DEVLOPMENT OF GLOBAL ALIGNMENT METHODOLOGY FOR DEPLOYABLE OPTICAL TELESCOPE SYSTEM USING IMAGE BASED SENSING

A Project Report ON

Rugnesh Vora
(20EL096)

Under the guidance of

Industry Guide: Shri NUMAN AHMAD
at
Space Applications Centre, (ISRO) Ahmedabad

Faculty Guide: Prof. MEHFUZA HOLIA

In fulfilment of the award of the
Degree Of
BACHELOR OF TECHNOLOGY
In
ELECTRONICS ENGINEERING

BIRLA VISHVAKARMA MAHAVIDYALAYA
ENGINEERING COLLEGE
[An Autonomous Institution]
ANAND – 388120
GUJARAT

Institute Vision:

"Produce globally employable innovative engineers with core values."

Institute Mission:

- Re-engineer curricula to meet global employment requirement
- Promote innovative practices at all levels.
- Imbibe core values
- Reform policies, systems and processes at all levels.
- Develop faculty and staff members to meet the challenges

Core Values:

Quality, Creativity, Team Work, Lifelong Learning, Pro-activeness,

B.Tech. Electronics program offered by the Department of Electronics Engineering

Program Vision:

"Produce globally employable innovative Electronics Engineers with core values"

Program Mission:

- Promote innovative practices to strengthen the teaching and learning process
- in Electronics engineering
- Develop faculty and staff members to meet challenges in Electronics
- Engineering
- Adapt engineering curricula to meet global requirements for the Electronics engineering program
- Reform policies, systems, and processes at all levels
- Imbibe core values.

Program Educational Objectives (PEOs):

1. Study and Analysis of Electronics Engineering Systems
2. Adapt state-of-the-art developments in Electronics Engineering and eco-friendly technologies
3. Design and Develop Electronics hardware and softw are-based applications

Birla Vishvakarma Mahavidyalaya

Engineering College

Department of Electronics Engineering

2023-24

CERTIFICATE

DATE:_____/_____/_____

This is to certify that the dissertation entitled **"Development of global alignment methodology for deployable optical telescope system using image base sensing"** has been carried out by **Rugnesh Tulshibhai Vora (20EL096)** under my guidance in fulfilment of the degree of Bachelor of Engineering in Electronics (8th Semester) of Birla Vishvakarma Mahavidyalaya Engineering College, Vallabh Vidyanagar during the academic year 2023-24.

Course Coordinator	Faculty Guide	Head of Department
Dr. Deepak Vala	Prof. Mehfuza Holia	Dr.Tanmay Pawar

INTERNSHIP CERTIFICATE

ACKNOWLEDGEMENT

I would like to take this opportunity to best of my acknowledgements to all the persons who have directly or indirectly been involved with me in making my research feasible and to run it up into a successful piece of work.

It is the product of many hands, and countless hours from many people. My thanks go to all those who helped, whether through their comments, feedback, edits or suggestions. I express a deep sense of gratitude to the **Head of the Electronics Engineering department, Dr. Tanmay. D. Pawar, Faculty guide Prof. Mehfuza Holia, and Course Coordinator Dr. D.L. Vala** for providing such a big opportunity to do internship at SAC, ISRO. Moreover, I would like to thank my **Project guide Shri Numan Ahmad,** Scientist/Engineer 'SE' and **Shri Naimesh Patel, Division Head, Optical Payload Mechanical Design Division, OPMG-MESA** and team of SAC who has helped me throughout my project work.

I would like to thank all the faculty members for their patience, understanding and guidance that gave me strength and will power to work for developing a project and preparing the report.

Last but not the least, I would also like to thank our friends and classmates, who have co-operated during the preparation of my project and report, without them this research has not been possible. Their ideas helped me a lot to improve my project work.

Rugnesh Vora

(20EL096)

About Organisation:

INDIAN SPACE RESEARCH ORGANISATION

Indian Space Research Organisation (ISRO) is the space agency of India. The organisation is involved in science, engineering and technology to harvest the benefits of outer space for India and the mankind. ISRO is a major constituent of the Department of Space (DOS), Government of India. The department executes the Indian Space Programme primarily through various Centres or units within ISRO.

The prime objective of ISRO/DOS is the development and application of space technology for various national needs. To fulfil this objective, ISRO has established major space systems for communication, television broadcasting and meteorological services; resources monitoring and management; space-based navigation services. ISRO has developed satellite launch vehicles, PSLV and GSLV, to place the satellites in the required orbits.

Alongside its technological advancement, ISRO contributes to science and science education in the country. Various dedicated research centres and autonomous institutions for remote sensing, astronomy and astrophysics, atmospheric sciences and space sciences in general function under the aegis of Department of Space. ISRO's own Lunar and interplanetary missions along with other scientific projects encourage and promote science education, apart from providing valuable data to the scientific community which in turn enriches science.

Table of Contents

List of Figures

Abstract

Deployable optical telescopes consist of segmented mirrors, which can be deployed after launching into space. After deployment of segmented mirrors, they can form a single primary mirror surface. This project aims at the “development of global alignment methodology based on image based sensing”. The deployable telescope considered in this study is built into Zemax optics design and simulation tool. This telescope consists of 6 segmented mirrors to form a primary mirror and one deployable secondary mirror. Each of these mirrors have provision of 6 degrees of freedom movement, which can be imparted to them in order to align the telescope.

Due to axisymmetric design of the system, only piston, tip and tilt motions of the mirrors are considered for global alignment simulation. In reality, global alignment caters to coarse level adjustment for the telescope, where major errors are corrected. Remaining residual error may further be corrected by different algorithms, which is not attempted in this work.

The performance metric for telescope alignment is the Point Spread Function of a simulated star at infinity. The PSF is obtained at the detector image plane in the software. Ideally, the goal is to find a single PSF spots of diffraction limited diameter by aligning all seven mirrors.

This project uses Python Zemax Dynamic Data Exchange library to interact with Zemax from python environment. This library has been used to correct the mirrors position in close loop feedback by reading the PSF image from Zemax. Finally, all the steps associated with global alignment are shown.

1. Project Introduction

1.1 Deployable Optical Space Telescope

Increased demands for deep space observation calls for high resolution optical observation in visible, infrared, and near infra-red spectrum. With need for high resolution optical observation, comes the need for large aperture space telescopes. The telescope size is usually limited due to limited space available in launch vehicles. In order to launch the larger aperture telescope, the launch vehicle capacity must be increased. But this comes with far greater challenges. Hence to deal with this problem it is a better alternative to design the large aperture telescopes in deployable configuration. Deployable telescope can be well accommodated in the launch fairing with stowed configuration; and can then be deployed to its full aperture in space after launch is completed. A deployable space telescope is a type of astronomical observatory that can be launched into space and unfolded or expanded to achieve its full operational size and capability [1]. Unlike traditional ground-based telescopes, deployable space telescopes are not limited by the distortion of the Earth's atmosphere, which can affect the quality of observations.

Deployable space telescopes, like the one in Figure 1.1, are made to be small and light for launch, but they may be extended or unfurled using mechanical, pneumatic, or inflatable mechanisms once they're in orbit. Unlike conventional space telescopes, which are constrained by the size of the rocket fairing, it is able to attain significantly bigger apertures and fields of view as a result.

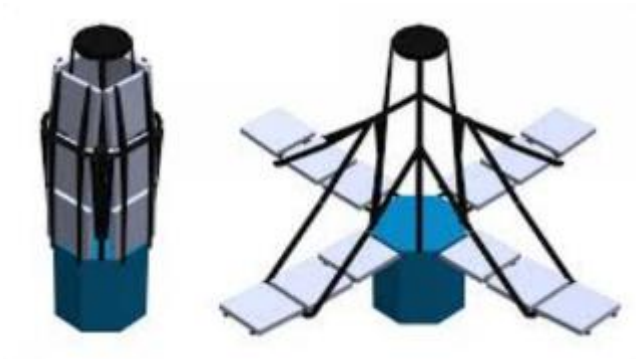


Figure 1.1 Deployable Optical Space Telescope

In order to observe a variety of astronomical phenomena, including stars, galaxies, and other celestial objects, deployable space telescopes are employed. They are also used to discover and characterize space and to investigate dark energy and dark matter. Over the past few decades, a number of deployable space telescopes have been launched, including the Chandra X-ray Observatory, the James Webb Space Telescope, and the Hubble Space Telescope. These telescopes have fundamentally changed how is perceived the universe.

This project deals with the alignment methodology of a deployable optical telescope system. This telescope consists of six segmented primary mirrors, which after perfect alignment can form a single surface mirror. A simplified representation of the telescope is shown below:

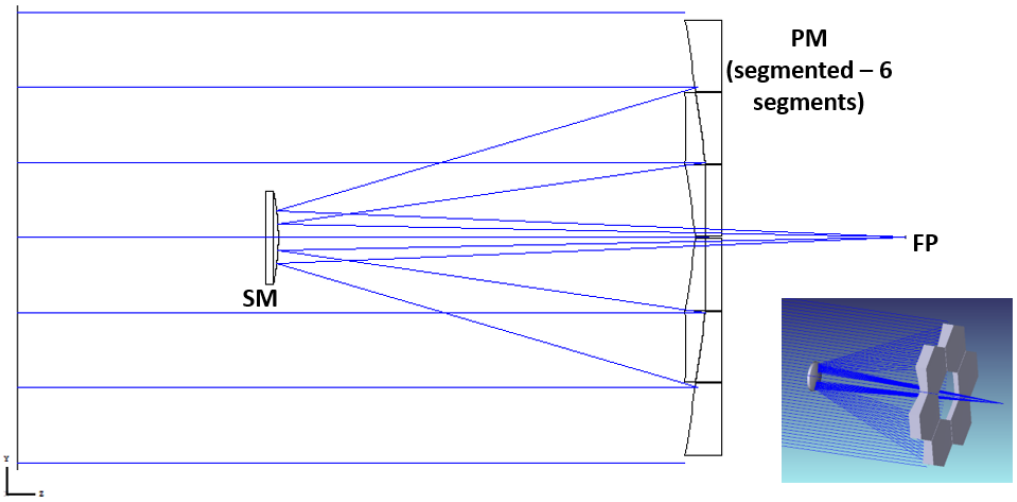


Figure 1.2Deployable Telescope Optical Layout

The primary mirror segment to secondary mirror separation is 650 mm, the image plane is located at 950 mm from secondary mirror. The effective focal length of the telescope is 5440 mm with F-number of 8. It is designed for wavelength 633 nm in visible range. All of the segmented mirrors and secondary mirror have six degrees of freedom system, which can provide motions to align the telescope.

With deployable optics comes the challenges associated with the alignment of telescope. In order to reach single mirror telescope kind of performance, we have to make sure that the secondary and segmented primary mirrors are well aligned, so that the segmented mirrors form a single surface, and the secondary mirror is placed at its perfect position.

In order to properly align the telescope, we are presenting a coarse level of alignment methodology, which can align the telescope with major corrections that can be provided to the mirrors. Since there are total 7 mirrors and each of them has 6 degrees of freedom, single shot alignment is not possible. Therefore, we have derived a methodology inspired from JWST alignment methodology [2].

This alignment method has been implemented on software level, to develop the alignment algorithm. We have used Zemax software for simulation of the deployable telescope and its associated optical elements.

Zemax software stands as a tool for optical design and analysis. The optical model of deployable optical telescope has been designed in Zemax software. The software simulates a star at infinite location, and the star's image forms on the image plane as Point Spread Function (PSF). This software allows us to provide independent motions to each of the mirror segments and secondary mirror elements. Further, based on the positions of mirror elements, PSF image of simulated star is obtained at the image plane. The PSF image then can be accessed by Python Zemax Dynamic Data Extension (PyZDDE) library, which gives us the provision to correct the mirror position with real time feedback.

In a way Zemax along-with PyZDDE enable us to simulate a virtual deployable optical telescope, where we can model real time default positions of the primary segmented mirrors and secondary mirror. The mirrors are randomly placed initially, to simulate the actual assembly condition that is possible in real hardware. Then, the PyZDDE library interacts with Zemax software and feedback corrections can be provided to the software.

