Individual Report

Visual Servoing Using Camera.

Shobhit Gumber (12143424)

Contents

Scope of Project 2

Updated Deliverables: 2

Methodology 2

Approaches 3

Code Flowchart 5

Challenges and Outcome 5

Conclusion 6

References: 6

Appendix 7

Appendix 1.1 (GANTT Chart of Project) 7

Appendix 1.2 (Large Angle Algorithm Output) 7

Appendix 1.3 (Small Angle Algorithm Output) 8

Scope of Project

Requirements:

The requirements of out project included the use of MATLAB and the computer vision toolbox (to access the camera) to be integrated with visual servoing. The requirement also dictates to use an algorithm which allows the current pose of the pattern to give directions to obtain the reference pose. The reference pose was decided to be in the centre of the camera frame with positions in 3d.

In relation to the requirements, a handheld camera could not be obtained due to COVID-19. The suppliers estimated a month for the camera to be delivered to us. To save both time and money, a collective group decision was taken to use webcams. The theory behind the camera would remain the same and would still work with a handheld camera.

Updated Deliverables:

1. Select Camera
2. Calibrate Camera and determine its capabilities
3. Research and choose preferred algorithm for visual servoing
4. Download relevant MATLAB Toolboxes on PC: ensures that bugs and codes can be tested and eliminated.
5. Design pattern to perceive desired position on the image
6. Mid-review report – 10th May 2020
7. Output camera image live to MATLAB
8. Design pattern
9. Identify desired position of camera in relation to the pattern
10. Determine, if any, the degree of human error allowed in adjusting camera position
11. Create the method of output for control instructions
12. Test and troubleshoot
13. Record video of demo of code
14. Project submission – 7th June 2020
15. Summarise and format documentation into presentation
16. Format documentation into report
17. Project presentation – 9th June 2020
18. Final Report – 10th June 2020

Methodology

How does the system work?

The system uses MATLAB toolboxes such as the Computer Vision Toolbox to ensure the external/internal webcam can be used with the algorithm. It allows the system to take a screenshot to allow the algorithm to run through it and make appropriate changes. The system uses a pinhole camera method and also uses end effector mounted or eye in hand visual servoing. These things together along with functions produced in MATLAB create velocities in three dimensions to allow the used to change the current pose and directing them to the reference pose.

How does the code work?

In order for the code to work the camera being used needs to be calibrated. Parameters such as the principle point, local length and the camera resolution need to be determined. Furthermore, an ideal z value needs to be inputted as depth is an issue with monocular camera. Furthermore, parameters of the detectCheckerboard function such as sensitivity are also adjusted in order to ensure operation in different lighting conditions. The desired coordinates of the checkerboard and other coordinates are also inputted in order to maintain uniformity and robustness.

Using the detectCheckerboard function within the loop ensures that the checkerboard is detected within the live feed. When in the loop the webcam takes a snapshot image from the webcam and converts it into grayscale.

After the image has been converted into grayscale a function scans the input image for checkerboard corners at two different levels of details. The function then outputs image points and the size of the board. An if statement is used if the checkerboard is found and increments the counter. This generates coordinates using the checkerboard size and the size of the squares. Using the coordinates, the coordinates of the centre of the checkerboard are calculated.

These points are then plotted within the image that is being used. Different colours are used in order to ensure that the correct corner is being displayed.

The coordinates are used to calculate distances between the camera and the reference image coordinate excluding the rotation. Another function then is used to estimate z distances of corner diagonals and ideal distance between centre and corner point of the checkerboard. The function also calculates the rotation relative to the camera and uses the rotation to calculate more effectively the distance between the camera and each corner of the checkerboard. The function then outputs z distances of corner diagonals to one another with reference to the rotation of the checkerboard. Then these coordinates are plotted on the image.

Global coordinates are subsequently calculated using the 4 corners and the centre. The error between the desired and the current position of the corners are also determined. Lamda was used to reduce the error by 10% to make the calculations more accurate. The Jacobean features are calculated using LX functions and interaction matrixes. This allows us to calculate the velocity and angular velocities in the 3 dimensions. These are inputted in the GUI to help the user reach the desired pose.

