# 3D cube-world construction robot

Final Report

**C.H. Conroy**
18072918

Submitted as partial fulfilment of the requirements of Project EPR402
in the Department of Electrical, Electronic and Computer Engineering
University of Pretoria

November 2021

Study leader: Mr. H. Grobler

# Part 1. Preamble

This report describes work that I did in my final year project, developing a robotic system to construct arbitrary 3D shapes using small cubes.

*Project proposal and technical documentation*
This main report contains an unaltered copy of the approved Project Proposal (as Part 2 of the report).

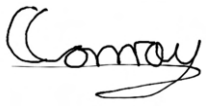
Technical documentation appears in Part 4 (Appendix).

All the code that I developed appears as a separate submission on the AMS.

*Project history*
This project makes use of existing algorithms in the traditional computer vision domain relating to object detection and 3D object localisation as a basis for the computer vision project component. However, the adaption and implementation of these algorithms in this project is my own work. This also applies to the algorithms used to create the coordinate system transformation matrices in the OpenGL 3D shape render component. A number of basic image processing and camera calibration methods were used from the OpenCV library. Furthermore, the C++ QT framework was used as the basis for the PC-based software component. Where other authors' work has been used, it has been cited appropriately, and the rest of the work reported on here, is entirely my own.

*Language editing*
This document has been language edited by a knowledgeable person. By submitting this document in its present form, I declare that this is the written material that I wish to be examined on.

My language editor was Christopher Henry Conroy.



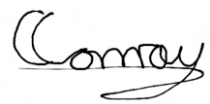
| | |
|---|---|
| *Language editor signature* | *Date* |
| | 21/11/2021 |

*Declaration*

I, Christopher Henry Conroy understand what plagiarism is and have carefully studied the plagiarism policy of the University. I hereby declare that all the work described in this report is my own, except where explicitly indicated otherwise. Although I may have discussed the design and investigation with my study leader, fellow students or consulted various books, articles or the Internet, the design/investigative work is my own. I have mastered the design and I have made all the required calculations in my lab book (and/or they are reflected in this report) to authenticate this. I am not presenting a complete solution of someone else.

Wherever I have used information from other sources, I have given credit by proper and complete referencing of the source material so that it can be clearly discerned what is my own work and what was quoted from other sources. I acknowledge that failure to comply with the instructions regarding referencing will be regarded as plagiarism. If there is any doubt about the authenticity of my work, I am willing to attend an oral ancillary examination/evaluation about the work.

I certify that the Project Proposal appearing as the Introduction section of the report is a verbatim copy of the approved Project Proposal.



| | |
|---|---|
| C.H. Conroy | 21/11/2021 |
| | Date |

# TABLE OF CONTENTS

# LIST OF ABBREVIATIONS

| | |
|---|---|
| **ADC** | Analog-to-digital converter |
| **API** | Application programming interface |
| **BRIEF** | Binary robust independent elementary features |
| **CAD** | Computer-aided design |
| **CNN** | Convolutional neural network |
| **DLT** | Direct Linear Transform |
| **DoF** | Degrees of Freedom |
| **EP*n*P** | Efficient P*n*P |
| **FAST** | Features from accelerated segment test |
| **GUI** | Graphical user interface |
| **IC** | Integrated circuit |
| **LSB** | Least significant bit |
| **MSB** | Most significant bit |
| **NDC** | Normalized device coordinates |
| **OpenGL** | Open Graphics Library |
| **ORB** | Oriented FAST and rotated BRIEF |
| **PCB** | Printed circuit board |
| **P*n*P** | Perspective-*n*-Point |
| **PSU** | Power supply unit |
| **PWM** | Pulse width modulation |
| **RF** | Radio frequency |
| **RGB** | Red, green and blue |
| **RGBD** | Red, green, blue and depth |
| **ROI** | Region of interest |
| **RX** | Receiver |
| **SCARA** | Selective compliance articulated robot arm |
| **SIFT** | Scale-invariant feature transform |
| **SURF** | Speeded-up robust features |
| **SWD** | Serial Wire Debug |
| **ToF** | Time-of-flight |
| **TQFP** | Thin quad flat pack |
| **TX** | Transmitter |
| **UART** | Universal asynchronous receiver-transmitter |
| **USB** | Universal serial bus |

# Part 2. Project definition: approved Project Proposal

This section contains the problem identification in the form of the complete approved Project Proposal, unaltered from the final approved version that appears on the AMS.

| For use by the Project lecturer | Approved | | Revision required | |
|---|---|---|---|---|

**Feedback**

Neatly done Project Proposal.

✓ *Approved*

Symbol awarded: A

To be completed by the student

| PROJECT PROPOSAL 2021 | Project no | HG2 | Revision no | 0 |
|---|---|---|---|---|

| Title | Surname | Initials | Student no | Study leader (title, initials, surname) |
|---|---|---|---|---|
| Mr | **Conroy** | **CH** | 18072918 | Mr H Grobler |

Project title

**3D cube-world construction robot**

| Language editor name | Language editor signature |
|---|---|
| Mr CH Conroy | Conroy |
| **Student declaration** I understand what plagiarism is and that I have to complete my project on my own. | **Study leader declaration** This is a clear and unambiguous description of what is required in this project |
| Student signature: Conroy | Study leader signature and date: 2021-05-19 |

## 1. Project description

What is your project about? What does your system have to do? What is the problem to be solved?

There are a wide range of tasks that humans find trivial to complete but prove challenging for an artificial system to perform. Solving 2D puzzles is an example of such a task which has been successfully emulated by robotic systems to a limited degree. This project aims to expand on this by exploring the task of building 3D puzzles using a robot. The 3D puzzles to be investigated in this project entail both novel and moderately complex 3D shapes consisting of small cubes. The shapes will be defined by an operator through a graphical user interface (GUI) and the robot will be required to manipulate the cubes to construct the specified shape.

In order to realise this functionality, a robot capable of manipulating small cubes will be developed along with an embedded platform to perform the processing required to control the robot and cube manipulation mechanism. A PC-based computer vision system will be implemented to allow the localisation of the robot and the detection of the construction pieces (cubes). Lastly, a PC-based GUI that facilitates the definition of the 3D shapes to be constructed will be designed and implemented along with the software necessary to plan and control the robotic actions required to build the shape.

## 2. Technical challenges in this project

Describe the technical challenges that are *beyond* those encountered up to the end of third year and in other final year modules.

### 2.1 Primary *design* challenges

A computer vision algorithm and model needs to be designed to detect the presence, location and pose of multiple small cubes with a high-degree of precision using input from a camera feed of the robotic manipulator's workspace. A mechanism also needs to be developed to allow the accurate localisation of the robotic manipulator with respect to its environment and the cubes. The mechanical design component of the robotic manipulator and end-effector needs to facilitate stable movement in 3D space. The end-effector needs to be designed to reliably manipulate the small cubes and integrate with the robotic manipulator such that rotation about the z-axis is supported. Lastly, the design of the 3D shape renderer in the GUI will require a mathematical understanding of computer graphic principles.

### 2.2 Primary *implementation* challenges

The high accuracy of the control required for the system as a whole to accurately position the end-effector and the cubes it manipulates in 3D space is the primary implementation challenge of this project. The construction of the mechanical components of the robotic system needs to be of a high standard to ensure precision of the robotic manipulator which needs to be implemented in a manner that minimises oscillations and vibrations generated by the mechanical drivers. The precision implementation and calibration of the cube detection and robotic localisation mechanisms using real-world data will also be challenging. Lastly, the integration of all the software and hardware components within the system is anticipated to be a challenge due to the number of subsystems on the I/O path.

## 3. Functional analysis

### 3.1 Functional description

Describe the design in terms of system functions as shown on the functional block diagram in section 3.2. This description should be in narrative format.

The system in Figure 1 below has two interfaces to receive data input from the external environment. The first interface exists at the image acquisition unit (FU1) which captures the light reflected off the workspace containing the cubes and robot and converts it to a digital image. The digital image is transmitted to the cube detection and localisation unit (FU2.2) which uses localisation techniques to align the image with the coordinate system of the robotic manipulator (FU3.4). Following this, FU2.1 detects all the cubes within the workspace image and passes this information to the system controller (FU2.3). The second interface exists at the shape definition unit (FU2.1) which is a GUI that facilitates the capture of the user's desired 3D shape input. The shape is defined through the specification of the relative positions of each individual cube in 3D space and this information is passed to FU2.3.

FU2.3 is the central system control point which coordinates all major system activities and integrates the computer vision, shape definition and robotic subsystems. FU2.3 forwards the shape definition and cube position information to the construction planner (FU2.4) which computes the translational and rotational transformations as well as the sequence of transformations that need to be applied to the cubes in order to realise the desired 3D shape in the coordinate system of FU3.4. The robotic motion planner (FU2.5) uses these transformation sequences to compute the robotic actions required to replicate these transformations with the physical cubes.

The communication unit (FU3.1) facilitates data exchange between the PC-based software and the embedded robotic controller (FU3.2) which converts the received robotic action sequence to digital signals for the drive systems of FU3.4 and the robotic end-effector (FU3.5). These signals first pass through the driver unit (FU3.3) which increases the signal power to a level sufficient to power the drive systems. FU3.4 manipulates the position of FU3.5 in 3D space according to the signal it receives from FU3.3 and the actuation state of FU3.5 changes in response to the signal it receives from FU3.3. FU3.5 is able to grip a cube when actuated and release it when not.

### 3.2 Functional block diagram

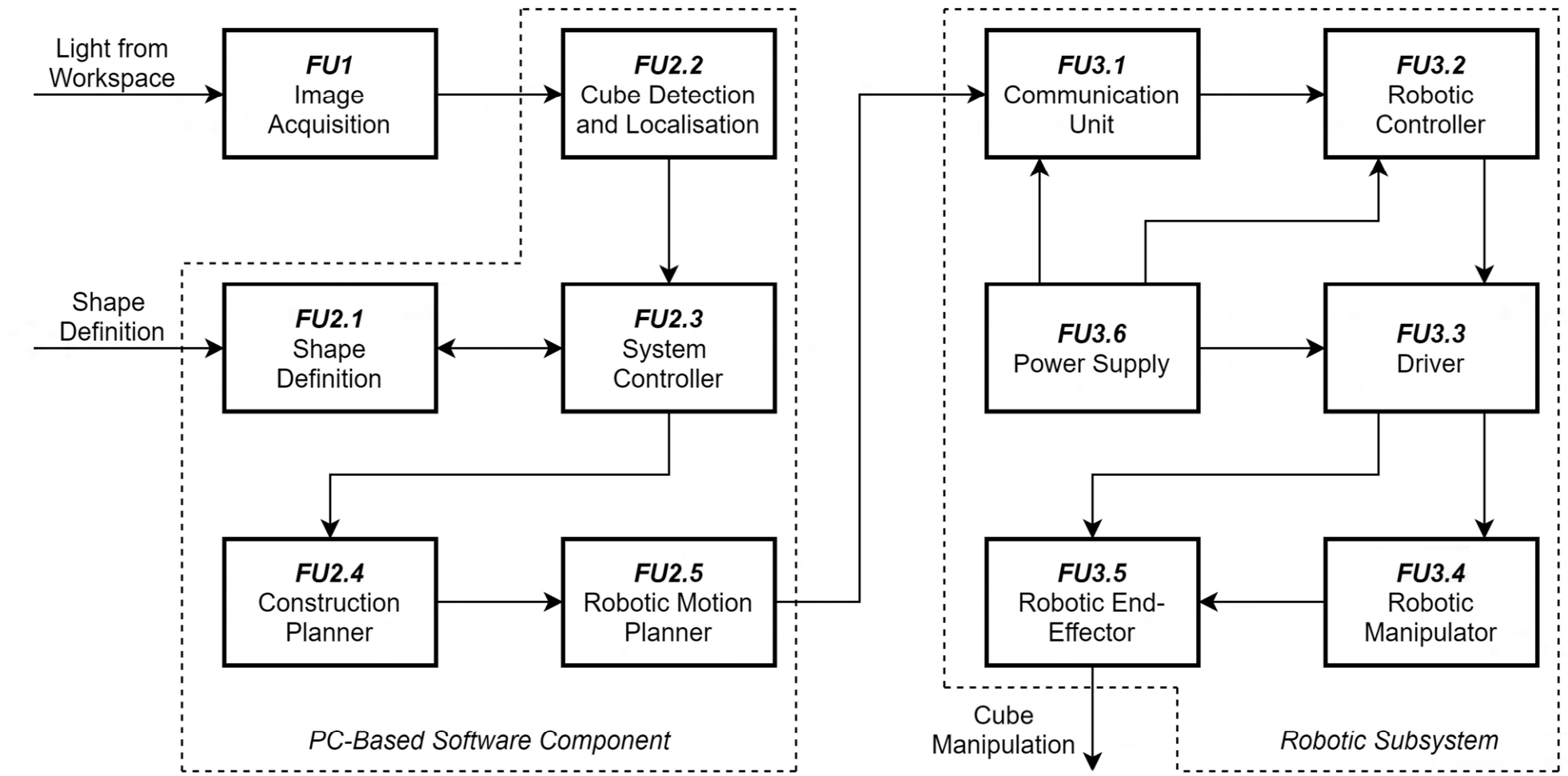



Figure 1. Block diagram showing the proposed functional structure of the system.

