IFAS USER GUIDE

Nowadays, multiple image processing areas have shown the necessity of what is called image *fidelity assessment*, i.e., the objective assessment of differences between a reference and a test image. More specifically, the objective of *image fidelity assessment* is to determine if there are visual differences between a given test image and the corresponding reference image considering certain features and if so to quantify those differences. The variety of image domains requiring image fidelity assessment implies the need for application specific or even application-tailored methodology rather than a one-size-fits-all solution. While many candidate algorithms already exist, testing them and identifying the best ones for a given use case is far from an easy exercise. Thus, it often takes a lot of time and effort to even prepare the test environment for benchmarking, validation and/or developing.

The image Fidelity Assessment Software (iFAS) is a software tool designed to assist researchers, engineers and other users in the process of *image fidelity assessment*. iFAS provides easy access to a range of state-of-the-art methods as well as intuitive visualizations that aid data analysis. The software is freely available to all for non-commercial use. iFAS includes a range of common mechanisms for image fidelity assessment including computation of fidelity measures in a database of images (reference set of images and their corresponding test images), scatter plots, correlation analysis between human scores as well as other measures and modelling. This user interface is intended to reduce the time spent on the fidelity assessment phase of an emerging technology and/or benchmarking of image fidelity assessment algorithms.

REQUIREMENTS

Operating system: Windows 10 64-bit x86 and 32-bit x86. Minimum 5 GB disk space to download and install Anaconda. To run iFAS: Minimum 1000 MHz processor, 2048 MB RAM, 512 MB of hard-drive space, 3D Acceleration Capable Video card with at least 512 MB.

Python virtual environment configuration including *pip, scipy, pandas, tk, matplotlib, opencv, pywavelets*

Install Anaconda | https://www.anaconda.com/ using the default settings

```
# to create enviroment
conda create --name ifas_venv --file requirements.txt
# To use environment
conda activate ifas_venv
cd /path/to/iFAS/folder
python main.py
# To use when finishing with environment
conda deactivate
```

IFAS USER INTERFACE

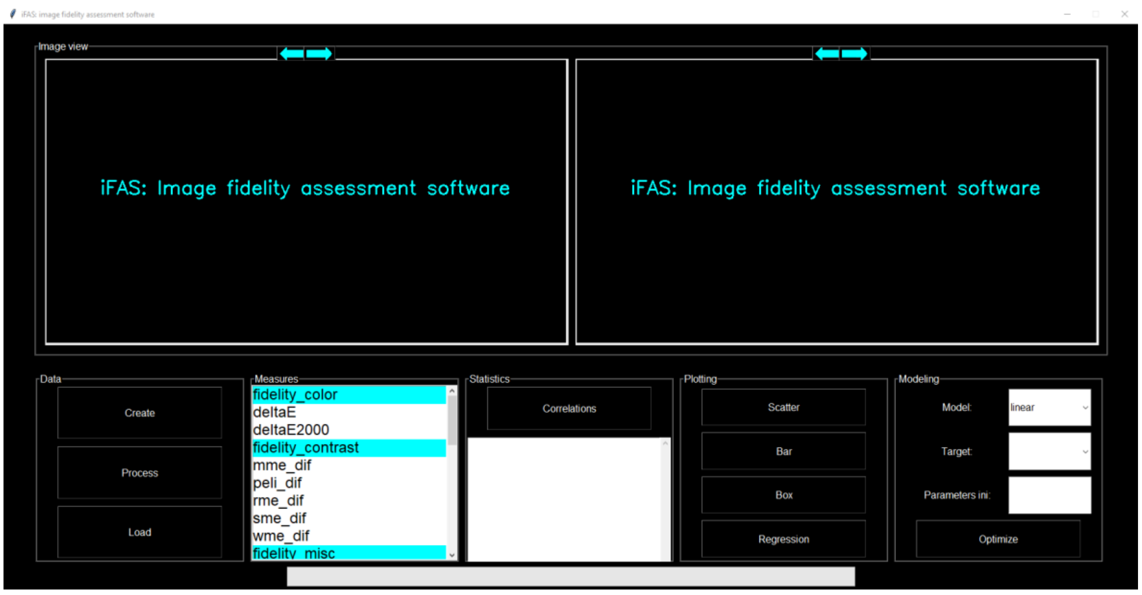


Figure 1 iFAS user interface

The iFAS interface is shown in Figure 1. The interface is composed by the following components:

