Moildev Documentation

Release 4.0.0

Perseverance Technology

July 28, 2023

CONTENTS:

CHAPTER

ONE

RELEASE NOTE

1.1 Moildev 4.0

Moildev is an advanced software development kit (SDK) designed specifically for processing fisheye images. It stands out from its competitors thanks to a groundbreaking and unique invention that completely transforms the way fisheye images are handled, setting new standards for quality and performance. With Moildev, you can expect an exceptional level of precision and accuracy in processing fisheye distortions, ensuring clear and distortion-free images like never before. Its user-friendly interface makes it easy for both experienced professionals and newcomers to utilize its advanced fisheye processing capabilities. Embrace Moildev to elevate your projects with outstanding results that represent the future of fisheye image processing.

1.2 Release Highlight

- Introducing the parameter camera, now become a property of the class Moildev, enhancing flexibility and control.
- Refactored function names in adherence to the PEP8 rule, ensuring improved code readability. For example, **getAnypointMaps** (V3.0) has been renamed to **maps_anypoint_mode1**. For detailed information, refer to the script help.
- Renamed the function reverseImage to recenter, making its purpose clearer and more intuitive.
- The current version of Moildev now specifies the panorama type by the application, with options for **panorama_car** and **panorama_tube**. This allows for a more tailored and efficient panorama creation process.

1.3 Copyright

CHAPTER

TWO

GETTING STARTED

This section is for you if you are a developer and are using Moildev Library for the first time.

2.1 Installation

You have two options for install Moildev library. The first option to access the Moildev library is directly install from the PyPI distribution. You just need to execute the command bellow:

```
$ pip install Moildev
```

Alternatively, you can choose the second option, which involves cloning the Moildev repository directly from GitHub.

```
$ git clone https://github.com/perseverance-tech-tw/moildev.git
```

2.2 Basic Concept

The Fisheye lens, also known as the Fisheye image sensor (FIS), is a unique ultra-wide-angle lens with a short focal length that generates considerable optical distortion and is designed to provide a wide, panoramic, or hemispherical image.

The large field of view is the most important characteristic. With a FOV of more than 180 degrees, a Fisheye camera (also known as a Fisheye image sensor, or FIS) can capture a clear image, but a severer barrel distortion comes along.

According to Prof. Chuang-Jan Chang, the approach to displaying Fisheye camera images incorporates multicollimator metrology and cartography in order to methodically characterize the Fisheye camera's projection mechanism.

The hemisphere coordinate system is produced by the Fisheye camera in our suggested technique. Hence, the position of an imaged point referring to the principal point on the image plane directly reflects its corresponding zenithal distance (alpha). and azimuthal distance (beta) of the sight ray in space to normalize the imaged point onto a small sphere presented in the following figure:

Based on the coordinate system, the angles respectively defined by incident rays and the optical axis are the zenithal angle (alpha) and the azimuthal angle (beta), which are the angles surrounding the optical axis.

It has a relationship with the coordinate system X, Y, and Z, where the optical axis is defined by the Z-axis. For the zenithal angle, it is the angle from the vertical optical axis to the X- and Y-axes, as shown in Figure A.

Whereas the azimuthal angle is defined as the angle of positive Y as the reference point with a value of 0 degrees and the Z-axis is used as the rotation axis, as shown in Figure B. The rotation around the optical axis is the angle of the Y axis, starting from the positive direction and clockwise around the X axis.