## 4. System requirements and specifications

These are the core requirements of the system or product (the mission-critical requirements) in table format IN ORDER OF IMPORTANCE. Requirement 1 is the most fundamental requirement.

| | **Requirement 1: the fundamental functional and performance requirement** | **Requirement 2** | **Requirement 3** |
|---|---|---|---|
| **1. Core mission requirements of the system or product.** Focus on requirements that are core to solving the engineering problem. These will reflect the solution to the problem. | The system should construct novel and moderately complex 3D shapes using small cubes. | The GUI should allow the user to define a wide range of 3D shapes to be constructed. | The end-effector should be able to grip a cube, maintain its grip during motion and release the cube. |
| **2. What is the target specification** (in *measurable* terms) to be met in order to achieve this requirement? | The system should handle shapes up to at least 4 cubes in height containing up to at least 30 cubes where each cube has a face parallel to the base plane. The system should handle equal size cubes with a side length between 10mm and 15mm. | For each constituent cube in the shape, the GUI should allow the position of each cube to be specified along each Cartesian axis with at least 1mm resolution as well as the rotation of each cube about the z-axis with at least 1 degree resolution. | The end-effector should maintain the cube in its grip when the robotic manipulator is at maximum acceleration. The end-effector should be able to maintain the cube in its grip for at least 20 seconds continuously. |
| **3. Motivation:** *how* or *why* will meeting the specification given in point 2 above *solve the problem*? (Motivate the *specific* target specification selected) | The shape specifications are based off the requirements for a pyramid with a 4x4 base and allow for a variety of 3D shape arrangements of varying complexity to be constructed. The cube size is based off a qualitative interpretation of the term small. | Allowing the position of each cube to be specified with 4 degrees of freedom with the stated resolutions facilitates relatively precise control over the 3D shape definition. This allows a wide range of different shape definitions. | The greatest force exerted on a constant mass occurs at the point of maximum acceleration which is the most-likely grip failure instant. 20 seconds was based off twice the average time components are held for in similar existing systems. |
| **4. How will you demonstrate at the examination** that this requirement (point 1 above) and specification (point 2 above) has been met? | The system will construct a selection of 3D shapes satisfying the shape specification using a set of cubes all having the same side length in the specified range. Each construction will either be demonstrated live or using a time-lapsed video. | A pre-defined arbitrary 3D shape that contains the positional and rotational details at the specified resolution will be input into the GUI and the captured shape will be rendered as a 3D graphic to confirm the correct shape was captured. | The system will execute a demonstration sequence in which it grips a cube and moves between the extremes on all axes for a period of at least 20 seconds before releasing the cube. |
| **5. Your own design contribution:** what are the aspects that *you will design and implement yourself* to meet the requirement in point 2? If none, *remove this requirement.* | The high-level structure of the overall system to construct the shapes will be designed. The software component that plans the construction and robotic movement will be developed from first principles and be executed on a PC. | The GUI structure, logic and flow will be designed. The 3D render of the captured shape will be designed from first principles with only calls to a low-level graphics API. A PC will be used for the processing platform for this component. | The mechanical component of the end-effector mechanism and its attachment to the robotic manipulator will be designed from first principles. |
| **6. What are the aspects to be taken off the shelf** to meet this requirement?<br>If none, indicate "none" | The construction cubes and PC will be taken off the shelf. | A GUI framework will be used to provide the base window for the GUI as well as to provide the basic GUI components. The PC will be taken off the shelf. | The motor that actuates the end-effector as well as the appropriate motor driver will be taken off the shelf. |

## System requirements and specifications page 2

| | Requirement 4 | Requirement 5 | Requirement 6 |
|---|---|---|---|
| **1. Core mission requirements of the system or product.** Focus on requirements that are core to solving the engineering problem. These will reflect the solution to the problem. | The robotic manipulator should accurately translate the end-effector in 3D space and rotate it about its vertical axis. | The computer vision component should detect and localise the construction cubes in the workspace to facilitate re-gripping dropped cubes and identifying damage to the 3D shape under construction to signal a construction halt condition. | The system should detect when a cube is unintentionally dropped by the end-effector. |
| **2. What is the target specification** (in *measurable* terms) to be met in order to achieve this requirement? | The robotic manipulator should have a repeatability of at least 2mm for each Cartesian axis and a repeatability of at least 5 degrees for the rotation about the z-axis. | Only the cubes whose faces are visible from a vertical perspective need to be detected and localised. Cubes that need to be gripped should be localised with a positional accuracy of 2mm and a rotational accuracy of 5 degrees about the z-axis. | The system should detect a cube has been dropped before the end-effector grips the next cube to be placed. |
| **3. Motivation:** *how* or *why* will meeting the specification given in point 2 above *solve the problem*? (Motivate the *specific* target specification selected) | The repeatability needs to be a relatively small fraction of the cube size to ensure minor positional placement variations do not significantly compromise the basic 3D shape. Similar reasoning applies for rotational repeatability. | Only cubes visible from the vertical perspective are required to identify 3D shape damage that prohibits further placement or are candidates as dropped cubes. Dropped cubes need to be localised with an accuracy similar to the robotic repeatability. | The dropped cube may be required as support for the next cube to be placed. Therefore, it is important that the system deals with the dropped cube situation (either by re-gripping the cube or replacing it if absent) before the next cube is placed. |
| **4. How will you demonstrate at the examination** that this requirement (point 1 above) and specification (point 2 above) has been met? | The robotic manipulator will be instructed to travel from the origin of its coordinate system to an arbitrary point on each axis 5 times with the maximum deviation measured. This will be repeated in the z-axis rotational coordinate system. | The system will be instructed to detect and localise cubes with 5 different positions and orientations. The detected orientations and positions will be displayed on the GUI and compared to the corresponding physical measurements. | The end-effector will be instructed to drop the cube it has in its grip. The system will proceed as normal until it detects that the cube has been dropped. At this point, the system will show a GUI notification indicating a dropped cube was detected. |
| **5. Your own design contribution:** what are the aspects that *you will design and implement yourself* to meet the requirement in point 2? If none, *remove this requirement.* | The mechanical component of the robotic manipulator will be designed. The firmware to control the robotic manipulator will be developed from first principles and will be executed on an embedded platform. | The algorithms required to process the images, detect the cubes in the images and localise the cubes will be developed from first principles. The algorithms will be executed on a PC. | The firmware to monitor the dropped cube sensor and the system control software that determines the response to the dropped cube will be designed and run on an embedded system and PC respectively. |
| **6. What are the aspects to be taken off the shelf** to meet this requirement?<br>If none, indicate "none" | The motors to drive the robotic manipulator's movement as well as the motor drivers will be taken off the shelf. The microcontroller will be taken off the shelf. The robotic subsystem's power supply will be taken off the shelf. | The camera hardware and firmware required to capture the images will be taken off the shelf. The PC will be taken off the shelf. | The sensor required to monitor for a dropped cube will be taken off the shelf. The microcontroller will be taken off the shelf. |

## 5. Field conditions

These are the REAL WORLD CONDITIONS under which your project has to work and has to be demonstrated.

| | Field condition 1 | Field condition 2 | Field condition 3 |
|---|---|---|---|
| **Field condition requirement.** In which field conditions does the system have to operate? Indicate the one, two or three most important field conditions. | The system should work under laboratory conditions. | The image background should be controlled. | |
| **Field condition specification.** What is the specification (in measurable terms) for this field condition? | The ambient lighting level should be approximately 500 lux. | The immediate background of the construction cubes in the captured images should be non-reflective and of a single hue. | |

## 6. Student tasks

### 6.1 Design and implementation tasks

List your primary design and implementation tasks in bullet list format (5-10 bullets). These are *not* product requirements, but *your* tasks.

- The mechanical structure of the robotic manipulator needs to be designed, constructed and its accuracy tested.
- The robotic end-effector mechanism needs to be tested with small cube manipulation and integrated with the robotic manipulator.
- Computer vision techniques and models for object detection and robotic localisation need to be researched.
- Many images need to be captured of the small cubes from various angles to design and develop the object recognition software.
- A shape definition GUI needs to be implemented and research needs to be performed for the selection of the most appropriate GUI framework and graphics API.
- A single piece of software needs to be developed that integrates the computer vision, GUI and robotic path planning functionality.
- A circuit for the robotic controller embedded system and its peripherals needs to be designed, constructed and integrated along with a power supply.
- A communication protocol needs to be developed for communication between the embedded system and PC.
- The most appropriate microcontroller for the robotic controller needs to be researched and selected and firmware needs to be developed for it.

### 6.2 New knowledge to be acquired

Describe what the theoretical foundation to the project is, and which new knowledge you will acquire (beyond that covered in any other undergraduate modules).

- Computer vision forms a large part of the theoretical basis for this project. The primary computer vision concept to be investigated is object detection as well as all the mathematical concepts, models and image processing techniques relevant to this.
- Robotic localisation involves the identification of the robotic system's location with respect to the environment. The theory associated with this in a 3D context will be investigated deeply to achieve as precise an alignment of the robotic end-effector with the cubes as possible.
- Robotics is at the core of this project and therefore a great deal of theoretical knowledge relating to robotic systems will have to be aquired. One subtopic that will require investigation is robotic path planning which involves planning the actions of the robot optimally.
- This project involves a degree of mechanical design which will require the acquisition of knowledge in this domain to produce a mechanically precise system.
- The construction planner for the 3D shapes requires a theoretical background on modelling geometric systems and objects in three dimensions.
- Knowledge of techniques relating to 3D computer graphic rendering needs to be acquired to implement the 3D shape renderer in the GUI.

# Part 3. Main Report

# 1. Literature study

## 1.1 Background

### *1.1.1 Overview*

The use of artificial systems to emulate tasks that humans find straightforward to perform, such as solving 2D puzzles, has been a long-standing practice since components of the solution system are often relevant in industrial applications [1]. This work focuses on a similar task that involves the construction of 3D shapes using small cubes. Such a task bears similarity to those tasks in the domain of pick and place robotics with the variation that object placement is dependent on the location of previously placed objects. Existing solutions in this domain typically consist of two primary components: a computer vision system to detect and localise the object of interest as well as a robot to alter the location and orientation of the object in 3D space [2].

### *1.1.2 Robotic System*

A robot can be viewed as the combination of two core components, namely the robotic manipulator and the end-effector. The end-effector is the physical interface between the robot and the object of interest and is referred to as a robot gripper when its purpose is to grip the object to facilitate pose manipulation. The nature of the robot gripper depends on the physical characteristics of the object of interest and as such a wide variety of grippers have been developed. These include stiff finger grippers, flexible finger grippers, magnetic grippers and vacuum grippers which are best suited for objects with a flat surface [3]. The function of the robotic manipulator is to alter the position and orientation of the end-effector in 3D space. Robotic manipulators are categorised by the coordinate systems used to describe their movement mechanics which includes polar, cylindrical, articulate and Cartesian coordinates [4]. Cartesian robots have the benefit that accuracy of the robot is uniform throughout the robot's work envelope.

### *1.1.3 Object Detection*

The purpose of the computer vision system is to detect the object of interest and localise it using the input image data captured from the robot's workspace, such that the robot has sufficient information to interact with the object. It is important that a distinction is made between object detection and object recognition. Object detection refers to the process of locating instances of a given object within an image while object recognition refers to the identification and classification of an object. This work is concerned with the former as the object of interest is known to be a cube. There are a wide range of approaches to the object

detection problem. These can be broadly classified as being part of either the traditional computer vision domain or the deep learning domain [5]. In both cases, various techniques are used to extract information from the image data in the form of features which are subsequently used to detect the objects of interest [6].

The distinguishing factor between traditional and deep-learning domains lies in the method of feature extraction. Traditional approaches incorporate a manual feature extraction step before the data is processed further. Deep learning approaches, on the other hand, integrate this step as part of the underlying model, such as a convolutional neural network (CNN). In this sense, such models can be viewed as highly integrated structures which take images as input after minimal preprocessing and produce the object recognition information as output. A deep-learning approach has been developed to detect generic rectangular cuboid objects in everyday scenes captured from a single perspective [7]. However, the generic nature of this task requires a highly sophisticated approach to achieve reasonable success.

The best solutions to computer vision problems that arise from unconstrained environments and require a great degree of generality are almost always found in the deep learning domain. However, when the problem is sufficiently constrained, solutions based on traditional techniques often exhibit performance that is comparable or even superior to that of deep learning approaches. In such cases, traditional approaches are often preferable since, unlike deep learning approaches, they do not require a massive training data set or a large degree of computational power. In general, a feature can be considered to be a piece of information present within the image input data. Edges, corners, blobs and ridges are examples of common low-level features that are considered within the traditional computer vision domain. Feature detectors are used to locate these fragments of information in the image input data. The Canny edge detector [8] and Harris corner detector [9] are examples of such methods which are popular for detecting edge and corner features respectively within an image.

The blurring of an image is a preprocessing step commonly employed with traditional approaches. This operation acts as a low-pass filter and filters out high-frequency noise which manifests itself as outlier pixel intensities. Improved performance of the feature detection stage is generally observed as a result. The conversion of an image to grey-scale is another common preprocessing step which reduces the complexity of subsequent operations when the color information of the image is insignificant. Thresholding is a useful method for obtaining shape-level feature information through image segmentation and is often applied following the preprocessing phase. The application of this technique to a grey-scale image results in a binary image which lends itself to further shape-level feature extraction. Since there are exposure inconsistencies that arise between images due to environmental light variability, it is usually prudent to incorporate an automatic threshold level determination mechanism when thresholding. In the ideal case, a grey image will exhibit a bimodal distribution of pixel intensities where the minima between the peaks corresponds to the ideal threshold value. However, such pixel intensity distributions are not necessarily guaranteed in most practical applications and, as a result, more robust automatic thresholding techniques have been developed, such as Otsu's method [10]. Existing automatic thresholding methods are usually classified as either histogram shape-based, clustering-based, entropy-based, object attribute-based, spatial or local methods [11].

Contours are another useful shape-level feature that can be used in service of traditional

approaches to object detection. The bounding outline that captures the shape of an object in an image is considered to be a contour. There are many different approaches to contour detection which, in general, can be categorised as either pixel-based, edge-based or region-based methodologies. Contours in images frequently correspond with discontinuities in grey-scale pixel intensity, particularly in the case of contours arising from luminance changes, which are detectable through the corresponding gradient magnitude information. A common approach to extract this information is to make use of a local filter which is convolved with the image. This results in a gradient space where the greatest gradient magnitudes are indicative of potential contours. However, this method is unreliable as it usually produces discontinuous contours and, therefore, often requires supplementary high-level feature information [12].

The contour detection problem is significantly simplified when the problem space is constrained to only binary images. In this case, gradient information is not required as with grey-scale images, as pixel intensity discontinuities can be determined using only adjacent pixels. Furthermore, a binary image can be interpreted as consisting of a number of connected components where a connected component is defined as a set of pixels with identical intensity values which are interconnected through either 4-pixel or 8-pixel connectivity. Within this framework, the concept of a contour can be reduced to the sequence of pixels that define the boundary between adjacent but dissimilar connected components. The advantage of such contours is that they are guaranteed to be continuous, in contrast to the grey-scale image case. The border following algorithm is a longstanding approach to the detection of these binary image contours [13]. An extension to this approach exists whereby a more advanced border labelling method is employed to facilitate the extraction of topological structure information. Such information includes the hierarchical relationship between borders as well the distinction between outer and hole borders. This approach has also been adapted such that only the top-level outer borders in the hierarchy are detected which offers improved computational performance for applications that only require such information [14].