IMAGE VIEW PANE

The image view pane is used to display the current reference and test images from the loaded database, left and right sub-panes, respectively. When the user clicks inside the image sub-pane, the image is opened in full resolution for a better inspection. The arrows on the top of the images are the controls to navigate through the image database. Click Left or Right arrow to move across the reference image (left sub-pane) and the test images (right sub-pane).

DATA PANE

The data pane contains iFAS' functionalities for data creation, processing and loading. This will be explained in more detail in Section iFAS database.

MEASURES PANE

The measures pane is the list of the fidelity measures available to be computed in any given database. The measures are selected by clicking on them. Note that the Python scripts named *fidelity_*.py* are highlighted using cyan color. If the user selects one of those lines, the software automatically selects all the functions within that *.py* file. Currently iFAS includes the following list of fidelity measures:

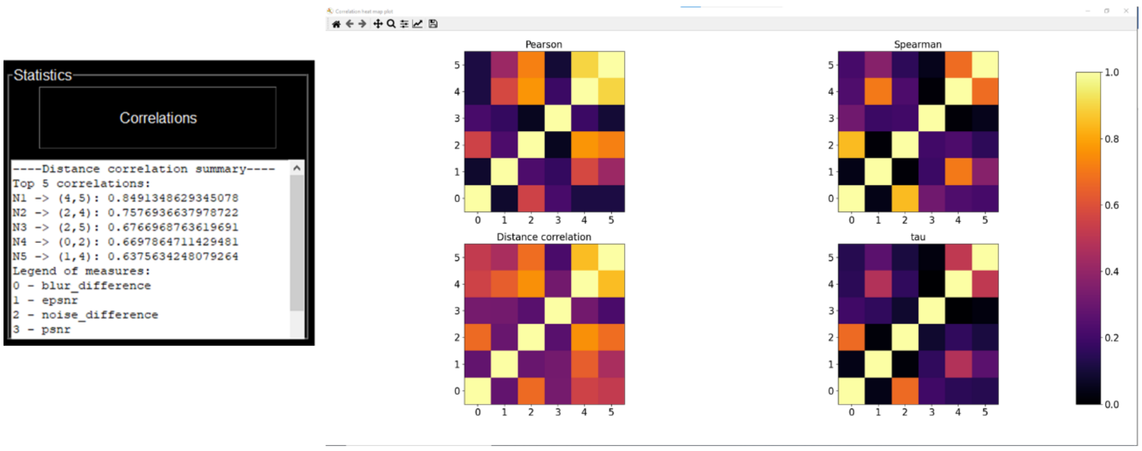
1. **Five miscellaneous measures**: (1) Peak signal to noise ratio (PSNR), (2) Structural similarity index measure (SSIM), (3) Wavelet domain noise difference, (4) variance of Laplacian blur difference, (5) Edge Peak signal to noise ratio (EPSNR)
2. **Twelve texture-based measures**: (1) autoregressive model difference, (2) Gaussian Markov random fields difference, (3) grey level cooccurrence matrix difference, (4) acf2d_dif, (5) local binary patterns difference, (6) Laws filter bank difference, (7) Eigen-filter bank difference, (8) wedge and ring filter bank based difference, (9) Gabor filter bank based difference, (10) Laplacian filter bank based differences, (11) steerable filter bank based differences, (12) Granulometry moments difference