1.2. Deployable Optical Telescope in Zemax Software

Zemax is a powerful optical design and simulation tool, which offers simulation of various optical performance parameters such as Wavefront Map, Point Spread Function (PSF), Modulation Transfer Function (MTF) etc. We have modeled this optical telescope in Zemax software. The image formed at the image plane is obtained from PSF of a simulated star at the entrance of the primary mirror. We have used PSF at image plane for our calculation during algorithm development, since it gives a representation of a distant star imaging for the telescope alignment.

The Zemax has Lens Data Editor interface, which models the fundamental elements of the optical system, e.g. primary mirror, secondary mirror and image plane etc. (representation view shown below [ref fig]).

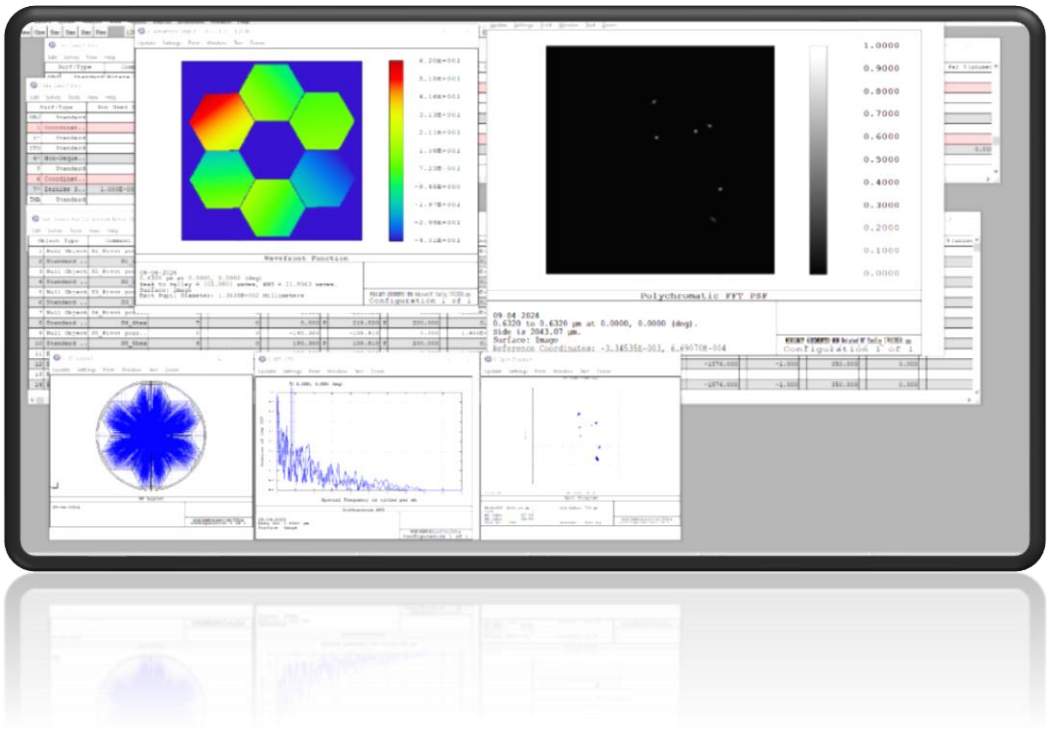


Figure 1.3Zemax Software View

Zemax also provides different visualization modes for the optical design, e.g. wavefront map, 3D Layout, spot diagram etc. some of which are shown in Figure. [1.4].

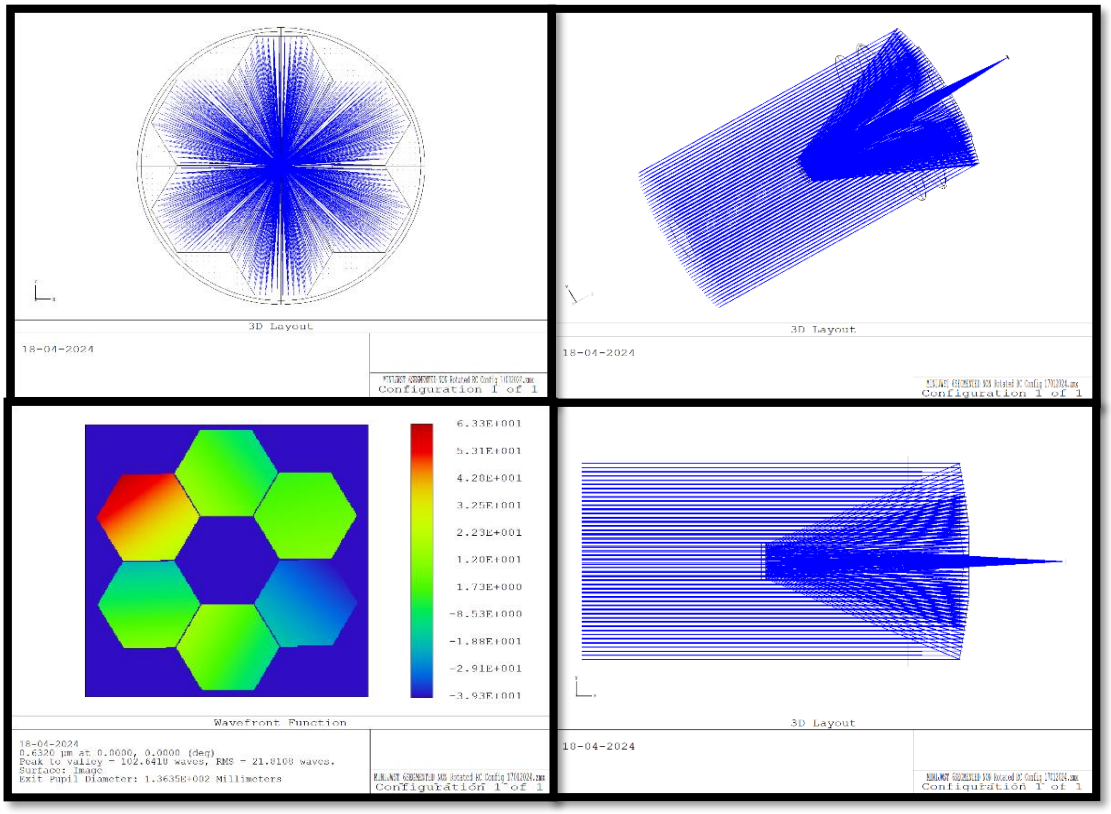


Figure 1.4 Telescope Views

- The telescope is made up of six primary mirrors and one secondary mirror, with each having six degrees of freedom to achieve the best possible view of objects. The modeling of six segments as a single primary mirror is done by modeling these as non-sequential components. These segments are then modeled using non-sequential data editor of Zemax software.

- Lens Data Editor:

Lens Data Editor

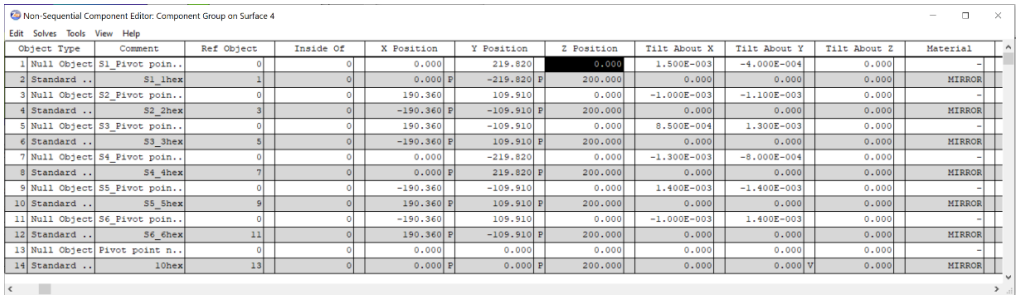
Edit Solves View Help

	Surf:Type	Comment	Radius	Thickness	Glass	Semi-Diameter		Conic	Extrapolate	2nd Or
OBJ	Standard	Rotate Telesc..	Infinity	Infinity		0.000		0.000		
1	Coordinat..	Tele Rotation		0.000	-	0.000				
2*	Standard		Infinity	1045.000		340.000		0.000		
STO	Standard	stop	Infinity	0.000		340.000		0.000		
4*	Non-Seque..	NSQ ENTRANCE ..	Infinity	-		350.000	U	0.000	3	
5	Standard	NSQ EXIT PORT	Infinity	-349.894		350.000	U	0.000		
6	Coordinat..	SM movement		0.000	V -	0.000				
7*	Zernike S..	SECONDARY MIR..	-323.000	953.434	MIRROR	56.992		-1.792	1	
IMA	Standard		Infinity	-		10.000	U	0.000		

Figure 1.5 Lens Data Editor Window

This window is a part of Zemax shows the current status of the secondary mirror. also shows thickness the Z-direction of the mirror.

- Non-Sequential Component Editor:



	Object Type	Comment	Ref Object	Inside Of	X Position		Y Position		Z Position	Tilt About X	Tilt About Y	Tilt About Z	Material
1	Null Object	S1_Pivot poin..	0	0	0.000		219.820		0.000	1.500E-003	-4.000E-004	0.000	-
2	Standard ..	S1_1hex	1	0	0.000	P	-219.820	P	200.000	0.000	0.000	0.000	MIRROR
3	Null Object	S2_Pivot poin..	0	0	190.360		109.910		0.000	-1.000E-003	-1.100E-003	0.000	-
4	Standard ..	S2_2hex	3	0	-190.360	P	-109.910	P	200.000	0.000	0.000	0.000	MIRROR
5	Null Object	S3_Pivot poin..	0	0	190.360		-109.910		0.000	8.500E-004	1.300E-003	0.000	-
6	Standard ..	S3_3hex	5	0	-190.360	P	109.910	P	200.000	0.000	0.000	0.000	MIRROR
7	Null Object	S4_Pivot poin..	0	0	0.000		-219.820		0.000	-1.300E-003	-8.000E-004	0.000	-
8	Standard ..	S4_4hex	7	0	0.000	P	219.820	P	200.000	0.000	0.000	0.000	MIRROR
9	Null Object	S5_Pivot poin..	0	0	-190.360		-109.910		0.000	1.400E-003	-1.400E-003	0.000	-
10	Standard ..	S5_5hex	9	0	190.360	P	109.910	P	200.000	0.000	0.000	0.000	MIRROR
11	Null Object	S6_Pivot poin..	0	0	-190.360		109.910		0.000	-1.000E-003	1.400E-003	0.000	-
12	Standard ..	S6_6hex	11	0	190.360	P	-109.910	P	200.000	0.000	0.000	0.000	MIRROR
13	Null Object	Pivot point n..	0	0	0.000		0.000		0.000	0.000	0.000	0.000	-
14	Standard ..	10hex	13	0	0.000	P	0.000	P	200.000	0.000	0.000	0.000	MIRROR

Figure 1.6 Non-Sequential Component Window

This window is also part of zemax, shows the current status of the 6 primary mirrors in x, y, z positions and shows which position the mirror is currently tilted.

1.2. PyCharm IDE

We have used python as programming language to interact with the Zemax software. Zemax itself has in built IDE to program macros, however, there is one open source library called PyZDDE which provides the provision to interact with Zemax from Python IDE. We have used PyCharm IDE for python Programming in this project. Alternatively, in higher versions of Zemax, there are inbuilt APIs to interact with MATLAB and Python.

1.3. Python Libraries:

The project relies on various Python libraries to implement image processing techniques and mathematical algorithms essential for alignment methodology development.

- **OpenCV:** OpenCV is an open-source library for computer vision and image processing. It provides a comprehensive set of functions and algorithms for image manipulation, feature detection, and object recognition. OpenCV is widely used in applications such as face detection, gesture recognition, and medical imaging. In this project, OpenCV is utilized for tasks such as image enhancement, spot detection.

- **NumPy:** NumPy is a fundamental library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions for array manipulation and linear algebra operations. NumPy is widely used in scientific computing, data analysis, and machine learning. In this project, NumPy is utilized for array manipulation, mathematical operations, and numerical analysis tasks.

- **Matplotlib:** Matplotlib is a plotting library for Python that provides a flexible and versatile platform for creating static, interactive, and animated visualizations. It offers a wide range of plotting functions and customization options for creating publication-quality figures and charts. Matplotlib is widely used in fields such as scientific research, data analysis, and engineering. In this project, Matplotlib is utilized for visualizing data, displaying images, and presenting analysis results.

