# Code description

August 14, 2017

## 1 cluster_test.py

This code is used for analyzing properties of all 100 images from `RealGalaxyCatalogue` sample provided by `galsim`. Currently it goes through all the galaxies and obtains their decomposition in the specified `basis` with specified `solver` and corresponding input parameters. Later on, the decomposition is analyzed through the coefficients obtained by `_visualize` function, which just plots the histogram distribution of the coefficient values. Also, it is used for perturbation of the initial galaxy vector, within the $\sigma_{noise}$ range, obtained from the stability tests (**??**).

It is intended to merge this with `test_script.py` (§ 7), after different visualization and clustering techniques are tested.

## 2 plotting_routines.py

This routine does all the plotting. Initial image is expected to be reshaped into `[size_X, size_Y]`, and all the others - reconstruction, residual, coefficient array - are given as 1D arrays. The plotting is a bit different for Cartesian and Polar basis, but can be actually implemented into one function.

The function plot_decomposition is used for only inner product decomposition, and plot_solution is used inside the `solver_routines` (§ 3).

## 3 solver_routines.py

Here, I select `solver` for obtaining the coefficients for the given `D`. After the coefficients are obtained they are plotted with the plot_solution routine (§ 2).

If this is called from within `stability_test.py`(**??**) then `noise_scale` - variance of the noise matrix - param must be specified, otherwise `noise_scale` param must be set to `None`. The reason for this is that during stability tests coefficients of the initial - zero noise - image are compared to the noised image coefficients, but if stability test is not under way then no comparison is needed and the obtained coefficients from one of the solvers are passed to the `plot_solution` from § 2.

## 4 sompy_custom/

This is the Self-Organizing-Map library, taken from this github repo. I started adding some topographic error estimations and I changed the visualization scripts to better suit me.

## 5 stability_test.py

This script is only used for stability tests of the given algorithm used for decomposition. Initial image is selected from the `galsim` *RealGalaxyCatalogue*, without Poisson noise added, and decomposed into the selected shapelet `basis`. After the initial decomposition, the obtained galaxy vector in shapelet space is compared to the average vector in the shapelet space of different noise realizations (current calculation of SNR needs to be corrected **!**). The averaging is done over the

number of different Gaussian noise matrices used for noising the original image, but keeping the SNR to the selected value. SNR range is from 20 - 50.

# 6 shapelet_dicts.py

This python routine initializes the basis dictionary `D` in the given `basis` and then calls the `_select_solver_do_fitting_plot` routine from `solver_routines` (§ 3) to do the decomposition with a selected `solver` with given input parameters.

If no image for decomposition is passed to this routine it selects image 91 from `galsim` *RealGalaxyCatalogue* sample data and does decomposition of that image.

All the calculation are done on 1D arrays, which are interpreted as vectors (e.g. 2D image passed is flattened to 1D array with `.flatten()`.

# 7 test_script.py

This script is made with a purpose to test various clustering and visualization algorithms. Currently under consideration are `Multi Dimensional Scaling` algorithm, `Self Organizing Map` algorithm and a simple scalar product based distance estimate (this should be deprecated). Intention is to merge this with the `cluster_test.py` code (§ 1).

# 8 utils/

All these codes are used as calling functions from one of the above scripts.

## 8.1 galsim_utils.py

Module with functions used only for generating weight image (best fit elliptical Gaussian) and extracting data from the `galsim.hsm.ShapeData.observedshape`, obtained from the `galsim` library.

## 8.2 I_O_utils.py

This contains only a handy I/O function for checking / creating a given folder path.

## 8.3 shapelet_utils.py

This script contains all the decomposition functions for Polar and Cartesian basis, single beta and compound. All the `decompose_*` functions first initialize a meshgrid.

Initially the X and Y 1D arrays are passed defining the size of the grid. If the chosen basis is Cartesian ($XY^*$) then meshgrid is simply obtained by doing `np.meshgrid(`$(X - x_0), (Y - y_0)$`)`, where $x_0$ and $y_0$ define the centroid of the initial image. Later on, if the basis is XY_Elliptical - elliptical Cartesian shapelet basis - then a coordinate transform is done within `elliptical_shapelet` routine, with specified `q` and `theta`, ellipticity and orientation of the ellipse.

For Polar basis, grid is R-$\phi$, where R is parameterized through $R = \sqrt{(X - x_0)^2 + (Y - y_0)^2}$ and $\phi =$ `np.atan2(`$Y - y_0, X - x_0$`)` for circular basis, and if elliptical basis is specified then $R$ is parameterized through $R = R_{circ}\sqrt{q \sin(\phi + \theta)^2 + \cos(\phi + \theta)^2/q}$, where $q$ and $\theta$ are ellipticity and orientation of the ellipse and $R_{circ}$ is the $R$ value in case of circular shapelet basis.