Contour detection, in conjunction with contour template matching, has been successfully used in robotic object detection and grasping applications using a single monocular camera [15]. There exist a number of other feature detectors which have applicability to cube detection. For objects with straight edges, the Hough transform is a useful image processing tool that can be used to capture these edges with parameterised straight lines in 2D space which is useful to determine the orientation of the object [16]. A more advanced and robust approach to determining useful features within an image involves the use of a feature descriptor which is a vector of values that describes the local region about a given image point. A number of feature descriptor algorithms have been developed such as the scale-invariant feature transform (SIFT) [17], speeded-up robust features (SURF) [18], features from accelerated segment test (FAST) [19], binary robust independent elementary features (BRIEF) [20] and finally the oriented FAST and rotated BRIEF (ORB) [21] algorithms. These features can be used to detect instances of objects within the input image data through the process of template matching. This method has been successfully applied as part of the detection process for cubes marked with alphabetical and numerical characters [22].

### *1.1.4 Object Localisation*

The detection of an object within an image only forms the first stage in the robot's computer vision system. In order for the robot to interact with the object of interest in the physical world, the detected object needs to be localised such that its pose with respect to the robot's coordinate system is known. The object localisation methods available for robots when only red, green and blue (RGB) image input data is available can be categorised as either monocular vision or stereo vision approaches. With the monocular vision case, only a single RGB image is available as input at each time instance while in the stereo vision case, two or more RGB images are available [23]. The primary drawback of monocular vision approaches is the loss of depth information that arises during the projection of the 3D world onto a single 2D image. An additional piece of information, such as the size or world plane of the object, is required in order to recover the depth information. Stereo vision approaches, on the other hand, are able to recover the depth data based on the disparity between images that arises due to difference in pose of the cameras used to capture the images [24]. However, stereo vision approaches require more hardware and greater computational resources than monocular vision approaches. An alternative approach is to make use of a device that captures red, green, blue and depth (RGBD) data directly, such as a time-of-flight (ToF) camera or integrated binocular stereo camera.

In order to relate the object detection information derived the from camera input data to the world frame, the pose of the camera with respect to the world coordinate system needs to be determined. This information is represented by an extrinsic camera matrix which encompasses the rotation and translation parameters of the camera's pose with respect to the world frame. The extrinsic matrix can be used to map points in the world coordinate system to the 3D camera coordinate system. In order to map points from the 3D camera coordinate system to the 2D homogeneous coordinates in the image, an intrinsic camera matrix is used [25]. Intrinsic parameters describe internal properties of the camera and are based on the pinhole camera model. These include the camera's inherent principal point offset, focal length and axis skew. The skew of the sensor axes occurs as a result of the optical axis not being exactly perpendicular to the sensor plane. However, for practical purposes this parameter is often discarded. The extrinsic matrix and intrinsic matrix can be multiplied to form the projection matrix which is used to project any point in the world frame to the image frame provided that the pinhole camera model is used and no lens distortion effects are present [26].

The intrinsic and extrinsic camera parameters need to be determined in order to make use of the pinhole camera model in practical applications. Camera calibration is used to estimate the intrinsic characteristics of the camera while camera localisation is used to estimate the extrinsic parameters of the camera. A popular approach to camera calibration involves the use of a planar pattern with known dimensions of which multiple images are captured at various different poses [27]. Either the pose of the planar pattern or the camera may be altered between calibration images. The application of this algorithm to a given set of such images will produce an estimate of the intrinsic parameters of the camera as well as the radial distortion of the camera. Real-world cameras have lens-induced distortion effects that are not included as part of the pinhole camera model. Radial distortion is observed when the degree to which light rays bend when incident on the lens is not consistent with the distance from the

optical centre of the lens. Tangential distortion is observed when a degree of misalignment exists between the image plane and lens. These distortion effects are described by the radial and tangential distortion coefficients respectively [28].

In order to determine the extrinsic camera matrix, the rotation and translation of the camera with respect to the world coordinate system needs to be calculated. A popular approach to this problem involves the use of $n$ 3D to 2D point correspondences to calculate the camera pose for six degrees of freedom (DoF). A point correspondence refers to the situation where the 3D location of a given point in the world coordinate system, as well as the corresponding 2D location in the homogeneous image frame, are known. The most general formulation of this problem requires the computation of the camera's intrinsic parameters as well as the camera's extrinsic parameters. The Direct Linear Transform (DLT) algorithm is a well-known solution to this problem when at least six point correspondences are given [29]. However, this approach suffers from a degree of inaccuracy due to the need to estimate the intrinsic parameters of the camera.

If the assumption is made that the camera is calibrated, such that the intrinsic parameters of the camera are known, the problem reduces to the Perspective-$n$-Point (P$n$P) problem which has been deeply explored in literature. As such, a number of iterative and non-iterative solutions to the P$n$P problem have been developed. The efficient P$n$P (EP$n$P) algorithm is a popular non-iterative approach for the case when four or more point correspondences are given. Although four point correspondences is sufficient for EP$n$P to find a solution, a greater number is preferred to provide a degree of redundancy and reduce the solution's sensitivity to noise. It is also noted that the algorithm is capable of solving for the case where the points used for the point correspondences have a planar arrangement in the world coordinate system [30].

Within the context of the P$n$P problem, the first step to solve for the camera's extrinsic parameters requires the creation of a set of point correspondences. The use of fiducial markers is a popular approach to this task, notably in the augmented reality domain. In general, a marker is an object that is placed as a known point of reference in the scene that is processed by the computer vision system. A fiducial marker system consists of three core components, namely the markers, a fiducial detector and an encoding scheme. The fiducial detector typically makes use of traditional computer vision techniques and are therefore usually characterised by simple designs that are distinct within the scene [31]. Morphological operations often form part of the traditional methods applied during the marker detection stage [32].

The design of the marker ensures it is rotationally asymmetric while encoding a piece of information such as the fiducial identifier. Some existing fiducial families, such as ARToolKitPlus [33] and AprilTag [34], make use of black and white square grids to encode binary data where each cell represents a binary digit. In order to extract information from the markers, perspective distortion needs to be eliminated. In the case of square markers, the corners of each marker candidate are used to compute a homography that is used to remove this distortion [35]. Finally, a unique marker with a known location in the world coordinate system can be used to create a point correspondence through the combination of this information with its detected location in the image coordinate frame. The use of multiple markers facilitates the creation of the requisite point correspondence set.

## 1.2 Applicability to Project

There exists a substantial amount of literature exploring the field of artificial systems and their application in various problem domains. Problems addressed by artificial system solutions in the domain of pick and place robotics are generally the most closely related to the 3D shape construction task explored in this project. It was noted that, at a system level, solutions in this domain typically consist of a computer vision system used in conjunction with a robot. This information, in addition to the implementation details of these systems, was used as a starting point to guide the system-level design in this project.

In terms of the robotic system, the general approach to the mechanical design of a robot can be partitioned into the design of two distinct components, namely the robotic manipulator and the end-effector. The literature highlights a wide variety of approaches to the design of both these mechanical facets. In both cases, the selection of a particular approach depends substantially on the the nature of the problem space. Fortunately, there is an expansive body of knowledge in terms of the strengths and weaknesses for each of the approaches for both the end-effector and the robotic manipulator with respect to a number of problem domains. As such, the literature was used as the basis for the mechanical design approach selected decisions made in this project. In addition, the machine design procedure frequently appears as a core component of the mechanical design process in similar projects. Therefore, this procedure was also incorporated into the robot design process utilised in this project.

The detection and localisation of the constituent construction cubes by the computer vision system is an essential component of this project that is required to facilitate interaction of the robot with dropped cubes. Fortunately, there exists a vast range of literature in the computer vision field that has direct applicability to this problem. For the cube detection component of this problem, there are two broad potential solution domains, namely the deep-learning and traditional computer vision domains. Solutions in the deep-learning domains are typically best applied to problems that require a significant degree of generality while traditional solutions are best suited to constrained problems. Traditional detection solutions are based on the manual extraction of selected features from the object of interest which are subsequently used to identify instances of the object in arbitrary images. A number of methods that are useful for extracting such features were identified in this literature study and many of these were trialed in application to the cube detection problem in this project. The methods that exhibited the greatest degree of accuracy and robustness when applied were included as part of the final computer vision system implementation.

Once a cube has been detected within the input image data, the location and orientation of the cube needs to be determined with respect to the robot in the physical world. This relates to the problem of object localisation which has been explored extensively in literature, particularly in the augmented reality and robotic system domains. The exact nature of the solution depends on whether a monocular or stereo vision approach is used. In general, existing solutions build on the mathematical foundation provided by the pinhole camera model. This model formed the basis of describing the relationship between the robot and camera coordinate system's in this project. The intrinsic and extrinsic camera parameters outlined within this model are sufficient to facilitate the projection of points between the coordinate systems. This was

taken advantage of to determine the location of cube points with respect to the robot from the corresponding points in the input image data.

In order to make practical use of the above object localisation approach, the intrinsic and extrinsic camera parameters need to be determined. The camera calibration techniques identified in literature offer an avenue to determine the intrinsic matrix of the camera and these were subsequently applied to calibrate the camera used in this project. Similarly, solutions to the P*n*P problem provide a means to determine the extrinsic camera parameters and, as such, one of the identified solutions from the literature was utilised to determine the camera extrinsics in this project. Furthermore, any solution to the P*n*P problem requires a set of point correspondences between the world frame and the image frame. Existing approaches make use of fiducial markers to obtain these correspondences and the same approach was applied in this project. Furthermore, a number of the papers included as part of this literature study detail a wide variety of fiducial marker design considerations. These were used to inform the design of the fiducial markers used in this project.

Overall, there have been a number of research projects into artificial systems, consisting of a computer vision system used in conjunction with a robot to perform tasks such as 2D puzzle building or generic pick and place operations. However, the specific task of constructing moderately complex 3D shapes using small cubes does not appear to have been explored. A similar project involved the use of larger cubes marked with identification artifacts to assist in the detection of the cubes [22]. In contrast, the cubes used in this project exhibit a plain appearance and a greater degree of reflectivity due to their metallic nature. In general, the majority of the approaches only focus on the development of a particular sub-component of the robotic and computer vision system while the bulk of the system comprises of an off-the-shelf implementation adapted to support the task in question. This work aims to develop a robot in conjunction with a computer vision system with all components tailored to fulfil the 3D shape construction task.

# 2. Approach

## 2.1 Problem Space

The nature of the selected approach to the cube construction task explored in this project is completely dependent on the characteristics of the problem space. The problem space was broadly defined as part of the project proposal. However, further details regarding the cube component of the problem space are required in order to justify the chosen approach. A number of general materials were considered to form the construction cubes including hard plastic, wood, aluminium and steel. Hard plastic cubes have the advantage that they are widely available off-the-shelf while wood offers ease in cube manufacturing. However, the low density of these materials means the cubes are more likely to shift in the shape structure when exposed to vibrations. Therefore, aluminium cubes were chosen due to their greater density. Aluminium was selected over steel due to its superior machinability and inability to rust.

## 2.2 Robotic Subsystem

The robotic subsystem (FU3) was identified as one of the major components of the solution system from a functional perspective in the project proposal. The high-level purpose of FU3 is to facilitate the manipulation of the construction cube's pose in 3D space. The robotic end-effector (FU3.5) acts as the robot's physical interface with the cube. Grippers are commonly used for the end-effector components. However, there exist planar arrangements of adjacent cubes that prevent access to at least one opposite face pair of the target cube which is required by the gripper to exert a grip. Therefore, a vacuum-based suction cup end-effector mechanism was selected as it only requires access to the top face of the target cube to exert a grip. Furthermore, the non-porous and smooth nature of the aluminum cubes render the cube amenable to this mechanism.

There are a wide range of approaches to the robotic manipulator component (FU3.4) which is required to manipulate the end-effector pose in 3D space. These include the articulated robot, selective compliance articulated robot arm (SCARA), delta robot and the Cartesian robot. The articulated robot offers the greatest flexibility in the range of poses the robot can attain while the delta robot offers excellent movement speed. However, these kinematics of these robots are complex and minor imperfections in their implementation creates inaccuracy. Furthermore the precision of these robots, in addition to SCARA robots, varies throughout the workspace. Cartesian robots, on the other hand, exhibit consistently high precision throughout the workspace and are suited to Cartesian-based problems. A Cartesian gantry robot was selected as the robotic manipulator approach for these reasons. The robot was designed outwards from its interface with the cube in an iterative fashion using computer-aided design (CAD) software and a mathematical base in dynamics.

The robotic controller (FU3.2) was designed as an embedded platform to provide an interface to control the robotic manipulator and end-effector. This component was first designed and

prototyped using a breadboard before a more robust PCB implementation was developed. The embedded software for this controller was prototyped using the STM32 hardware abstraction layer (HAL) library before being converted to first principles. The power supply (FU3.6) was taken off-the-shelf as well as the motor drivers (FU3.3). Finally, the communication unit (FU3.1) was based on the universal asynchronous receiver-transmitter (UART) and realised as a custom communication protocol used in conjunction with an off-the-shelf CH340 serial converter integrated circuit (IC).

## 2.3 PC-Based Software Component

The PC-based software component (FU2) was developed as a graphical user interface (GUI) application using C++ and the QT framework. C++ was selected due to the suitability of its performance to the computationally intensive nature of the image processing required in this project. QT was selected due to its maturity and support for OpenGL integration. FU2 was explored and developed in a bottom up approach. This means that the shape definition component (FU2.1) and cube detection and localisation component (FU2.2) were designed and developed first followed by the system controller (FU2.3), construction planner (FU2.4) and robotic motion planner (FU2.5) which all depend on FU2.1 and FU2.2. OpenGL was selected as the low-level graphics API to implement the 3D shape model render required by the shape definition component. This component was originally developed externally to the QT ecosystem to verify its functionality before being integrated adapted for integration with the QT OpenGL interface.

A number of object detection approaches were investigated for cube detection in FU2.2. However, a binary thresholding approach based on the reflection intensity of light from the top cube faces was found to be the most robust and used in conjunction with contour detection. A pin-hole camera model based approach was used to map points between the image and world frames for the purpose of cube localisation in FU2.2. These components were initially developed using the OpenCV library before being progressively replaced with first principle implementations.

Finally the system controller was developed simultaneously with the construction planner and the robotic motion planner in a tightly integrated manner. As the system controller acts as the central point where the information pathways from the shape definition unit, cube detection and localisation unit and the robotic subsystem intersect, the system controller was developed as the top-level component in the PC-based software that integrates the other software components. As a result, the primary high-level software design work took place as part of the development of this component.

