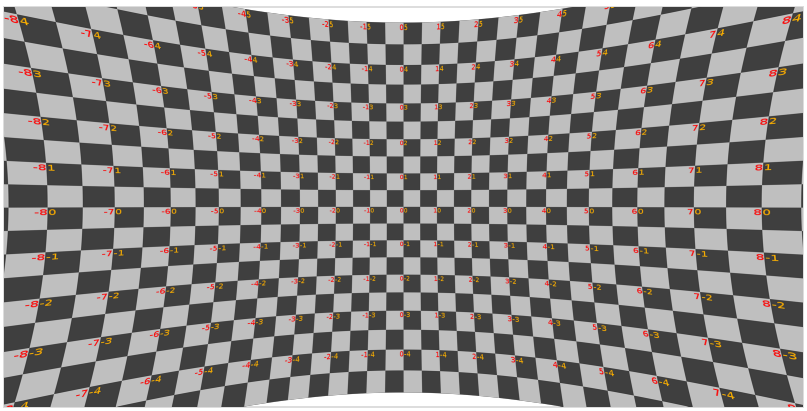


Compute difference image with respect to original image, and amplify:

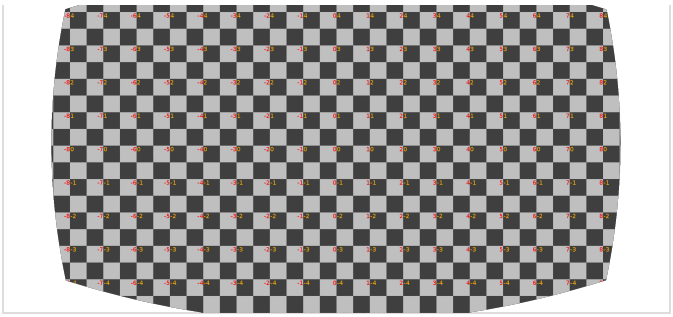


2.2.4.4 Radial - Fisheye, Degree 8

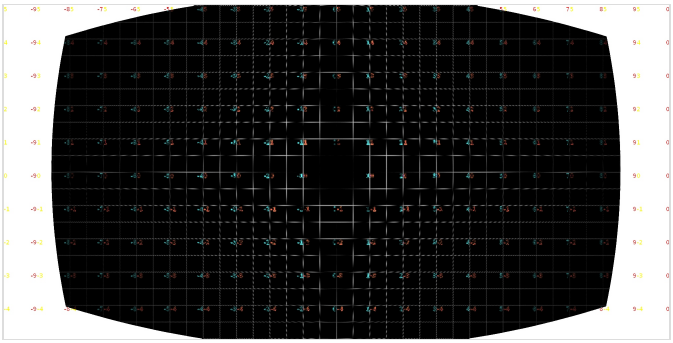
```
$SDV_IMAGE_WARP          -model LD_3DE4_Radial_Fisheye_Degree_8\
                         -direction remove\
                         -output_mode image\
                         -interpolation cubic\
                         -filmback 2.0 1.0\
                         -focal_length 1.2\
                         -field_of_view 0.05 0.025 .975 .95\
                         -lens_center_offset -0.03 0.02\
                         -overscan 200 100\
                         -parameters .1 -.05 .02 -.005\
                         -original_size 2000 1000\
                         -infile $MYDIR/grid_2000x1000.png\
                         -outfile $MYDIR/grid_remove_radial_fisheye_degree_8_2400x1200.png
```



```
$SDV_IMAGE_WARP          -model LD_3DE4_Radial_Fisheye_Degree_8\
                         -direction apply\
                         -output_mode image\
                         -interpolation cubic\
                         -filmback 2.0 1.0\
                         -focal_length 1.2\
                         -field_of_view 0.05 0.025 .975 .95\
                         -lens_center_offset -0.03 0.02\
                         -crop 200 100\
                         -parameters .1 -.05 .02 -.005\
                         -original_size 2000 1000\
                         -infile $MYDIR/grid_remove_radial_fisheye_degree_8_2400x1200.png\
                         -outfile $MYDIR/grid_reapply_radial_fisheye_degree_8_2000x1000.png
```