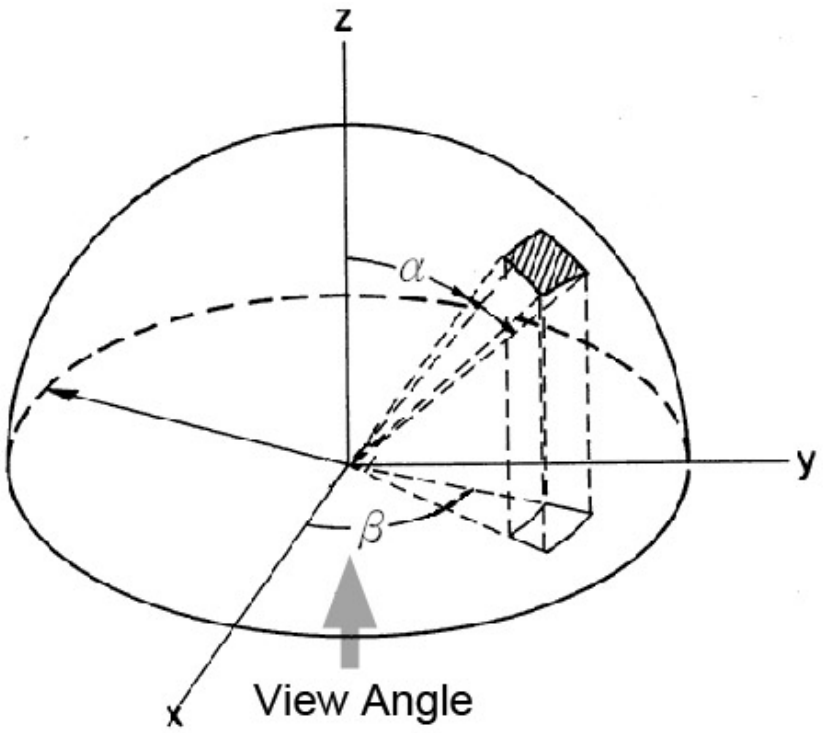


Fig. 1: Moil dash camera

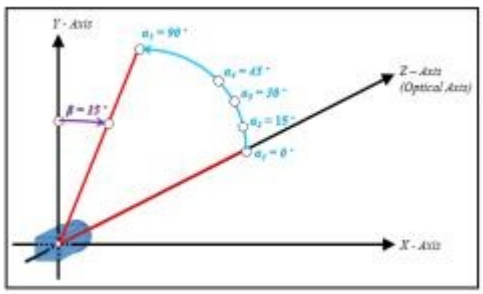


Fig. 2: A.Zenithal Angle

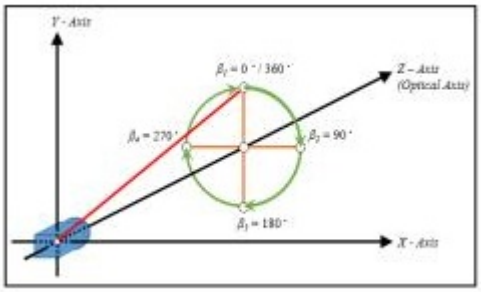


Fig. 3: B.Azimuthal Angle

CHAPTER

THREE

QUICK TUTORIALS

If you're working on fisheye image processing applications, especially using the Python programming language, look no further than the Moildev Library. This tutorial is designed to give you a straightforward and comprehensive understanding of the incredible capabilities that Moildev brings to the table. So, let's dive in and discover the exciting possibilities that await you with Moildev!

For complete reference, see the *API documentation*.

3.1 Import moildev library

```
from moildev import Moildev
```

3.2 Create moildev object

To create the object from Moildev, you have to provide the parameter. The camera parameter is the result from calibration camera by MOIL laboratory that will store on **.json** file. Bellow this is the example:

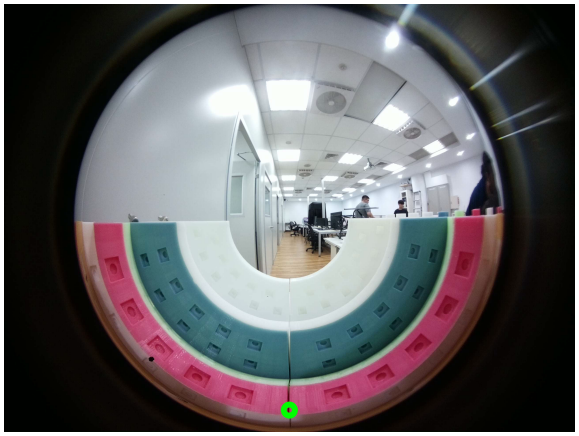
```
moildev = Moildev("camera_parameter_path.json")
```

3.3 Load Image

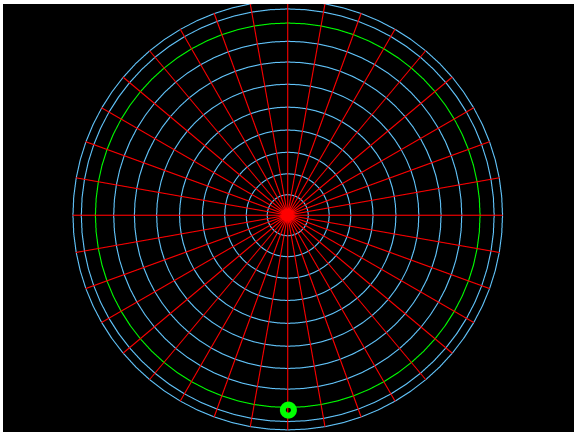
We start by loading an image from a specified file using the OpenCV Library. Next, we process this image using Moildev to achieve our desired results.

```
image = cv2.imread("Image_Path in your computer")
```

Below is an example of a fisheye image. We are providing two images: the first one is from an experiment, and the second one is created based on the given parameter. Both images have a field of view of 220 degrees.



Fisheye image from experiment



Phantom fisheye image

3.4 Create maps anypoint mode 1

This function serves to generate a set of X-Y Maps based on the provided alpha, beta, and zoom parameters. These maps are then used to remap the original fisheye image to the desired angle image. The function specifically creates maps for the "anypoint mode 1" which is ideal for tube applications. Below are some examples showcasing how this function can be used.

```
map_X, map_Y = moildev.maps_anypoint_mode1(alpha, beta, zoom)
```

Example:

```
map_X, map_Y = moildev.maps_anypoint_mode1(90, 180, 2)
anypoint_maps_m1 = cv2.remap(image, map_X, map_Y, cv2.INTER_CUBIC)
anypoint_maps_m1 = cv2.resize(anypoint_maps_m1, (400, 300))
cv2.imshow("anypoint using maps mode 1", anypoint_maps_m1)
```

3.5 Create anypoint mode 1

This function generates an anypoint view mode 1 image. The resulting image is rotated by beta degrees around the Z-axis (roll) after an alpha degree rotation around the X-axis (pitch). To use this function, you'll need to provide an image as a parameter, and it will return the remapped image as the result. Below, you'll find an example implementation to help you understand how to use it.

```
anypoint_m1 = moildev.anypoint_mode1(image, alpha, beta, zoom)
```

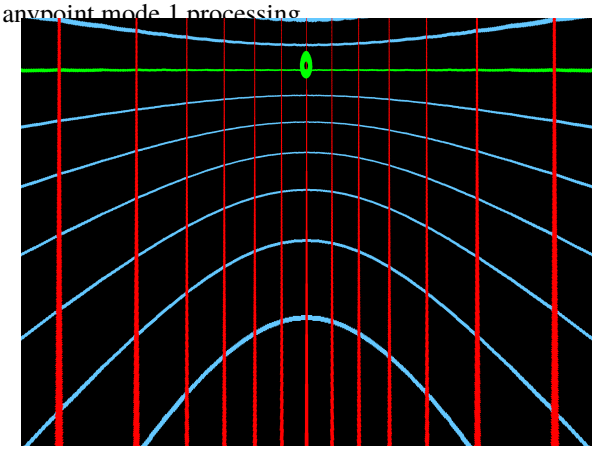
Example:

```
anypoint_m1 = moildev.anypoint_mode1(image, 90, 180, 2)
anypoint_m1 = cv2.resize(anypoint_m1, (400, 300))
cv2.imshow("anypoint mode 1", anypoint_m1)
```

Here is the resulting image obtained after applying the anypoint mode 1 processing. This image showcases the outcome achieved through the use of the anypoint mode 1 algorithm.