The approach reasoning and considerations discussed here correspond directly to the structure of the system design presented in Section 3. Specifically the design of the robotic subsystem is first presented in two parts, namely the mechanical design in Section 3.2 followed by the embedded controller design in Section 3.3. The design of the PC-based software is then presented in three parts. The first two parts are the shape definition component in Section 3.4 and the computer vision component in Section 3.5 on which the system controller depends. Finally the system controller and the use of this component as the basis for system integration is presented in Section 3.6.

# 3. Design and implementation

## 3.1 Design summary

This section presents a summary of the project design tasks as well as the implementation of these tasks (see Tables 1-2).

| **Deliverable or task** | **Implementation** | **Completion of deliverable or task, and section in the report** |
|---|---|---|
| Mechanical design and construction of robotic manipulator | The mechanical design of the robot manipulator was completed from first principles using the Fusion 360 CAD software. This was constructed by the student both using 3D printing and metal machining technologies. | Completed. Section 3.2 |
| Mechanical design and construction of robotic end-effector mechanism | The mechanical design of the robotic end-effector mechanism was completed from first principles using the Fusion 360 CAD software. This was constructed by the student using 3D printing technologies. | Completed. Section 3.2.1 |
| Design of embedded robot controller circuit | The design was completed from first principles and a prototype was implemented on a breadboard. | Completed. Section 3.3 |
| Design of printed circuit board (PCB) for embedded robot controller | The PCB design was completed using the KiCAD software package from first principles. | Completed. Section 3.3.2 |
| Development of firmware for embedded robot controller | Firmware was developed from first principles using C. | Completed. Section 3.3.4 Section 3.3.3 |
| Design of communication protocol for communication between the embedded robot controller and the PC-based software | The communication protocol design was completed from first principles and implemented between the embedded robot controller and PC. | Completed. Section 3.3.4 |

**Table 1. Design summary.**

| **Deliverable or task** | **Implementation** | **Completion of deliverable or task, and section in the report** |
|---|---|---|
| Design and implementation of shape definition GUI based on a low-level graphics application programming interface (API) | The shape definition GUI was designed and implemented from first principles using the OpenGL graphics API. | Completed. Section 3.4 |
| Development of computer vision cube detection algorithm | The computer vision cube detection algorithm was developed from first principles using C++. A few basic image processing functions were used from OpenCV. | Completed. Section 3.5.1 Section 3.5.3 Section 3.5.5 Section3.5.6 |
| Development of computer vision cube localisation algorithm | The computer vision cube localisation algorithm was developed from first principles using C++. A few basic image processing functions and camera calibration functions were used from OpenCV. | Completed. Section 3.5.2 Section 3.5.3 Section 3.5.5 Section 3.5.4 |
| Development of construction planner software component | The construction planner software component was developed from first principles. | Completed. Section 3.6.2 |
| Development of robotic motion planner software component | The robotic motion planner software component was developed from first principles | Completed. Section 3.6.3 |
| Development of system control software that integrates the shape definition, computer vision, construction planner and robotic motion planner software components | The system control software was developed from first principles using the QT C++ framework. | Completed. Section 3.6.1 |

**Table 2. Design summary continued.**

## 3.2 Mechanical Robotic Component

### *3.2.1 End-Effector Mechanism*

For the purposes of this project, the end-effector subsystem refers to the components responsible for enabling the direct manipulation of the cube. These components include the vacuum pad, tubing and vacuum generation system. The end-effector subsystem is attached to the gantry robot by means of an end-effector mechanism. In order to design this component, the machine design procedure was followed. The first step of this procedure involves understanding the requirements of the machine. The end-effector mechanism requirements are listed below.

- It should attach the end-effector to the gantry robot.
- It should maintain the suction-pad component in a vertical orientation.
- It should allow limited linear buffered motion of the vertical suction-pad component along the z-axis w.r.t the gantry robot.
- The linear buffer should facilitate at least a 5mm range of linear motion.
- The purpose of the linear buffer is to allow the robot to target a z-axis position slightly below the intended z-axis position to ensure the cube is definitely touching the placement surface so the cube is not released in mid-air.
- It should allow the vertical suction-pad components to rotate about the z-axis.
- It should allow the connection of a drive mechanism to drive the rotation about the z-axis.
- It should allow the vacuum tubing to be routed to the gantry robot.

The initial design investigated for the end-effector mechanism was centred around the requirement of a 5mm linear displacement buffer. A buffer can be implemented as a linear rod with guide holes as well as a ridge on the rod that allows a spring to be placed between the ridge and one of the guide holes. This provides a linear buffering action along the axis to which the rod is aligned. The design idea with respect to the end-effector mechanism is to mount the vacuum pad on the end of the rod opposite to the spring and position the rod in a vertical orientation with the vacuum pad at the bottom where it can access cubes on the plane below it. When the vacuum pad is not in contact with anything, the spring and the force of gravity push the vacuum pad into its lowest position. When a force is applied vertically upwards against the vacuum pad, as is the case when the vacuum pad is pressed against a cube, the spring will compress if the force is greater than the gravitational force on the moving rod as well as the spring force at that length.

An issue with this simple design is that the vacuum tube needs to be routed to the vacuum pad as well as the fact that the vacuum pad needs to be rotatable by an external motor. The tubing routing issue is solved by making the rod hollow and routing the tube through the rod and out the top of the rod. Furthermore, it is noted that the system only needs to be able to rotate a minimum of 90 degrees to be able to realise any orientation of a cube in terms of rotation about the z-axis. The rubber tubing is comfortably able to absorb this degree of torsion and

therefore, no additional mechanisms are required to route the tube.

The second design consideration is how the rod will be rotated to rotate the vacuum pad given that the rod has linear motion of 5mm. The design solution to this investigated in the proof of concept design is the use of a gear attached to the rod which can be driven by a motor gear. The use of gears with their axes aligned with the rod axis allows the linear motion to be absorbed by the linear freedom of movement between the gear teeth. In order to absorb this motion, the height of the gear attached to the rod needs to be at least 5mm greater than the height of the motor gear is the maximum overlap is to be always maintained assuming the position of the motor is fixed with respect to the component supporting the rod. An alternative design that could be explored in the future, is fixing the linear position of the motor with respect to the spring to reduce the gear height and using the weight of the motor to act in a similar manner to the spring force.

The rod designed to facilitate the rotation and linear buffer motion of the vacuum pad was further developed. Intuitively, the rod requires a relatively low amount of torque to initiate and maintain rotary motion. Therefore, the smallest class of NEMA stepper motors, namely NEMA 8 motors, were considered as a guide for the motor footprint in the mechanical design. A preliminary decision to use a stepper motor over a servo was made based on the fact that rotary stability is essential once the vacuum pad has reached its required orientation. Servos may pulsate at standstill which is undesirable as the gripped cube may displace adjacent cubes. Furthermore, the rotation speed required is low and stepper motors exhibit the best torque characteristics at low speed. Lastly, in full step mode, NEMA 8 stepper motors typically exhibit a full step resolution of $1.8°/\text{step}$ which falls within the rotary accuracy specifications of $5°$.

The width and breadth of the front face of the NEMA 8 stepper motor is 20 mm and 20 mm. The diameter of the designed rotary rod is 13 mm. In order to facilitate a sufficient space between the motor and the rotary rod for spring and rotary rod gear, the centres of rotation of the rotary rod and the NEMA 8 stepper motor were placed 20 mm apart. Since the step accuracy of the motor is sufficient, no gearing ratio is required. Therefore, the most space efficient manner of connecting these two rotary centres is with two gears with a pitch diameter of 20mm each. Since this is not a specialised application of these gears, relatively standard parameters were selected for their design. The gear parameters as designed in Fusion 360 and used for both gears are as follows:

- Pressure Angle = 20°
- Module = 1
- Number of Teeth = 20
- Backlash = 0.3 mm
- Root Fillet Radius = 0.3 mm
- Pitch Diameter = 20 mm

A relatively large backlash of 0.3 mm was selected due to the high tolerance required by 3D printed parts. A pressure angle of 20° and 25° are the most commonly used angles in gear design. A smaller pressure angle has weaker teeth but runs quieter. Due to the low torque nature of this gear application, a 20° pressure angle was selected to take advantage of the qualitative low noise benefit.

The point of entry for calculating the various mass elements that need to be translated and rotated is the aluminium cube that needs to be manipulated. The density of aluminum is $2.7g/cm^3$. Since the cube has a maximum side length of $1.3cm$, the cube has a maximum mass of $5.94g$. Using the updated design of the end-effector rod, the 3D model was converted to a triangle mesh and exported to the 3D printing slicer Cura. Using PLA 1.75mm filament, Cura estimated the weight of the part to be 7g when printed at 50% infill. The total rotating mass in the end-effector was 20.74 g.

The system does not have any explicit angular velocity and angular acceleration specifications that it is required to meet from the project proposal. Therefore, it is decided that on a qualitative basis that the end-effector should be capable of completing a single rotation once every 4 seconds when at maximum velocity. Similarly, it is also decided that the end-effector should be capable of reaching this speed in 0.5 seconds from standstill. The angular velocity is computed as shown below in Equation 1.

$$\omega = \frac{\Delta\theta}{\Delta t} = \frac{\pi}{2} \text{ rad/s} \tag{1}$$

The maximum angular acceleration the motor should be capable of driving is as calculated in Equation 2 below assuming that the system accelerates linearly to the maximum velocity from standstill.

$$\alpha = \frac{\Delta\omega}{\Delta t} = \frac{\pi}{4} \text{ rad/s}^2 \tag{2}$$

In order to calculate the torque required to rotate the end-effector component, the moment of inertia $I_O$ about the centre of rotation $O$ of the component needs to be computed. The component is modeled as a cylinder with a radius of 10 mm and all of its mass located in its outer shell only. This is the cylindrical configuration that has the greatest moment of inertia and therefore requires the greatest torque to rotate. This guarantees that a motor capable of rotating this is capable of rotating the actual component. The moment of inertia for the cylindrical model is calculated as shown in Equation 3 below

$$I_O = mr^2 = 2.074 \times 10^{-6}\, kg \cdot m^2 \tag{3}$$

where $m$ is the mass of the cylinder and $r$ is the radius of the outer shell of the cylinder. In order to relate the kinematic motion of the rotary end-effector component to the external forces applied to it, the moment equation for rotation about a fixed axis $O$ shown in Equation 4 below can be used

$$\sum M_{Oi} = \sum F_i r_i = I_O \alpha \tag{4}$$

where $M_{Oi}$ is the moment that results from the application of force $F_i$ at the $i^{th}$ point at perpendicular distance $r_i$ from the axis of rotation $O$. Two external forces to the rotary end-effector component are considered, namely the force of static friction $F_{fs}$ and the force $F_A$

applied to the component gear by the gear on the motor's drive shaft. Only static friction is considered as it is generally greater than kinetic friction and therefore more challenging to overcome. The coefficient of static friction for plastic on plastic used in these calculations is $\mu_s = 0.4$. Furthermore, it is assumed that the friction only acts along the outer edge of the cylinder model since this requires the most torque to overcome. Using Equation 4, the force required to accelerate the end-effector component from standstill at the required rate can be calculated as shown in Equation 5

$$F_A r_A - F_{fs} r_{fs} = I_O \alpha \tag{5}$$

$$F_A = 81.52 \times 10^{-3} N \tag{6}$$

where $r_A$ and $r_{fs}$ are the distances between the point of force application and the centre of rotation O and for the forces $F_A$ and $F_{fs}$ respectively. Since the gear on the motor shaft is to be 3D printed, its mass is taken to be negligible. Since the pitch circle radius $r_m$ of the motor gear is 10 mm, the torque $\tau$ required to be generated by the motor in order to exert $F_A$ on the end-effector rotor is calculated as shown in Equation 7 below.

$$\tau = F_A \times r_m = 815.2 \times 10^{-6} N \cdot m \tag{7}$$

$$\tag{8}$$

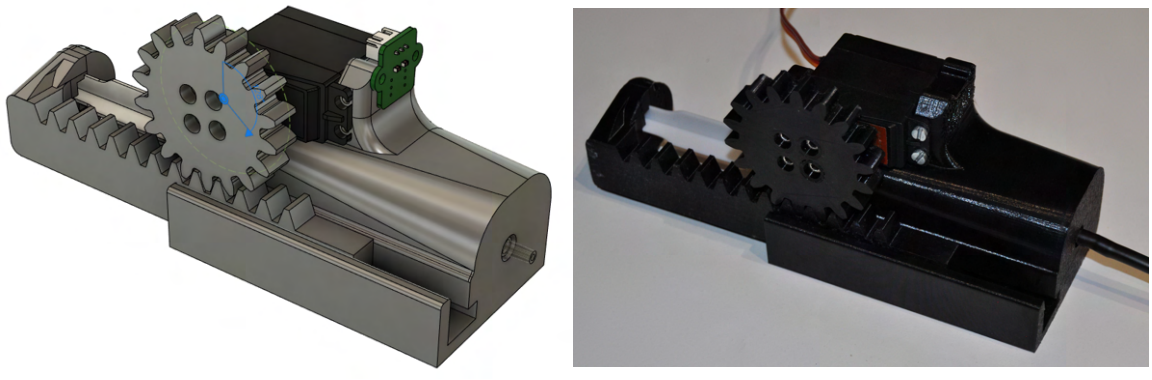
Using an engineering safety factor of 2 to account for inaccuracies in modeling the end-effector rotor and to ensure there is at least a 50% torque margin for the motor, a minimum required motor holding torque of $1.63 \times 10^{-3}\ N \cdot m$ or $16.63\ g \cdot cm$. Using the Wantai Motor product line as a reference, the smallest available stepper motor is the 20BYGH202 model which has a holding torque of $\tau_H = 140\ g \cdot cm$, detent torque of $\tau_D = 20\ g \cdot cm$ and a mass of $50\ g$. The detent torque of the motor refers to the amount of torque generated by the motor when the windings of the motor are not energized. The holding torque, on the other hand, is the amount of torque required to rotate the motor one step when the rotor is stationary but the windings are energized. The running torque $\tau_R$ of the motor is limited by the motor's current rating and at low speeds is equal and can be calculated using Equation 9 below.

$$\tau_R = \tau_H - 2\tau_D \tag{9}$$

The running torque of the 20BYGH202 model is calculated as $\tau_R = 100\ g \cdot cm$ which comfortably meets the torque requirement of $16.63\ g \cdot cm$. The excess torque can be used to implement greater acceleration or micro-stepping functionality. In the interest of keeping costs down for the project, the ISG component bank was considered when sourcing the motor. The smallest NEMA 8 motor contained in the bank was the 20BYGH406 which has a holding torque of $\tau_H = 260\ g \cdot cm$, detent torque of $\tau_D = 50\ g \cdot cm$ and a mass of $80\ g$ which yields a running torque of $\tau_R = 160\ g \cdot cm$ which also comfortably meets the torque requirement. The additional $30\ g$ of mass was considered acceptable for the project and, therefore, the 20BYGH406 model was selected as the end-effector motor.