- **Math:** The Math module in Python provides a set of mathematical functions for performing common mathematical operations. It includes functions for basic arithmetic, trigonometry, logarithms, exponentials, and more. The Math module is a standard part of the Python language and is widely used in mathematical computations and scientific programming. In this project, Used in the calculation for creating hexagons within the image, the Math module is utilized for performing mathematical calculations and operations required for alignment methodology development.

2. Python-Zemax Communication

2.1. Understanding Python Zemax Dynamic Data Exchange (PyZDDE):

PyZDDE is python library serves as the intermediary link between Python and Zemax, enabling users to control and interact with Zemax from within Python scripts. This module essentially acts as a wrapper around the Dynamic Data Exchange (DDE) protocol, which Zemax supports for external communication.

2.2. Setup

Before utilizing PyZDDE, it's essential to ensure that both Python and Zemax OpticStudio are Open on the system. Additionally, PyZDDE must be installed and configured to establish a connection with Zemax.

PyZDDE can be imported into Python scripts using the following import statement:

```
import pyzdde.zdde as pyz
```

2.3. Initializing the DDE Link

The first step in using PyZDDE is to initialize the DDE link between Python and Zemax. This is achieved using the zDDEInit() method, as demonstrated below:

```
ln = pyz.PyZDDE()
start = ln.zDDEInit()
if start != 0:
    raise Exception("Unable to initialize DDE link")
```

2.4. Interacting with Zemax

With the DDE link established, users can now interact with Zemax by invoking various methods provided by PyZDDE. These methods enable tasks such as setting and retrieving parameters of optical components, analysing optical performance, and generating reports.

For example, the following code snippet demonstrates setting and retrieving the position of a non-sequential component in Zemax.

```
ln.zSetNSCPosition(surfNum=4, objNum=1, code=4, data=)
ln.zGetNSCPosition(surfNum=4, objNum=1)
```

Similarly, the **zSetNSCObjectData()** and **zGetNSCObjectData()** methods allow manipulation and retrieval of object data in non-sequential components.

```
ln.zSetNSCObjectData(surfNum=4, objNum=i, code=1)
ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1)
```

users can access thickness information of optical surfaces using the **zSetThickness()** and **zGetThickness()** methods.

```
ln.zSetThickness(surfNum=6)
ln.zGetThickness(surfNum=6)
```

PyZDDE facilitates the computation of Point Spread Functions (PSF) using the **zGetPSF()** method, providing insights into the optical performance of the system.

```
ln.zGetPSF('fft')
```

3. Brief overview of the alignment methodology

The segmented primary mirrors and secondary mirror have 6 degrees of freedom movement available. Therefore, there are total 42 variables for which the alignment has be optimized. Due to symmetry in the optical design, we have omitted rotational symmetries, thereby reducing the total variable to 21. These variables are Piston (Z – out of plane motion of the mirror), Tip (Rotation about X-axis of the mirror), Tilt (Rotation about Y-axis of the mirror). Since optimizing for these many variables is not possible simultaneously, there is need for a careful and thorough methodology.

Our methodology is inspired from alignment methodology of James Webb Space Telescopes (JWST). However, we are optimizing only for piston, tip and tilt motions in iterative manner to find the sharpest PSF of the image. This corresponds to some coarse alignment of the telescope, which will not remove the alignment errors completely, but will provide a somewhat aligned mirrors; on which further different fine level adjustment can be made for final alignment. This methodology is expressed in form of a flow chart in figure below.

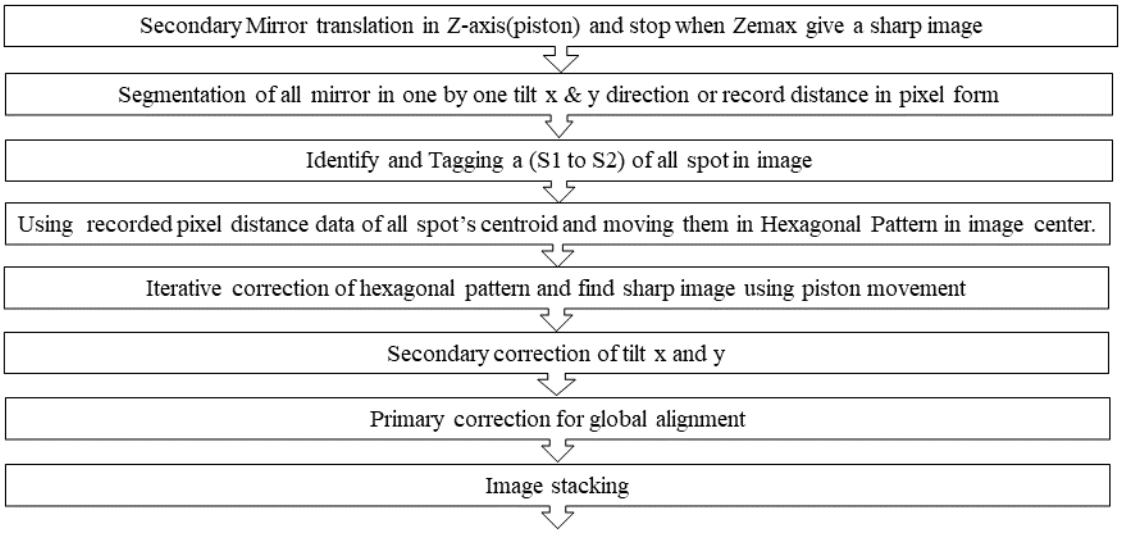


Figure 3.1 methodology Flowchart

Global alignment methodology first corrects the secondary mirror piston errors by finding the sharpest image for all segment spots on the image plane, thereby removing a large portion of secondary piston movement. Then all six segments are identified by providing small tip & tilt movements and correlating their movements on the image plane. Afterwards, in order to remove the large portion of tip/tilt errors from primary segments, hexagonal arrays are formed.

From hexagonal array formation, small movements in secondary mirror in terms of piston, tip and tilt are given, while maintaining the hexagonal array, further reduced the errors due to secondary mirror positioning. Further, all six segment spots are merged one-by-one onto image plane center such that all the spot centroids fall in subpixel area of the image plane center. This process is called image stacking.

4. Spot Analysis in images

3.1. Otsu's Method:

Otsu’s method is a popular technique employed for image thresholding so use this Method.

Here's a brief overview of step by step how the Otsu's method works.

Step 1: Compute the Grayscale Histogram

The grayscale histogram of an image shows the distribution of pixel intensity values in it.

Step 2: Compute the Cumulative Distribution Function

The Cumulative Distribution Function (CDF) represents the probability that a pixel in the image has a grayscale intensity value less than or equal to a particular level.

Step 3: Compute the Mean Grayscale Intensity Value of the Image

The mean grayscale intensity value of the image is the process of averaging the grayscale intensity values of all the pixels in it

Step 4: Compute the Between-Class Variance for Each Possible Threshold Value

The between-class variance measures the separation between the foreground and background regions.

Step 5: Find the Threshold Value That Maximizes the Between-Class Variance

To find the optimal threshold for image segmentation, we need to define the threshold value that maximizes the between-class variance.

3.2. Moment Analysis:

3.2.1. Zeroth-Order Moment:

area of the contour can be calculated using the zeroth order moment (**M["m00"]**). The zeroth order moment represents the total mass or area of the object. Mathematically, it's the sum of all the pixel intensities within the contour.

$$\mathbf{M_{00}} = \sum \mathbf{x} \sum \mathbf{y}\ \mathbf{I(x,y)}$$

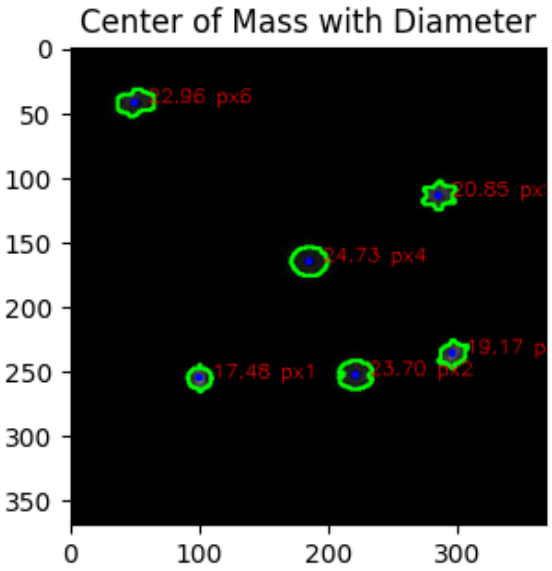


Figure 4.1Center of mass with diameter

3.2.2. First-Order Moment:

The center of mass (centroid) of the contour can be calculated using the first-order moments (**M["m10"]** and **M["m01"]**). The first-order moments represent the "center of gravity" of the object in the x and y directions respectively

$$\boldsymbol{M_{10}} = \sum \boldsymbol{x} \sum \boldsymbol{y}\ \boldsymbol{x}\ \boldsymbol{I(x,y)} \qquad \boldsymbol{cX} = \boldsymbol{M_{00}} / \boldsymbol{M_{10}}$$

$$\boldsymbol{M_{01}} = \sum \boldsymbol{x} \sum \boldsymbol{y}\ \boldsymbol{y}\ \boldsymbol{I(x,y)} \qquad \boldsymbol{cY} = \boldsymbol{M_{00}} / \boldsymbol{M_{01}}$$

3.2.3. Diameter:

The diameter of the contour can be estimated using the area of the contour (**M["m00"]**). Assuming the contour represents a circle (which might not be the case in reality), the diameter can be calculated from the area.

$$\boldsymbol{Diameter} = \sqrt{\frac{\mathbf{4\ x}\ \boldsymbol{Area}}{\boldsymbol{\pi}}}$$

5. Optimization for Sharpest Image

In this chapter introduce a how to find a sharp image, when secondary mirror position change via piston, here I show the how to find out sharp image by one experiment.

First of all, move piston initial position to 0.06 μm. This value is set lens data editor in zemax software by using python with the help of PyZDDE library. And see the result in step 1.

Step 1:

Move the secondary piston by 0.06 μm and analyse the centroids and diameters of spots in the image.

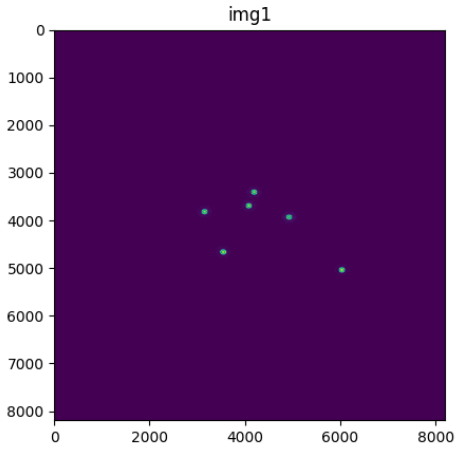


Figure 5.1: 1st Position of piston

Step 2:

Move the secondary piston by 0.03 μm and repeat the analysis.

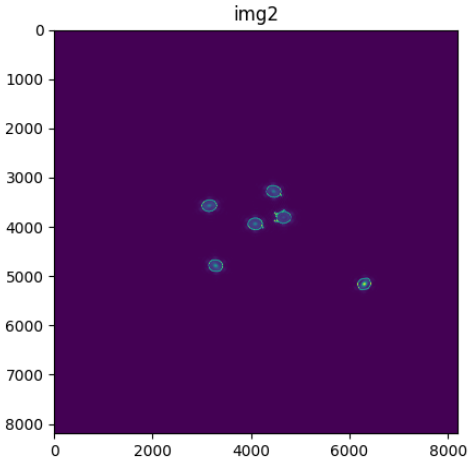


Figure 5.2: 2nd Position of piston

Step 3:

Move the secondary piston by 0.00 µm and perform analysis.

