Name: Manasa Rayachoti
E-mail: manasa.r.2811@gmail.com
Mobile: 8105233239

ASSIGNMENT- ML Level -1

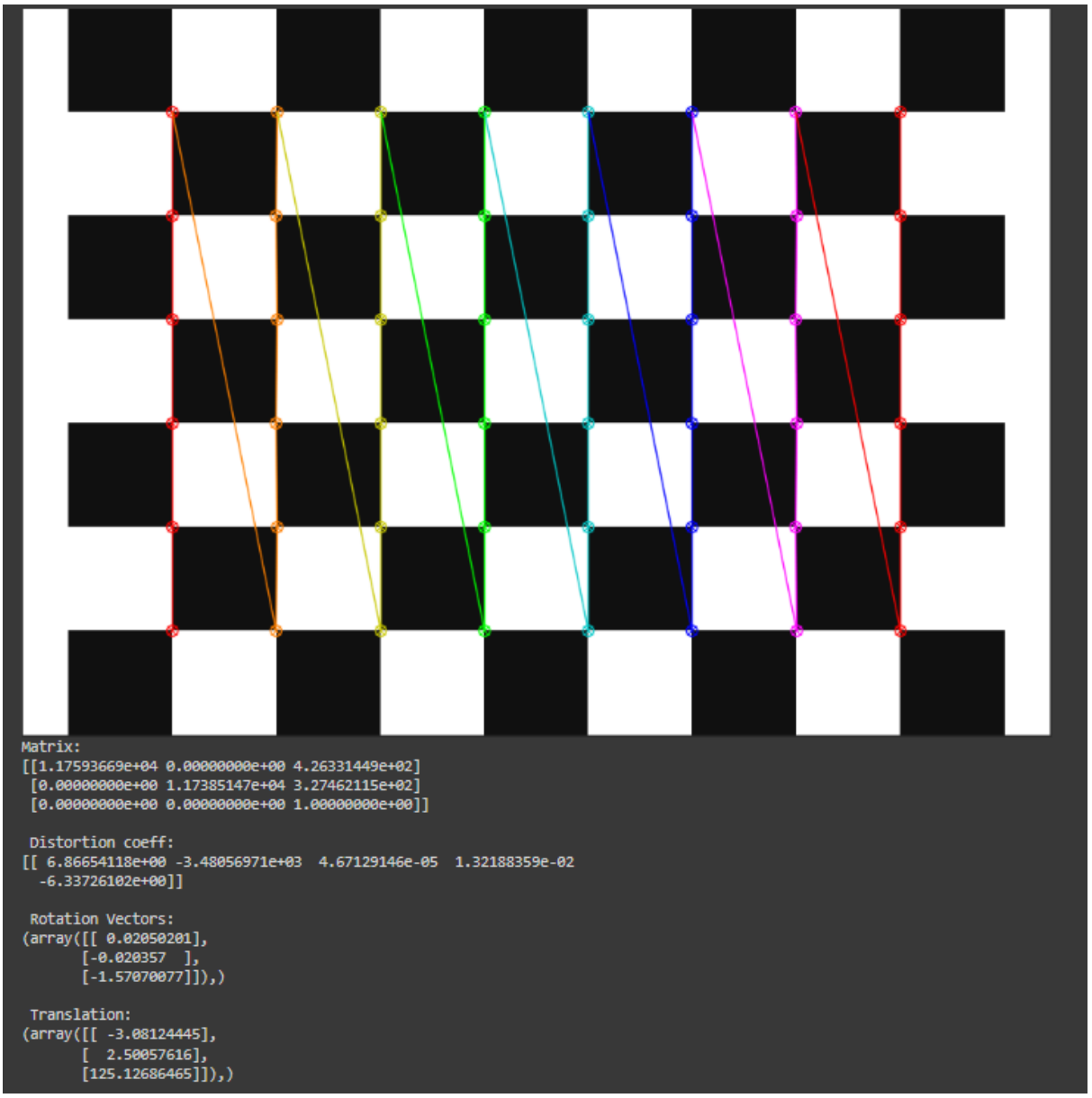
1. Installing OpenCV:

```
[1] !pip install opencv-python

    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
    Requirement already satisfied: opencv-python in /usr/local/lib/python3.8/dist-packages (4.6.0.66)
    Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.8/dist-packages (from opencv-python) (1.21.6)
```