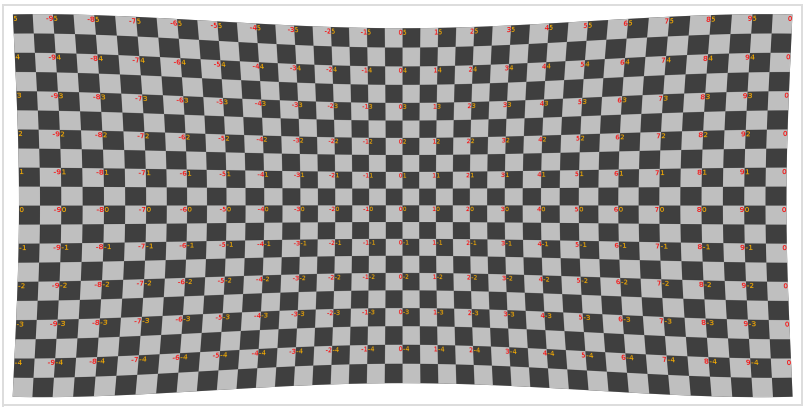


Compute difference image with respect to original image, and amplify:

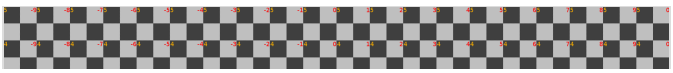


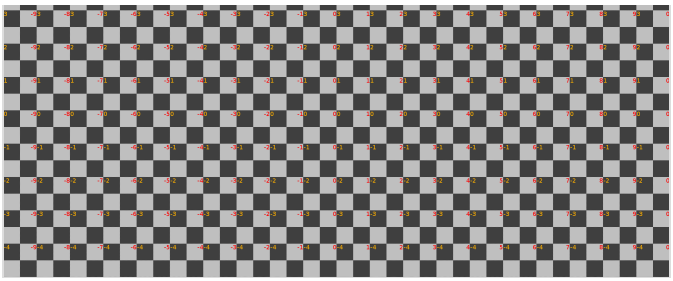
2.2.4.5 Classic LD Model

```
$SDV_IMAGE_WARP          -model LD_3DE_Classic_LD_Model\
                         -direction remove\
                         -output_mode image\
                         -interpolation cubic\
                         -filmback 2.0 1.0\
                         -field_of_view 0.05 0.025 .975 .95\
                         -lens_center_offset -0.03 0.02\
                         -overscan 200 100\
                         -parameters 0.3 1.1 0.06 -0.03 -.13\
                         -original_size 2000 1000\
                         -infile $MYDIR/grid_2000x1000.png\
                         -outfile $MYDIR/grid_remove_classic_ld_model_2400x1200.png
```

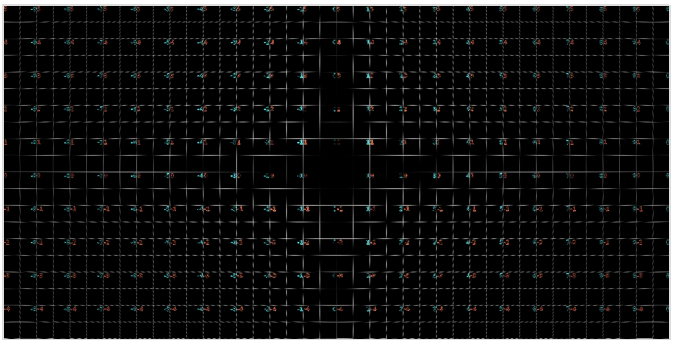


```
$SDV_IMAGE_WARP          -model LD_3DE_Classic_LD_Model\
                         -direction apply\
                         -output_mode image\
                         -interpolation cubic\
                         -filmback 2.0 1.0\
                         -field_of_view 0.05 0.025 .975 .95\
                         -lens_center_offset -0.03 0.02\
                         -crop 200 100\
                         -parameters 0.3 1.1 0.06 -0.03 -.13\
                         -original_size 2000 1000\
                         -infile $MYDIR/grid_remove_classic_ld_model_2400x1200.png\
                         -outfile $MYDIR/grid_reapply_classic_ld_model_2000x1000.png
```