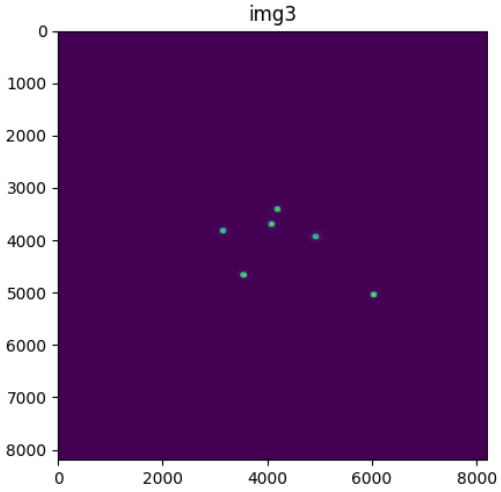


Figure 5.3: 3rd Position of piston

Step 4:

Move the secondary piston by -0.03 µm and analyse the image.

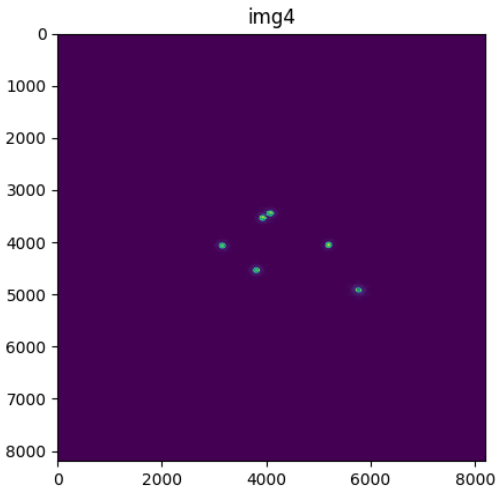


Figure 5.4: 4th Position of piston

Step 5:

Move the secondary piston by -0.06 μm and perform analysis.

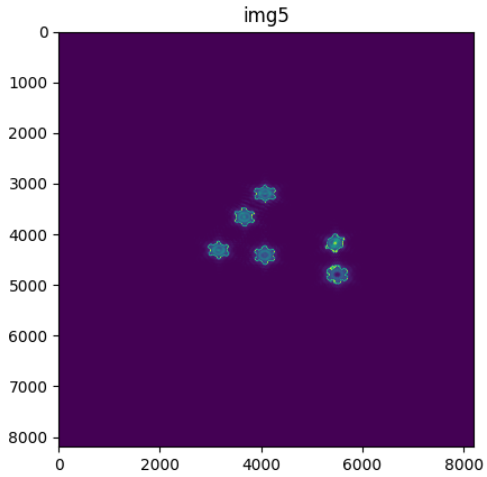


Figure 5.5: 5th Position of piston

Step 6:

Selection of the sharpest image: Calculate the variance of the diameter of spots in all five images. The image with the smallest variance is identified as the sharpest image.

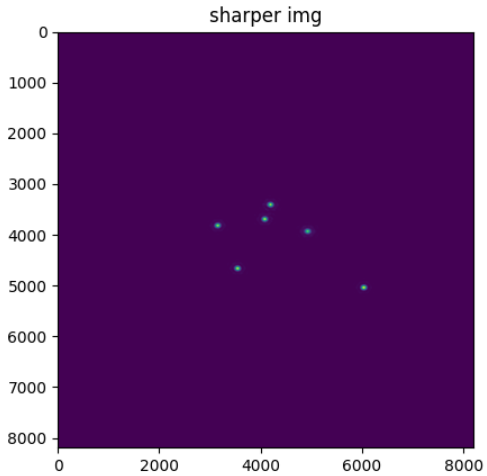


Figure 5.6: Selected sharp image

By systematically comparing the variance of spot diameters across the images obtained from different piston movements, the image with the least variation in spot size is deemed to be the sharpest. Once the sharpest image is found then the piston is moved to the position at which the sharpest image was formed.

6. Segment ID

Step1: Spot Segmentation:

- Identify and segment the spots within the image corresponding to the primary mirror segments (s1 to s6).
- Due to potential ambiguity in spot-mirror correspondence, develop a systematic methodology to accurately attribute each spot to its respective mirror segment.

Step 2: Mirror Tilt Measurement Procedure:

- Begin with the first mirror segment, s1.
- Introduce a small tilt in the x-direction while maintaining the other segments stationary.
- Observe the spot that moves within the image due to this tilt adjustment and designate it as s1.
- Utilize centroid analysis to quantify the displacement of s1 in terms of pixels in both the x and y directions.

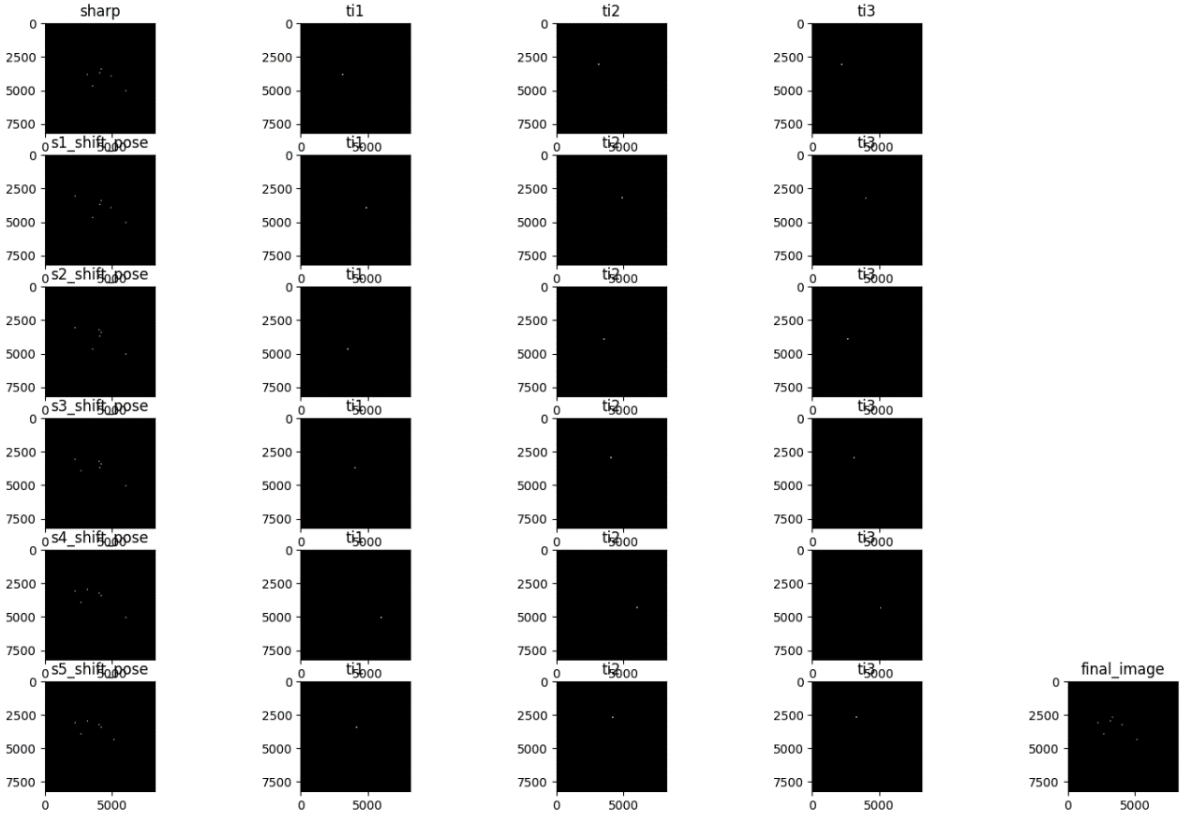
Step 3: Iterative Tilt Analysis:

- Repeat the tilt adjustment process for each mirror segment, one at a time.
- For each segment (s2 to s6), introduce tilts in both the x and y directions separately.
- Observe the resulting movements of the corresponding spots and record the pixel distances traversed in each direction.

Step 4: Spot Tagging:

- Assign unique identifiers to the spots corresponding to each mirror segment (S1 to S6) for clarity.

Result:


sharp
ti1
ti2
ti3
s1_shift_pose
s2_shift_pose
s3_shift_pose
s4_shift_pose
s5_shift_pose
final_image
0
2500
5000
7500
0
5000

7. Hexagonal Pattern Adjustment for Spot Alignment

In this chapter, we focus on aligning the coordinates of the six spots with a hexagonal pattern. Here's what we'll do:

Step 1: Imagining the Hexagon:

Imagine a six-sided shape (like a hexagon) right in the middle of our image. We want each spot to sit perfectly on of the six corners of this imaginary hexagon.

Ideal Hexagon construction and find a 6-corner coordinates

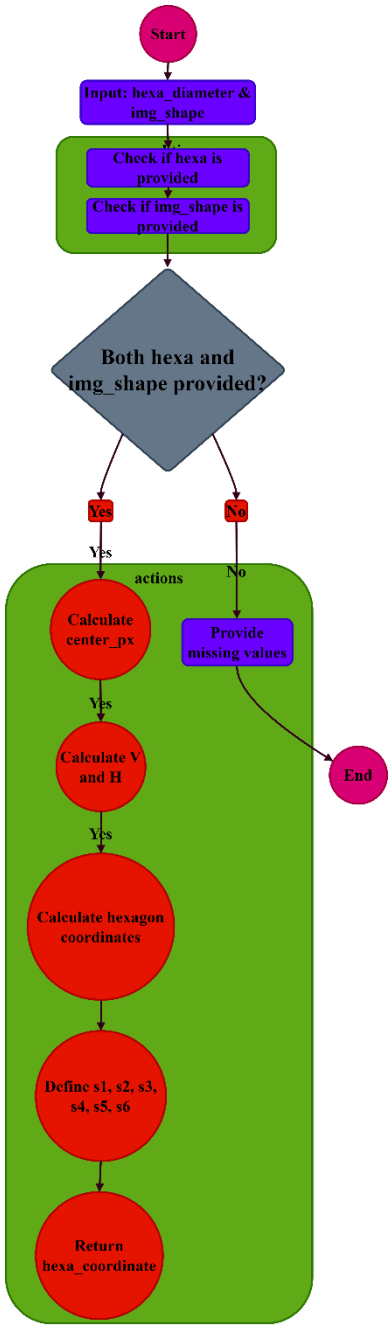


Figure 7.1 Ideal Hexagon Construction Flowchart

Step 2: Finding the Right Tilt:

We'll look at how much pixels do the spots move when we tilt the mirrors (which is already found in chapter 5). Then, we'll figure out how much more we need to tilt them to move the spots to the desired 6-coordinates of the imaginary hexagon one by one.

Step 3: Aligning Spot Coordinates:

By adjusting the tilts of the mirrors, we'll make sure each spot lines up perfectly with one of the corner coordinates of our imaginary hexagon. This way, all the spots will come together to form a neat hexagon in our image.

