Polarisation image analysis

Table of Contents

Installation

Requirements

For the project to run these are the pre-requisites to be installed in the systems.

- Python 3.9 The project is fully written in python programming language. For installation, click here
- Pip. for installing various libraries used in the project. For installation click here
- Git. To access the project and run locally. For installation click here
- Any python IDE like Pycharm (optional but recommended , for ease of use)

Running locally

After you have successfully installed the pre-requisites, do the follow the instructions to set the project up and running.

1. clone the repository.
    a. Go to the command terminal.

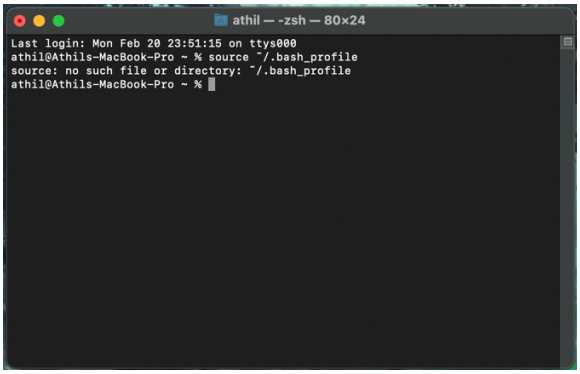


Figure 1 A command terminal may look like this

Note: based on the type of OS the terminal may look different

    b. Write the following command on the terminal.

```
git clone https://github.com/athilalthaf/polarisation_image_analysis
```

this will clone the repository and create folder polarisation_image_analysis on the respective directory. This folder will be contains some scripts and folders with some image inside it.

    c. to change the directory and access the contents inside. On terminal itself write the following command.

```
cd polarisation_image_analysis
```

2. install the python libraries needed.
    a. on the terminal write the following command

```
pip install -r requirements.txt
```

this will read the text file requirements.txt and will install the library packages and its versions. Now you have everything that you needed for the project to run on your system.

Project contents

This section includes the content description of the project. Skip to examples /usage to start running the program.

Scripts

pol_img_func.py

Script act as a function library for processing skylight images. The functions in this script can be used for elevation distortion correction, map elevation and azimuth of the sky regions. Along with that we can project the skylight images into a equirectangular projection for the ease of processing. There is a partial implementation gaussian kernel function to convolve a equirectangular image to further analyse the images. Functions can be tested with some of the sample images given in test_images.

lib_importer.py

An import file that imports all the necessary python libraries needed for the project. Use :

```
from lib_importer import *
```

If needed, any new libraries can be added to this script making it automatically available for all the other files in this project.

plots_examples.py

This script loads all functions from pol_img_func, sample images from test_images and then it plots the resulting figures. Running this file gives you a series of plots. By default, the image loaded is a low res image which can be changed to higher resolution in the script itself.

bee_eye_subsampling.py

A modified version of create_bee_eye.py written by Evripidis Gkanias. plots ommatidial co-ordinates of bee right eye.

dummy_file_create_bee_eye.py

A dummy file of create_bee_eye.py just to refer the original version

Folders

Plot_examples

Contains sample plots of based on the functions in pol_img_functions.py .

- blend_mid_equi_after.png: An equirectangular projection of elevation corrected sample image
- calib_img_subsampled_azimap_fig.png: An azimuth map of the sample image
- calib_img_subsampled_elevationmapcorr.png : An elevation map of the sample image corrected for the elevation distortion while capturing a fisheye images
- calib_img_subsampled_elevationmapsrc.png : An elevation map of the sample image with elevation distortion
- simple_gauss_kernel_60.png : A simple gaussian kernel with elevation correction at the 60 degree elevation
- tiled_img_sample_01.png : A tiled equirectangular image with 45 degree tile at its azimuths and 30 degree tile at elevation

Test_images

Folder that holds some test images to play with , there are different qualities of sample image. Ideally use the one which low quality for trials and troubleshooting , then proceed to the higher quality ones.

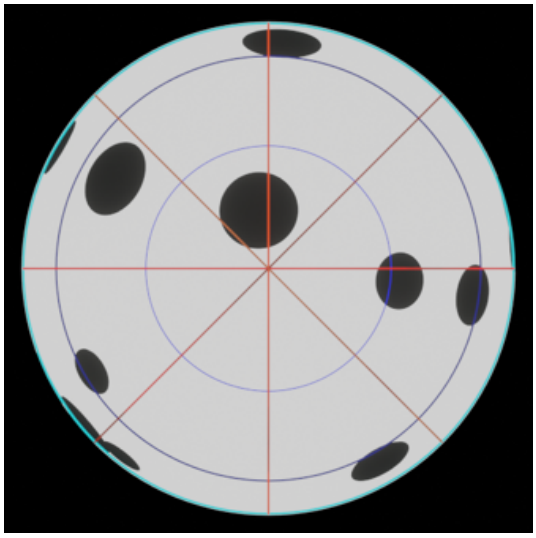


Figure 2 This is the sample image generated in blender.