Compute difference image with respect to original image, and amplify:

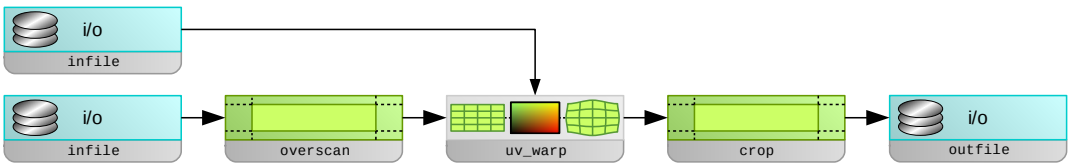


2.3 Image processing scheme

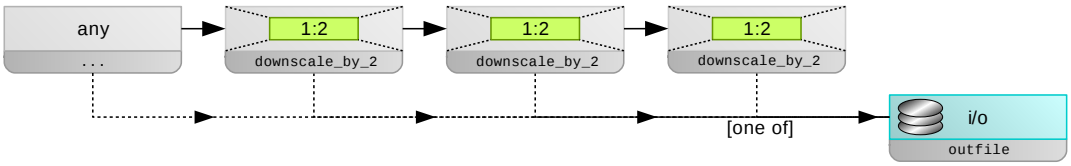
sdv_image_warp basically controls a light-weight, node-based image processing system. In this section we have look at the underlying image processing pipeline. The basic task of sdv_image_warp is of course removing and applying lens distortion to footage represented in the following diagram by an *infile*-node. At your choice, the domain of this footage is extended symmatrically by means of option `-overscan` or `overscan_size`, which is represented by the *overscan*-node. Speaking in terms of data window and display node, the overscan node extends the data window, but leaves the display window unmodified. The *distortion*-node (i.e. a node which uses one of 3DE4's lens distortion models as a warp function) acts on the display window and produces an image having the same display window. Pixel data mapped to somewhere outside this display window will not get lost as long as you extend the data window by overscan as mentioned before. Usually, when removing distortion the result would now be written to the file system (*outfile*-node). When re-applying lens distortion you might be interested in cropping the overscanned images back to their original sie (or to some other size), which is represented by the *crop*-node in the diagram. The *crop*-node is controlled by one of the options `-crop` or `-crop_size`. The node is nothing else but an *overscan*-node, yet margins are interpreted with reverse sign. The pipeline is given by the following diagram:

As we have seen in the use-cases, sdv_image_warp can also be used to create ST-maps (as they are called in e.g. in Nuke) by setting `-output_mode` to `stmap`. The only difference in the pipeline is, that instead of an *infile*-node we now have a procedural image generator node named *standard_uv_ramp*. The size of this ramp image is given by option `-original_size`. It is interpreted as an image with equal data and display window. Hence, in order to create a proper ST-map for removing lens distortion you probably want to apply the same amount of overscan as you would to in the direct, parameter-based approach. This leads to the following image processing pipeline:

Once you have created an ST-map, you want to apply it to footage, either by means of sdv_image_warp or in some of the well-known compositing systems. If you decide to do this in sdv_image_warp, the pipeline would look as shown below. The upper *infile*-node represents the ST-map you previously created, the lower one represents the footage. Instead of an *LD-3DE4-distortion*-node we now have a *uv_warp*-node (names may change in the future) which interprets the ST-map as a warp function and applies it to the (overscanned) footage:



It had been requested by users to provide the functionality of downscaling the output images by factors 2, 4, or 8, in order to create proxy material. For this purpose `sdv_image_warp` offers the option `-downscale` with possible values 1 (the default, does nothing), 2, 4, 8. In terms of our image processing pipeline, the downscale is alway applied last, directly before writing the image to the filesystem. Therefore the downscale acts upon all pixel-sized quantities envolved: data window, display window, overscan margins and crop margins:



Document created with CDML from /server/devel/sdv/privat/uwe/source/tde4_minil_warp/doc/cdml/tde4_minil_warp.xml

---