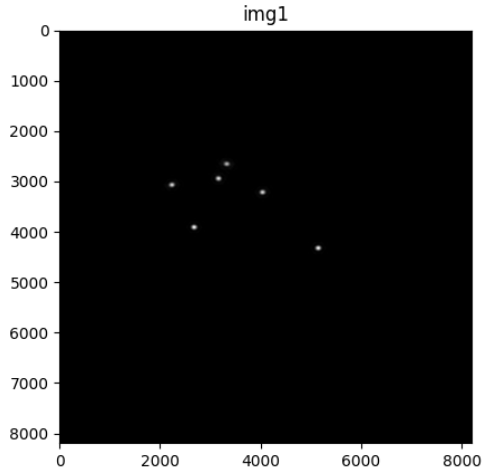


Figure 7.2: Segmented Image before hexagonal alignment

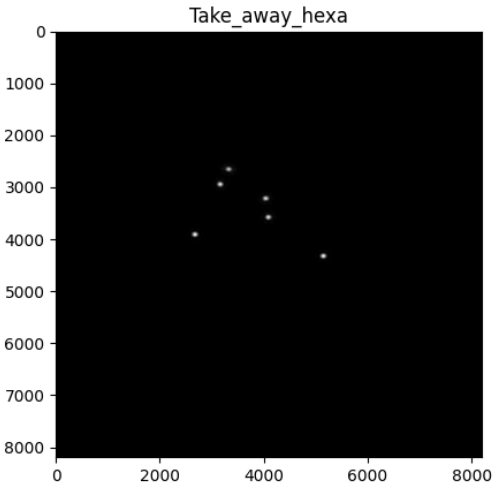


Figure 7.3: Alignment of s1 spot to form hexagonal array

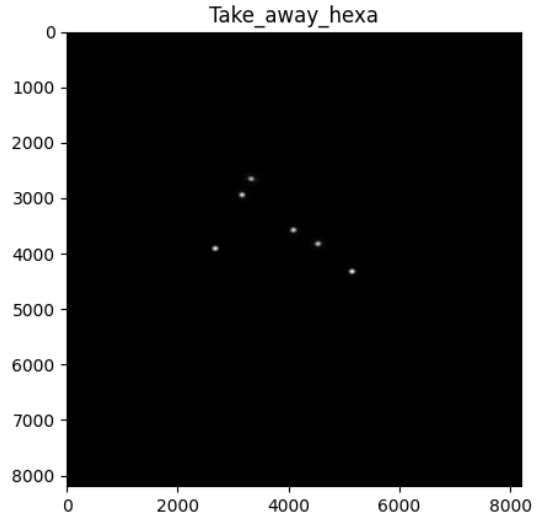


Figure 7.4: Alignment of s2 spot to form hexagonal array

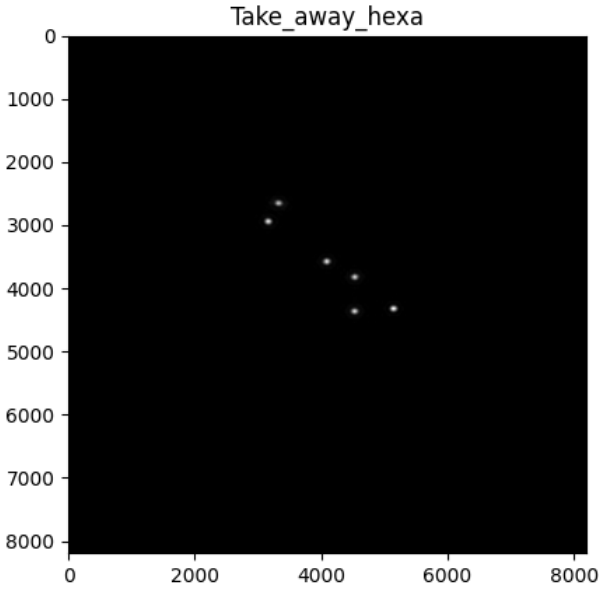


Figure 7.5: Alignment of s3 spot to form hexagonal array

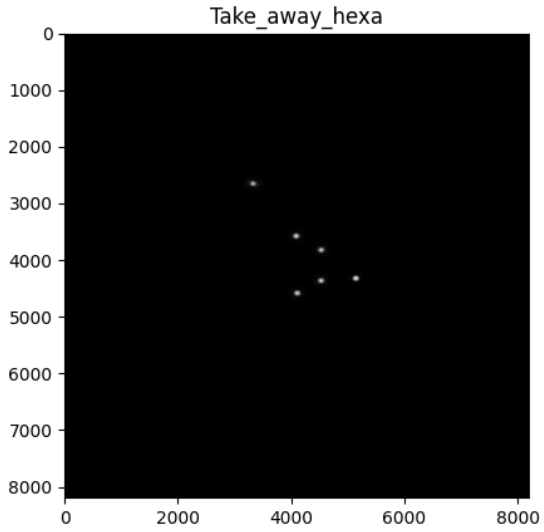


Figure 7.6: Alignment of s4 spot to form hexagonal array

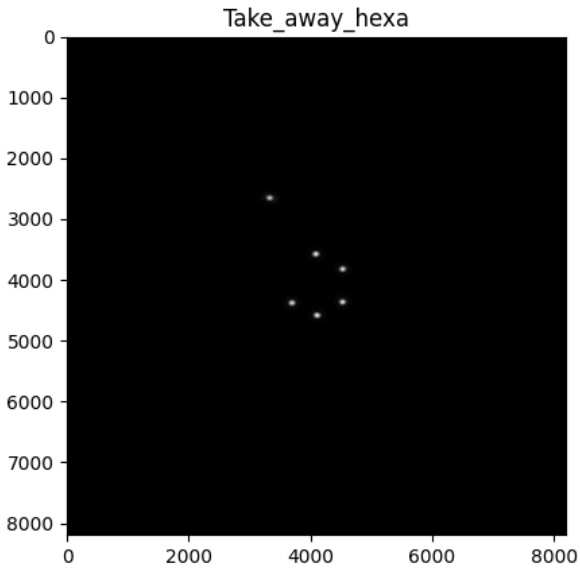


Figure 7.7: Alignment of s5 spot to form hexagonal array

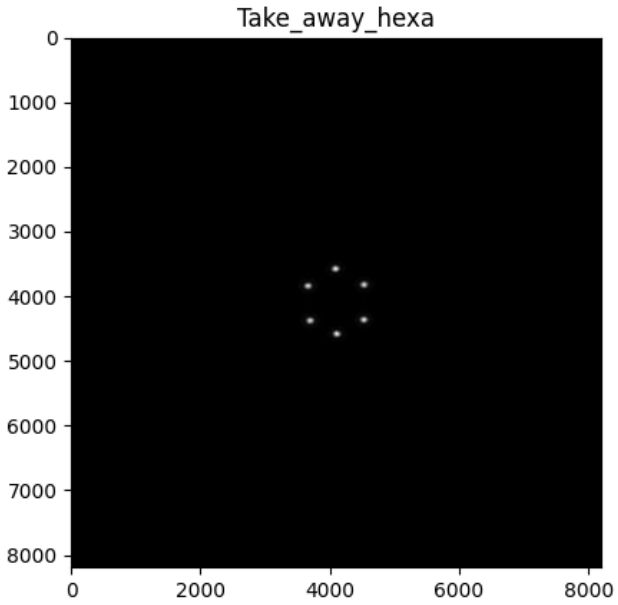


Figure 7.8: Alignment of s6 spot to form hexagonal array

8. Image Stacking(Centroids)

In this chapter, we build upon the x and y tilt measurements conducted in Chapter 5 to bring all spots to the center of the image.

1. **Analysis of Tilt Measurements:**
    - Review the x and y tilt measurements obtained in Chapter 5 for each mirror.
    - Understand how these measurements relate to the displacement of spots from their current positions to the center of the image.
2. **Calculating Tilt Adjustments**:
    - Determine the necessary tilt adjustments for each mirror to move their respective spots towards the center of the image.
    - Use a proportional relationship between tilt measurements and spot displacement to calculate the required adjustments.
3. **Applying Mirror Adjustments:**
    - Implement the calculated tilt adjustments for each mirror using the mirror control system.
    - Ensure precise control and coordination to achieve the desired movement of spots towards the image center.

