CITS4402
COMPUTER VISION

HOLOGRAPHIC ACQUISITION RIG AUTOMATIC CALIBRATION

ERWIN BAUERNSCHMITT, ALIAN HAIDAR, LUKE KIRKBY
22964301, 22900426, 22885101

OVERVIEW

- Automatic calibration of a holographic acquisition Rig. The purpose of this project is to implement the automatic calibration procedure for a holographic acquisition Rig. The inputs in play are a series of images taken by specially located cameras inside a room taking images of the same subject from different angles.

HOW IT IS CREATED

- The program is written in Python 3.8.5 and uses the following libraries:
- – OpenCV 4.5.1
- – Numpy 1.19.5
- – Matplotlib 3.3.4
- – Pillow 8.1.0
- – Scipy 1.6.0
- – Scikit-Image 0.18.1
- – PySimpleGUI 4.45.0
- Main framework of the program's GUI is based on Qt5 and PySimpleGUI.

HOW TO SETUP

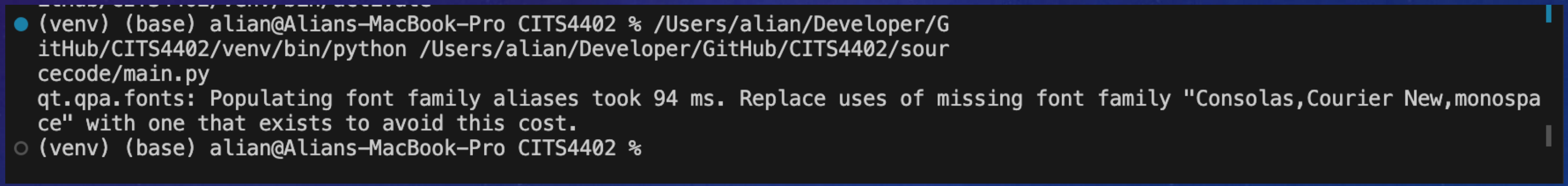
- A virtual environment is required to run the project.\
- To create a virtual environment, run the following command:

-
  ```
- python3 -m venv virtual_venv | python -m venv virtual_venv
- ```
- To install the required packages, run the following command:
- ```
- pip install -r requirements.txt
- ```
- To exit the virtual environment, run the following command:
- ```
- deactivate
- ```
- For Windows:
- Set powershell permissions to allow all bypass
- ```
- Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope Process -Force\
- .\venv\bin\activate.ps1 | .\env\script\activate.ps1
  ```