Code:

```
[10] import cv2
     import numpy as np
     import os
     import glob
     from google.colab.patches import cv2_imshow


     CHECKERBOARD = (6, 8)

     criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
     threedpoints = []
     twodpoints = []
     objectp3d = np.zeros((1, CHECKERBOARD[0]
                           * CHECKERBOARD[1],
                           3), np.float32)
     objectp3d[0, :, :2] = np.mgrid[0:CHECKERBOARD[0],
                                    0:CHECKERBOARD[1]].T.reshape(-1, 2)
     prev_img_shape = None

     images = glob.glob('/content/Checkerboard.PNG')

     for filename in images:
         image = cv2.imread(filename)
         grayColor = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

         ret, corners = cv2.findChessboardCorners(
                         grayColor, CHECKERBOARD,
                         cv2.CALIB_CB_ADAPTIVE_THRESH
                         + cv2.CALIB_CB_FAST_CHECK +
                         cv2.CALIB_CB_NORMALIZE_IMAGE)


         if ret == True:
             threedpoints.append(objectp3d)


             corners2 = cv2.cornerSubPix(
                 grayColor, corners, (11, 11), (-1, -1), criteria)

             twodpoints.append(corners2)

             image = cv2.drawChessboardCorners(image,
                                               CHECKERBOARD,
                                               corners2, ret)

         cv2_imshow(image)
         cv2.waitKey(0)

     cv2.destroyAllWindows()

     h, w = image.shape[:2]

     ret, matrix, distortion, r_vecs, t_vecs = cv2.calibrateCamera(
         threedpoints, twodpoints, grayColor.shape[::-1], None, None)


     print("Matrix:")
     print(matrix)

     print("\n Distortion coeff:")
     print(distortion)

     print("\n Rotation Vectors:")
     print(r_vecs)

     print("\n Translation:")
     print(t_vecs)
```

Output:



2. Checkerboard pattern is the most used pattern for camera calibration. This is because the planar grid structure makes sure that the control points can be the corners within the board.The calibrator assigns x and y direction. The longer side is x- direction. The distance between the detected points and reprojected points is looked at. This method is widely used as it gives high accuracy.
   Another method that can be used is Field Camera Calibration.

3. Implementation of single linked list:

```c
#include<stdlib.h>
#include<stdio.h>

struct Node{
    int data;
    struct Node *next;
};

void add(struct Node** head, int data){

    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = *head;
    *head = newNode;
    printf("Added %d\n",newNode->data);
}

void deletePos ( int pos)
{
    struct Node* head;
    struct Node* temp = head;
    int i;
    if (pos == 0) {
        printf("\nElement deleted is : %d\n", temp->data);
        head = head->next;
        temp->next = NULL;
        free(temp);
    }
    else {
        for (i = 0; i < pos - 1; i++) {
            temp = temp->next;
        }


        struct Node* del
            = temp->next;
        temp->next = temp->next->next;
        printf("\nElement deleted is : %d\n", del->data);
        del->next = NULL;
        free(del);
 }
}
```

```c
void deleteAll()
{
    struct Node *temp;


    struct Node *head;
    while(head != NULL)
    {
        temp = head;
        head = head->next;

        free(temp);
    }

    printf("SUCCESSFULLY DELETED\n");
}

void display(struct Node* node)
{
    printf("Linked List: ");
    while(node!=NULL){
        printf("%d ",node->data);
        node = node->next;
    }

    printf("\n");
}
```

```c
int main()
{
    struct Node* head = NULL;


    add(&head,100);
    add(&head,80);
    add(&head,60);
    display(head);

    deletePos(1);
    deleteAll();
    display(head);

    return 0;
}
```