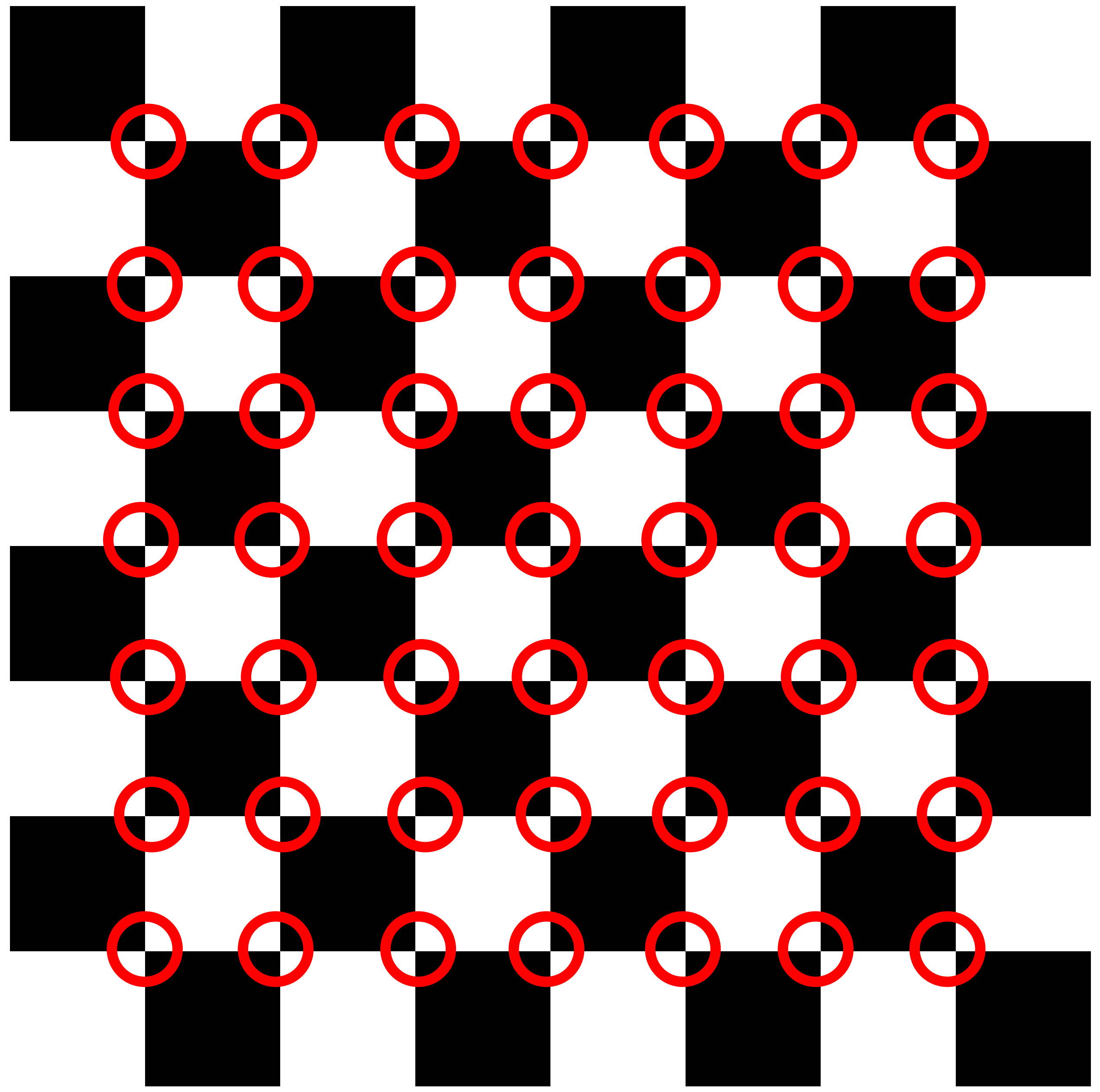
Numerical Software Development Final Project