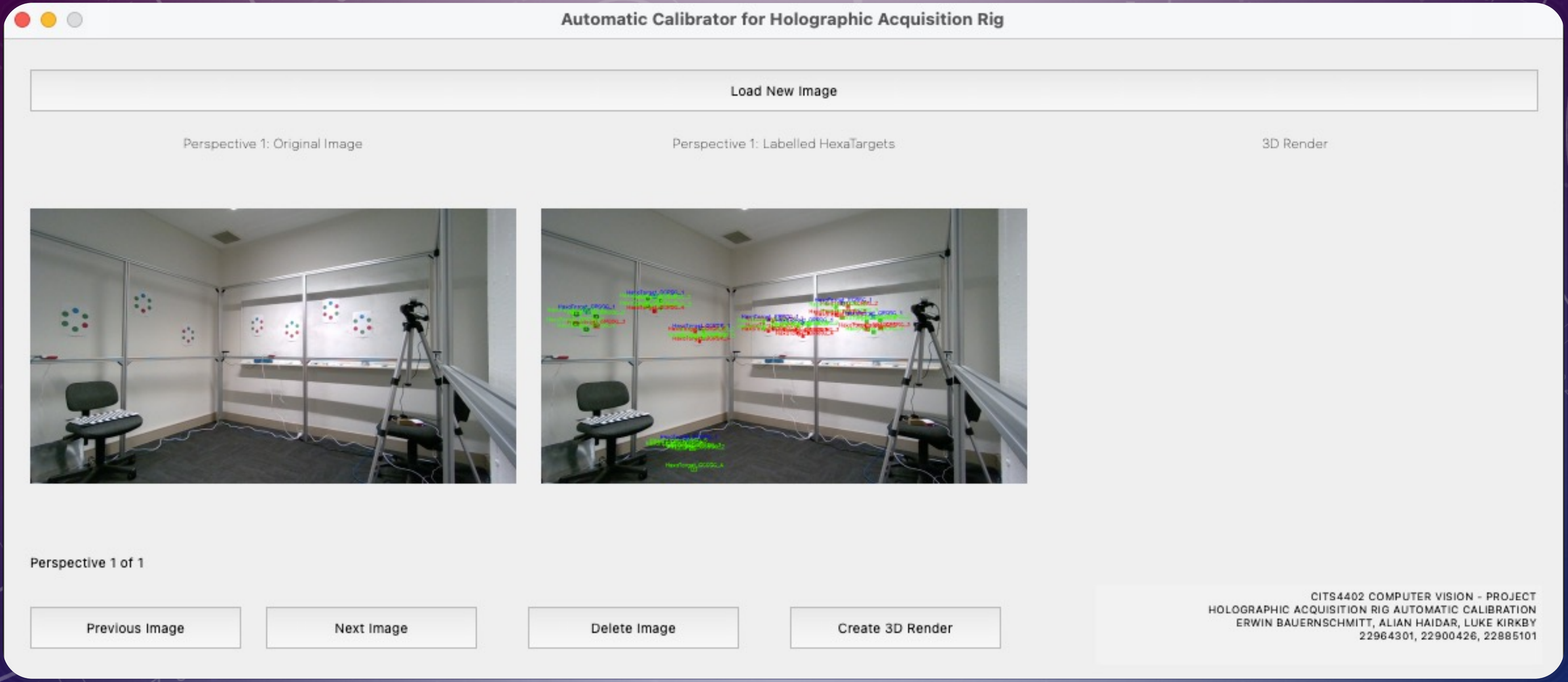
HOW TO RUN

- Assume that all dependencies are installed and terminal is on working virtual environment
- To run the program, run the following command:

-
  ```
- python3 main.py | python main.py
- ```