3. **Five fidelity measures based on contrast**: (1) Simple measure of enhancement, (2) Weber's measure of enhancement, (3) Michelson's measure of enhancement, (4) Root mean squared measure of enhancement, (5) Peli's measure of enhancement
4. **### image color difference measures**: (1), (2), (3), (4), (5) -> To be implemented

The fidelity measures are Python functions which have two inputs and one output. The inputs are two RGB images of the same size (reference and test image) to be compared by the fidelity measure and the output is a float representing the differences between the two images. It is possible to add new fidelity measures to iFAS. To do so it is necessary to add a new Python script file *.py* under the folder *fidelity_measures.* This process will be explained in more detail in Section Advanced iFAS functionalities.

STATISTICS PANE

The statistics pane shows the available set of statistics to be computed using the data set. iFAS currently includes four correlation matrices: Pearson, Spearman, tau and correlation of distances. After loading a database, it is possible to compute correlations between the measures available in the *.csv* file that should be together with the database. The result of clicking the Correlation button is the legend of the measures together with a heat map of the correlations. The correlations are exported as *.csv* files in the database directory: **_ifas_dist_corr.csv*, **_ifas_pearson.csv*, **_ifas_spearman.csv* and **_ifas_tau.csv*. In the correlations pane the information related to the top five correlations is shown. Figure 2 shows the Statistics pane and the heat map of correlations. These correlations are computed using the values available in the *.csv* file in the database.



```
----Distance correlation summary----
Top 5 correlations:
N1 -> (4,5): 0.8491348629345078
N2 -> (2,4): 0.7576936637978722
N3 -> (2,5): 0.6766968763619691
N4 -> (0,2): 0.6697864711429481
N5 -> (1,4): 0.6375634248079264
Legend of measures:
0 - blur_difference
1 - epsnr
2 - noise_difference
3 - psnr
```

Figure 2 Statistics pane and matplotlib heat map of correlations

PLOTTING PANE

After loading a database, the plotting pane has the following functionalities:

1. Scatter plot of the data in the *.csv* file with one of the measures in the x-axis and another measure in the y-axis (the axes are labelled correspondingly to the measures). The plot can be controlled using the left, right, up and down keys in order to change between the different measures (Figure 3 top-left).
2. Bar plot of the correlations of each measure against the target variable selected in the Modeling pane (Figure 3 bottom-left).
3. Box plot of the correlation against the target variable for each fidelity measure. Each box represents the correlation distribution for each reference image (Figure 3 bottom-right).

Scatter plot and regression line between the available fidelity measures and the target variable selected in the Modeling pane. The regression line is explained with more detail in Section Modeling pane.