Table 1: Image result from anypoint mode 1 processing

Fisheye image from experiment



result from Phantom fisheye image

3.6 Create maps anypoint mode 2

This function is designed to generate a set of X-Y Maps based on the provided pitch, yaw, roll, and zoom parameters. These maps are then used to remap the original fisheye image to the desired target image. By utilizing these X-Y Maps, the function allows for precise and accurate adjustments to the perspective of the fisheye image, ensuring that the final result aligns with the intended visual specifications.

```
map_X, map_Y = moildev.maps_anypoint_mode2(pitch, yaw, roll, zoom)
```

Example:

```
map_X, map_Y = moildev.maps_anypoint_mode2(-90, 0, 0, 2)
anypoint_maps_m2 = cv2.remap(image, map_X, map_Y, cv2.INTER_CUBIC)
anypoint_maps_m2 = cv2.resize(anypoint, (400, 300))
cv2.imshow("anypoint maps mode 2" anypoint_maps_m2)
```

3.7 Create anypoint mode 2

This function generates an anypoint view mode 2 image. To use this function, you'll need to provide an image as a parameter, and it will return the remapped image as the result. Below, you'll find an example implementation to help you understand how to use it.

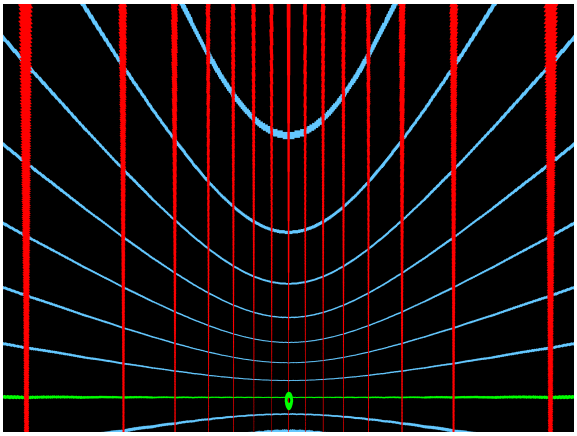
```
anypoint_m2 = moildev.anypoint_mode2(image, pitch, yaw, roll, zoom)
```

Example:

```
anypoint_m2 = moildev.anypoint_mode2(image, -90, 0, 0, 2)
anypoint_m2 = cv2.resize(anypoint_m2, (400, 300))
cv2.imshow("anypoint mode 2", anypoint_m2)
```

Here is the resulting image obtained after applying the anypoint mode 2 processing. This image showcases the outcome achieved through the use of the anypoint mode 2 algorithm.

Fisheye image from experiment



Phantom fisheye image

3.8 Create panorama tube

This function enables the creation of an image with a panoramic view. It allows you to capture a wide-angle perspective, providing a seamless and immersive visual experience.

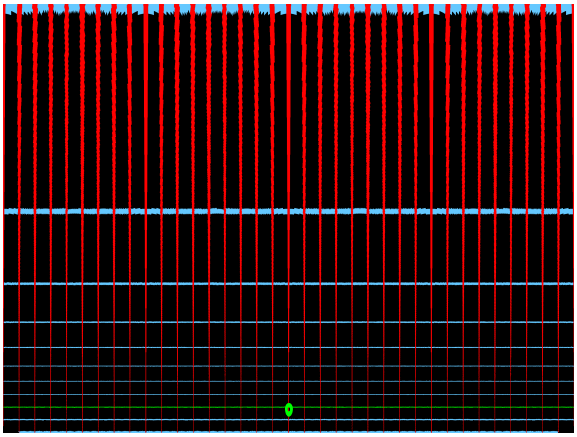
```
panorama_tube = moildev.panorama_tube(image, alpha_min, alpha_max)
```

Example:

```
panorama_tube = moildev.panorama_tube(image, 10, 110)
panorama_tube = cv2.resize(panorama_tube, (400, 300))
cv2.imshow("panorama tube", panorama_tube)
```

Here is the resulting image obtained after applying the panorama tube processing. This image showcases the outcome achieved through the use of the panorama tube algorithm.

Fisheye image from experiment



Phantom fisheye image

3.9 Create panorama car

This function is designed to generate a panorama image from a fisheye camera. You can control the image's pitch direction by adjusting the alpha parameter and the yaw direction by modifying the beta parameter. Additionally, to select a specific region of interest (ROI), you have the flexibility to adjust the left, right, top, and bottom parameters. This enables you to precisely customize and tailor the resulting panorama image to suit your specific requirements and preferences.

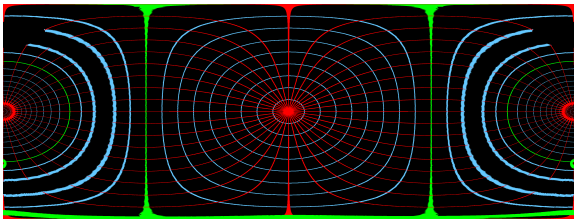
```
panorama_car = moildev.panorama_car(image, alpha_max, alpha, beta, left, right, top,␣
↪bottom)
```

Example:

```
panorama_car = moildev.panorama_car(image, 180, 80, 0, 0.25, 0.75, 0, 1)
panorama_car = cv2.resize(panorama_car, (400, 300))
cv2.imshow("panorama car", panorama_car)
```

Here is the resulting image obtained after applying the panorama car processing. This image showcases the outcome achieved through the use of the panorama car algorithm.