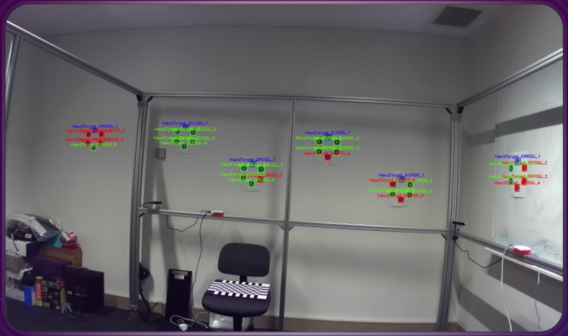
```
● (venv) (base) alian@Alians-MacBook-Pro CITS4402 % /Users/alian/Developer/G
  itHub/CITS4402/venv/bin/python /Users/alian/Developer/GitHub/CITS4402/sour
  cecode/main.py
  qt.qpa.fonts: Populating font family aliases took 94 ms. Replace uses of missing font family "Consolas,Courier New,monospa
  ce" with one that exists to avoid this cost.
○ (venv) (base) alian@Alians-MacBook-Pro CITS4402 %
```
```

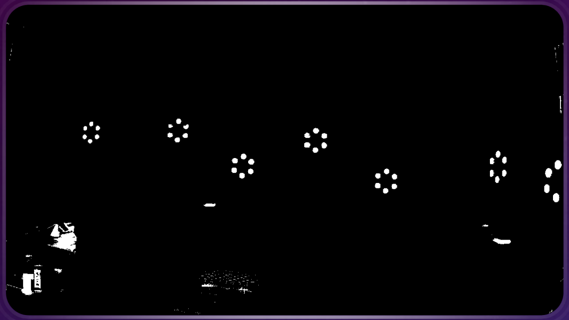

Automatic Calibrator for Holographic Acquisition Rig

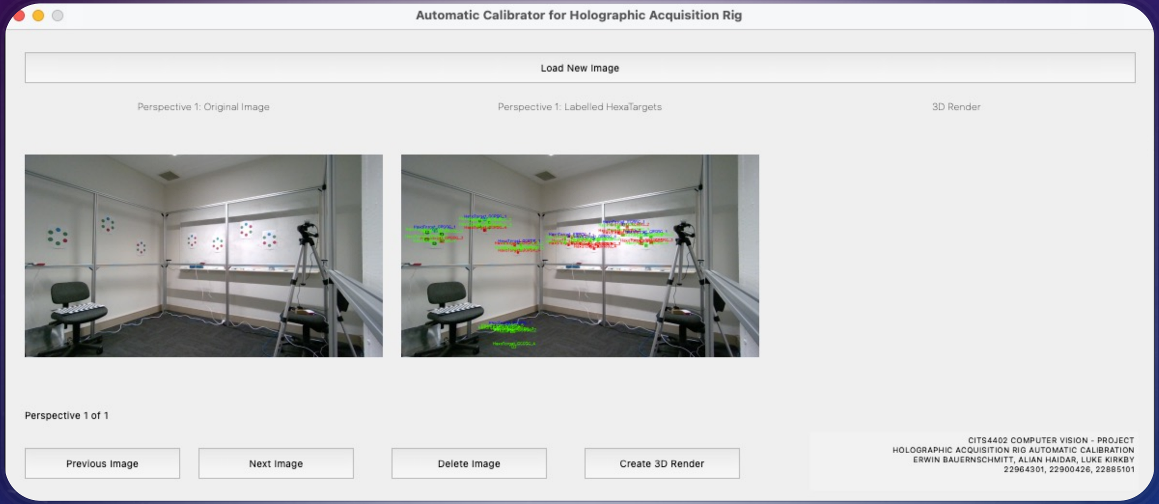
CITS4402 COMPUTER VISION - PROJECT
HOLOGRAPHIC ACQUISITION RIG AUTOMATIC CALIBRATION
ERWIN BAUERNSCHMITT, ALIAN HAIDAR, LUKE KIRKBY
22964301, 22900426, 22885101

HOW TO USE: GUI

WHEN RAN SUCCESSFULLY, A GUI WINDOW SHOULD OPEN UP






Automatic Calibrator for Holographic Acquisition Rig
Load New Image
Perspective 1: Original Image
Perspective 1: Labelled HexaTargets
3D Render
Perspective 1 of 1
Previous Image
Next Image
Delete Image
Create 3D Render

CITS4402 COMPUTER VISION - PROJECT
HOLOGRAPHIC ACQUISITION RIG AUTOMATIC CALIBRATION
ERWIN BAUERNSCHMITT, ALIAN HAIDAR, LUKE KIRKBY
22964301, 22900426, 22885101

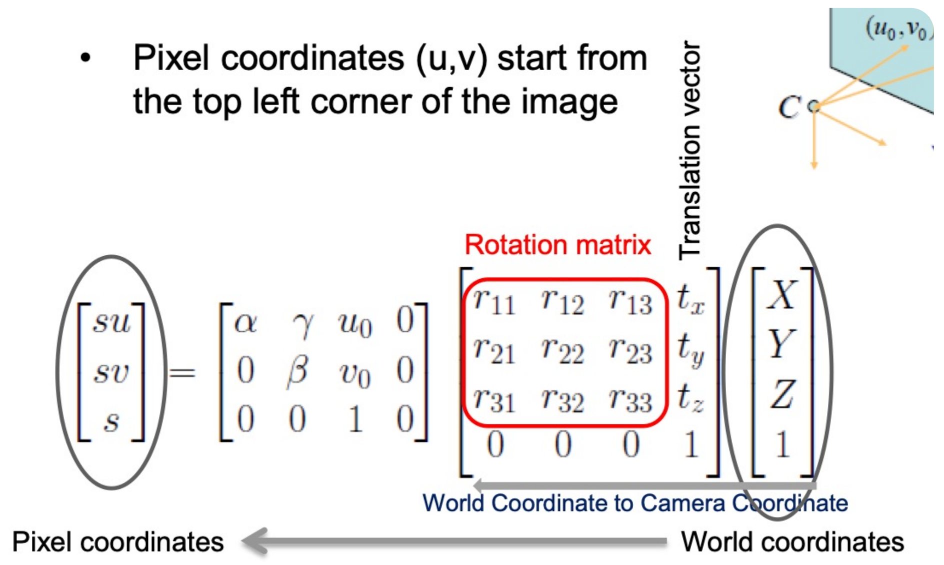
HOW TO USE: GUI