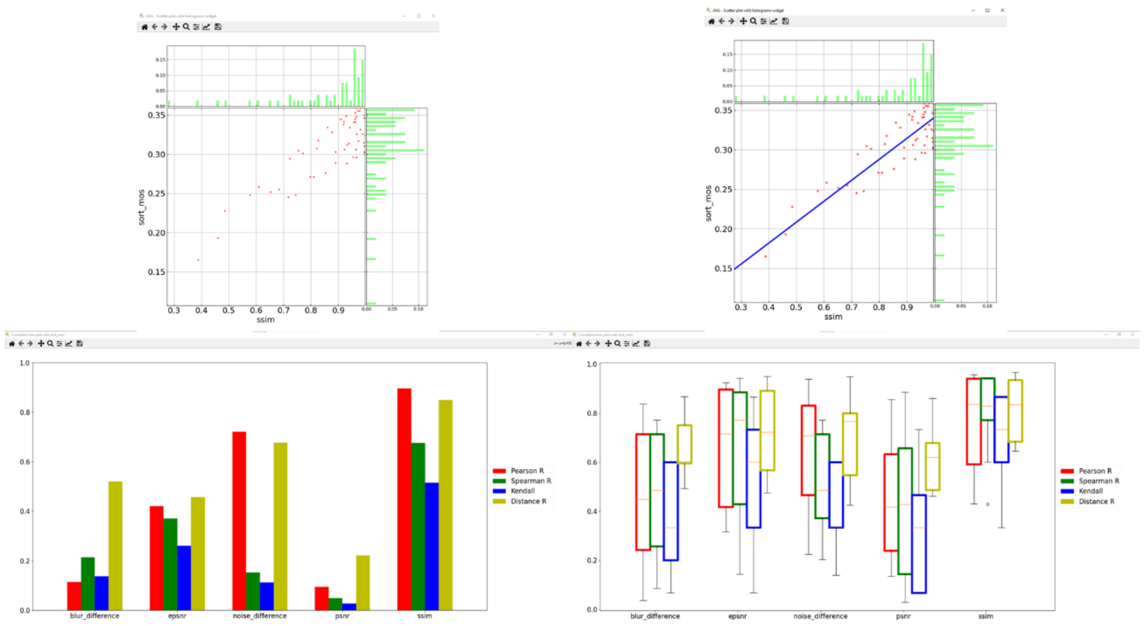


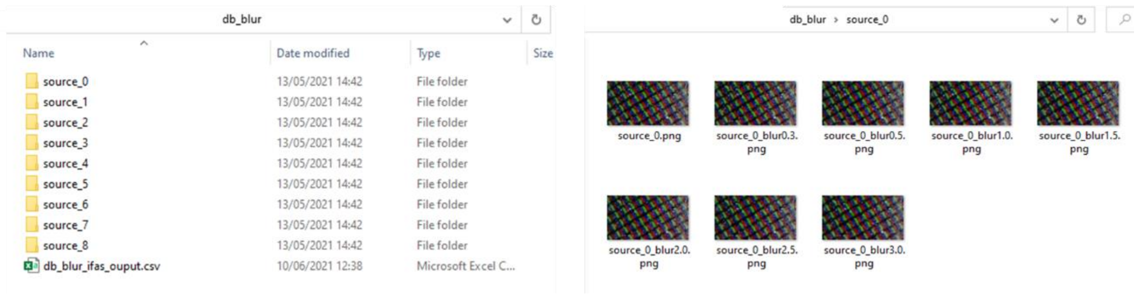
Figure 3 Available plotting tools in iFAS. From left to rigth and top to bottom: Scatter plot, Scatter plot and regression line, correlations bar plot, correlations box plot per-source

MODELING PANE

The modeling pane contains the settings for creating models between a target variable and the given fidelity measures. It is possible to select between **six** different **regression models** in the drop box list Model: (1) linear, (2) quadratic, (3) cubic, (4) exponential, (5) logistic, (6) complementary error. The target is a drop box list of the available computed fidelity measures. The parameters ini is a text input reserved for comma separated values of the initial parameters for the optimization process executed with the Optimize button. After optimization has been done, it is possible to plot the results of the optimization using the Regression plot. The regression plot shows the Scatter plot and regression line between the available fidelity measures and the target variable (Figure 3 top-right). The ini parameters expects the following number of float values separated by coma:

1. Two parameters for the linear model: $y = a_0 + a_1 x$, e.g., *0.5,-0.5*
2. Three parameters for the quadratic model: $y = a_0 + a_1 x + a_1 x^2$, e.g., *-0.5,1,-0.5*
3. Four parameters for the cubic model: $y = a_0 + a_1 x + a_2 x^2 + a_3 x^3$, e.g., *-0.5,1,-1,0.5*
4. Three parameters for the exponential model: $y = a_0 + \exp\left(a_1 x + a_2\right)$, e.g., *0.5,-0.5,1*
5. Three parameters for the logistic model: $y = \frac{a_0}{1+\exp\left(-(a_1 x + a_2)\right)}$, e.g., *0.5,-0.5,1*
6. Two parameters for the complementary error model: $y = 0.5 - 0.5\,\mathrm{erf}\left(\frac{x-a_0}{\sqrt{2}a_1}\right)$, e.g., *0.5,-0.5*