Fisheye image from experiment



Phantom fisheye image

3.10 Create recenter image

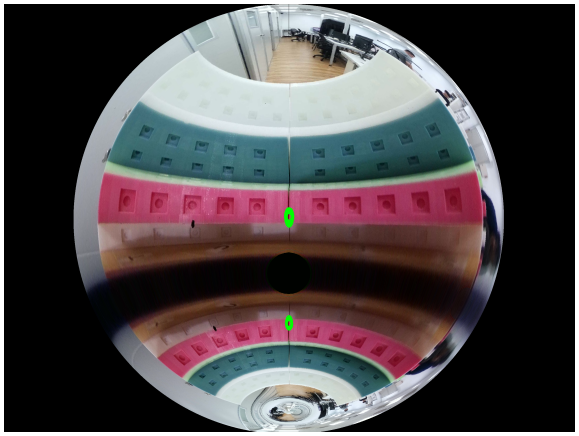
This function serves a crucial purpose in the realm of fisheye image processing by facilitating the alteration of the optical point. It allows users to redefine the center of view, offering the ability to shift the focal point or pivot of the fisheye image. This feature proves invaluable for various applications, as it empowers developers and creatives to fine-tune the perspective and focus of their fisheye images according to their specific needs and artistic vision. Whether it involves adjusting the field of view or repositioning the optical center, this function unlocks a world of possibilities and unleashes the full potential of fisheye image processing.

```
recenter = moildev(image, alpha_max, IC_alpha_degree, IC_beta_degree)
```

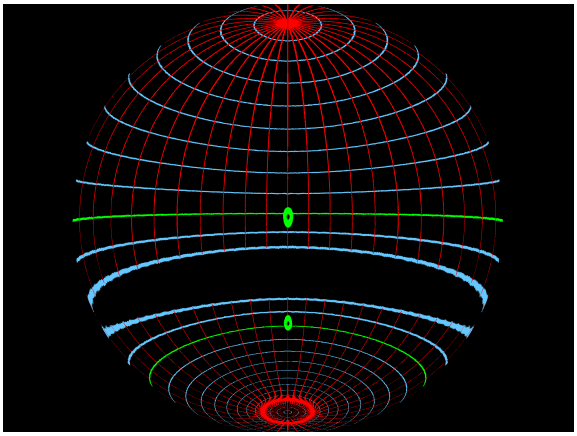
Example:

```
recenter = moildev.recenter(image, 110, 25, 10)
recenter = cv2.resize(recenter, (400, 300))
cv2.imshow("show recenter", recenter)
```

Here is the resulting image obtained after applying the recenter optical point processing. This image showcases the outcome achieved through the use of the recenter optical point algorithm.