### *3.2.2 Vacuum Actuation Mechanism*

A syringe was used as the basic component to generate the vacuum within the vacuum system. The syringe has a maximum range of linear travel of 78 mm. The servo has a maximum range of rotational motion of 180°. Therefore, in order to achieve the full range of linear motion for the syringe, the mechanism connecting the servo to the syringe needs to translate the servo's 180° of rotational motion into 78 mm of linear motion. A rack and pinion system is a mechanism that is commonly used for converting translational motion into rotational motion. Therefore, half of the circumference of the pitch diameter circle of the gear needs to be equal to the length of linear travel for these specifications to be achieved. A gear with a pitch diameter of 49.66 mm satisfies this requirement. Therefore, it was selected to use a gear with a pitch diameter of 50 mm. The mechanical connection mechanism to connect the syringe to the servo motor is shown as a 3D model in Figure 1a as well as the physical realisation in Figure 1b below.



**(a) 3D vacuum actuator model**

**(b) Physical vacuum actuator**

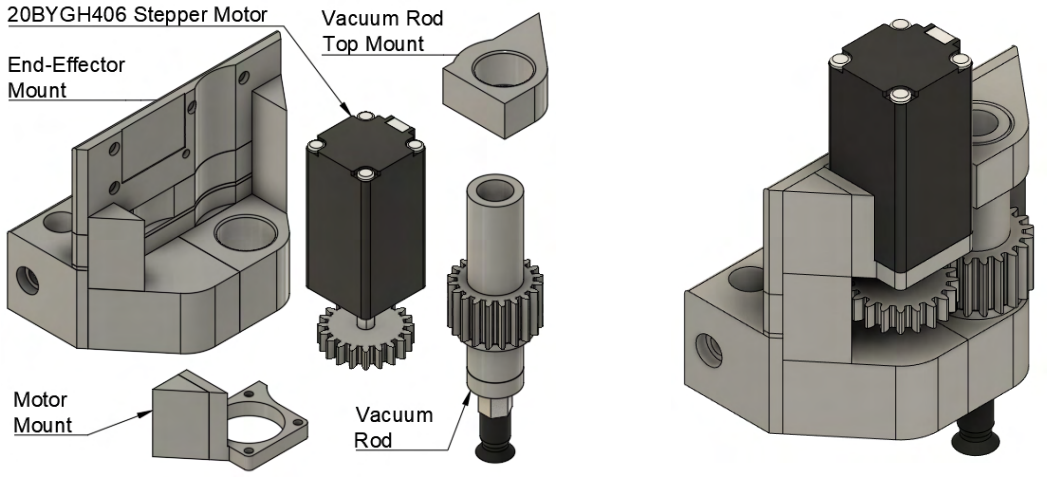
**Figure 1. Vacuum generation mechanism for the end-effector mechanism.**

### *3.2.3 End-Effector Assembly*

The end-effector mount was designed to fulfil to following requirements:

- It must provide a mounting structure for both the 20BYGH406 stepper motor and the vacuum rod such that the gear components of each are aligned and able to mesh correctly.
- It must facilitate at least 5mm of translational motion of the vacuum rod along the z-axis.
- It must provide a connection point for connecting the end-effector mount to the rest of the robotic subsystem.
- It must facilitate assembly of the end-effector mount, 20BYGH406 stepper motor and vacuum rod components.

Figure 2a shows the end-effector mount and supporting components that were designed to meet these requirements along with the 20BYGH406 stepper motor and vacuum rod. The end-effector mount, motor mount and vacuum rod top mound were originally designed as a single piece. The vacuum rod top mount was separated as a component to facilitate the insertion of the vacuum rod into the end-effector assembly. The motor mount was similarly separated to facilitate the manufacturing of this part using FDM 3D printing methods as the overhang was not capable of being printed without support. Figure 2b shows the same components when assembled to form the complete end-effector mount assembly.




**(a) Disassembled end-effector mount**

**(b) Assembled end-effector mount.**

**Figure 2. End-effector mount assembly that provides the mechanical connection between the end-effector mechanism and the robotic manipulator.**

### *3.2.4 Z-Axis Assembly*

For the purposes of this project, the z-axis assembly is defined as the collection of mechanical components that move linearly along the z-axis when the z-axis drive mechanism is activated. In the case of this project, the z-axis drive mechanism is included in the z-axis assembly. There are many different approaches to the z-axis mechanism. One of the most common approaches which is often used in CNC applications involves the use of a linear guide which is fixed with respect to the motion of the z-axis assembly. The z-axis drive mechanism is also fixed with regards to the z-axis assembly. In other words, when the z-axis drive is activated, the z-axis assembly moves relative to the linear guides and the linear drive. The advantage of this approach is that the moving mass of the z-axis assembly is minimised as the linear guides and the z-axis drive mechanism are fixed relative to the motion. The linear bearings form part of the z-axis assembly in this case. The disadvantage of this approach is that a connecting component is required to connect the end-effector to the point at which the linear bearings connect to the linear guides. As such, the greater the range of motion along the z-axis that is required, the longer this connecting element has to be. This introduces a potential area of play as the connecting component is prone to a greater degree of flex the greater its length is.

Therefore, this approach is only suited to tasks in which the range of motion along the z-axis is minimal.

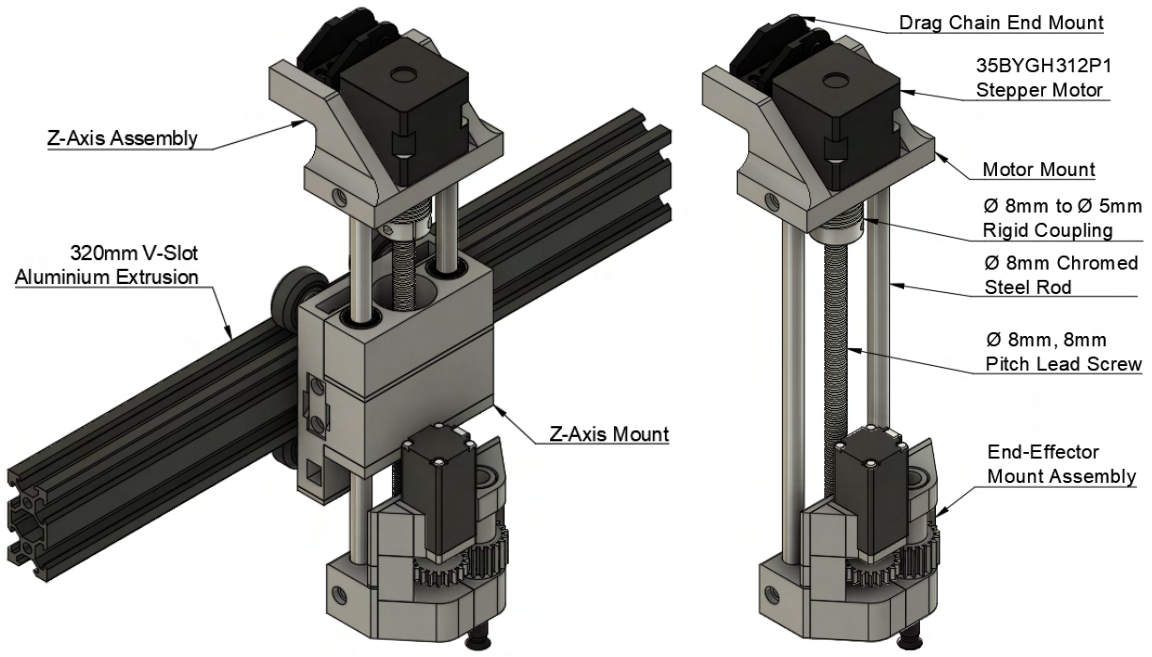
Another variation of this approach is to include the linear guides as well as the z-axis drive as part of the z-axis assembly that moves along the z-axis. In this arrangement, the linear bearings are fixed with regards to the motion of the z-axis assembly. The disadvantage of this approach is that there is greater moving mass along the z-axis but this is allows a greater range of motion to be achieved along the z-axis. Since the nature of this project requires a reasonably large range of motion along the z-axis, this latter approach was selected.

With the approach including both the z-axis drive mechanism as well as the linear guides as part of the z-axis assembly, there are two primary design decisions. These are with regards to the selection of the linear guide components as well as the selection of the linear drive mechanism. The most popular drive mechanisms are the lead screw drive solution and the timing belt based drive mechanism. Timing belts are beneficial when the load needs to be moved over a relatively large distance at a relatively high speed. However, they generally require larger motors with more torque. Therefore, they are suited to motion along directions where they are not working against gravity. Lead screw drives, on the other hand, generally require small motors with less torque to drive the same load and are suited to moving loads slowly and accurately over small distances. Since the z-axis motor forms part of the moving z-axis assembly in the design approach discussed above, a smaller motor is preferable to reduce the moving mass of the assembly. Secondly, the range of motion along the z-axis is relatively small compared to the range of motion required along the x and y axes. Therefore a lead screw drive approach was selected for the z-axis assembly.

There are two popular linear guide mechanisms with regards to motion along the z-axis. The first is the linear rail guide which has excellent characteristics when it comes to resisting forces and torque in any other direction than its line of travel. Unfortunately, these preferable characteristics come at a cost. The linear rails in themselves are not completely stiff and need to be mounted to a supporting structure such as an aluminium v-slot extrusion which would increase the moving mass of the z-axis assembly to an unreasonable level. Secondly, the financial cost of the linear rails per relatively high. Therefore, linear rails were not selected for linear motion along the z-axis. Rather the alternative linear guide mechanism, namely linear rods, was selected instead. Specifically, 8mm diameter chromed steel linear rods were selected since they are one of smallest linear rod diameters available and the vertical nature of the z-axis assembly does not exert much torque on the linear rails. Figure 3b shows how all of these components are integrated to form the final z-axis assembly.

### *3.2.5 Z-Axis Mount*

It was decided to use aluminium v-slot extrusions as the foundation of the robotic subsystem primarily due to the design flexibility offered this structure. The profile of the extrusion offers considerable stiffness required by the frame structure as well as mounting locations along any position of the structure by means of T-nuts. Furthermore, the v-shape present along the grooves in the extrusion allow the extrusion to be used as a linear guide. The extrusion was not considered as an option for a linear guide in the z-axis assembly due to its comparatively large mass compared to the mass of the z-assembly. However, in comparison to the mass of




**(a) X-axis assembly.** **(b) Z-axis assembly.**

**Figure 3. Assembled z-axis assembly and its integration into the x-axis assembly.**

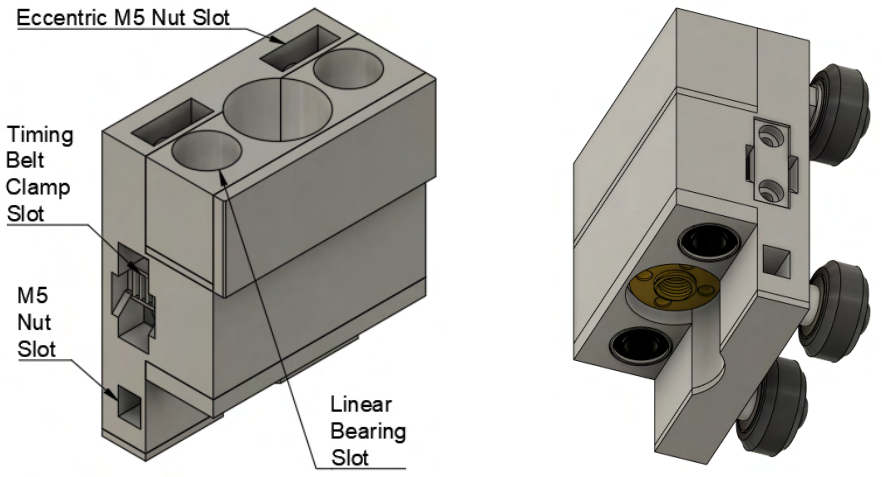
the x-axis assembly, the mass of the extrusion is much more reasonable and offers greater structural support in comparison to other linear guides such as linear chromed steel rods. The fact that the aluminium v-slot extrusion approach was selected for the rest of the robotic subsystem structure combined with the reasons discussed earlier was considered sufficient to select an aluminium extrusion as the x-axis linear guide. To assist in countering the torque generated about the x-axis by the z-axis assembly, it was decided to use a 2040 aluminium v-slot extrusion as opposed to a 2020 aluminium v-slot extrusion.

Linear motion along the aluminium v-slot extrusion is facilitated by v-slot wheels that run along the v-shaped grooves of the extrusion. Therefore, a need for a component to connect the z-axis assembly to the x-axis aluminium v-slot extrusion arose. The specific requirements of the required component are outlined below:

- The component needs to connect the 4 v-slot wheels positioned to run along the 2040 aluminium v-slot extrusion to the 4 linear bearings through which the linear chromed steel rods of the z-axis assembly run.
- The component needs to accommodate the excess movement required by the eccentric nuts used by half of the v-slot wheels.
- The component needs to provide mounting points for the timing belt on either side of the component.
- The component needs to provide a mounting point for the lead screw nut.
- The component needs to accommodate the length of the 8mm diameter to 5mm diameter rigid coupling used to attach the lead screw to the z-axis motor in the z-axis assembly. This accommodation allows the z-axis motor to move lower along the z-axis relative to

the fixed linear bearings. This results in a shorter moment arm to the z-axis motor and reduces the torque about the x-axis.
- The component needs to be capable of being manufactured using FDM 3D printing techniques.
- The component needs to support assembly with all of its supporting components.

Figure 4a shows the z-axis mount developed to meet these requirements with various design features relating to these requirements highlighted. The walls of the eccentric spacer measure 1.76 mm and 0.34 mm. That means the bolt may be offset a maximum of 0.71 mm in any direction from the centre of the outer radius of the eccentric spacer which is accommodated in the design by the feature identified as the eccentric M5 nut slot. Figure 4b shows the z-axis mount assembled. The 8mm pitch lead screw nut can be seen centred near the bottom of the component. One of the timing belt clamps can be seen placed over the timing belt clamp slot on the side of the z-axis mount.