IFAS DATABASE



db_blur

Name	Date modified	Type	Size
source_0	13/05/2021 14:42	File folder	
source_1	13/05/2021 14:42	File folder	
source_2	13/05/2021 14:42	File folder	
source_3	13/05/2021 14:42	File folder	
source_4	13/05/2021 14:42	File folder	
source_5	13/05/2021 14:42	File folder	
source_6	13/05/2021 14:42	File folder	
source_7	13/05/2021 14:42	File folder	
source_8	13/05/2021 14:42	File folder	
db_blur_ifas_ouput.csv	10/06/2021 12:38	Microsoft Excel C...	

db_blur › source_0

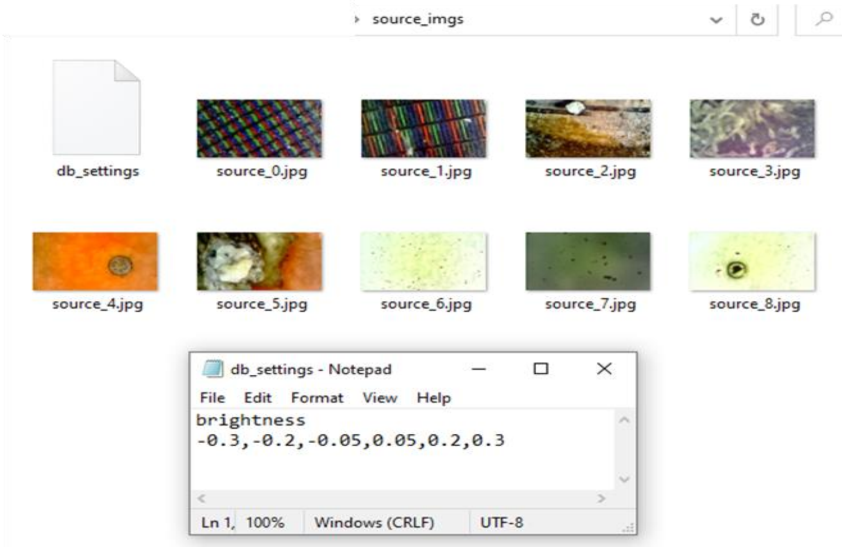
source_0.png
source_0_blur0.3.png
source_0_blur0.5.png
source_0_blur1.0.png
source_0_blur1.5.png
source_0_blur2.0.png
source_0_blur2.5.png
source_0_blur3.0.png

Figure 4 Example database file hierarchy

An iFAS database is simply several reference images and their respective test images as well as a *.csv* file where the computed fidelity measure values are permanently stored. Figure 4 shows an example of the database file hierarchy used in iFAS. iFAS uses the following folder hierarchy for the database:

1. Main Database folder contains
   a. Subfolder per reference image: the folder is named as the reference image (Figure 4 left)
      i. *.png* files of the reference image with its corresponding test images (Figure 4 right)
   b. A *.csv* file with the following columns:
      reference image name, test image name, fidelity measure 1, …, fidelity measure n

CREATING IFAS DATABASE FROM SOURCE IMAGES



```
brightness
-0.3,-0.2,-0.05,0.05,0.2,0.3
```

Figure 5 Example folder and db_settings file used by iFAS

1. Click on Create button
2. In the first prompt window select the folder with a set of source images
3. In the second prompt window select the destination folder and give a name to the database
4. The bar will start filling while creating the database
5. A window will prompt when database was created

Note: It is mandatory to have a *db_setings* file in the selected directory as shown in Figure 5. The first line name of the type of distortion to be applied to the reference images and the second line comma separated values of distortion levels. The folder with the source images also has a settings file named *db_settings* with the following rows:

1. row one contains the name of the distortion which should be at *add_distortions.py* script
2. row two contains the levels of distortion in comma separated values format

PROCESSING IFAS DATABASE

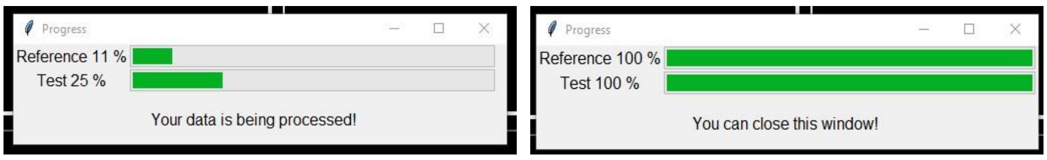


Figure 6 Database processing window

1. Select the measures in the measures pane. Note that the Python scripts named *fidelity_*.py* are highlighted using cyan color.
2. Click on Process
3. In the window prompt, select the folder of the set of source and test images as described above
4. A popup window with progress bar shows the progress of the computations (Figure 6)
5. The pop window will indicate when the window can be closed (process finished)
6. Results in a csv file on the database folder named **_ifas_ouput.csv*

Note: Multiple processes can be spawn depending on your system capabilities

LOADING DATABASE

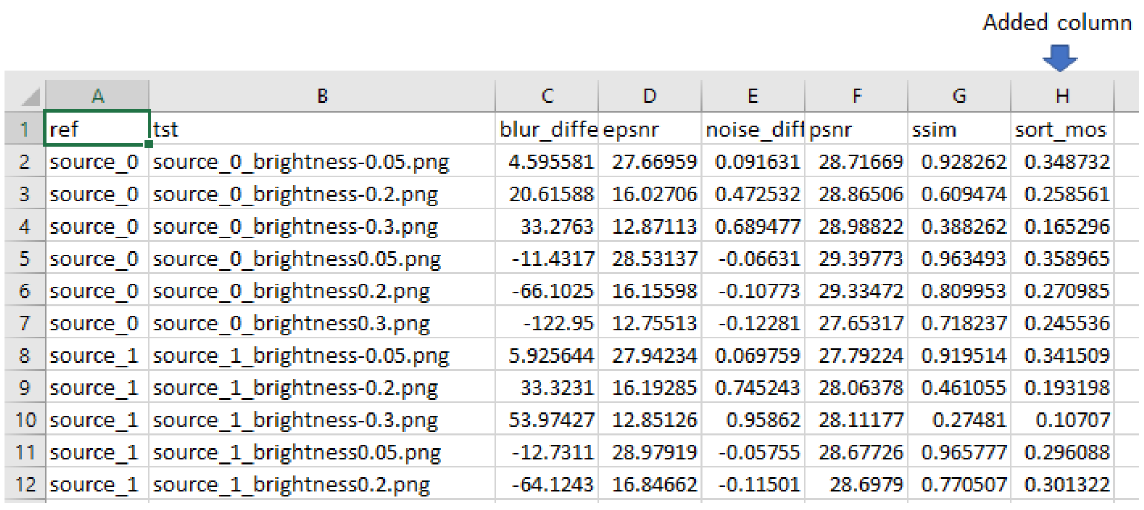
1. Select the folder of the set of source and test images as well as the *.csv* file
2. Prompt will indicate if the data was loaded successfully

After loading the database, it is possible to analyze the data using the functionalities explained in Sections Image view pane, Statistics pane, Plotting pane, Modeling pane.

ADVANCED IFAS FUNCTIONALITIES

This section describes iFAS advanced functionalities which requires Python scripting knowledge. iFAS advanced functionalities includes adding new distortion types, adding precomputed measures and/or adding new fidelity measures.