Approaches

Original Approach

Due to COVID-19, originally, we delegated tasks within our groups to work individually. Each member was required to do their specific part. An extra two week was given to compile each task to integrate with each other. As everyone finished their tasks, we realised that certain features will not work with the pattern chosen. Using a checkerboard, it was found that certain features of the image if symmetrical would be mismatched with other features using surf detection. The surf detection will use image based visual servoing to determine calculations for velocities. Furthermore, if mismatched, all the calculations and velocities produced would be highly inaccurate. The interaction matrix will be extremely unreliable, and the project will have failed. After careful consideration from each group member it was decided that this approach will be highly effective and decided to use the different approaches to complete the project within the specified timeframe.

Alternate Approach

After researching, it was found that the detectCheckerBoard function was the best and easily available feature that could be used in our project. Furthermore, it was found to be extremely reliable and efficient. Alternate methods that were discussed by the groups included Image based visual servoing which generates errors directly from the image frame. In Image based visual servoing the error signal is dictated by the image feature parameters. Another possible alternate method that was discovered included the use of position based visual servoing. Position based visual servoing uses visual data to reconstruct a 3d pose of the object and generates an error in Cartesian space and generates commands. After conducting research, it was also found that position based had advantages such as defining tasks within the cartesian frames. However, it was also found that the control law strongly depended on optical parameters of the camera and is also highly susceptible to calibration errors. Whereas, image based visual servoing is less susceptible to calibration. A problem encountered with image based visual servoing included the onboard calculation of the image Jacobean which is dependent on the distance between the camera and the target.

From the figure below it can be seen that PBVS (position based visual servoing) and IBVS(Image based visual servoing) work better at a slower speed as it gives the system to be given enough time to the algorithm to catch up. The following figure is from a study that describes the difference of each technique and how it affects the overall outcome. It can be seen in PBVS that when the slow mode is used the measured to actual coordinates is overall lower than IBVS slow and fast modes.

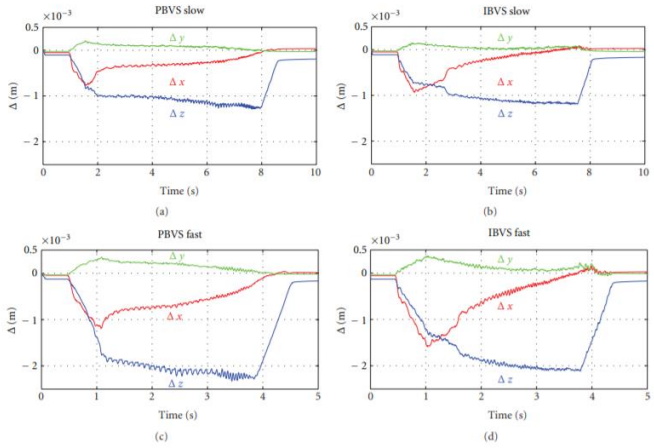
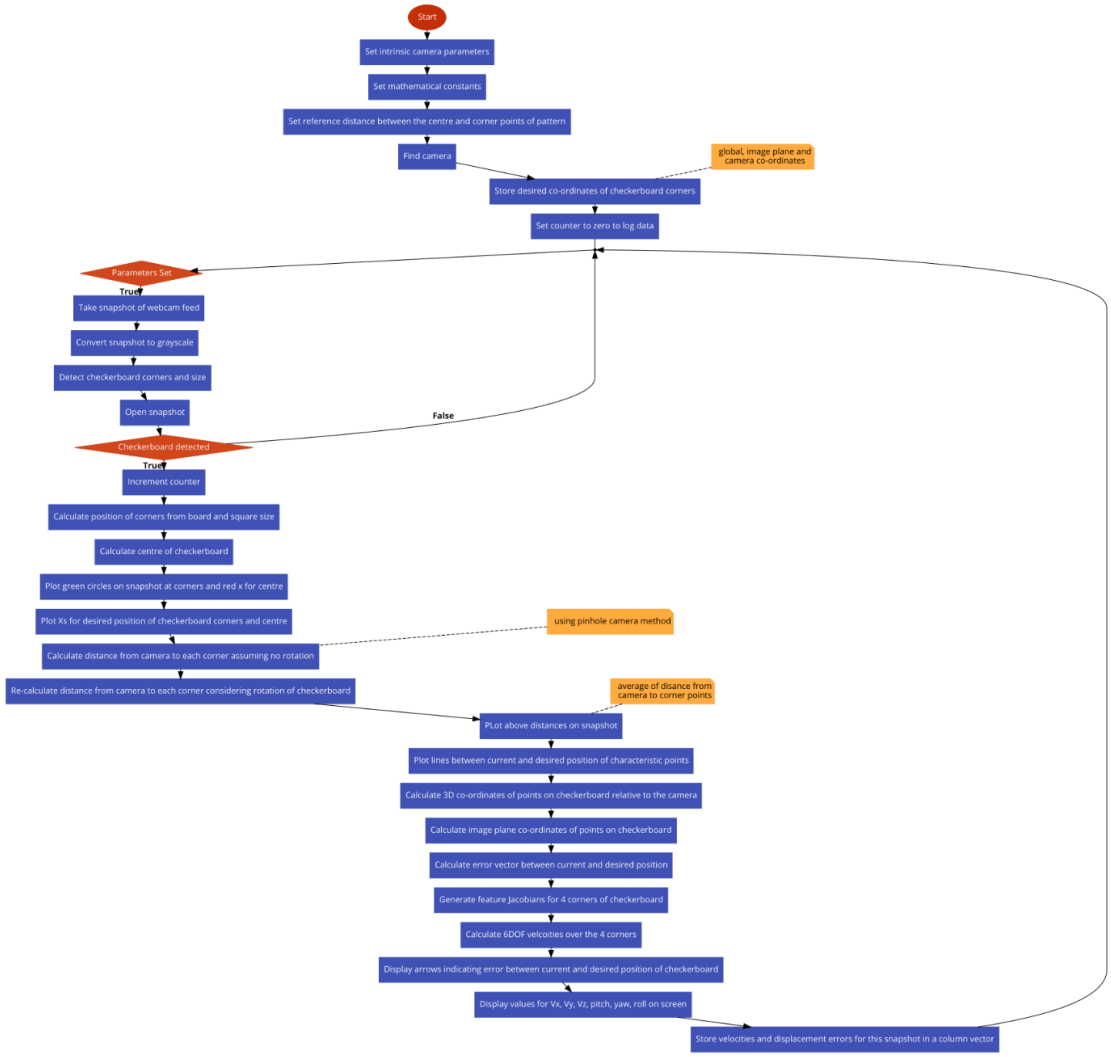