**(a) Plain z-axis mount** **(b) Assembled z-axis mount**

**Figure 4. 3D design of the z-axis mounting component and its integration with supporting components.**

### *3.2.6 X-Axis Assembly*

For the purposes of this project, the x-axis assembly is defined in a similar manner to the z-axis assembly. Specifically, the x-axis assembly is the collection of components that move linearly along the x-axis when the x-axis drive motor is activated. Similar to the z-axis assembly, a decision needs to be made regarding whether to position the x-axis drive motor as part of the x-axis assembly or not. There is no significant benefit to including the x-axis motor as part of the x-axis assembly either than it potentially reduces the complexity of the x-axis mount which no longer needs to support the motor as well. However, this is offset by the complexity increase in the z-axis mount which would need to include a mounting point for the x-axis motor. Furthermore, mounting the motor in this manner would increase the mass of the x-axis

assembly which would require a larger motor with more torque to drive. For these reasons, it was selected to not include the x-axis motor in the x-axis assembly. Note, the x-axis assembly does not include the x-axis linear guide in the form of the 2040 aluminium v-slot extrusion as it does not move when the x-axis drive is activated.

The second design consideration is the selection of the drive mechanism. Again, a lead screw approach was considered against a timing belt based approach. In this situation, the drive mechanism does not need to work directly against gravity as was the case with z-axis assembly. Furthermore, the range of motion of the x-axis assembly is significantly greater than that of the z-axis assembly. Based on these factors, in conjunction with the discussion of the drive mechanisms covered during the z-axis assembly analysis, the timing belt approach was selected. Due to the existence of the v-wheels to connect the z-axis mount to the aluminium extrusion, it was identified as being significantly more complicated to route the timing belt with the flat face parallel to the base plane. However, there were no obvious restrictions in routing the timing belt with the front face parallel to the x-axis aluminium v-slot extrusion. Furthermore, the aluminium v-slot extrusion facilitates the routing of the far side of the belt through the centre of the extrusion. For these reasons, it was decided to route the timing belt in this manner. Therefore, with all of the above considered, the x-axis assembly is defined to only consist of the z-axis mount as well as the z-axis assembly. Figure 3a shows x-axis assembly positioned on the x-axis aluminium v-slot extrusion.

### *3.2.7 Y-Axis Assembly*

For the purposes of this project, the y-axis assembly is defined in a similar manner to the z-axis and x-axis assemblies. Specifically, the y-axis assembly is the collection of components that moves linearly along the y-axis when the y-axis drive is activated. At this point in the design, the x-axis assembly is the highest level component of the robotic subsystem. The x-axis assembly runs along the x-axis aluminium v-slot extrusion. In order to introduce linear motion along the y-axis, both of these components need to be translated and therefore will both form part of the y-axis assembly.

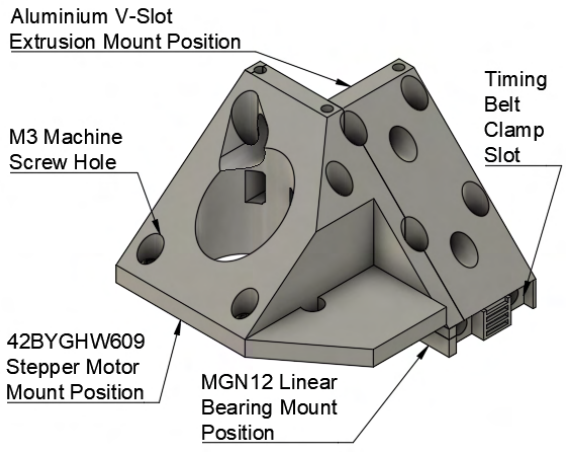
In order to facilitate linear motion along the y-axis, linear guides need to be introduced into the design. Again, the most popular two linear guides were considered, namely the linear rail and the linear chromed steel rod. A consideration that applies to the y-axis motion that did not apply to the linear guides used on the other axes is the fact that the y-axis linear guides will always be fixed relative to the robotic structure. Therefore, weight is not a consideration in the selection of the linear guides. Furthermore, it has already been noted that aluminium v-slot extrusions are used to form the frame of the robotic subsystem. These extrusions facilitate simple mounting of linear rails by means of T-nuts. Furthermore, all of the mechanical advantages of the linear rail over the linear chromed steel rod as discussed earlier still apply. Since the y-axis assembly has the greatest mass of any of the moving components, the mechanical advantages offered by the linear rail were considered more relevant here than in earlier parts of the design. V-wheels were also considered as a means of linear motion along the v-slot aluminium extrusion. However, exploration of potential designs using this mechanisms exhibited many issues with routing the timing belt for the x-axis as well as the y-axis. Furthermore, the aluminium spacers would introduce additional length along the

x-axis without any increase in the range of motion along the x-axis.

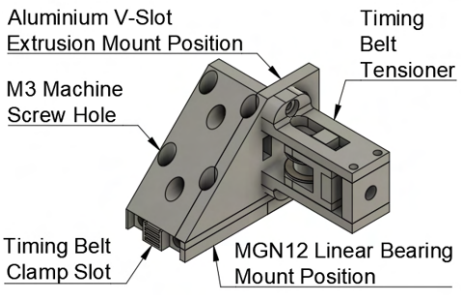
For these reasons, the linear rail was selected as the linear guide along the y-axis. Furthermore, since both sides of the x-axis aluminium v-slot extrusion need to be supported, it was decided to use two linear rails, one on either side of the x-axis aluminium extrusion. The MGN12H linear bearing acts as the connection between the linear rail and the load item. Therefore, a requirement for two components to connect both ends of the x-axis aluminium extrusion to the MGN12H bearings arose. The requirements of the first component are as follows:

- The component needs to connect the left side of the x-axis 2040 aluminium v-slot extrusion to the left MGN12H linear bearing.
- The component needs to provide a mounting point for the 42BYGHW609 stepper motor such that the motor is in a position to drive the x-axis timing belt.
- The component needs to provide a timing belt clamping point on both sides of the component for the left y-axis timing belt.
- The component needs to facilitate the routing of the x-axis timing belt.
- The item needs to provide a mounting point for the end piece of the drag chain to facilitate the routing of wires and tubing originating from the z-axis assembly as well as the 42BYGHW609 stepper motor cables.
- The component needs to be capable of being manufactured using FDM 3D printing techniques.

Figure 5a shows the left x-axis mount that was designed to meet these requirements along with indications of the roles of the various parts of the design. Similarly, the right x-axis mount was designed in a similar manner and the result is shown in Figure 5b.




**(a) X-axis assembly left mount.**

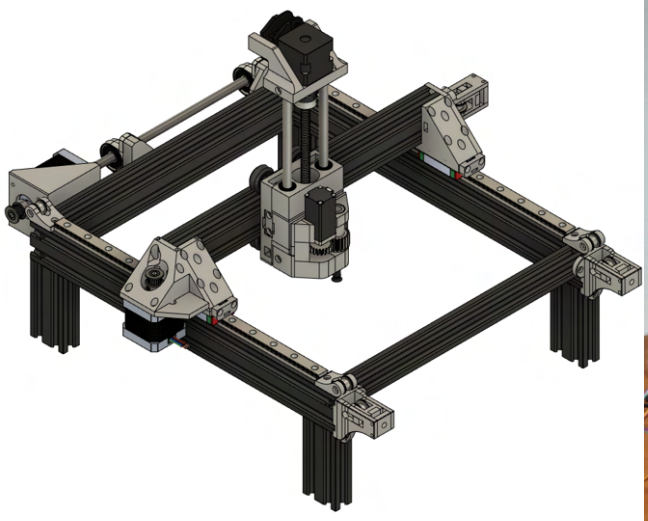



**(b) X-axis assembly right mount.**

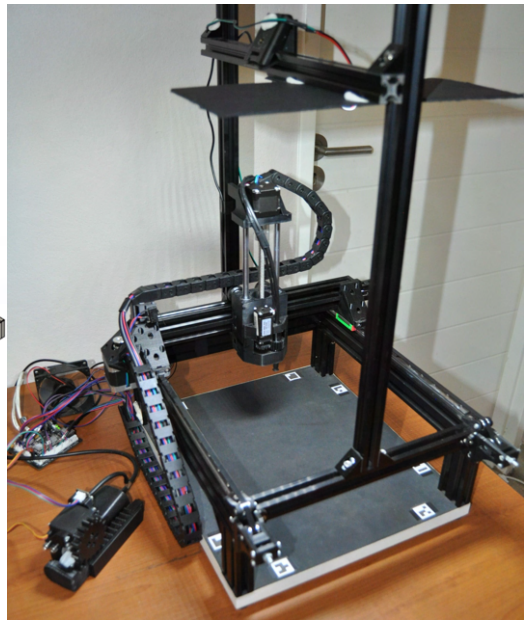
**Figure 5. Mounting pair for the left and right sides of the x-axis assembly.**

### *3.2.8 Final Assembly*

The primary requirement of the y-axis drive mechanism is that it needs to be capable of driving the y-axis timing belts on both the left and right side of the robotic subsystem. The obvious basis of this mechanism is to use a dual shaft stepper motor with each shaft used to drive one of the y-axis timing belts. The 42BYGHW920L21B2 stepper motor was selected for this characteristic. Unfortunately, the length of the stepper motor is not sufficient to drive each belt directly off each shaft. Instead it was decided to drive only the left side y-axis timing belt directly off the shaft using a 20 tooth pulley for a 6mm GT2 timing belt with a 5mm bore. In order to drive the right side y-axis timing belt, the torque needs to be transferred from the stepper motor shaft to a pulley connected to the timing belt. It was decided to use an 8mm linear chromed steel rod in order to transfer this torque to a 20 tooth pulley for a 6mm GT2 timing belt with a 8mm bore. An 8mm diameter to 5mm diameter coupling is required to connect the stepper motor shaft to the 8mm linear rod. The rigid coupling is not sufficient to support the linear rod along the drive axis. Therefore, two KP08 8mm pillow block bearings were introduced to support the linear rod. In order to connect the pillow blocks to the aluminium v-slot extrusion frame of the robotic subsystem, two custom connector components needed to be designed. Similarly, a component needed to be designed to connect the 42BYGHW920L21B2 stepper motor to the frame as well. All of the components discussed above form the y-axis drive mechanism. The final robotic system assembly design is shown in Figure 6a. The physical realisation of this design is shown in Figure 6b.



**(a) Complete robot 3D model**



**(b) Constructed robot**

**Figure 6. The *Robotic System* in both the final designed and final constructed states.**

## 3.3 Embedded Robot Controller

The *Robotic System* required an embedded controller to drive the end-effector mechanism and the robotic manipulator developed as part of the mechanical robotic component in Section 3.2 as well as to facilitate communication with the *PC System*. Specifically, in service of these requirements, the embedded controller needed to fulfill the following functions:

- The controller should provide control signals to control the four robotic manipulator stepper motors as well as the vacuum mechanism servo motor.
- The controller should capture the pressure sensor reading in the vacuum system.
- The controller should facilitate bi-directional communication with the *PC System* such that the *PC System* can send control commands to and receive status information from the *Robotic Subsystem*.
- The controller should distribute energy from a power supply to the components in the *Robotic System* including the stepper motors, servo motor, cooling fan (for the embedded controller) and robot workspace lights.
- The controller should monitor the robotic manipulator limit switches on each Cartesian axis.

The hardware considerations of the circuit designed to fulfill these requirements are outlined in Section 3.3.1 while the development of the physical realisation of this circuit in the form of a printed circuit board (PCB) is presented in Section 3.3.2. Following this, the firmware implementations developed to fulfill various facets of the embedded controller's functional requirements are discussed.

### *3.3.1 Circuit Design*

As noted in the project proposal, the motor drivers for the robotic manipulator stepper motors were taken off-the-shelf. The DRV8825 stepper motor driver was selected for this purpose since the driver is capable of supplying up to 1.5 A to the motor which is sufficient for the selected robotic manipulator motors. Specifically, the DRV8825 driver breakout board was selected for use in this project. Each DRV8825 driver exposes four output drive pins to control the current in both coils in the connected stepper motor. Furthermore, the DRV8825 driver exposes a number of input pins to control the operation of the motor and the driver.

The DRV8825 driver control inputs and status outputs of interest for this project are shown in Table 3 and their use in this project is detailed in Section 3.3.3. Therefore, the microcontroller around which the embedded controller is based needs to provide seven digital output pins and one digital input pin per stepper motor. Furthermore, the STEP pin needs to be controlled with time-sensitive signals and as such one timer peripheral is required for each stepper motor. Since the robotic manipulator makes use of four stepper motors, a total of 28 digital output pins, four digital input pins and four timer peripherals are required in the microcontroller to control the motors.

| Pin No. | Pin Name(s) | Type | Description |
|---|---|---|---|
| 2,3,4 | M0, M1, M2 | Input | Microstepping resolution selection |
| 5 | $\overline{\text{RESET}}$ | Input | Reset the driver's step indexer |
| 6 | $\overline{\text{SLEEP}}$ | Input | Place the motor in a low-power sleep state |
| 7 | STEP | Input | Motor rotation direction selection |
| 8 | DIR | Input | Advance the motor one step on rising edge |
| 10 | $\overline{\text{FAULT}}$ | Output | Overheat and overcurrent event flag |

**Table 3. DRV8825 breakout board control and status pins utilised.**[1]

In order to facilitate serial communication with the *PC Subsystem*, the UART protocol was selected as the basis for this communication. The selection was made as the communication with the *PC System* was expected to only consist of simple asynchronous control commands and status updates which are easily supported by UART. The universal serial bus (USB) interface of the *PC System* is not directly compatible with the UART serial interface peripheral and, as such, the CH340G IC was selected as the communication data conversion point between these two interfaces. The CH340G exposes a transmitter (TX) pin and receiver (RX) pin for the microcontroller to interface with. Therefore, the microcontroller requires one digital input pin, one digital output pin and a UART peripheral to support serial communication. The serial communication component is discussed further in Section 3.3.4

In addition to the stepper motor and UART control pins, a single digital output pin was required to control the vacuum system servo motor along with a timer peripheral to act as the time base for the pulse width modulation (PWM) signal. An analog input pin, in conjunction with a corresponding analog-to-digital converter (ADC) peripheral, was required to capture the ADP5111 gauge pressure sensor reading. Lastly, for the purpose of reducing the power consumption of the system in idle state, it was elected to make both the robot workspace lighting and embedded controller cooling fan controllable through software. To this end, two additional digital output pins were required in the microcontroller.