Camera Calibrator

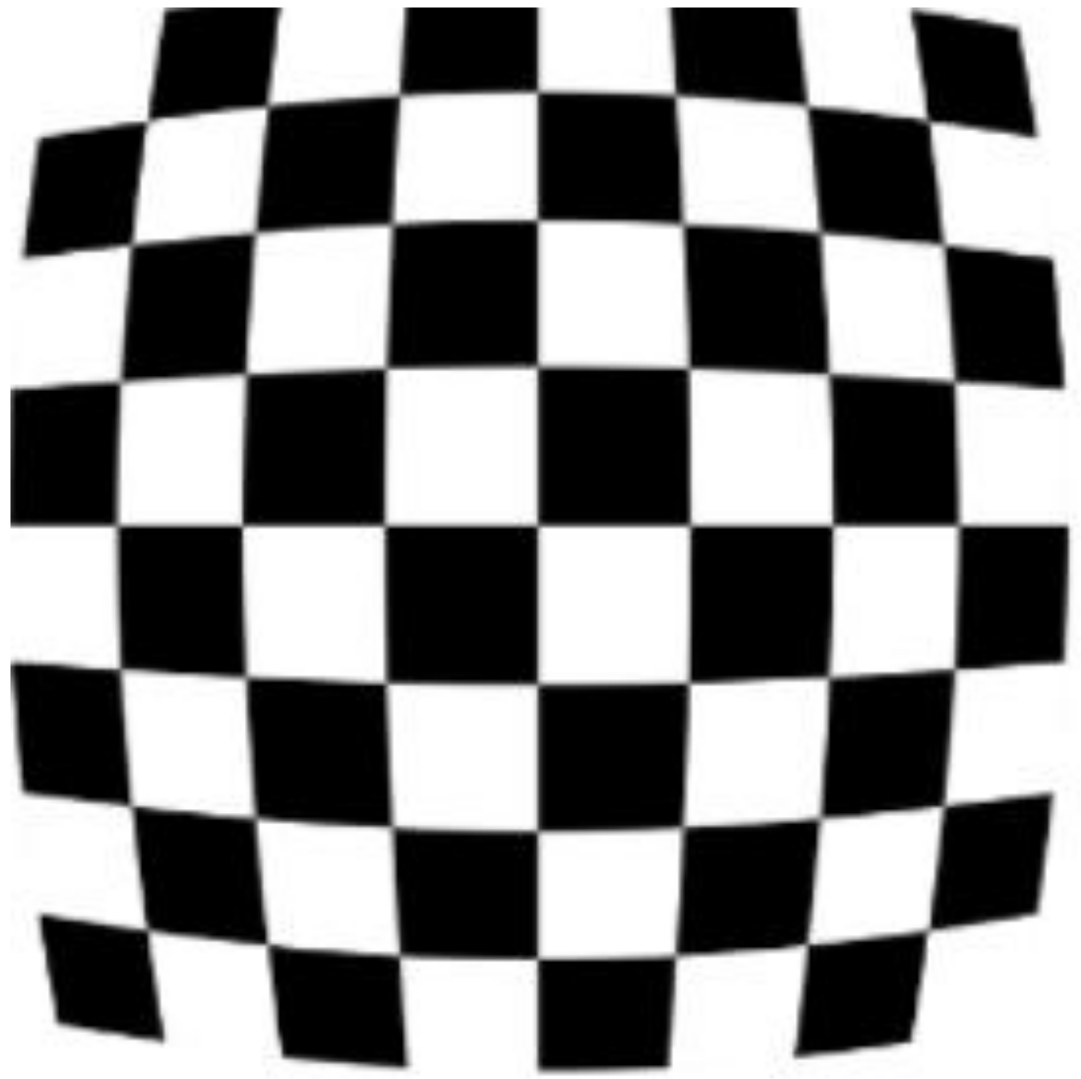
Speaker：張皓雲

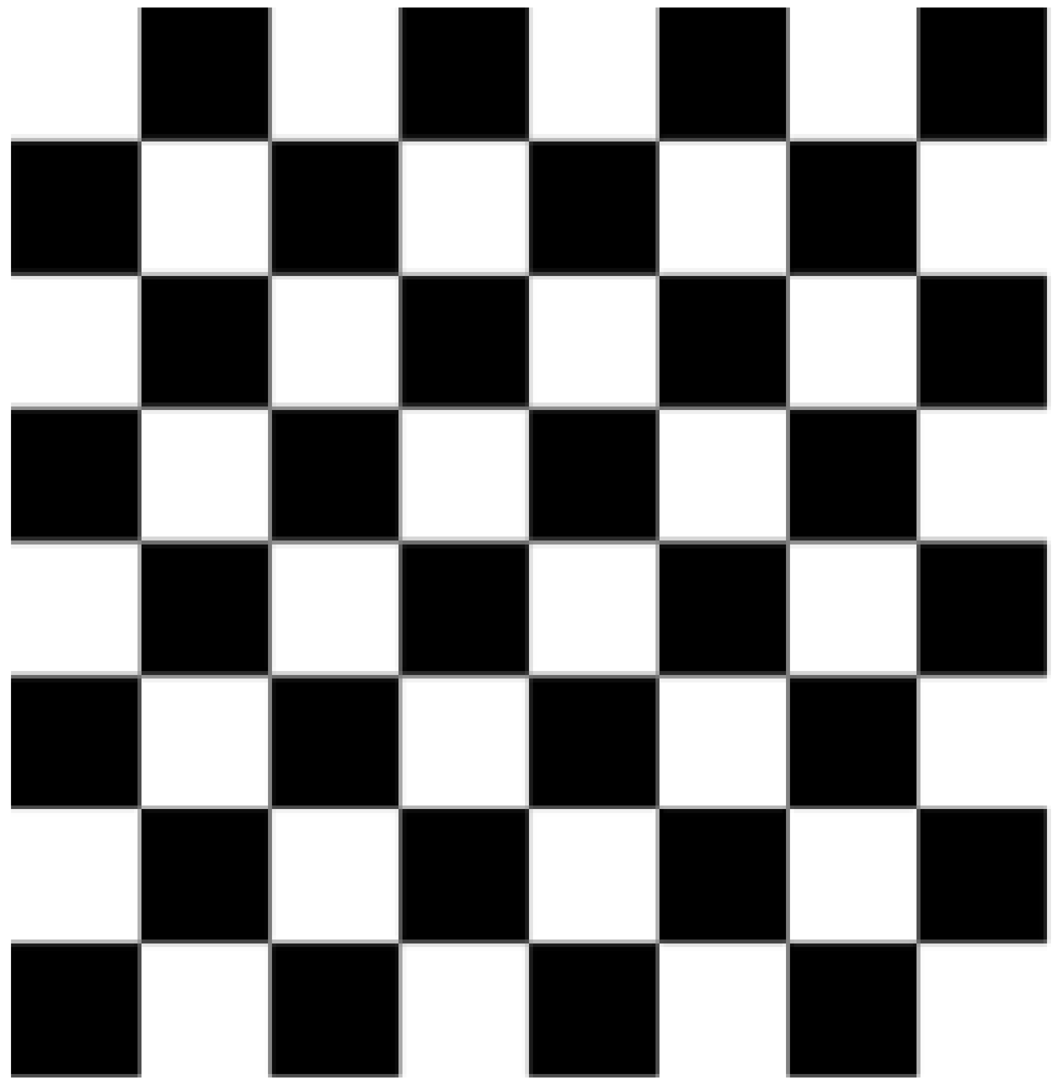
Department：資料工碩一

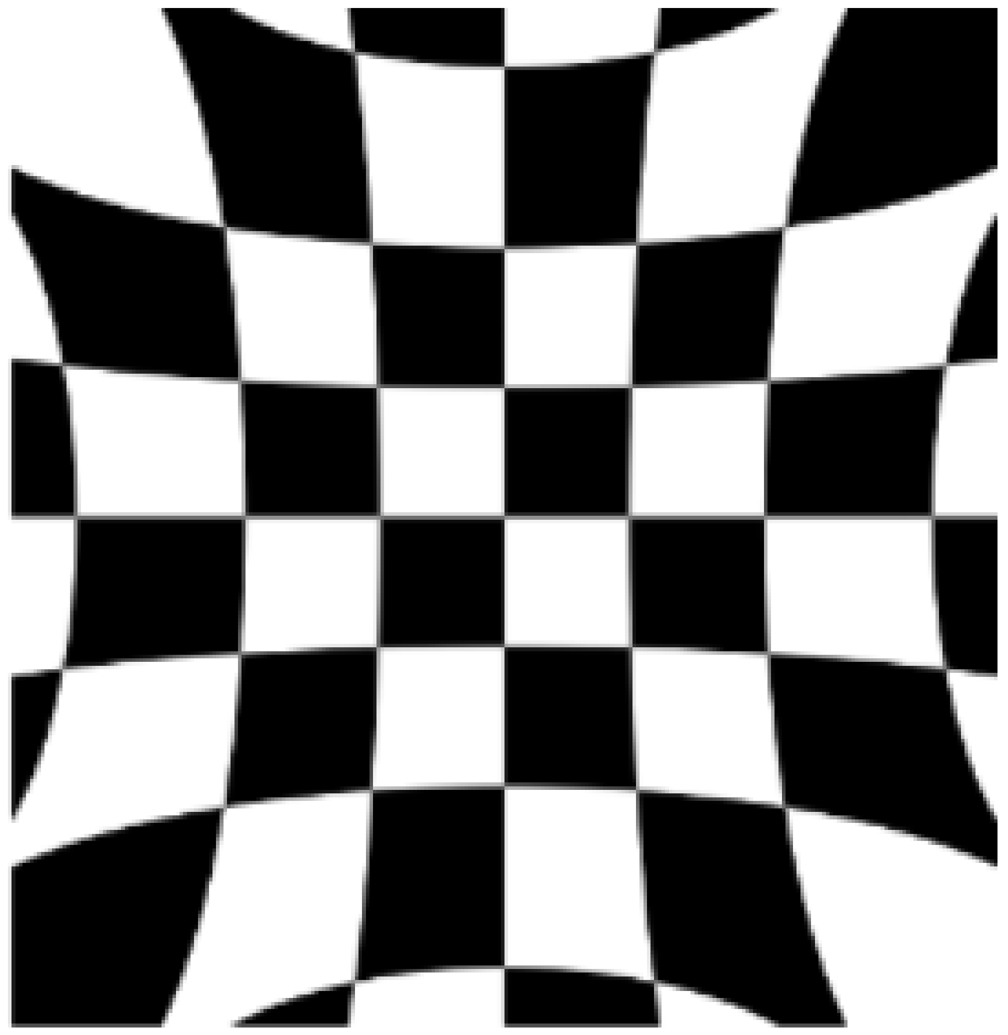
Outline



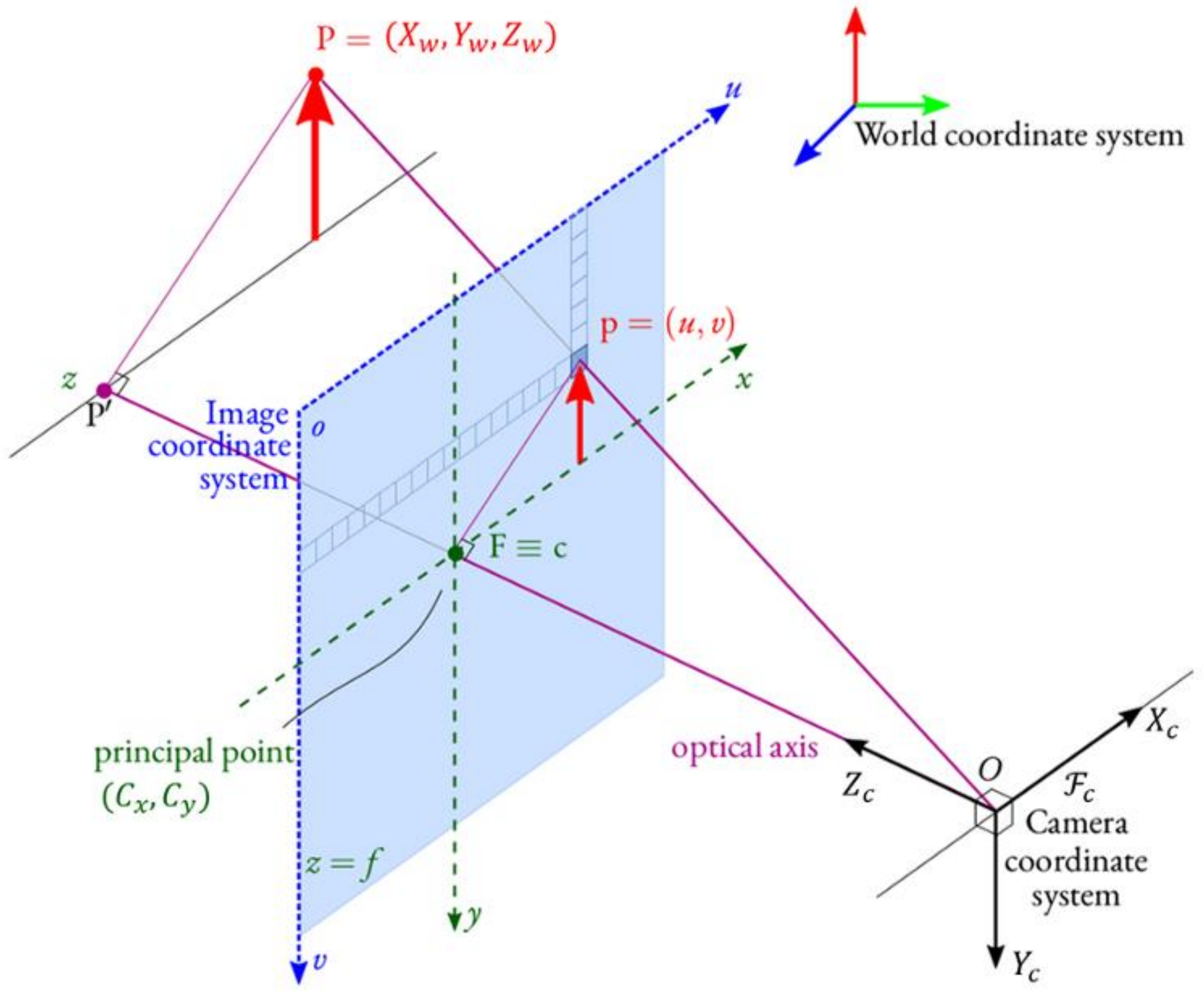
Problem to Solve


Positive radial distortion


Normal


Negative radial distortion

Problem to Solve



Problem to Solve

Distorted Coefficient

$$s\underset{\substack{\text{成像平面座標}\\ \text{Image Coordinate}}}{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}} = \underset{\substack{\text{內部參數矩陣}\\ \text{Intrinsic Parameter Matrix}}}{\begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}} \underset{\substack{\text{外部參數矩陣}\\ \text{Extrinsic parameters}}}{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}} \underset{\substack{\text{世界座標}\\ \text{World Coordinate}}}{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}} = K\,[R \mid T]\,W$$

Supplement

The camera calibration usually needs to capture real-time image on the same camera, and there are two points to be careful. This project has already deleted this part and collected the image.

1. Camera calibration needs the chessboard coordinates to appear in every corners of the screen.

2. Camera calibration requires the chessboard coordinates to be presented on the same x-y plane.

Supplement video: https://youtu.be/Bty191pcCvw

Methods Description

Image Reader	Read lots of frames from a directory.
Color Converter	Converts the image from RGB to grayscale.
Chessboard detector	The chessboard coordinates will be retrieved from the picture.
Camera Matrix Generator	The camera matrix is generated by chessboard coordinate points and real coordinate points.
Image Undistort	The image will be undistorted by camera matrix and distorted coefficient.

API Description

1. CameraCalibrate (Constructor)

https://github.com/secondlevel/Social-distance-detector/blob/main/README.md#cameracalibrate