Figure1: Error between planned and measured coordinates from PBVS and IBVS in fast vs slow mode [1]

Justification of Final Approach

As described above PBVS was the most accurate in slow mode. As a reason a variation of both PBVS and IBVS was chosen in order to compute the image in real time and also providing reasonable errors. The image based visual servoing was used to produce a Jacobean feature which outputted velocities and IBVS was also used to produce an estimated position of each feature.

Code Flowchart



Challenges and Outcome

There were many challenges faced during this project. However, most of them were overcome by collaboration and research into various topics and articles. A few of theses problems are listed below.

Angles

One of the main concerns in our assignment was the angles produced being too high will generate inaccurate distances. This was a major concern going while testing and troubleshooting. We overcame this by calculating the actual centre point and not only the perceived centre point using trigonometry. An image in Appendix 1.2 shows the algorithm output with a large angle compared to Appendix 1.3 which shows the algorithm output with a small angle.

Lighting

Another major concern was the lighting under which the testing will take place. It was a fear that if the lighting is bad, the detectCheckerboard function will not work. After researching on the internet and adjusting the sensitivity of the camera we tested the algorithm under different lighting conditions to ensure the algorithm was robust and effective.

Computational Power

Another challenge faced included the computational power from each computer. Sometimes, my laptop froze and had to be restarted due the excessing CPU power required by MATLAB and its plugins. Furthermore, when the algorithm ran, we were roughly running the program round 5 frames per second due to a number of calculations running at the same time. Due to time constraints more modifications could be made in order to save memory from CPU processes.

Camera

Another concern that was feared by each member included our algorithm not working with the monocular camera. Due to COVID 19, an external camera could not be acquired and therefore needed to be tested on webcams whilst moving the pattern around. However, we all felt comfortable with the robustness of the algorithm and its affect on an external camera. The theory will still be effective in an external camera.

Conclusion

In conclusion, I learned a lot about image processing and visual servoing. Furthermore, the challenges faced by the team will help me in the future as more considerations need to be made when using algorithms. These include the CPU processing power of the visual servoing machine. Most robots, including dobot have memory constraints and processing constraints. These need to be considered before allocating memory and running the algorithm. In addition, lighting conditions and other parameters also need to be considered to ensure that the project is compatible with other machines with different specs.