In terms of performance, the microcontroller needed to have the computational power to control the four stepper motors almost simultaneously using acceleration profiles. At the same time, the microcontroller needed to control the vacuum system servo motor, monitor the vacuum system pressure and support serial communication with the *PC System*. Furthermore, the use of microstepping to achieve smooth stepper motor motion (see Section 3.3.3) also increased the computational load as more step pulses needed to be sent to the stepper motor drivers per unit time. The use of an 8-bit microcontroller was considered since it could potentially achieve this performance with highly optimised code. However, in general, 32-bit microcontrollers offer greater computational power which facilitate greater leeway in the stepper motor control computations. Furthermore, the current trend in industry, especially in the 3D printing space, is towards 32-bit controller boards. For these reasons, it was decided to use a 32-bit microcontroller as the basis for the embedded platform. Lastly, with regards

[1] The overline notation, $\overline{\text{PIN}}$, on a pin name indicates the control input or status output is active low.

to clock speed, controllers with similar requirements to the controller in this project have been successfully implemented using 16 MHz microcontrollers. Therefore, 16 MHz was considered to be the minimum acceptable microcontroller clock speed.

The STM32L072RZT6 microcontroller was selected as the computational foundation for the embedded controller in this project since it satisfied all the digital I/O, peripheral and computational speed requirements discussed above. Note that this microcontroller is part of the STMicroelectronics low-power microcontroller range. The low-power property of this range wasn't strictly necessary for this project since the *Robotic System* uses mains electricity for power. However, due to availability issues resulting from the global semi-conductor shortage, the STM32L072RZT6 was chosen over similar microcontrollers that would not be available for the foreseeable future.

### *3.3.2 PCB Design*

Before the development of the PCB began, a breadboard based prototype of the embedded controller was first created to identify any issues present with the circuit design. In order to provide breadboard access to the SMD STM32L072RZT6 microcontroller, the device was soldered to a TQFP breakout board with a pin pitch of 0.5mm. Due to the fine pitch of the pins, solder flux was required to be used in conjunction with the drag solder hand soldering technique for the process to be successful. Furthermore, several bridges formed during the initial drag solder pass which were removed using solder wick. Finally, the solder flux residue was cleaned using IPA. The final robotic controller prototype is shown in Figure 7.

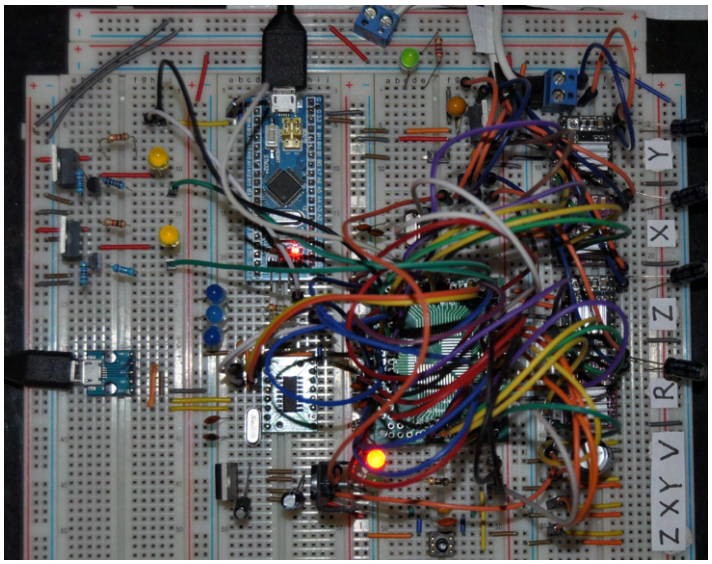



**Figure 7. Embedded robotic controller prototype.**

After the prototype controller was confirmed to be functioning correctly, the PCB design process was initiated. The PCB design was developed using the KiCAD schematic capture and PCB design software. The design process consisted of two steps. The first step involved the development of the electrical design in which all the electrical component pins were

assigned to electrical nodes. In other words, the component connections were defined in this step. This was followed by the PCB component layout and trace routing step. In order to determine the width of the power supply trace used on the PCB, the maximum current that could flow on that trace needed to be determined based on the current ratings of the constituent components. The maximum current ratings of each of the components are listed below:

- 42BYGHW920L21B2 stepper motor - 2.2 A
- 42BYGHW609 stepper motor - 1.7 A
- 35BYGH312P1 stepper motor - 1.2 A
- 20BYGH406 stepper motor - 0.6 A
- DS3118MG servo motor - 2 A
- STM32L072RZT6 microcontroller - 105 mA
- CH340G chip - 30 mA.

Based on the sum of these values, the initial power connection trace needed to support 5.835 A of current. A current value of 6 A was used to incorporate an engineering safety margin. Furthermore, a maximum temperature rise tolerance of 10 $^{\circ}C$ and a copper thickness of 1oz/ft$^2$ was used in the calculation of the trace width. Based on this, it was calculated that a trace width of at least 140 mil, or 3.56 mm, was required. A similar procedure was used to calculate the widths of power traces that supported fewer components. For the thickness of the PCB traces, it was noted that 2 oz outer copper weight is generally notably more expensive than 1 oz outer copper weight. Therefore, the latter was used for traces for the PCB developed in this project.

The D+ and D- lines for the USB portion of the USB to serial converter constitute a differential pair. As such, a number of PCB layout guidelines needed to be adhered to, to ensure the integrity of the differential pair signal. The following guidelines were followed for this purpose:

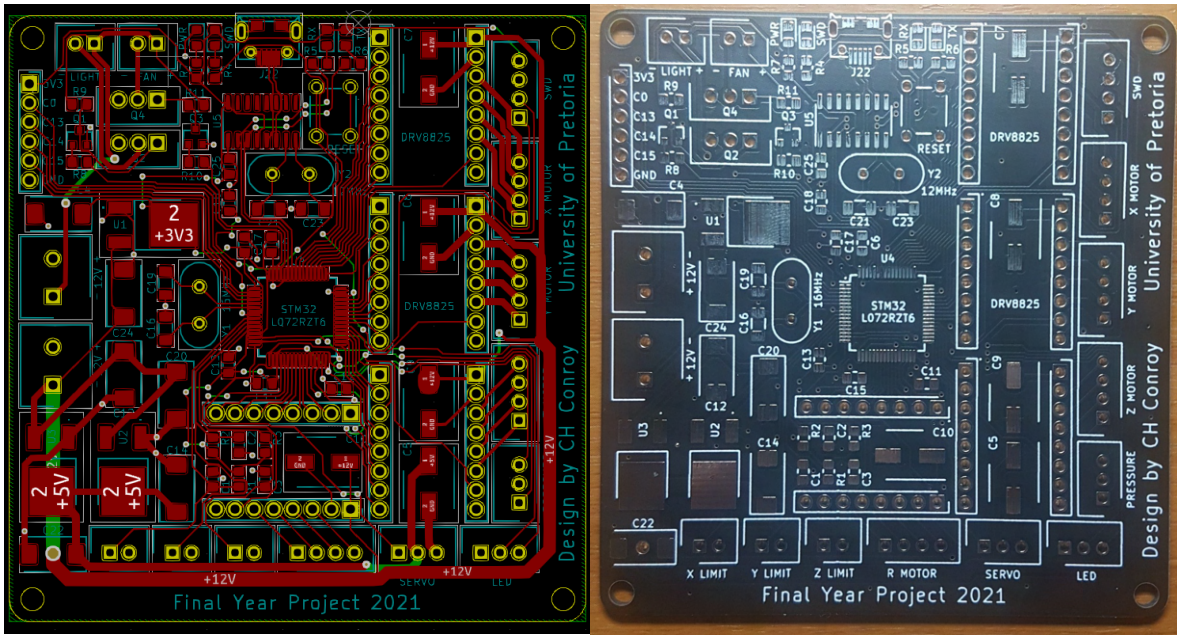
- The use of vias should be minimised.
- The differential pair should be isolated from the other traces.
- Differential pair traces should mirror each other as far as possible.
- The lengths of each trace in the differential pair must be identical even if symmetry needs be sacrificed to achieve this.

In this project, vias were not used at all for the differential USB signal traces. Additionally, the trace isolation condition was satisfied by enforcing a clearance of three standard trace widths from other traces for a total clearance of 0.75mm. Similarly, for both the STM32L072RZT6 external oscillator and the CH340G external oscillator, the following guidelines were adhered to during the PCB layout process:

- The crystal and its supporting capacitors should be placed as close as possible to the oscillator input and output pins on the microcontroller.
- The trace length in the oscillator circuit should be minimised.
- The traces in the oscillator circuit should not cross other signal lines.
- Traces should not incur right angle bends.
- The supporting capacitors should share a ground plane.
- The size of loops in the oscillator circuit should be minimised.
- The ground node should not pass under the crystal.

- Power and digital signal on other layers of the board should not pass under the crystal.

In addition to the PCB design components discussed above, the PCB was designed in such a manner that all the components were placed on the top layer of the two layer PCB. The top layer was referred to as the signal layer since traces were routed on the top layer as far as possible. Only when necessary, the traces were routed onto the bottom layer to pass below other traces before being returned to the signal layer by means of vias. This routing strategy allowed a ground plane fill to be applied to the majority of the bottom plane. This improved the radio frequency (RF) characteristics of the PCB as higher frequency signal components could return much more closely to their signal path which minimised the current loop size. Furthermore, the trace routing was simplified as an electrical node on the signal plane could be grounded by simply adding a via to the ground plane. The PCB layout with the routing complete is shown in Figure 8a while the physical realisation of this design after manufacturing is shown in Figure 8b. A 3D model was created using KiCAD's rendering functionality to identify any logistical issues that could arise during the assembly process due to the 3D form factor of the components. This model is shown in Figure 9a. The final manufactured PCB after assembly is shown in Figure 9b without the motor drivers inserted.

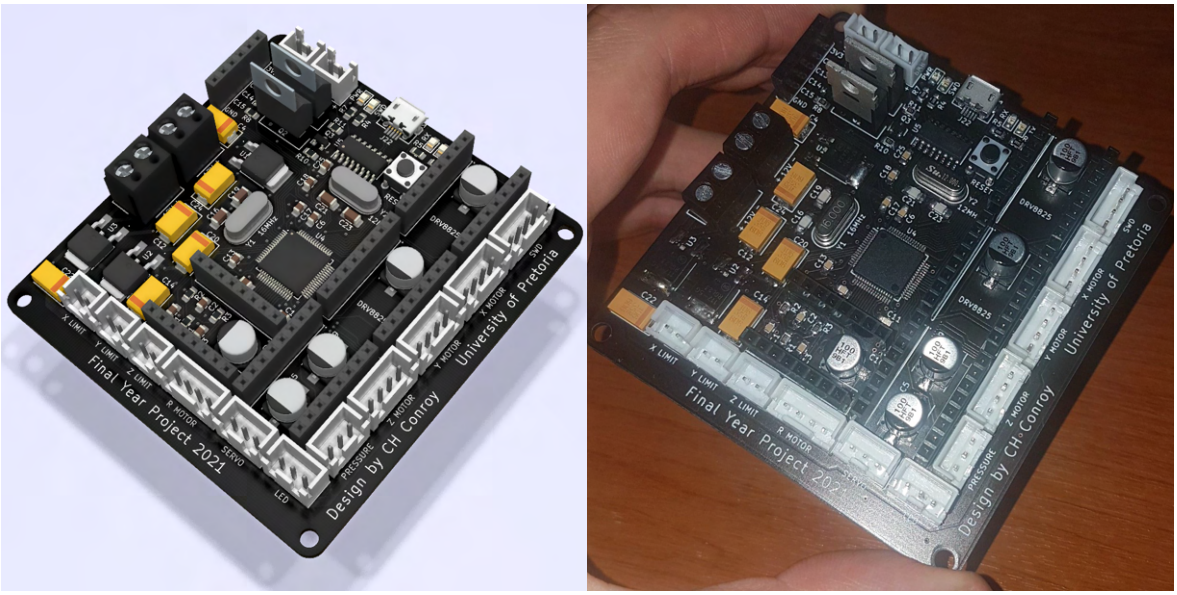



**(a) Embedded controller PCB routing** **(b) Manufactured controller PCB**

**Figure 8. Design of the PCB board component and trace layout before and after manufacturing.**

### *3.3.3 Motor Control*

The DS3118MG servo motor used as the actuation driver for the vacuum generation mechanism and has a angular rotation range of 180°. The medium of control for the servo motor is a PWM signal that has two degrees of freedom which motor responds to. Firstly, the period of




**(a) Embedded controller PCB 3D model** **(b) Assembled controller PCB**

**Figure 9. Final PCB after all components have been assembled.**

the PWM signal determines the fraction of the angular rotation range available to the servo motor. The input period can be set from 3 ms to 20 ms (333 Hz to 50 Hz) where a period of 20 ms corresponds to the full range of rotation for the servo. Since the vacuum generation mechanism was designed on the assumption that the servo operates with 180° of rotational motion, 20 ms was chosen as the PWM period.

The second degree of freedom in control of the servo offered by the PWM signal exists in the high-time of the pulse, or pulse width. The position of the servo is altered by changing this parameter between $500\mu s$ and $2500\mu s$ where the bounds of this range correspond to the 0° and 180° servo motor positions[2]. The release, idle and actuate positions of the vacuum generation mechanism were selected by setting the PWM pulse width to either $2400\mu s$, 2100 $\mu s$ or 1500 $\mu s$ respectively.

As mentioned in Section 3.3.1, the DRV8825 drivers were selected to provide an interface to the microcontroller to control the stepper motors. This interface includes the *STEP* control input, which moves the motor one step[3], as well as the *DIR* control input which indicates the rotational direction in which the step is taken. The first design consideration in the control of the stepper motors is the step resolution, or microstepping mode, employed. The microstepping mode selection control inputs $M0$, $M1$ and $M2$ were connected to the microcontroller as part of the circuit design which allowed the microstepping mode to be selected in software. In general, increasing the step resolution improves the smoothness of the stepper motor rotation but also increases the pulse rate required to move the motor at the same speed. This is turn increases the computational load on the microcontroller.

[2]This assumes the PWM period is set to 20 ms.

[3]The size of the step the motor takes is dependent on the microstepping mode configuration.

The DRV8825 supports a maximum of 1/32 microstepping.[4] Since this project has specifications relating the accuracy of the system but not the speed of the system, the maximum microstepping mode was selected to reduce vibrations as far as possible. It was noted that increasing the microstepping resolution did not correspond to an increase in position resolution. This is due to the fact that there is an exponential reduction in holding torque in microstep positions as the microstepping resolution increases. For this reason, the motor control firmware was designed such that the stepper motors only make use of microstep positions during the movement phase to reduce vibration and only rest in full step positions.