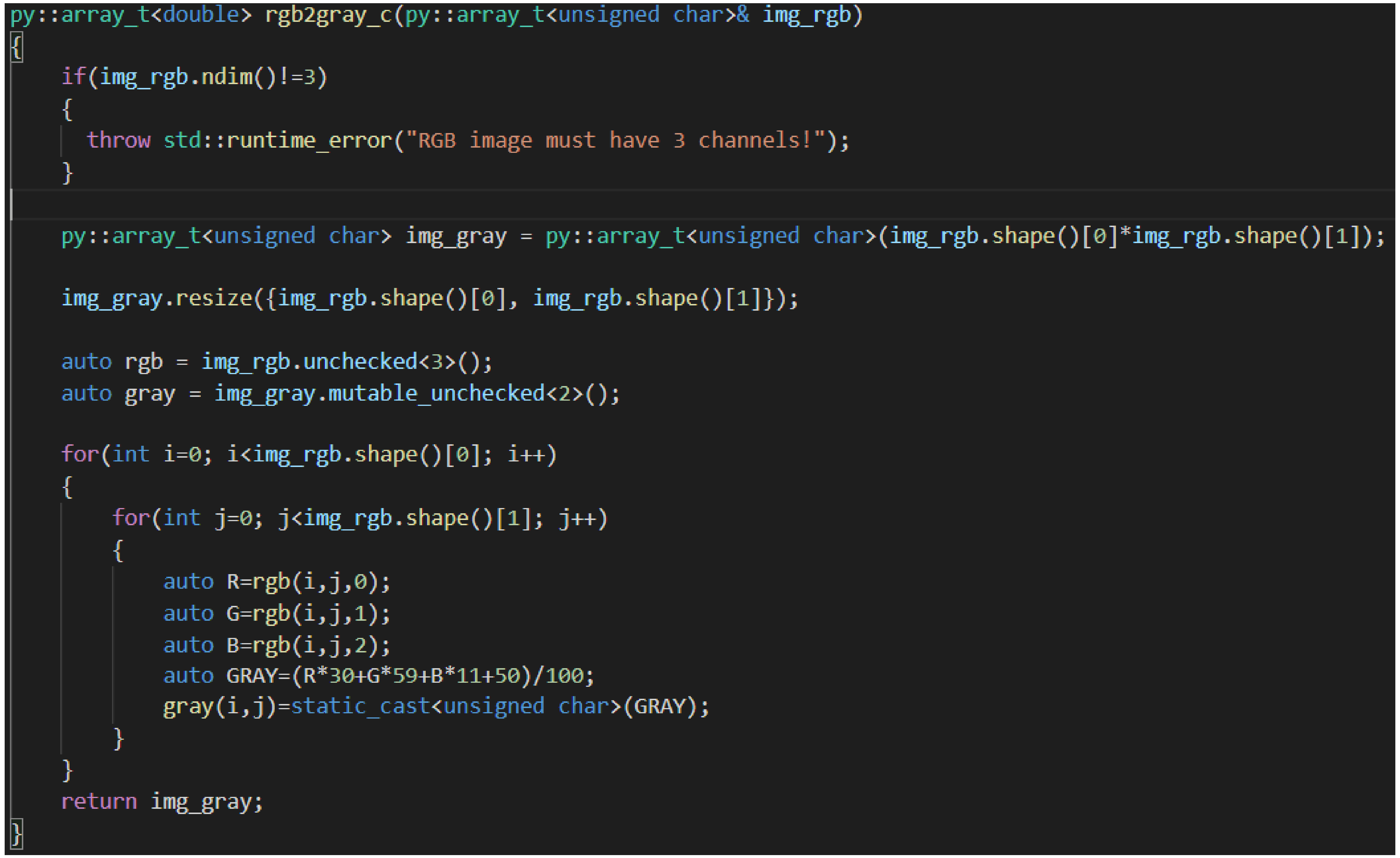
2. Additional function

$Gray = Red * 0.3 + Green * 0.59 + Blue * 0.11 + 0.5$

- BGR2GRAY(pybind::array_t)
- BGR2GRAY2(pybind::buffer_info)
- BGR2GRAY3(pybind::buffer_info and multithread)