Fisheye image from experiment



Phantom fisheye image

~Perseverance Technology~

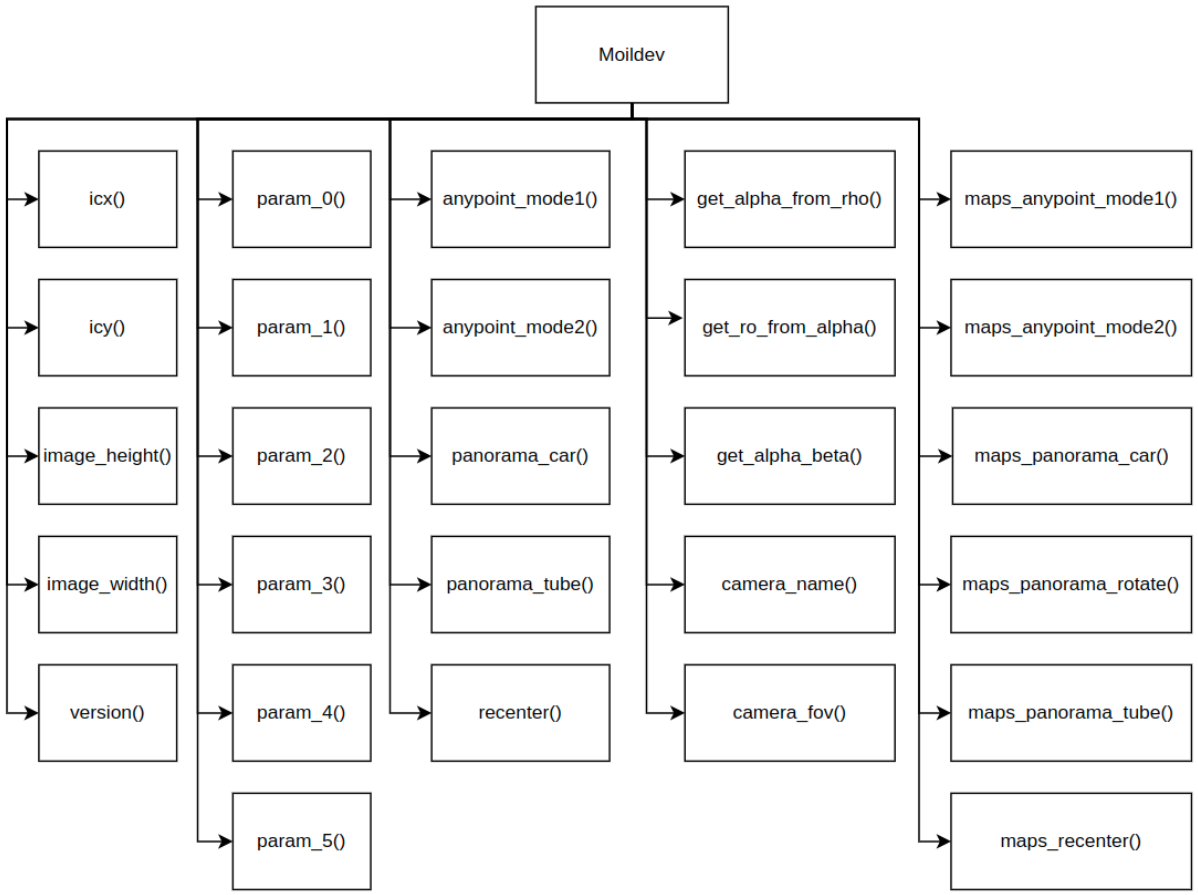
CHAPTER

FOUR

ARCHITECTURE

4.1 Overview

The Moildev Library is a comprehensive set of functions designed to facilitate the development of fisheye image applications. Originally created for the C++ programming language, the library has evolved to adapt to the changing preferences of developers over the past decade. Given Python's immense popularity among developers in recent years, we recognized the need to expand the Moildev library's capabilities and make it accessible to Python users as well. As a result, we are now actively developing and enhancing the Moildev Library for Python, ensuring that Python developers can harness its power and versatility for their fisheye image projects with ease and efficiency.



4.2 Supported Platform

This library is specifically designed to provide support for Python programming language, catering to both Windows x64 and Ubuntu systems with AMD architecture. By offering compatibility with these platforms, developers using Python can seamlessly leverage the capabilities of this library for their projects on Windows and Ubuntu operating systems running on 64-bit architectures and AMD processors. This ensures that a broader community of Python developers can take advantage of the library's functionalities and optimize their coding experience for enhanced performance and efficiency. Whether you're working on a Windows x64 or Ubuntu with AMD setup, this versatile library empowers you with a wealth of tools and features to create robust and cutting-edge applications.

CHAPTER

FIVE

SAMPLE APPLICATION

5.1 1. Car Application

There are several implementations of Moildev in car applications, including Driver Monitoring Systems (DMS), birds-eye view, and dash cameras. Below, you can find examples showcasing the impressive results achieved through our Moildev implementation in these car-related applications.

Driver Monitoring System (DMS)

https://youtu.be/Z5EtsPGONW8

Moildev Birds View

https://youtu.be/t0WoHAfyE9o

https://youtu.be/Vkep3_SpWkQ

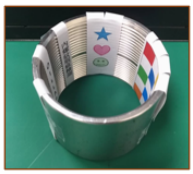
Advanced Moildev birds view configuration

https://youtu.be/9v3VWt_3rAI

5.2 2. Internal Thread Inspection

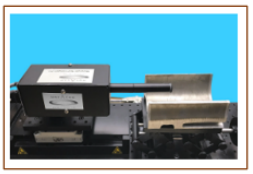
Internal thread inspection refers to the process of evaluating and assessing the quality and integrity of threads inside a cylindrical component, such as a nut or a threaded hole. This inspection is critical in various industries, including manufacturing, aerospace, automotive, and engineering, where threaded components play a vital role in assemblies and connections.

Moildev proves to be an excellent platform for processing fisheye images in internal thread inspection applications. With its ability to handle larger fields of view (FOV) offered by fisheye images, Moildev enables faster and more efficient inspection processes, resulting in a reduced error rate. Leveraging fisheye images in internal thread inspection allows for a more comprehensive and accurate assessment of threaded components, ensuring their proper fit and functionality.

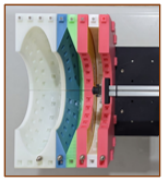


Bore specimen

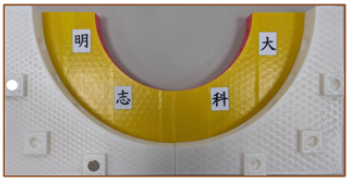
Moildev's reconstruction of the bore's inside wall



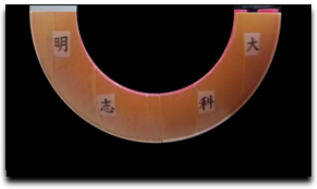
Existing Solution



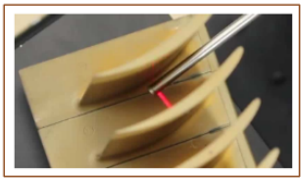
Trench structure



Target taken by normal camera



Hide Part Texture Rebuild



Existing Solution

5.3 3. 3D measurement

Moildev is not only proficient in fisheye image processing but also excels in high-accuracy 3D measurements. It enables measurements in any direction and covers areas captured by the fisheye lens. This versatility allows for precise and comprehensive 3D measurements, making Moildev a powerful tool in various applications that require accurate spatial analysis. Whether it's for industrial metrology, robotics, or other fields, Moildev's capabilities in 3D measurement add a new dimension of accuracy and efficiency to the process, ensuring reliable and valuable data insights.

5.4 4. Visual odometry

Visual odometry is a technique used in computer vision and robotics to estimate the movement and position of a camera or a vehicle by analyzing the changes in visual information captured by the camera over time. It works similarly to how humans use their eyes to judge their movement and orientation. By comparing consecutive images or frames, visual odometry algorithms can determine how much the camera or vehicle has moved and in which direction. This information is valuable for navigation, mapping, and localization tasks in various applications, such as autonomous vehicles, drones, and augmented reality systems.

Instead using single view, using fisheye camera and process with moildev, we can extend the single fisheye image into several view to improve the accuracy.