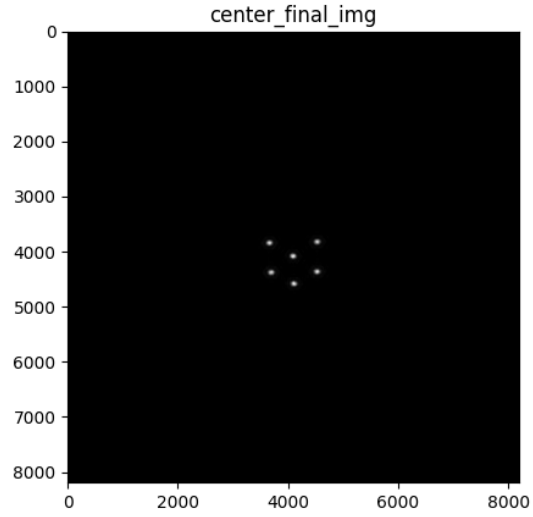


Figure 8.1 S1 Spot take away in center

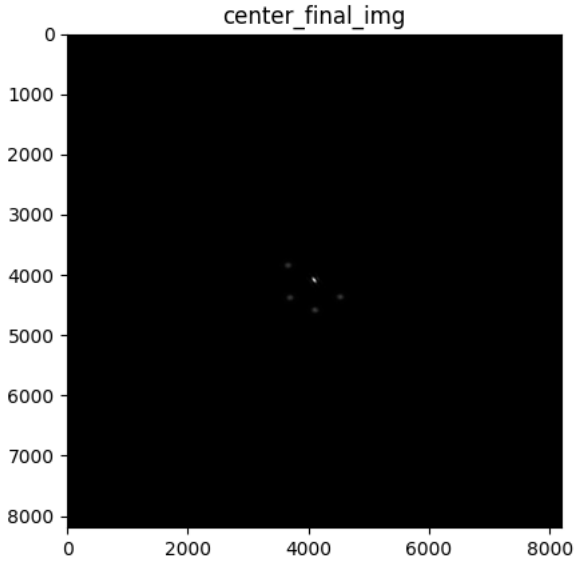


Figure 8.2 S2 Spot take away in center

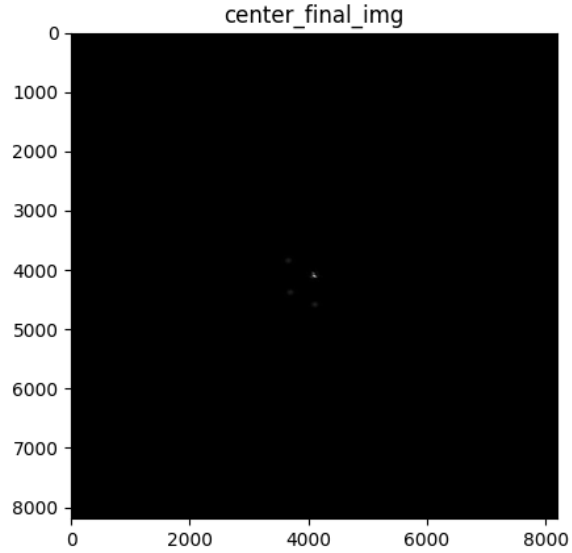


Figure 8.3 S3 Spot take away in center

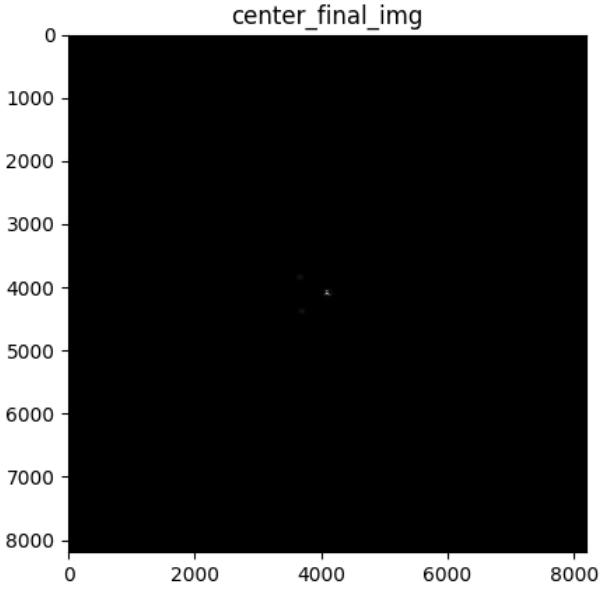


Figure 8.4 S4 Spot take away in center

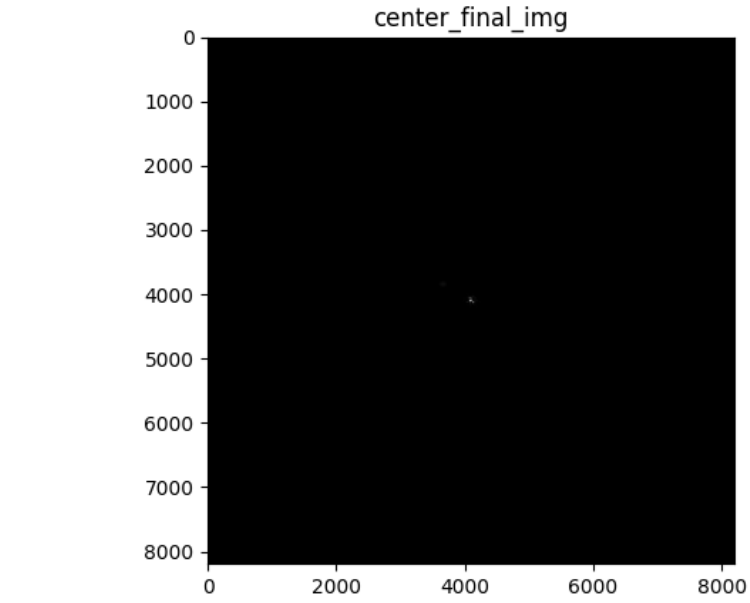


Figure 8.5 S5 Spot take away in center

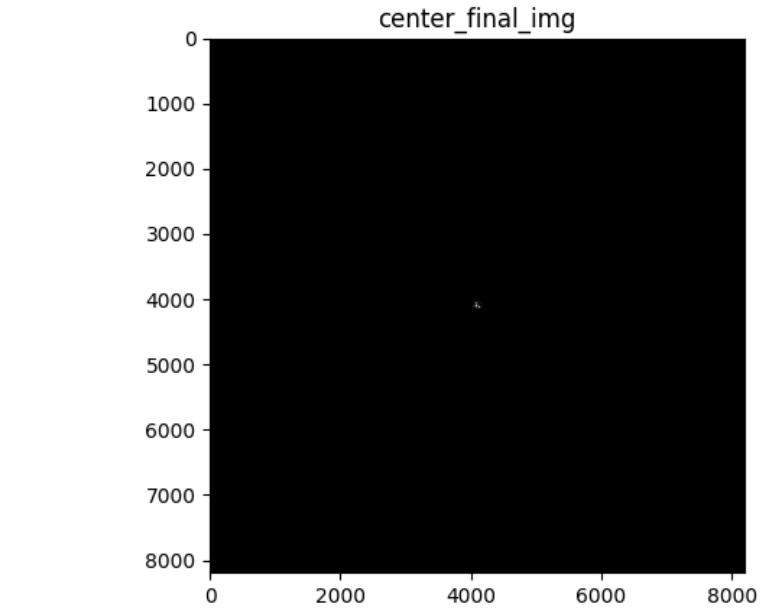


Figure 8.6 S6 Spot take away in center

9. Conclusion and Future Scope

This project consist of a alignment system which was able to globally align the primary and secondary mirror to study specific celestial objects.

The system can perform following process:

- Focus sweep: Find the best position of secondary mirror at which sharpest image of six spots from each primary mirrors can be achieved.
- Segmented ID: Identifying and tagging the spots parent primary mirror.
- Global Alignment: Forming hexagonal alignment of the six spots of primary mirrors.
- Image Stacking Centroids: Aligning primary mirrors to form a single spot at center from the six spots.