- When image loaded successfully, the original image will be shown on the left-hand side, whilst a coordinated pixel selected HexaTargets will appear on the middle

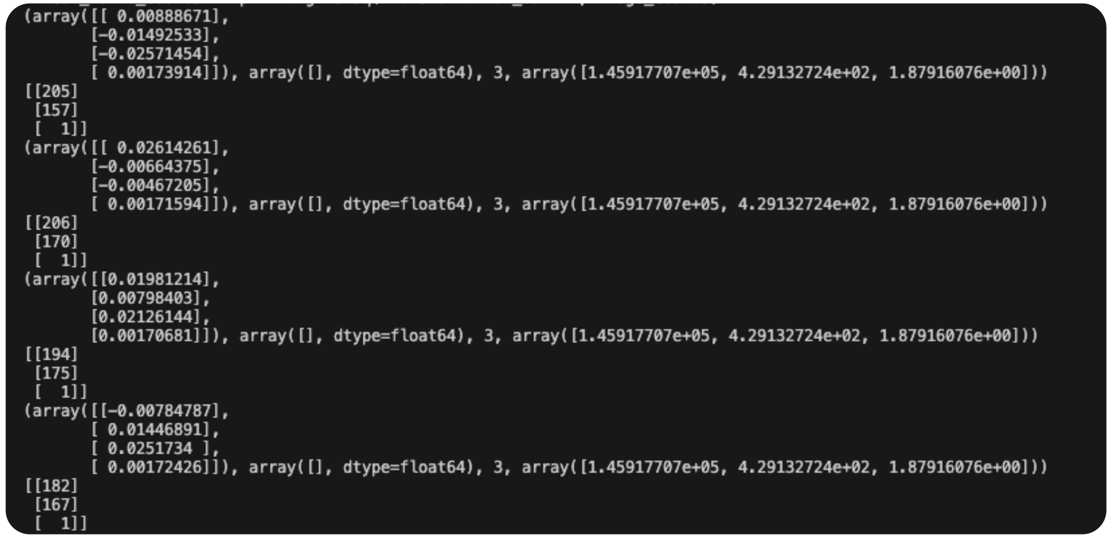
HOW TO USE: GUI

- The three panels are:
- **Perspective 0: Original Image** – Shows the original Image.
- **Perspective 0: Labelled HexaTargets** – Shows the location of the RGB dots on the screen.
- **3D Render** – This panel displays the 3D render of the selected images.
- When all images are selected successfully, it will ask the user to import the selected JSON files specific to the camera image used. This will import all calibration data, such as the coordinates of the RGB dots, the camera matrix, the distortion coefficients, the rotation and translation vectors.
- Once all the images are loaded and the JSON files are imported, the user can now select the image that they want to calibrate. The user can scroll through the images using the **Previous Image** and **Next Image** buttons. The user can also delete the selected image using the **Delete Image** button.

HOW TO FIND CAMERA PROPERTIES


• Pixel coordinates (u,v) start from the top left corner of the image
Translation vector
(u_0, v_0)
C
Rotation matrix
World Coordinate to Camera Coordinate
Pixel coordinates
World coordinates

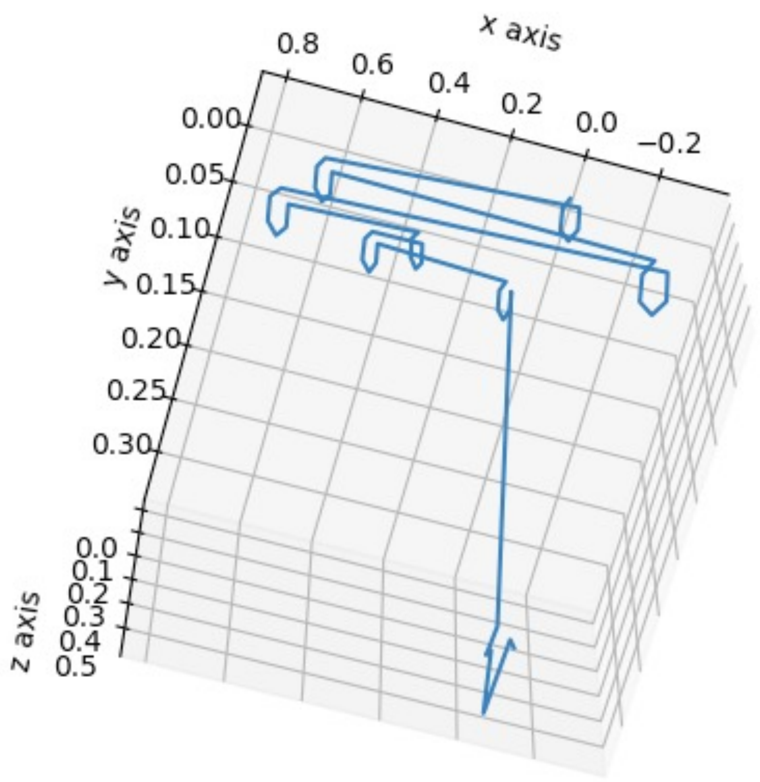
$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