ADDING PRE-COMPUTED MEASURES



	A	B	C	D	E	F	G	H
1	ref	tst	blur_diffe	epsnr	noise_difl	psnr	ssim	sort_mos
2	source_0	source_0_brightness-0.05.png	4.595581	27.66959	0.091631	28.71669	0.928262	0.348732
3	source_0	source_0_brightness-0.2.png	20.61588	16.02706	0.472532	28.86506	0.609474	0.258561
4	source_0	source_0_brightness-0.3.png	33.2763	12.87113	0.689477	28.98822	0.388262	0.165296
5	source_0	source_0_brightness0.05.png	-11.4317	28.53137	-0.06631	29.39773	0.963493	0.358965
6	source_0	source_0_brightness0.2.png	-66.1025	16.15598	-0.10773	29.33472	0.809953	0.270985
7	source_0	source_0_brightness0.3.png	-122.95	12.75513	-0.12281	27.65317	0.718237	0.245536
8	source_1	source_1_brightness-0.05.png	5.925644	27.94234	0.069759	27.79224	0.919514	0.341509
9	source_1	source_1_brightness-0.2.png	33.3231	16.19285	0.745243	28.06378	0.461055	0.193198
10	source_1	source_1_brightness-0.3.png	53.97427	12.85126	0.95862	28.11177	0.27481	0.10707
11	source_1	source_1_brightness0.05.png	-12.7311	28.97919	-0.05755	28.67726	0.965777	0.296088
12	source_1	source_1_brightness0.2.png	-64.1243	16.84662	-0.11501	28.6979	0.770507	0.301322

Figure 7 Adding existing data using Excel or LibreOffice

Adding pre-computed measures or human scores to the database is done by manually copying the entries into the **_ifas_ouput.csv* file. Add in the first row and last column of the *.csv* file the name of the measure and in the subsequent rows the values corresponding to the pair reference – test image indicated in the first and second columns. Note that the user has the responsibility to make sure that the test image corresponds in scene to the reference images and the corresponding values as in Figure 7 example.

ADDING A NEW DISTORTION TYPE

Adding a new distortion type requires to create a python function inside the *processing\add_distortions.py* script. The function format for adding a new distortion type is three inputs and one optional output: an input image path, a float indicating the level of distortion, an output folder to write the resulting images and optionally returns the image. For example, the following is an example where the input image is modified in the brightness:

```
# Adds a constant to the brightness of the image when image is in bgr
def brightness(in_img, lvl=0.25, out_folder=None):
    if isinstance(in_img, str):
        in_img_fol = in_img
        in_img = cv2.imread(in_img_fol)
    # converting image to HSV in order to modify only the V component
    img_hsv = cv2.cvtColor(in_img.astype("float32") / 255., cv2.COLOR_BGR2HSV)
    # Adding the constant to the Y component
    img_hsv[::, ::, 2] += lvl
    # Cliping the results and converting back to bgr
    img_hsv[::, ::, 2] = np.clip(img_hsv[::, ::, 2], 0, 1)
    img_out = cv2.cvtColor(img_hsv.astype("float32"), cv2.COLOR_HSV2BGR)
    img_out = (255 * img_out).astype("uint8")

    if out_folder is not None:
        cv2.imwrite(out_folder, img_out)

    return img_out
```

ADDING A NEW FIDELITY MEASURE

Adding a new fidelity measure requires to create a python script as *fidelity_measures\fidelity_*.py.* Inside the script functions can be created in order to compute image differences. The fidelity measures are Python functions which have two inputs and one output. The inputs are two RGB images of the same size (reference and test image) to be compared by the fidelity measure and the output is a float representing the differences between the two images. For example, the following is an example where the Peak Signal to Noise Ratio is implemented as fidelity measure:

```
def psnr(ref_img, tst_img):
    mse = np.mean(np.power(ref_img - tst_img, 2))
    if mse != 0:
        psnr = 10 * np.log10((255. ** 2) / mse)
    else:
        psnr = np.inf
    return psnr
```