Annexure

Program:

```
#-------------------------------------------------------------------------------
# Name:        alignment.py
# Purpose:     Global alignment for deployable telescope and Zemax Communication
#
# Author of original Code: Rugnesh Vora, Apr 2024
# Copyright:   (c) SAC ISRO
#-------------------------------------------------------------------------------

import cv2
import numpy as np
from matplotlib import pyplot as plt
import pyzdde.zdde as pyz
import math

def normalize(data1):
    """This function use a normalize image,  all pixel value set between 0 to 255"""
    Not_N_img = np.array(data1, dtype=float)
    # down = cv2.pyrDown(Not_N_img)
    N_img = cv2.normalize(Not_N_img, None, alpha=0, beta=255, norm_type=cv2.NORM_MINMAX,
dtype=cv2.CV_8U)
    print("Image Shape:", N_img.shape)
    return N_img

def otsus(image):
    """This function make a threshold image and find a spot and return contours or thresholded
image"""
    t, thresholded_image = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    # thresholed_val.append(t)
    contours, t = cv2.findContours(thresholded_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    return contours, thresholded_image

def find_coordinate2(SUBTRACT):
    """This is a function of find a coordinate for given image"""
    t1, ti = cv2.threshold(SUBTRACT, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    print("t1", t1)
    contours, p = cv2.findContours(ti, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    # cv2.drawContours(drow_spot, contours, -1, 255, 15)
    for contour in contours:
        M = cv2.moments(contour)
        if M["m00"] > 0:
            cX = int(M["m10"] / M["m00"])
            cY = int(M["m01"] / M["m00"])
            cX_cY = [cX, cY]

    return cX_cY

def sharpImage():
    """This function is perform five experiment and find sopt, centroide, mean, varianc and this
data using a return sharp image """
    Mean = []
    Var = []
    len_dia = []

    d = ln.zGetThickness(surfNum=6)
    c = 0.03
    Thicknesh_list = [d + 2 * c, d + 1 * c, d, d - 1 * c, d - 2 * c]
    print("Thickness-> ", Thicknesh_list)

    def findDiameter(contourss, im):
        cv2.drawContours(im, contourss, -1, 255, 10)
        dia_list = []
        for contour in contourss:
            M = cv2.moments(contour)
            if M["m00"] > 0:
                # cX = int(M["m10"] / M["m00"])
                # cY = int(M["m01"] / M["m00"])
                diameter = np.sqrt(4 * M["m00"] / np.pi)
                dia_list.append(diameter)
        print("diameters of spots are: ", len(dia_list), dia_list)

        m = np.mean(dia_list)
        Mean.append(m)
        v = np.var(dia_list)
        Var.append(v)
        len_dia.append(len(dia_list))

    # 1st Communiocation
    Thickness1 = Thicknesh_list[0]
    print("Thickness-> ", Thickness1)
```

```
img1 = ln.zGetPSF('fft')  # Get FFT
n1 = normalize(img1[1])
contours, o1 = otsus(n1)  # Make binary image Find Spot for image
findDiameter(contours, n1)  # Find Diameter

plt.figure()
plt.imshow(n1, cmap="viridis")
plt.title("img1")
plt.show(block=False)

Thickness2 = Thicknesh_list[1]
ln.zSetThickness(surfNum=6, value=Thickness2)
print("Thickness-> ", Thickness2)
img2 = ln.zGetPSF('fft')
n2 = normalize(img2[1])
contours, o2 = otsus(n2)
findDiameter(contours, n2)

plt.figure()
plt.imshow(n2, cmap="viridis")
plt.title("img2")
plt.show(block=False)

Thickness3 = Thicknesh_list[2]
ln.zSetThickness(surfNum=6, value=Thickness3)
print("Thickness-> ", Thickness3)
img3 = ln.zGetPSF('fft')
n3 = normalize(img3[1])
contours, o3 = otsus(n3)
findDiameter(contours, n3)

plt.figure()
plt.imshow(n3, cmap="viridis")
plt.title("img3")
plt.show(block=False)

Thickness4 = Thicknesh_list[3]
ln.zSetThickness(surfNum=6, value=Thickness4)
print("Thickness-> ", Thickness4)
img4 = ln.zGetPSF('fft')
n4 = normalize(img4[1])
contours, o4 = otsus(n4)
findDiameter(contours, n4)

plt.figure()
plt.imshow(n4, cmap="viridis")
plt.title("img4")
plt.show(block=False)

Thickness5 = Thicknesh_list[4]
ln.zSetThickness(surfNum=6, value=Thickness5)
print("Thickness-> ", Thickness5)
img5 = ln.zGetPSF('fft')
n5 = normalize(img5[1])
contours, o5 = otsus(n5)
findDiameter(contours, n5)

plt.figure()
plt.imshow(n5, cmap="viridis")
plt.title("img5")
plt.show(block=False)

l1 = [img1[1], img2[1], img3[1], img4[1], img5[1]]

print("Mean--> ", len(Mean), Mean)
print("Var---> ", len(Var), Var)
print("l1 len-> ", len(l1))
print("Befor-len_dia-> ", len(len_dia), len_dia)
print("befor_Thickness-> ", len(Thicknesh_list))

f_len_dia = []
f_Var = []
f_l1 = []
f_thicknes = []
for index, item in enumerate(len_dia):
    if item >= 6:
        f_len_dia.append(item)
        f_Var.append(Var[index])
        f_l1.append(l1[index])
        f_thicknes.append(Thicknesh_list[index])

len_dia = f_len_dia
Var = f_Var
l1 = f_l1
Thicknesh_list = f_thicknes
print("After-Mean--> ", len(Mean), Mean)
print("After-Var---> ", len(Var), Var)
print("After-l1 len-> ", len(l1))
```

```
        print("After-len_dia-> ", len(len_dia), len_dia)
        print("After_Thickness-> ", len(Thicknesh_list))

        for index, item in enumerate(Var):
            if item == min(Var):
                print(f"img {index} is sharper")
                print("selected_Thicknes-> ", Thicknesh_list[index])
                ln.zSetThickness(surfNum=6, value=Thicknesh_list[index])
                sharp_img = l1[index]
                # plt.subplot(3, 5, 6 + index)
                plt.figure()
                plt.imshow(sharp_img, cmap="viridis")  # viridis
                plt.title("for loop sharper image")
                plt.show(block=False)

        return sharp_img

    def main(image):
        """This function perform a segmentation of sharp image and tagging of each spot in image"""
        def find_coordinate1(SUBTRACT):
            t1, ti = cv2.threshold(SUBTRACT, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
            print("t1", t1)
            contours, p = cv2.findContours(ti, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
            # cv2.drawContours(drow_spot, contours, -1, 255, 15)
            for contour in contours:
                M = cv2.moments(contour)
                if M["m00"] > 0:
                    cX = int(M["m10"] / M["m00"])
                    cY = int(M["m01"] / M["m00"])
                    cX_cY = [cX, cY]

            return cX_cY, ti

        for i in range(1, 12, 2):
            print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
    ln.zGetNSCPosition(surfNum=4, objNum=i))

        tilt = 0.001
        # segment 1
        print("s1 X tilt start")
        ln.zSetNSCPosition(surfNum=4, objNum=1, code=4, data=list_x[0] + tilt)
        for i in range(1, 12, 2):
            print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
    ln.zGetNSCPosition(surfNum=4, objNum=i))
        read1 = ln.zGetPSF("fft")

        print("s1 Y tilt start")
        ln.zSetNSCPosition(surfNum=4, objNum=1, code=5, data=list_y[0] + tilt)
        for i in range(1, 12, 2):
            print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
    ln.zGetNSCPosition(surfNum=4, objNum=i))
        read2 = ln.zGetPSF("fft")

        s1_shift_pose1 = normalize(read1[1])
        s1_shift_pose2 = normalize(read2[1])

        s1_sub1 = cv2.subtract(image, s1_shift_pose1)
        s1_sub2 = cv2.subtract(s1_shift_pose1, image)
        s1_sub3 = cv2.subtract(s1_shift_pose2, s1_shift_pose1)


        s1_pos1_coordinate, ti1 = find_coordinate1(s1_sub1)
        s1_pos2_coordinate, ti2 = find_coordinate1(s1_sub2)
        s1_pos3_coordinate, ti3 = find_coordinate1(s1_sub3)
        print("s1_pos1_coordinate, s1_pos2_coordinate, s1_pos3_coordinate", s1_pos1_coordinate,
    s1_pos2_coordinate, s1_pos3_coordinate)

        plt.figure()
        plt.subplot(6, 6, 2)
        plt.imshow(image, cmap="gray")
        plt.title("sharp")
        plt.subplot(6, 6, 3)
        plt.imshow(ti1, cmap="gray")
        plt.title("ti1")
        plt.subplot(6, 6, 4)
        plt.imshow(ti2, cmap="gray")
        plt.title("ti2")
        plt.subplot(6, 6, 5)
        plt.imshow(ti3, cmap="gray")
        plt.title("ti3")

        # # segment 2
        print("s2 X tilt start")
        ln.zSetNSCPosition(surfNum=4, objNum=3, code=4, data=list_x[1] + tilt)
        for i in range(1, 12, 2):
            print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
    ln.zGetNSCPosition(surfNum=4, objNum=i))
        read1 = ln.zGetPSF("fft")
```

```
    print("s2 Y tilt start")
    ln.zSetNSCPosition(surfNum=4, objNum=3, code=5, data=list_y[1] + tilt)
    for i in range(1, 12, 2):
        print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
ln.zGetNSCPosition(surfNum=4, objNum=i))
    read2 = ln.zGetPSF("fft")

    s2_shift_pose1 = normalize(read1[1])
    s2_shift_pose2 = normalize(read2[1])

    s2_sub1 = cv2.subtract(s1_shift_pose2, s2_shift_pose1)
    s2_sub2 = cv2.subtract(s2_shift_pose1, s1_shift_pose2)
    s2_sub3 = cv2.subtract(s2_shift_pose2, s2_shift_pose1)

    s2_pos1_coordinate, ti1 = find_coordinate1(s2_sub1)
    s2_pos2_coordinate, ti2 = find_coordinate1(s2_sub2)
    s2_pos3_coordinate, ti3 = find_coordinate1(s2_sub3)
    print("s2_pos1_coordinate, s2_pos2_coordinate, s2_pos3_coordinate", s2_pos1_coordinate,
s2_pos2_coordinate, s2_pos3_coordinate)

    plt.subplot(6, 6, 8)
    plt.imshow(s1_shift_pose2, cmap="gray")
    plt.title("s1_shift_pose")
    plt.subplot(6, 6, 9)
    plt.imshow(ti1, cmap="gray")
    plt.title("ti1")
    plt.subplot(6, 6, 10)
    plt.imshow(ti2, cmap="gray")
    plt.title("ti2")
    plt.subplot(6, 6, 11)
    plt.imshow(ti3, cmap="gray")
    plt.title("ti3")

    # segment 3
    print("s3 X tilt start")
    ln.zSetNSCPosition(surfNum=4, objNum=5, code=4, data=list_x[2] + tilt)
    for i in range(1, 12, 2):
        print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
ln.zGetNSCPosition(surfNum=4, objNum=i))
    read1 = ln.zGetPSF("fft")

    print("s3 Y tilt start")
    ln.zSetNSCPosition(surfNum=4, objNum=5, code=5, data=list_y[2] + tilt)
    for i in range(1, 12, 2):
        print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
ln.zGetNSCPosition(surfNum=4, objNum=i))
    read2 = ln.zGetPSF("fft")

    s3_shift_pose1 = normalize(read1[1])
    s3_shift_pose2 = normalize(read2[1])

    s3_sub1 = cv2.subtract(s2_shift_pose2, s3_shift_pose1)
    s3_sub2 = cv2.subtract(s3_shift_pose1, s2_shift_pose2)
    s3_sub3 = cv2.subtract(s3_shift_pose2, s3_shift_pose1)

    s3_pos1_coordinate, ti1 = find_coordinate1(s3_sub1)
    s3_pos2_coordinate, ti2 = find_coordinate1(s3_sub2)
    s3_pos3_coordinate, ti3 = find_coordinate1(s3_sub3)
    print("s3_pos1_coordinate, s3_pos2_coordinate, s3_pos3_coordinate", s3_pos1_coordinate,
s3_pos2_coordinate, s3_pos3_coordinate)

    plt.subplot(6, 6, 14)
    plt.imshow(s2_shift_pose2, cmap="gray")
    plt.title("s2_shift_pose")
    plt.subplot(6, 6, 15)
    plt.imshow(ti1, cmap="gray")
    plt.title("ti1")
    plt.subplot(6, 6, 16)
    plt.imshow(ti2, cmap="gray")
    plt.title("ti2")
    plt.subplot(6, 6, 17)
    plt.imshow(ti3, cmap="gray")
    plt.title("ti3")

    # segment 4
    print("s4 X tilt start")
    ln.zSetNSCPosition(surfNum=4, objNum=7, code=4, data=list_x[3] + tilt)
    for i in range(1, 12, 2):
        print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
ln.zGetNSCPosition(surfNum=4, objNum=i))
    read1 = ln.zGetPSF("fft")

    print("s4 Y tilt start")
    ln.zSetNSCPosition(surfNum=4, objNum=7, code=5, data=list_y[3] + tilt)
    for i in range(1, 12, 2):
        print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
ln.zGetNSCPosition(surfNum=4, objNum=i))
```

```
    read2 = ln.zGetPSF("fft")

    s4_shift_pose1 = normalize(read1[1])
    s4_shift_pose2 = normalize(read2[1])

    s4_sub1 = cv2.subtract(s3_shift_pose2, s4_shift_pose1)
    s4_sub2 = cv2.subtract(s4_shift_pose1, s3_shift_pose2)
    s4_sub3 = cv2.subtract(s4_shift_pose2, s4_shift_pose1)

    s4_pos1_coordinate, ti1 = find_coordinate1(s4_sub1)
    s4_pos2_coordinate, ti2 = find_coordinate1(s4_sub2)
    s4_pos3_coordinate, ti3 = find_coordinate1(s4_sub3)
    print("s4_pos1_coordinate, s4_pos2_coordinate, s4_pos3_coordinate", s4_pos1_coordinate,
s4_pos2_coordinate, s4_pos3_coordinate)

    plt.subplot(6, 6, 20)
    plt.imshow(s3_shift_pose2, cmap="gray")
    plt.title("s3_shift_pose")
    plt.subplot(6, 6, 21)
    plt.imshow(ti1, cmap="gray")
    plt.title("ti1")
    plt.subplot(6, 6, 22)
    plt.imshow(ti2, cmap="gray")
    plt.title("ti2")
    plt.subplot(6, 6, 23)
    plt.imshow(ti3, cmap="gray")
    plt.title("ti3")

    # segment 5
    print("s5 X tilt start")
    ln.zSetNSCPosition(surfNum=4, objNum=9, code=4, data=list_x[4] + tilt)
    for i in range(1, 12, 2):
        print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
ln.zGetNSCPosition(surfNum=4, objNum=i))
    read1 = ln.zGetPSF("fft")

    print("s5 Y tilt start")
    ln.zSetNSCPosition(surfNum=4, objNum=9, code=5, data=list_y[4] + tilt)
    for i in range(1, 12, 2):
        print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
ln.zGetNSCPosition(surfNum=4, objNum=i))
    read2 = ln.zGetPSF("fft")

    s5_shift_pose1 = normalize(read1[1])
    s5_shift_pose2 = normalize(read2[1])

    s5_sub1 = cv2.subtract(s4_shift_pose2, s5_shift_pose1)
    s5_sub2 = cv2.subtract(s5_shift_pose1, s4_shift_pose2)
    s5_sub3 = cv2.subtract(s5_shift_pose2, s5_shift_pose1)

    s5_pos1_coordinate, ti1 = find_coordinate1(s5_sub1)
    s5_pos2_coordinate, ti2 = find_coordinate1(s5_sub2)
    s5_pos3_coordinate, ti3 = find_coordinate1(s5_sub3)
    print("s5_pos1_coordinate, s5_pos2_coordinate, s5_pos3_coordinate", s5_pos1_coordinate,
s5_pos2_coordinate, s5_pos3_coordinate)

    plt.subplot(6, 6, 26)
    plt.imshow(s4_shift_pose2, cmap="gray")
    plt.title("s4_shift_pose")
    plt.subplot(6, 6, 27)
    plt.imshow(ti1, cmap="gray")
    plt.title("ti1")
    plt.subplot(6, 6, 28)