```
(array([[ 0.00888671],
        [-0.01492533],
        [-0.02571454],
        [ 0.00173914]]), array([], dtype=float64), 3, array([1.45917707e+05, 4.29132724e+02, 1.87916076e+00]))
[[205]
 [157]
 [  1]]
(array([[ 0.02614261],
        [-0.00664375],
        [-0.00467205],
        [ 0.00171594]]), array([], dtype=float64), 3, array([1.45917707e+05, 4.29132724e+02, 1.87916076e+00]))
[[206]
 [170]
 [  1]]
(array([[0.01981214],
        [0.00798403],
        [0.02126144],
        [0.00170681]]), array([], dtype=float64), 3, array([1.45917707e+05, 4.29132724e+02, 1.87916076e+00]))
[[194]
 [175]
 [  1]]
(array([[-0.00784787],
        [ 0.01446891],
        [ 0.0251734 ],
        [ 0.00172426]]), array([], dtype=float64), 3, array([1.45917707e+05, 4.29132724e+02, 1.87916076e+00]))
[[182]
 [167]
 [  1]]
```

- Using SolvePNP, we can discover R-vector and T-vector
- Able to get camera matrix from JSON file
- Able to solve the system of linear equations relating 2-D and 3-D points

3D PLOT



- Once an image and respective JSON file was uploaded to the program, and the 3D render button was clicked, it will generate a new window of MatPlot3d that shows all points and camera in 3d space
- The image on the left shows all 7 points with Camera 71 as the initial point