Image Based Visual Servoing with a Handheld Monocular

Hamish McEwan – Sarah Abadir – Cameron Gordon

Code Function:

1st section:

The code was split into three major sections, the 1st section utilised camera parameters taken from a pre calibrated camera and completed calculations of the X, Y and Z variance from the set origin position.

This section of the code calculated the change in distances between the detected checkerboard points in combination with the focal length parameters to estimate a Z axis value.

2nd section:

The second 2nd section of the code is the main running loop for visual servoing, this section of the code utilizes a while loop to take a picture using the onboard webcam every 0.01 seconds. The algorithm utilises the detectcheckerboard points function to detect the points of the pattern.

Using the data from the 1st section of the code, the 2nd section then calculates the relation of the average checkerboard points position (checkerboard centre) and a user defined acceptable error. When the checkerboard location falls within the acceptable location values (X,Y,Z) the user interface displaying the values turns green and displays a message to inform the user.

The 2nd section of the code deals with the overlay for the checkerboard points, data values on the figure and the guidance arrow. The guidance arrow is a calculated vector from the original position to the current average position of the checkerboard.

3rd section:

The 3rd section of the code opens a predesignated folder of checkerboard images and runs a calibration process to determine the intrinsic and extrinsic parameters. The algorithm then uses this data to determine a series of world points and the board size of the checkerboard.

Following the calibration phase, the program then activates the inbuilt webcam and begins to run a while loop like section 2, a photo is taken every 0.5 seconds of a checkerboard, and the points are detected. Rather than displaying a live camera feed, a plot is displayed with x, y and z axis in a 3d simulation of the camera pose. These updates live to a figure allowing the user to see in real time the simulated pose of the camera.

All these sections are contained as functions within a GUI class created with the MATLAB App Designer. This allows the user to open the GUI and easily operate the system with the intuitive display.