    plt.imshow(ti2, cmap="gray")
    plt.title("ti2")
    plt.subplot(6, 6, 29)
    plt.imshow(ti3, cmap="gray")
    plt.title("ti3")

    # segment 6
    print("s6 X tilt start")
    ln.zSetNSCPosition(surfNum=4, objNum=11, code=4, data=list_x[5] + tilt)
    for i in range(1, 12, 2):
        print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
ln.zGetNSCPosition(surfNum=4, objNum=i))
    read1 = ln.zGetPSF("fft")

    print("s6 Y tilt start")
    ln.zSetNSCPosition(surfNum=4, objNum=11, code=5, data=list_y[5] + tilt)
    for i in range(1, 12, 2):
        print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
ln.zGetNSCPosition(surfNum=4, objNum=i))
    read2 = ln.zGetPSF("fft")

    s6_shift_pose1 = normalize(read1[1])
    s6_shift_pose2 = normalize(read2[1])

    s6_sub1 = cv2.subtract(s5_shift_pose2, s6_shift_pose1)
```

```
    s6_sub2 = cv2.subtract(s6_shift_pose1, s5_shift_pose2)
    s6_sub3 = cv2.subtract(s6_shift_pose2, s6_shift_pose1)

    s6_pos1_coordinate, ti1 = find_coordinate1(s6_sub1)
    s6_pos2_coordinate, ti2 = find_coordinate1(s6_sub2)
    s6_pos3_coordinate, ti3 = find_coordinate1(s6_sub3)
    print("s6_pos1_coordinate, s6_pos2_coordinate, s6_pos3_coordinate", s6_pos1_coordinate,
s6_pos2_coordinate, s6_pos3_coordinate)

    plt.subplot(6, 6, 32)
    plt.imshow(s5_shift_pose2, cmap="gray")
    plt.title("s5_shift_pose")
    plt.subplot(6, 6, 33)
    plt.imshow(ti1, cmap="gray")
    plt.title("ti1")
    plt.subplot(6, 6, 34)
    plt.imshow(ti2, cmap="gray")
    plt.title("ti2")
    plt.subplot(6, 6, 35)
    plt.imshow(ti3, cmap="gray")
    plt.title("ti3")

    print("-------------------------------------------------------------------------------------")
    print("s1_pos1_coordinate, s1_pos2_coordinate, s1_pos3_coordinate", s1_pos1_coordinate,
s1_pos2_coordinate, s1_pos3_coordinate)
    print("s2_pos1_coordinate, s2_pos2_coordinate, s2_pos3_coordinate", s2_pos1_coordinate,
s2_pos2_coordinate, s2_pos3_coordinate)
    print("s3_pos1_coordinate, s2_pos3_coordinate, s3_pos3_coordinate", s3_pos1_coordinate,
s3_pos2_coordinate, s3_pos3_coordinate)
    print("s4_pos1_coordinate, s4_pos3_coordinate, s4_pos3_coordinate", s4_pos1_coordinate,
s4_pos2_coordinate, s4_pos3_coordinate)
    print("s5_pos1_coordinate, s5_pos3_coordinate, s5_pos3_coordinate", s5_pos1_coordinate,
s5_pos2_coordinate, s5_pos3_coordinate)
    print("s6_pos1_coordinate, s6_pos3_coordinate, s6_pos3_coordinate", s6_pos1_coordinate,
s6_pos2_coordinate, s6_pos3_coordinate)

    plt.subplot(6, 6, 36)
    plt.imshow(s6_shift_pose2, cmap="gray")
    plt.title("final_image")
    coordinate_list = [s1_pos1_coordinate, s1_pos2_coordinate, s1_pos3_coordinate,
s2_pos1_coordinate, s2_pos2_coordinate, s2_pos3_coordinate,
                       s3_pos1_coordinate, s3_pos2_coordinate, s3_pos3_coordinate,
s4_pos1_coordinate, s4_pos2_coordinate, s4_pos3_coordinate,
                       s5_pos1_coordinate, s5_pos2_coordinate, s5_pos3_coordinate,
s6_pos1_coordinate, s6_pos2_coordinate, s6_pos3_coordinate]

    return coordinate_list, tilt, s6_shift_pose2
def hexagon(dia_px):
    """This function return ideal hexagon coordinate"""
    hexa = dia_px
    center_px = [img_shape[0]/2, img_shape[1]/2]
    V = hexa * math.sin(30 * math.pi / 180)
    H = hexa * math.cos(30 * math.pi / 180)
    s1 = [int(center_px[0]), int(center_px[1] - hexa)]
    s2 = [int(center_px[0] + H), int(center_px[1] - V)]
    s3 = [s2[0], int(center_px[1] + V)]
    s4 = [center_px[0], int(center_px[1] + hexa)]
    s5 = [int(center_px[0] - H), s3[1]]
    s6 = [s5[0], s2[1]]
    hexa_coordinate = [s1, s2, s3, s4, s5, s6]
    return hexa_coordinate


if __name__ == '__main__':
    # step 01: connect Zemax software
    ln = pyz.PyZDDE()
    start = ln.zDDEInit()
    if start != 0:
        raise Exception("unable to initialize DDE link")

    # step 02: set parameter
    s1_x, s1_y = np.random.randn(1) * 0.001, np.random.randn(1) * 0.001
    s2_x, s2_y = np.random.randn(1) * 0.001, np.random.randn(1) * 0.001
    s3_x, s3_y = np.random.randn(1) * 0.001, np.random.randn(1) * 0.001
    s4_x, s4_y = np.random.randn(1) * 0.001, np.random.randn(1) * 0.001
    s5_x, s5_y = np.random.randn(1) * 0.001, np.random.randn(1) * 0.001
    s6_x, s6_y = np.random.randn(1) * 0.001, np.random.randn(1) * 0.001
    s1_6_tz = np.random.randn(6)*0.001

    list_x = [round(s1_x[0], 5), round(s2_x[0], 5), round(s3_x[0], 5), round(s4_x[0], 5),
round(s5_x[0], 5), round(s6_x[0], 5)]
    list_y = [round(s1_y[0], 5), round(s2_y[0], 5), round(s3_y[0], 5), round(s4_y[0], 5),
round(s5_y[0], 5), round(s6_y[0], 5)]
    list_tz =
[round(s1_6_tz[0],5),round(s1_6_tz[1],5),round(s1_6_tz[2],5),round(s1_6_tz[3],5),round(s1_6_tz[4],
5),round(s1_6_tz[5],5)]

    print("list_x =", list_x)
```

```
    print("list_y =", list_y)
    print("list_tz", list_tz)

    ln.zSetNSCPosition(surfNum=4, objNum=1, code=4, data=list_x[0])
    ln.zSetNSCPosition(surfNum=4, objNum=1, code=5, data=list_y[0])
    ln.zSetNSCPosition(surfNum=4, objNum=1, code=3, data=list_tz[0])

    ln.zSetNSCPosition(surfNum=4, objNum=3, code=4, data=list_x[1])
    ln.zSetNSCPosition(surfNum=4, objNum=3, code=5, data=list_y[1])
    ln.zSetNSCPosition(surfNum=4, objNum=3, code=3, data=list_tz[1])

    ln.zSetNSCPosition(surfNum=4, objNum=5, code=4, data=list_x[2])
    ln.zSetNSCPosition(surfNum=4, objNum=5, code=5, data=list_y[2])
    ln.zSetNSCPosition(surfNum=4, objNum=5, code=3, data=list_tz[2])

    ln.zSetNSCPosition(surfNum=4, objNum=7, code=4, data=list_x[3])
    ln.zSetNSCPosition(surfNum=4, objNum=7, code=5, data=list_y[3])
    ln.zSetNSCPosition(surfNum=4, objNum=7, code=3, data=list_tz[3])

    ln.zSetNSCPosition(surfNum=4, objNum=9, code=4, data=list_x[4])
    ln.zSetNSCPosition(surfNum=4, objNum=9, code=5, data=list_y[4])
    ln.zSetNSCPosition(surfNum=4, objNum=9, code=3, data=list_tz[4])

    ln.zSetNSCPosition(surfNum=4, objNum=11, code=4, data=list_x[5])
    ln.zSetNSCPosition(surfNum=4, objNum=11, code=5, data=list_y[5])
    ln.zSetNSCPosition(surfNum=4, objNum=11, code=3, data=list_tz[5])


    for i in range(1, 12, 2):
        print(ln.zGetNSCObjectData(surfNum=4, objNum=i, code=1), "=====",
ln.zGetNSCPosition(surfNum=4, objNum=i))

    # step 03: find sharp image
    sharp = sharpImage()
    sharp1 = normalize(sharp)
    img_shape = sharp1.shape
    print("Sharp_image_shape-> ", img_shape)
    plt.imshow(sharp, cmap="viridis")  # viridis
    plt.title("sharper img")
    plt.show(block=False)

    # step 04: segment id
    coordinate_list1, tilt1, segment_img = main(sharp1)

    # step 05: find a hexa coordinate
    hexa_coordinate = hexagon(500)

    # step 06: final Calculation for tilt x and y
    print("hexa_coordinate-> ", hexa_coordinate)
    print("coordinate_list1-> ", coordinate_list1)

    s_tiltx_list = []
    s_tilty_list = []
    final_sumX = []
    final_sumY = []

    for i in range(len(hexa_coordinate)):
        s_sub_distx = np.array(coordinate_list1[3 * i + 1] - np.array(coordinate_list1[3 * i]))
        s_sub_disty = np.array(coordinate_list1[3 * i + 2] - np.array(coordinate_list1[3 * i]))
        s_sub_hexa = np.array(hexa_coordinate[i]) - np.array(coordinate_list1[3 * i + 2])

        print(coordinate_list1[3 * i], coordinate_list1[3 * i + 1], coordinate_list1[3 * i + 2],
hexa_coordinate[i])
        print("s_sub_distx for s{}:".format(i + 1), s_sub_distx)#[1]
        print("s_sub_disty for s{}:".format(i + 1), s_sub_disty)#[0]
        print("s_sub_hexa for s{}:".format(i + 1), s_sub_hexa)

        s_tiltx = round((s_sub_hexa[1] * tilt1) / (s_sub_distx[1]), 5)
        s_tiltx_list.append(s_tiltx)
        s_tilty = round((s_sub_hexa[0] * tilt1) / (s_sub_disty[0]), 5)
        s_tilty_list.append(s_tilty)

        print("s_tiltx for s{}:".format(i + 1), s_tiltx)
        print("s_tilty for s{}:".format(i + 1), s_tilty)

        hx = round(list_x[i] + tilt1 + s_tiltx, 5)
        hy = round(list_y[i] + tilt1 + s_tilty, 5)
        ln.zSetNSCPosition(surfNum=4, objNum=2 * i + 1, code=4, data=hx)
        ln.zSetNSCPosition(surfNum=4, objNum=2 * i + 1, code=5, data=hy)
        print(f"Total_tilt x for hexa array s{i + 1}: {list_x[i]} + {tilt1} + {s_tiltx}  = {hx}")
        print(f"Total_tilt y for hexa array s{i + 1}: {list_y[i]} + {tilt1} + {s_tilty}  = {hy}")



        while_loop_x = []
        while_loop_y = []
        j = 0
        while True:
```

```
            print(f"how many time run while loop?--> {j}")
            img2 = ln.zGetPSF("fft")
            img2_n = normalize(img2[1])
            plt.figure()
            plt.imshow(img2_n, cmap="gray")
            plt.title("img2")

            img_sub = cv2.subtract(img2_n, segment_img)
            plt.figure()
            plt.imshow(img_sub, cmap="gray")
            plt.title("img_sub")

            s1_hexa1_coordinate = find_coordinate2(img_sub)
            print(f"s{i + 1}_hexa1_coordinate", s1_hexa1_coordinate)

            if s1_hexa1_coordinate == hexa_coordinate[i]:
                break
            else:
                s_sub_hexa = np.array(np.array(hexa_coordinate[i]) -
np.array(s1_hexa1_coordinate))

                s_tiltx = round((s_sub_hexa[1] * tilt1) / (s_sub_distx[1]), 9)
                while_loop_x.append(s_tiltx)
                s_tilty = round((s_sub_hexa[0] * tilt1) / (s_sub_disty[0]), 9)
                while_loop_y.append(s_tilty)

                print("s_tiltx", s_tiltx)
                print("s_tilty", s_tilty)

                sum_x = sum(while_loop_x)
                sum_y = sum(while_loop_y)
                ln.zSetNSCPosition(surfNum=4, objNum=2 * i + 1, code=4, data=hx + sum_x)  # 2 * i
+ 1
                ln.zSetNSCPosition(surfNum=4, objNum=2 * i + 1, code=5, data=hy + sum_y)  # 2 * i
+ 1
                print("while_loop_x", while_loop_x)
                print("while_loop_y", while_loop_y)
                print("sum_x =", sum_x)
                print("sum_y =", sum_y)
                print("final_tilt = ", hx + sum_x, hy + sum_y)

                for m in range(1, 12, 2):
                    print(ln.zGetNSCObjectData(surfNum=4, objNum=m, code=1), "=====",
ln.zGetNSCPosition(surfNum=4, objNum=m))

            j += 1
        final_sumX.append(sum_x)
        final_sumY.append(sum_y)
        while_loop_x = []
        while_loop_y = []
        segment_img = img2_n
        print("-------------------------------------------------------------------------------------
---------------")

    for m in range(1, 12, 2):
        print(ln.zGetNSCObjectData(surfNum=4, objNum=m, code=1), "=====",
ln.zGetNSCPosition(surfNum=4, objNum=m))

    # step 7: show final image
    last_img = ln.zGetPSF("fft")
    final_img = normalize(last_img[1])
    plt.figure()
    plt.imshow(final_img, cmap="gray")
    plt.title("final_img")

    # step 08: all spot at center
    print("start all spot take away in center")
    coordinate_list2 = hexa_coordinate
    hexa_coordinate1 = hexagon(0)
    print("hexa_coordinate1-> ", hexa_coordinate1)
    print("coordinate_list2-> ", coordinate_list2)

    print("s_tiltx_list ", s_tiltx_list)
    print("s_tilty_list ", s_tilty_list)

    for i in range(len(hexa_coordinate1)):
        s_sub_hexa_center = np.array(hexa_coordinate1[i]) - np.array(coordinate_list2[i])

        print(coordinate_list1[3 * i], coordinate_list1[3 * i + 1], coordinate_list1[3 * i + 2],
hexa_coordinate[i], hexa_coordinate1[i])
        # print("s_sub_dist for s{} take away in center:".format(i + 1), s_sub_dist)
        print("s_sub_hexa_center for s{} take away in center:".format(i + 1), s_sub_hexa)

        s_tiltx_center = round((s_sub_hexa_center[1] * tilt1) / (s_sub_distx[1]), 9)
        s_tilty_center = round((s_sub_hexa_center[0] * tilt1) / (s_sub_disty[0]), 9)

        print("s_tiltx for s{} take away in center:".format(i + 1), s_tiltx_center)
        print("s_tilty for s{} take away in center:".format(i + 1), s_tilty_center)
```

```
        tx = round(list_x[i] + tilt1 + s_tiltx_list[i] + final_sumX[i] + s_tiltx_center, 9)
        ty = round(list_y[i] + tilt1 + s_tilty_list[i] + final_sumY[i] + s_tilty_center, 9)
        ln.zSetNSCPosition(surfNum=4, objNum=2 * i + 1, code=4, data= tx)
        ln.zSetNSCPosition(surfNum=4, objNum=2 * i + 1, code=5, data= ty)

        print(f"Total_tilt x s{i + 1}: {list_x[i]} + {tilt1} + {s_tiltx_list[i]} + {final_sumX[i]}
+ {s_tiltx_center} = {tx}")
        print(f"Total_tilt y s{i + 1}: {list_y[i]} + {tilt1} + {s_tilty_list[i]} + {final_sumY[i]}
+ {s_tilty_center} = {ty}")
        print("-------------------------------------------------------------------------------------
---------------")

    center_img = ln.zGetPSF("fft")
    center_final_img = normalize(center_img[1])
    plt.figure()
    plt.imshow(center_final_img, cmap="gray")
    plt.title("center_final_img")

    plt.show()
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

References

[1] Reynolds, P., Atkinson, C., & Gliman, L. (2004, May). Design and development of the primary and secondary mirror deployment systems for the cryogenic jwst. In *37th Aerosp. Mech. Symp* (pp. 29-44).

[2] Wavefront Sensing And Control For The James Webb Space Telescope, By D. Scott Acton, Bruce Dean, Lee Feinberg**,** 34th Space Symposium, Technical Track, Colorado Springs, Colorado, United States of America Presented on April 16, 2018

- Zemax Manual
- Opencv documentation
- Matplotlib documentation
- PzDDE documentation
- Numpy documentation

DEVLOPMENT OF GLOBAL ALIGNMENT METHODOLOGY FOR DEPLOYABLE OPTICAL TELESCOPE SYSTEM USING IMAGE BASED SENSING

A Project Report ON

Rugnesh Vora
(20EL096)

Under the guidance of

Industry Guide: Shri NUMAN AHMAD
at
Space Applications Centre, (ISRO) Ahmedabad

Faculty Guide: Prof. MEHFUZA HOLIA

In fulfilment of the award of the
Degree Of
BACHELOR OF TECHNOLOGY
In
ELECTRONICS ENGINEERING

BIRLA VISHVAKARMA MAHAVIDYALAYA
ENGINEERING COLLEGE
[An Autonomous Institution]
ANAND – 388120
GUJARAT