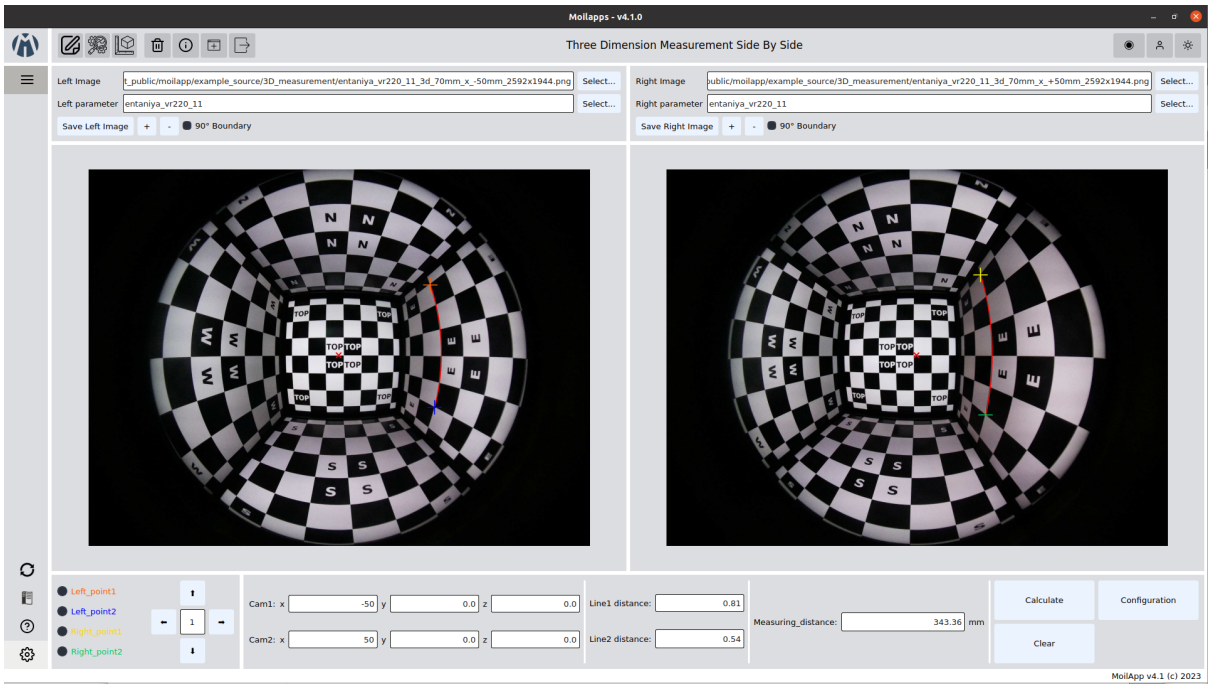


Fig. 1: Example 3D measurement in vertical direction

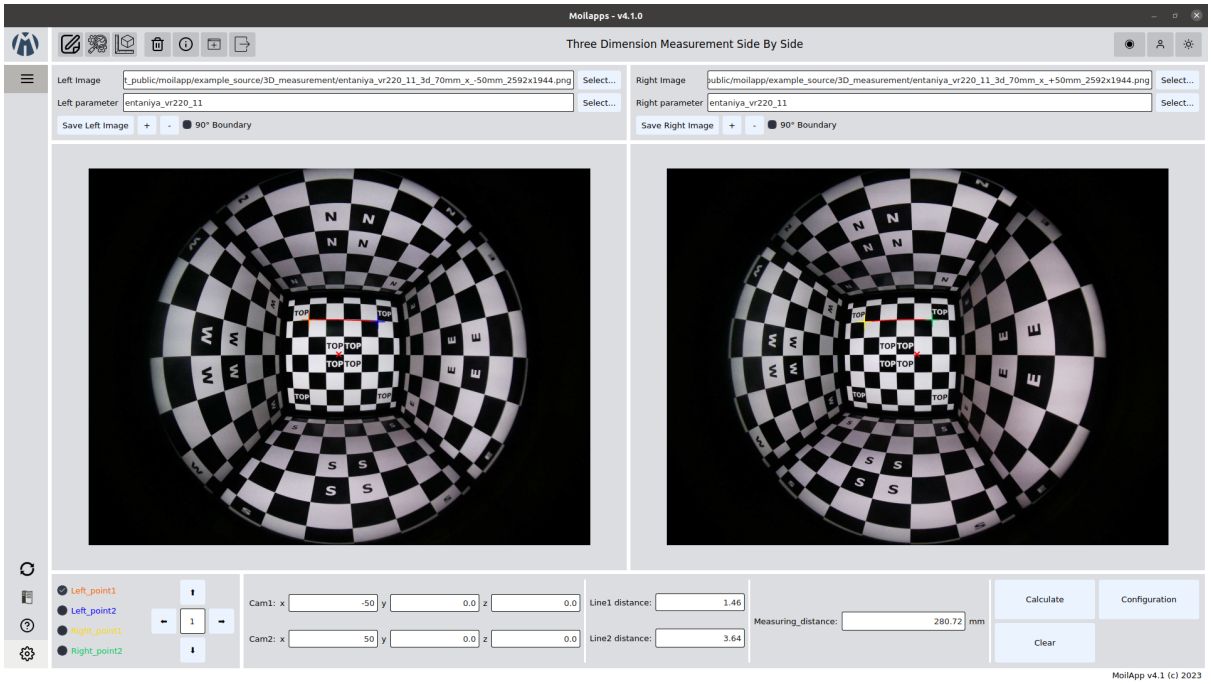


Fig. 2: Example 3D measurement in horizontal direction

https://youtu.be/gh66k5fZd90

5.5 5. Colonoscopy

Colonoscopy is a medical procedure used to examine the inside of the colon (large intestine) and rectum. During the procedure, a long, flexible tube with a camera at the tip, known as a colonoscope, is inserted through the anus and guided through the colon. The camera allows the healthcare provider to view the lining of the colon on a monitor, enabling them to identify and diagnose various conditions, such as colon polyps, inflammation, and colorectal cancer. Colonoscopy is an essential tool for detecting and preventing colorectal diseases and is often recommended for individuals above a certain age or those with specific risk factors. It plays a crucial role in early detection and treatment, promoting better health outcomes for patients.