API Description

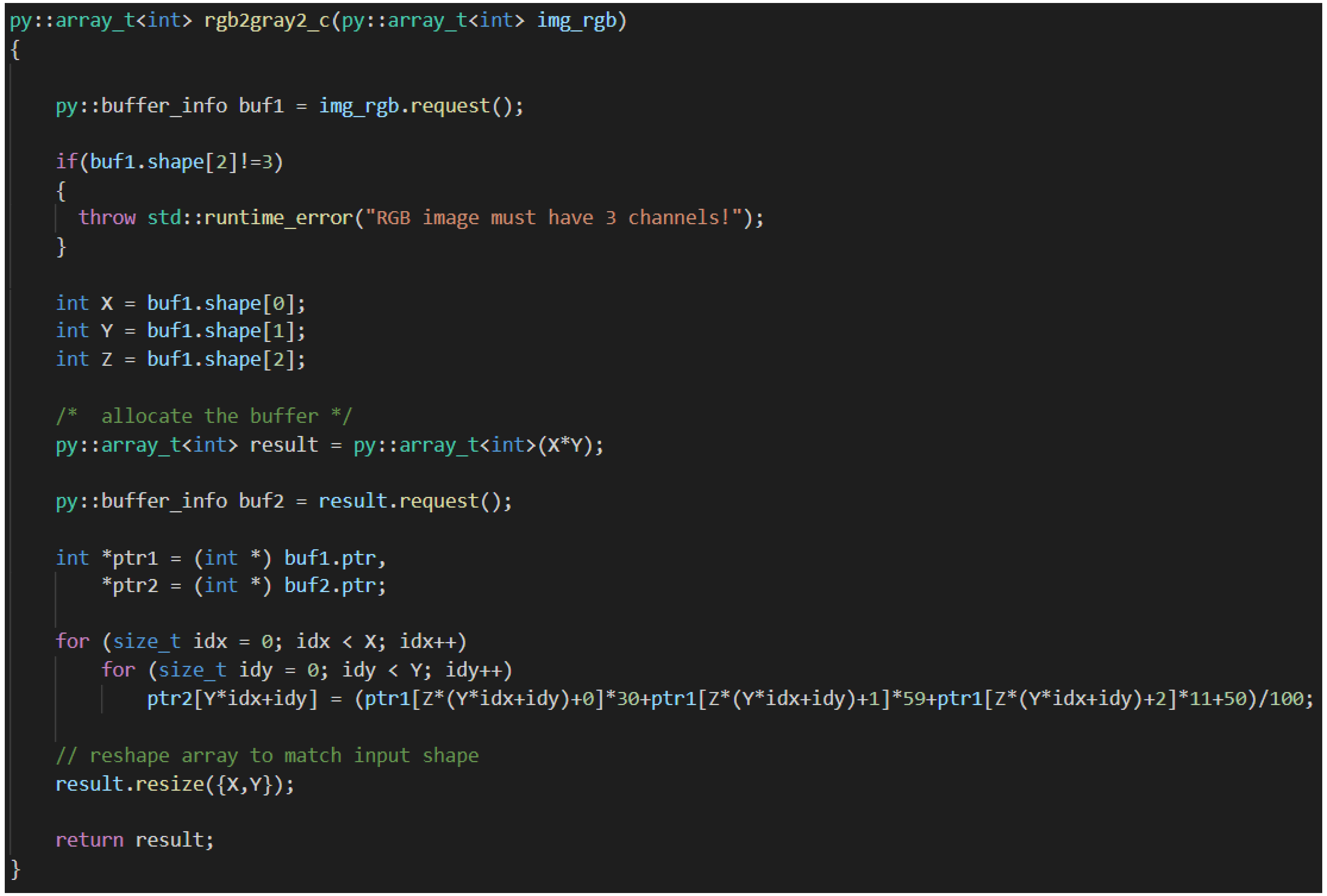
2. Additional function- BGR2GRAY(pybind::array_t)



```cpp
py::array_t<double> rgb2gray_c(py::array_t<unsigned char>& img_rgb)
{
    if(img_rgb.ndim()!=3)
    {
      throw std::runtime_error("RGB image must have 3 channels!");
    }

    py::array_t<unsigned char> img_gray = py::array_t<unsigned char>(img_rgb.shape()[0]*img_rgb.shape()[1]);

    img_gray.resize({img_rgb.shape()[0], img_rgb.shape()[1]});

    auto rgb = img_rgb.unchecked<3>();
    auto gray = img_gray.mutable_unchecked<2>();

    for(int i=0; i<img_rgb.shape()[0]; i++)
    {
        for(int j=0; j<img_rgb.shape()[1]; j++)
        {
            auto R=rgb(i,j,0);
            auto G=rgb(i,j,1);
            auto B=rgb(i,j,2);
            auto GRAY=(R*30+G*59+B*11+50)/100;
            gray(i,j)=static_cast<unsigned char>(GRAY);
        }
    }
    return img_gray;
}
```