References:

[1] Palpacelli, M., 2016. *A Comparison Between Position-Based And Image-Based Dynamic Visual Servoings In The Control Of A Translating Parallel Manipulator | Emarefa*. [online] Search.emarefa.net. Available at: <https://search.emarefa.net/detail/BIM-446635/a-comparison-between-position-based-and-image-based-dynamic-visual-servoings-in-the-control-of-a-translating-parallel-manipulator/1> [Accessed 26 May 2020].

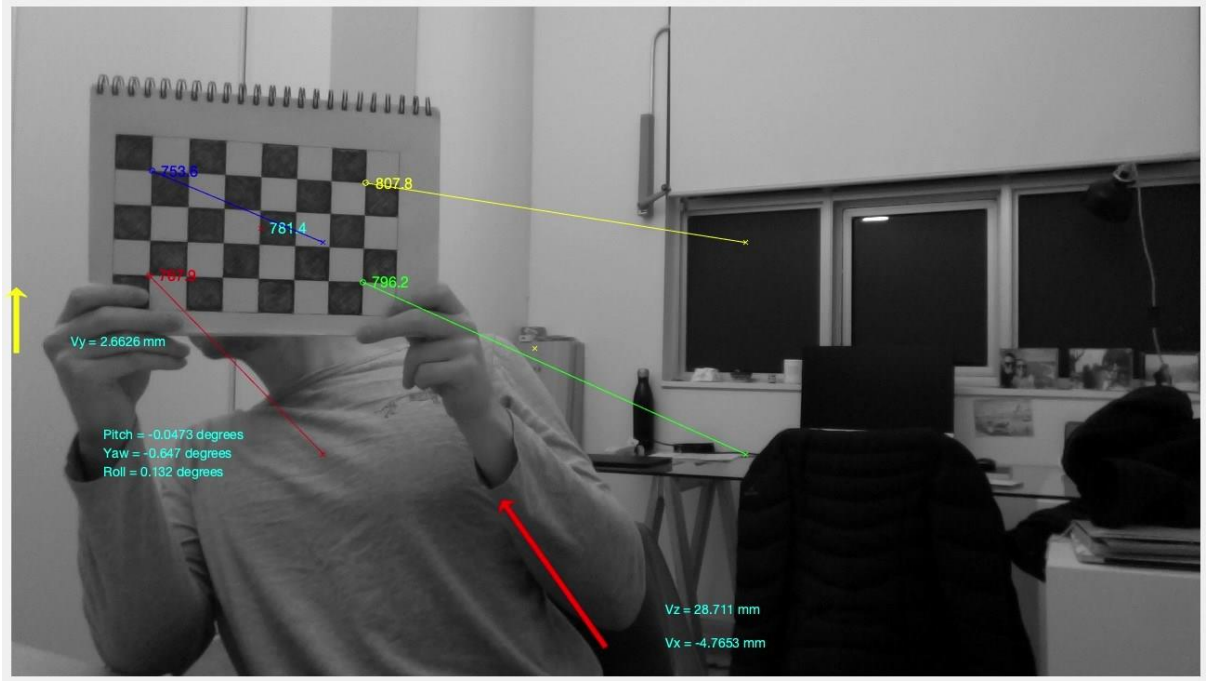
[2] Zhou,Liang.2020. Sensors and Control for Mechatronics System, Week 9. Available at : <https://online.uts.edu.au/webapps/blackboard/content/listContent.jsp?course_id=_42043_1&content_id=_3780446_1 > [Accessed 26th May 2020]

Appendix

Appendix 1.1 (GANTT Chart of Project)

		Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9
Task	Due Date	13/4/20	20/4/20	27/4/20	4/5/20	11/5/20	18/5/20	25/5/20	1/6/20	8/6/20
Project Proposal	19/4/20									
Design pattern to perceive desired position on the image										
Output position, velocity, displacement of camera										
Mid-review report – 10th May 2020	10/5/20									
Output camera image live to MATLAB										
Output live directions to user										
Record video of demo of code										
Project submission – 31st May 2020	31/5/20									
Summarise and format documentation into presentation										
Format documentation into report										
Project presentation – 9th June 2020	9/6/20									
Final Report - 10th June 2020	10/6/20									

Appendix 1.2 (Large Angle Algorithm Output)



Appendix 1.3 (Small Angle Algorithm Output)