By utilizing fisheye images and processing them with Moildev, we can offer improved perspectives during medical examinations. This approach helps reduce errors for doctors and enhances the detection rate of abnormalities, leading to more accurate diagnoses and improved patient outcomes.

Bellow video is some of the experiment result colonoscopy image process by Moildev

https://youtu.be/hgYHjykljwE

https://youtu.be/ndQIB7Dn0c4

CHAPTER

SIX

API DOCUMENTATION

6.1 Moildev Module

class Moildev(*file_camera_parameter=None*, *camera_type=None*, ***kwarg*)

Bases: `object`

Before the ensuing functions can function properly, the camera parameter must be configured at the start of the program. The camera parameter is the outcome of the MOIL laboratory's calibration camera.

Parameters

- **file_camera_parameter** – *.json file
- **camera_type** – the name of the camera type used (use if yore pass the parameter using *.json file)
- **cameraName** – the name of the camera used
- **cameraFov** – camera field of view (FOV)
- **sensor_width** – size of sensor width
- **sensor_height** – size of sensor height
- **Icx** – center image in x-axis
- **Icy** – center image in y-axis
- **ratio** – the value of the ratio image
- **imageWidth** – the size of width image
- **imageHeight** – the size of height image
- **calibrationRatio** – the value of calibration ratio
- **parameter5** (*parameter0 ..*) – intrinsic fisheye camera parameter get from calibration

```
for more detail, please reference `https://github.com/perseverance-tech-tw/
↪moildev`
```

classmethod version()

Showing the information of the version moildev library.

Returns

Moildev version information

property camera_name

Get camera name used.

Returns

Camera name (string)

property camera_fov

Get Field of View (FoV) from camera used.

Returns
FoV camera (int)

property icx

Get center image x-axis from camera used.

Returns
Image center X (int)

property icy

Get center image y-axis from camera used.

Returns
Image center Y(int)

property image_width

Get the width of the image used.

Returns
image width(int)

property image_height

Get the height of the image used.

Returns
image height(int)

property param_0

Get the value of calibration parameter_0 from camera used.

Returns
Parameter_0 (float)

property param_1

Get the value of calibration parameter_1 from camera used.

Returns
Parameter_1 (float)

property param_2

Get the value of calibration parameter_2 from camera used.

Returns
Parameter_2 (float)

property param_3

Get the value of calibration parameter_3 from camera used.

Returns
Parameter_3 (float)

property param_4

Get the value of calibration parameter_4 from camera used.

Returns
Parameter_4 (float)

property param_5

Get the value of calibration parameter_5 from camera used.

Returns
Parameter_5 (float)

maps_anypoint_mode1(*alpha*, *beta*, *zoom*)

Generate a pair of X-Y Maps for the specified alpha, beta and zoom parameters, and then utilize the resulting X-Y Maps to remap the original fisheye image to the target angle image. This function has 2 mode to generate maps anypoint, mode 1 is for tube application and mode 2 usually for car application

Parameters

- **alpha** – value of zenith distance(float).
- **beta** – value of azimuthal distance based on cartography system(float)
- **zoom** – value of zoom(float)

Returns
the mapping matrices X mapY: the mapping matrices Y

Return type
mapX

```
please reference: `https://github.com/perseverance-tech-tw/moildev`
```

maps_anypoint_mode2(*pitch*, *yaw*, *roll*, *zoom*)

Generate a pair of X-Y Maps for the specified pitch, yaw, and roll also zoom parameters, and then utilize the resulting X-Y Maps to remap the original fisheye image to the target image.

Parameters

- **pitch** – pitch rotation (from -110 to 110 degree)
- **yaw** – yaw rotation (from -110 to 110 degree)
- **roll** – roll rotation (from -110 to 110 degree)
- **zoom** – zoom scale (1 - 20)

Returns
the mapping matrices X mapY: the mapping matrices Y

Return type
mapX

```
please reference: `https://github.com/perseverance-tech-tw/moildev`
```

maps_panorama_tube(*alpha_min*, *alpha_max*)

To generate a pair of X-Y Maps for alpha within 0 … alpha_max degree, the result X-Y Maps can be used later to generate a panorama image from the original fisheye image.

Parameters

- **alpha_min** – the minimum alpha degree given
- **alpha_max** – the maximum alpha degree given. The recommended value is half of camera FOV. For example, use 90 for a 180 degree fisheye images and use 110 for a 220 degree fisheye images.

Returns
the mapping matrices X mapY: the mapping matrices Y

Return type
mapX

```
please reference: `https://github.com/perseverance-tech-tw/moildev`
```

maps_panorama_car(*alpha_max*, *iC_alpha_degree*, *iC_beta_degree*, *p_alpha_from*, *p_alpha_end*)

To generate a pair of X-Y Maps for alpha within 0 alpha_max degree, the result X-Y Maps can be used later to generate a panorama image from the original fisheye image. The panorama image centered at the 3D direction with alpha = iC_alpha_degree and beta = iC_beta_degree.