API Description

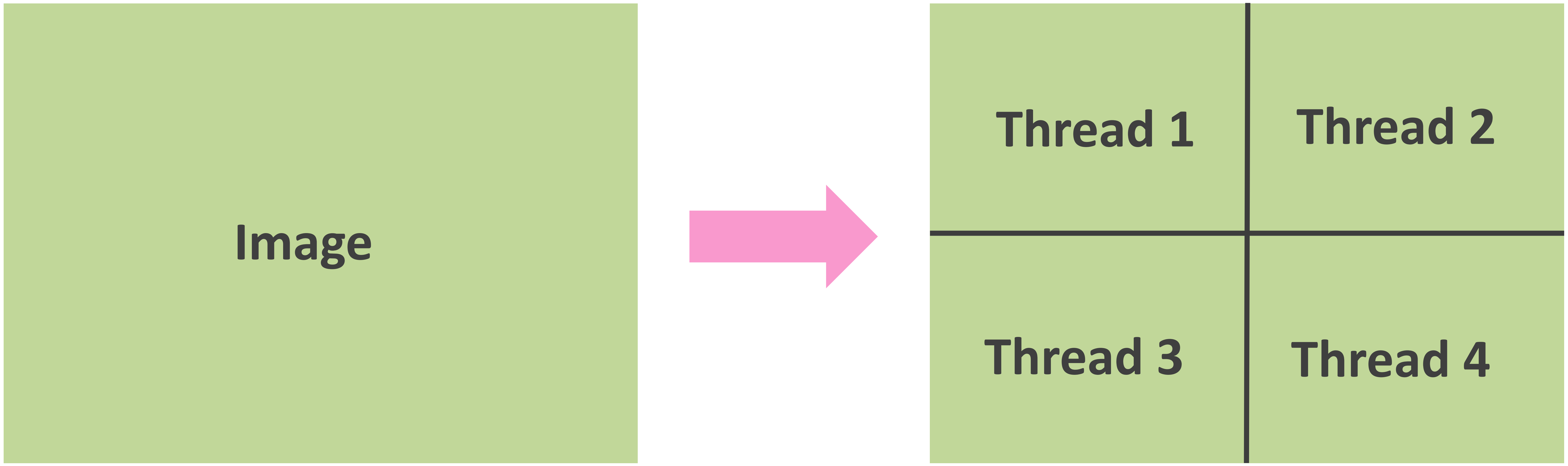
2. Additional function- BGR2GRAY2(pybind::buffer_info)



```
py::array_t<int> rgb2gray2_c(py::array_t<int> img_rgb)
{

    py::buffer_info buf1 = img_rgb.request();

    if(buf1.shape[2]!=3)
    {
      throw std::runtime_error("RGB image must have 3 channels!");
    }

    int X = buf1.shape[0];
    int Y = buf1.shape[1];
    int Z = buf1.shape[2];

    /*  allocate the buffer */
    py::array_t<int> result = py::array_t<int>(X*Y);

    py::buffer_info buf2 = result.request();

    int *ptr1 = (int *) buf1.ptr,
        *ptr2 = (int *) buf2.ptr;

    for (size_t idx = 0; idx < X; idx++)
        for (size_t idy = 0; idy < Y; idy++)
            ptr2[Y*idx+idy] = (ptr1[Z*(Y*idx+idy)+0]*30+ptr1[Z*(Y*idx+idy)+1]*59+ptr1[Z*(Y*idx+idy)+2]*11+50)/100;

    // reshape array to match input shape
    result.resize({X,Y});

    return result;
}
```

API Description

2. Additional function- BGR2GRAY3(pybind::buffer_info)



Build System

There have three build instruction to build the system.

1. make clean

Clean up the unnecessary directory such as build, .pytest,.etc.

2. make

Construct directory and tigger the cmake to build the system.

3. make test

Test the function and its performance.

Unit test

1. Camera Calibration test

- Exception Input test.
- Take the camera matrix to opencv function to undistorted the same image, and check whether the undistorted image is 50% same as the image undistorted by the function.

2. BGR to Gray test

- Exception Input test.
- Check the pixel value is same as the opencv function converted.
- Test performance with opencv function, python version function.,etc

Unit test

2. BGR to Gray test

- Test performance with opencv function, python version function and so on.

Method	1	2	3	4	5
python	1.5749 s	3.1212 s	4.7028 s	6.2647 s	7.7854 s
array_t	0.0325 s	0.0643 s	0.0955 s	0.1262 s	0.1564 s
bufferinfo	0.0084 s	0.0115 s	0.0209 s	0.0240 s	0.0291 s
bufferinfo_multithread	0.0085 s	0.0173 s	0.0254 s	0.0326 s	0.0413 s
opencv	0.0039 s	0.0085 s	0.0107 s	0.0139 s	0.0182 s

Thank you for your listening.