There are 3 different scaled version of these images. Based on the image used, image related arguments, i.e. its centre radius, etc. will change

Name	Dimensions [height, width]	Centre [x,y] coordinates	Radius (pixels)	Thres hold
test_img_voronoi_image_low_res.png	[108,108]	[54,54]	51	1
test_img_voronoi_image_half_res.png	[540,540]	[270,270]	250	.1
test_img_voronoi_image_high_res.png	[1080,1080]	[540,540]	503	.05

Examples / usage

Using the functions

Functional part of the project lies on the script pol_img_functions.py . So whenever using the functions we need to import the necessary functions from it. To check and work non-destructively, it is better to start a new python file in the same path. (Or use the python terminal on a python IDE). Here we will try an example of correcting an image's elevation distortion and then project this image as an equirectangular projection. Elevation distortion happens when are taking a fisheye image, where the image squishes a bit as we go from the centre to the horizon of the fisheye image.

Correcting for elevation distortion

1. Start with the following python command to import all the necessary python libraries.
   ```
   from lib_importer import *
   ```
2. To import the necessary functions from the pol_img_functions , use the following command
   ```
   from pol_img_functions import azimuth_mapping, elevation_mapping , pixel_map_func
   ```
3. Import a fisheye image or a sample image that mimics it, here we have some test images in the folder test_images which will use here.

   ```
   blend_img_low = cv2.imread("test_images/test_img_voronoi_image_low_res.png")
   ```
4. to invert the color channels from BGR to RGB:
   ```
   blend_img_low = blend_img_low[:,:,[2,1,0]]
   ```

5. these functions require some arguments, like the zenith or the centre of the image in pixel coordinates, radius or the distance from centre to the horizon of the image in pixel distance. This can be found from using a image editing/processing software like fiji calculating the radius of circle that encompasses the horizon in pixel distance. And centre is the normally the centre most pixel coordinate for a fisheye image. Thresh is threshold value while mapping the pixels based on the azimuth and elevation mapping. The lower the value, stricter the mapping. Normally low res images (image with dimension 108 X 108) have higher threshold values like 1, and it scales down by factor of 20 , when the image scales by a factor of 10 (along the width and height). [very crude method, play around with the value to tune the mapping]

```
centre = [54, 54]
radius = 51
thresh = 1
```

6. To map the image, we need to find the azimuth map and the elevation map , so first we generate these maps.

```
azimuth_map = azimuth_mapping(blend_img_low, radius, centre)
elevation_map_src, elevation_map_corr = elevation_mapping(blend_img_low, radius,
centre)
```

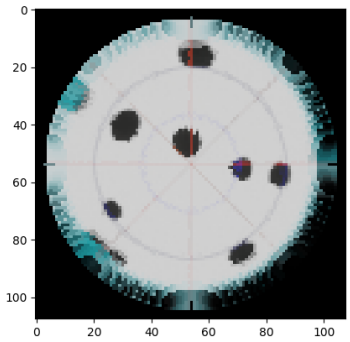
the functions return a matrix like array with azimuth and elevation values of each corresponding pixels.

7. Now we map the image, ie. Correct the elevation distortion that is inherent to a fisheye image.

```
mapped_img = pixel_map_func(blend_img_low, centre,
radius,elevation_map_src,elevation_map_corr,azimuth_map,thresh)
```

8. After that we can plot the image.

```
plt.imshow(mapped_img)
plt.show()
```



Equirectangular projection

1. Now that we have a corrected image which can use this an image to project it in an equirectangular manner. From north in an anticlockwise fashion

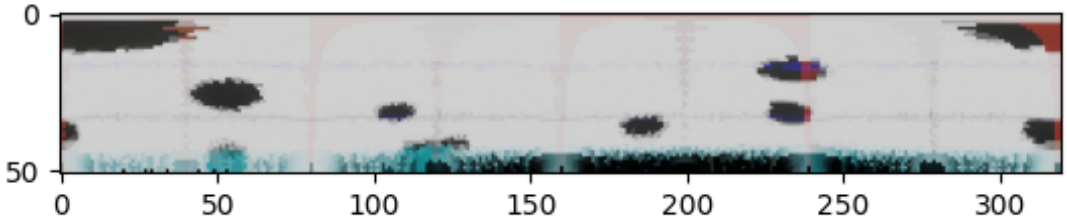
```
equi_plot_after = pol_2_equirect(mapped_img, radius, centre)
```

this returns a np.ndarray with circular boundary being the width and radius being the height

2. To visualize the projection simply follow the previous plotting commands.

```
plt.imshow(equi_plot_after)
```

```
plt.show()
```



as you can see the image dimensions have changed , going from top to down of this image is equivalent as to going from centre to periphery of the previous sample image. And going from left to right is like rotating anticlockwise from north of the sample image.

Editing or making changes

- to add new library or module, add the import command inside the lib_importer.py which act as a global library importer
- to edit or add any more functionality, add or make changes inside pol_img_func.py
- to make a new script own which uses any of the pol_img_functions, make a python file on the same path import the base libraries by lib_importer.py and specific functions from pol_img_functions.py
- to access a sample DRA coordinates data of the bee , use the bee_eye_subsampling.
- To plot any functions run plot_examples.py