Parameters

- **alpha_max** – max of alpha. The recommended value is half of camera FOV. For example, use 90 for a 180 degree fisheye images and use 110 for a 220 degree fisheye images.
- **iC_alpha_degree** – alpha angle of panorama center.
- **iC_beta_degree** – beta angle of panorama center.
- **p_alpha_from** –
- **p_alpha_end** –

Returns
mapX mapY

```
please reference: `https://github.com/perseverance-tech-tw/moildev`
```

maps_panorama_rt(*alpha_max*, *iC_alpha_degree*, *iC_beta_degree*)

To generate a pair of X-Y Maps for alpha within 0..alpha_max degree, the result X-Y Maps can be used later to generate a panorama image from the original fisheye image. The panorama image centered at the 3D direction with alpha = iC_alpha_degree and beta = iC_beta_degree.

Parameters

- **alpha_max** – max of alpha. The recommended value is half of camera FOV. For example, use 90 for a 180 degree fisheye images and use 110 for a 220 degree fisheye images.
- **iC_alpha_degree** – alpha angle of panorama center.
- **iC_beta_degree** – beta angle of panorama center.

Returns
mapX mapY

```
please reference: `https://github.com/perseverance-tech-tw/moildev`
```

maps_recenter(*alpha_max*, *beta_degree*)

Create maps for reverse image. this can work using input panorama rotation image

Parameters

- **alpha_max** – max of alpha. The recommended value is half of camera FOV. For example, use 90 for a 180 degree fisheye images and use 110 for a 220 degree fisheye images.
- **beta_degree** – beta angle.

Returns
maps_x_reverse, maps_y_reverse

```
please reference: `https://github.com/perseverance-tech-tw/moildev`
```

anypoint_mode1(*image*, *alpha*, *beta*, *zoom*)

Generate anypoint view image. for mode 1, the result rotation is betaOffset degree rotation around the Z-axis(roll) after alphaOffset degree rotation around the X-axis(pitch). for mode 2, The result rotation is thetaY degree rotation around the Y-axis(yaw) after thetaX degree rotation around the X-axis(pitch).

Parameters

- **image** – source image given
- **alpha** – the alpha offset that corespondent to the pitch rotation
- **beta** – the beta offset that corespondent to the yaw rotation

- **zoom** – decimal zoom factor, normally 1..12

Returns
anypoint image

```
please reference: `https://github.com/perseverance-tech-tw/moildev`
```

anypoint_mode2(*image*, *pitch*, *yaw*, *roll*, *zoom*)

Generate anypoint view image. for mode 1, the result rotation is betaOffset degree rotation around the Z-axis(roll) after alphaOffset degree rotation around the X-axis(pitch). for mode 2, The result rotation is thetaY degree rotation around the Y-axis(yaw) after thetaX degree rotation around the X-axis(pitch).

Parameters

- **image** – source image given
- **pitch** – the alpha offset that corespondent to the pitch rotation
- **yaw** – the beta offset that corespondent to the yaw rotation
- **roll** – the beta offset that corespondent to the yaw rotation
- **zoom** – decimal zoom factor, normally 1..12

Returns
anypoint image

```
please reference: `https://github.com/perseverance-tech-tw/moildev`
```

panorama_tube(*image*, *alpha_min*, *alpha_max*)

The panorama image

Parameters

- **image** – image source given
- **alpha_min** –
- **alpha_max** –
- **flip_h** – Flip horizontal axis (boolean True or False)

Returns
Panorama view image

```
please reference: `https://github.com/perseverance-tech-tw/moildev`
```

panorama_car(*image*, *alpha_max*, *alpha*, *beta*, *left*, *right*, *top*, *bottom*)

The function that generate a moil dash panorama image from fisheye camera. the image can control by alpha to change the pitch direction and beta for yaw direction. in order to select the roi, we can control by the parameter such as left, right, top, and bottom.

Parameters

- **image** – input fisheye image
- **alpha_max** –
- **alpha** – change the pitch direction(0 ~ 180)
- **beta** – change the yaw direction(-90 ~ 90)
- **left** – crop the left image by scale(0 ~ 1)
- **right** – crop the right image by scale(0 ~ 1)
- **top** – crop the top image by scale(0 ~ 1)
- **bottom** – crop the bottom image by scale(0 ~ 1)

- **flip_h** – Flip horizontal axis (boolean True or False)

Returns
Panorama image

```
please reference: `https://github.com/perseverance-tech-tw/moildev`
```

recenter(*image*, *alpha_max*, *iC_alpha_degree*, *iC_beta_degree*)

Change the optical point of fisheye image.

Parameters

- **image** – input image
- **alpha_max** – max of alpha. The recommended value is half of camera FOV. For example, use 90 for a 180 degree fisheye images and use 110 for a 220 degree fisheye images.
- **iC_alpha_degree** – alpha angle of panorama center
- **iC_beta_degree** – beta angle of panorama center

Returns
reverse image

```
please reference: `https://github.com/perseverance-tech-tw/moildev`
```

get_alpha_from_rho(*rho*)

Get the alpha from rho image.

Parameters
rho – the value of rho given

Returns
alpha

get_rho_from_alpha(*alpha*)

Get rho image from alpha given.

Parameters
alpha – the value of alpha given

Returns
rho image

get_alpha_beta(*coordinateX*, *coordinateY*, *mode=1*)

Get the alpha beta from specific coordinate image.

Parameters

- **coordinateX** –
- **coordinateY** –
- **mode** –

Returns
alpha, beta (if you get none, the coordinate is out of range that can cover)

```
please reference: `https://github.com/perseverance-tech-tw/moildev`
```

6.2 Testing code

To gain a comprehensive understanding of the moildev capabilities, you have the option to explore the entire testing code, accessible within the **unittest** folder.

PYTHON MODULE INDEX

m

moildev.Moildev, 16

INDEX