The simplest form of control for the stepper motors is constant speed control where the rate at which pulses are sent to the motor is always constant. This was the control initially implemented. However, when the pulses are started or stopped, a near instantaneous change in velocity occurred which resulted in a peak in acceleration that induced a mechanical jolt in the *Robotic System*. This limited the maximum motor speed that could be used to ensure the mechanical jolt generated was minimal. To overcome this issue, a linear acceleration profile was implemented for each stepper motor. This was achieved by altering the period of the underlying microcontroller timer peripheral for each stepper motor.

### *3.3.4 Serial Communication*

A communication protocol needed to be developed between the robotic controller and the PC running the system control software. It was noted that the maximum data unit that needed to be transmitted between the *Robotic Subsystem* and the PC-based software component was a command instruction for the robot to move to a target position. Since the position of each of the robot's axes was specified by a 16 bit number, eight data bytes were required to contain this information. Furthermore, an additional control byte was required to distinguish between the purpose of each data byte for a given packet. Each packet was formed by specifying the control byte followed with a concatenation of the four data bytes. The definitions assigned to each of the control bytes are also determined by the direction of transmission and given in Table 4.

For the serial communication component required to facilitate the use of this protocol, a word length of 8 bits was selected with no parity bit and 1 stop bit as this is the most common structure used and there was no reason to choose an altered structure. A baud rate of 115200 bits/s was also selected with the maximum oversampling rate of 16 samples chosen to minimise the effective noise during data reception.

[4] For each step in 1/32 microstepping mode, the motor rotates 1/32 of the full step angle.

| Control Byte | Command | Description |
|---|---|---|
| ***PC System* to *Robotic System*** | | |
| 0x1 | Wake | Place the stepper motors into active state from sleep state. |
| 0x2 | Calibrate | Initiate the robot calibration sequence to move along each and find the limit switch on linear axis. |
| 0x3 | Set target position | Set the desired step position on each axis to define the position to which the robot will move. |
| 0x4 | Actuate vacuum mechanism | Move the vacuum system servo motor to the suction position. |
| 0x5 | Delay | Delay the robot's command complete return packet by a short period of time. |
| 0x6 | Request pressure update | Trigger the *Robotic System* to respond with a packet containing the pressure sensor reading. |
| ***Robotic System* to *PC System*** | | |
| 0x1 | Command complete | Indicates the previously received packet has been processed successfully. |
| 0x6 | Pressure update | Sends the current reading of the pressure sensor. |

**Table 4. Serial communication protocol utilised between the *PC System* and the *Robotic System*.**

## 3.4 Shape Definition Interface

### *3.4.1 Background*

The second system specification for this project requires the development of a PC-based GUI software component that allows the user to define 3D shapes to be constructed by the robot using the small construction cubes. Furthermore, this specification also indicates that this software component must make use of graphical primitives to generate a 3D render of the shape as part of this process. To this extent, the OpenGL specification was selected as the basis for the 3D rendering component. DirectX was considered as an alternative to OpenGL. However, DirectX is only supported for the Windows operating system and XBox while OpenGL is cross-platform. The C++ QT framework used as the basis for the PC-based software component is inherently cross-platform and also offers better support for OpenGL integration which justified its selection for use in this project. It is noted that OpenGL is only a specification for a graphics API and not an implementation in itself. This API is usually implemented by the graphics card manufacturers. Furthermore, it is noted that OpenGL can be considered to be a state machine which is referred to as the graphical context. The behaviour

of various OpenGL instructions depends on the state of the OpenGL context.

The foundation of rendering using OpenGL is centred around the use of vertices. Specifically, a number of vertices are defined as the starting point for various objects to be rendered in 3D. These vertices are assembled into primitive shapes such as lines, triangles, quadrilaterals and other polygons depending on the OpenGL context. Triangles were used as the primitive shape created from vertices in this project since it is straightforward to define cubes through the use of triangles and most OpenGL hardware supports render acceleration for triangles. Surfaces are created by generating a number of these shapes adjacent to each other with varying sizes and orientations. These vertices and shapes encounter a number of processes and transformations when being converted from a description in 3D space to a collection of pixels on a 2D computer screen. There are approximately six such steps in this process which are known as the six stages of the graphics pipeline. Two of these steps require an implementation to be defined when using OpenGL, namely the vertex shader and the fragment shader. These shaders were written using the OpenGL Shading Language (GLSL)[5]. In order to define the positions and orientations of the various objects to be rendered as well as how they are transformed to 2D space on the screen, a number of coordinate systems and transformation matrices are required.

The object is described in the local frame $l$ with the origin of the frame of reference usually located somewhere on or within the object. The world frame $w$ is the space that relates the position, orientation and scale of all the objects to be rendered together. The matrix that maps the local frame to the world frame is known as the model matrix, denoted here by $\boldsymbol{M}^{wl}$. Similarly, the view of the world, usually thought of as a camera, has its own coordinate system called view space $v$. The world space is mapped to this view space by means of the view matrix $\boldsymbol{M}^{vw}$. This defines the point of view the world is seen from. Lastly, it is noted that the coordinates that are mapped to the screen need to be normalized device coordinates (NDC) where all vertex values of the coordinate axes are between 0 and 1. Any vertex outside of this space will not be projected onto the screen. As such, this space is called clip space $c$ and the projection matrix $\boldsymbol{M}^{cv}$ is used to map from view space to clip space. The projection matrix can be used to define the nature of this clip space and as such can be used to control the projection perspective.

Lastly, the viewport transform is performed to convert the 3D coordinates to 2D coordinates. The former three transforms, namely the model, view and projection matrices, are the transforms that were manipulated to transform the vertices accordingly. Each of these matrices consist of a translation, rotation and scale sub-component of which the order of operations is important to ensure the correct transform result.

### *3.4.2 3D Shape Render*

The 3D shape to be rendered was approached by first considering how each individual constituent cube within the shape should be rendered. Since triangles were selected as the graphical shape primitive, the cube was formed by defining the six cube faces using two

[5]GLSL is a high-level language with C-style syntax that executes on graphics hardware and allows modification of the graphics pipeline.

triangles per face. Each triangle was defined by specifying the location of each vertex in the local coordinate system for a total of 36 vertices to define the entire cube as shown in Figure 10b. Furthermore, in order to map a texture to the cube face, each triangle vertex was associated with an NDC in the texture image file shown in Figure 10a such that the entire texture image was mapped to each cube face. Let $\boldsymbol{p}_i^l \in \mathbb{R}^3$ denote the position of the $i$th vertex in the local coordinate system and let $\boldsymbol{t}_i \in \mathbb{R}^2$ denote the NDC in the texture image associated with $\boldsymbol{p}_i^l$. In order to capture the structure and texture of the cube to be rendered, a vertex information array was defined as

$$vertices = [\boldsymbol{p}_0^l, \boldsymbol{t}_0, \boldsymbol{p}_1^l, \boldsymbol{t}_1, ..., \boldsymbol{p}_{35}^l, \boldsymbol{t}_{35}]. \tag{10}$$

Vertex attribute pointers were created to indicate the location of the position and texture coordinates within the *vertices* array in Equation 10 for the OpenGL context. To render the cube using this vertex information the *vertices* array was bound to the current OpenGL context using various calls to the OpenGL API. However, this resulted in only the front 2D cube face being rendered on the screen. In order for multiple cubes to be rendered from different perspectives, the model, view and projection matrices need to be derived and utilised. The model matrix is constructed by noting that a given cube is mapped from local space to world space through scaling, rotation and translation operations. Let the matrices $\boldsymbol{S} \in \mathbb{R}^{4\times4}$, $\boldsymbol{R} \in \mathbb{R}^{4\times4}$ and $\boldsymbol{T} \in \mathbb{R}^{4\times4}$ represent these operations respectively in homogeneous coordinates. Furthermore, let $\boldsymbol{v}_i^l \in \mathbb{R}^4$ denote the position of the $i$th vertex in homogeneous coordinates in the local frame. Since each cube only needed to be rotated about the y-axis, $\boldsymbol{R}$ can be simplified for only this rotation as $\boldsymbol{R}_y$. Additionally, Euler angles were chosen to parameterise the cube's orientation since Gimbal lock is not possible with the rotation restricted as such. Based on this, the model matrix is calculated as

$$\boldsymbol{M}^{wl} = \boldsymbol{T}\boldsymbol{R}_y\boldsymbol{S}, \tag{11}$$

$$= \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

where $(T_x, T_y, T_z)$ is the translation vector, $(S_x, S_y, S_z)$ is the scaling vector and $\theta$ is the angle of rotation of the cube about the y-axis. The *vertices* vector was defined in such a manner that the cube has a side length of 1 unit in the local coordinate system. Since the end-effector in the *Robotic System* can only be positioned at a number of discrete *step* positions, it was chosen to use *steps* as the measurement unit for the world space coordinate system in the *Shape Definition* component. A physical construction cube has a side length of 64 *steps*. Therefore, a scaling factor of 64 *steps* was used for all axes to form $\boldsymbol{M}^{wl}$. The translation vector values were obtained from the position state of the centre of a given cube in *steps*. Furthermore, the translation vector element $T_y$ was offset by 32 *steps* to ensure no part of the cube exists below the plane $z = 0$ in the world frame.

The cubes in the world space are mapped to the view space next using the view matrix. The result of this transformation gives the impression that the world space is rendered from the

perspective of a camera. This idea was used as the basis to form the view matrix[6] using the camera's position $\boldsymbol{p}_c^w \in \mathbb{R}^3$, the camera's focal point $\boldsymbol{p}_f^w \in \mathbb{R}^3$ and the camera's up direction vector $\hat{\boldsymbol{u}}_u^w \in \mathbb{R}^3$, all with respect to the world frame. The camera direction[7] vector $\hat{\boldsymbol{u}}_z^w \in \mathbb{R}^3$ (i.e. camera's positive z-axis) is calculated as

$$\boldsymbol{u}_z^w = \boldsymbol{p}_c^w - \boldsymbol{p}_f^w, \tag{13}$$

$$\hat{\boldsymbol{u}}_z^w = \frac{\boldsymbol{u}_z^w}{||\boldsymbol{u}_z^w||}. \tag{14}$$

With the direction of the camera known, the camera's right vector $\hat{\boldsymbol{u}}_x^w \in \mathbb{R}^3$ (i.e. camera's positive x-axis) in the world frame is calculated as

$$\boldsymbol{u}_x^w = \hat{\boldsymbol{u}}_u^w \times \hat{\boldsymbol{u}}_d^w, \tag{15}$$

$$\hat{\boldsymbol{u}}_x^w = \frac{\boldsymbol{u}_x^w}{||\boldsymbol{u}_x^w||}. \tag{16}$$

Finally, the direction of the camera's positive y-axis $\hat{\boldsymbol{u}}_y^w \in \mathbb{R}^3$ with respect to the world frame is calculated as

$$\hat{\boldsymbol{u}}_y^w = \hat{\boldsymbol{u}}_z^w \times \hat{\boldsymbol{u}}_x^w. \tag{17}$$

The results of Equations 14, 16 and 17 are used to calculate the view matrix as

$$\boldsymbol{M}^{vw} = \begin{bmatrix} \hat{u}_{x,0}^w & \hat{u}_{x,1}^w & \hat{u}_{x,2}^w & 0 \\ \hat{u}_{y,0}^w & \hat{u}_{y,1}^w & \hat{u}_{y,2}^w & 0 \\ \hat{u}_{z,0}^w & \hat{u}_{z,1}^w & \hat{u}_{z,2}^w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -p_{c,0}^w \\ 0 & 1 & 0 & -p_{c,1}^w \\ 0 & 0 & 1 & -p_{c,2}^w \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

where $\hat{u}_{x,i}^w$, $\hat{u}_{y,i}^w$, $\hat{u}_{z,i}^w$ and $p_{c,i}^w$ are the $i$th elements of $\hat{\boldsymbol{u}}_x^w$, $\hat{\boldsymbol{u}}_y^w$, $\hat{\boldsymbol{u}}_z^w$ and $\boldsymbol{p}_c^w$ respectively. The final transformation that was implemented is the mapping of view space to clip space using the projection matrix. The points that are clipped are selected based on their location with respect to a frustum defined in the view frame. Points that fall within this frustum are kept and converted to NDCs while points outside of this frustum are clipped. The shape of this frustum defines the nature of the projection. Specifically, if the frustum is a rectangular prism, the resulting projection is an orthographic projection. However, if the frustum is not a uniform prism, the resulting projection is a perspective projection. The perspective projection is how the world appears to the human eye and, therefore, was selected as the projection method for this project to ease the process of comparing the 3D shape model and the constructed shape. The projection matrix is defined as

[6] In this case, the view matrix is also frequently referred to as the *LookAt* matrix.

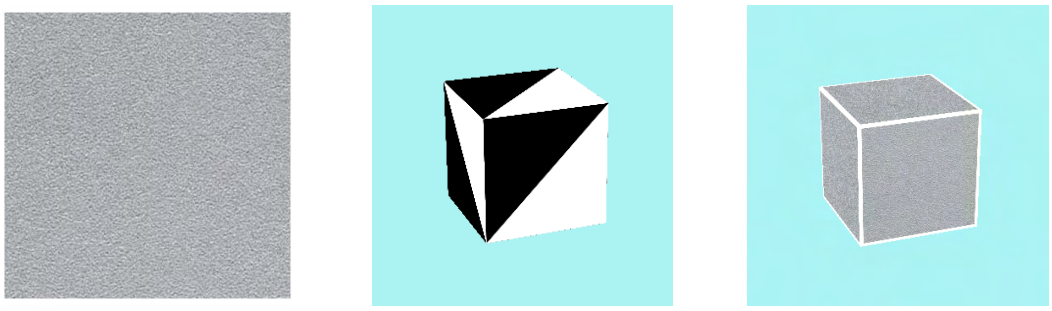
[7] Unintuitively, the camera direction vector points in direction opposite to the direction the camera is facing.

$$\boldsymbol{M}^{cv} = \begin{bmatrix} \frac{n}{r} & 0 & 0 & 0 \\ 0 & \frac{n}{t} & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{19}$$

where $n$ and $f$ are the near and far z-planes of the projection frustum respectively in the view frame. Furthermore, $r$ and $t$ are the right and top bounding x- and y-planes respectively for the near frustum z-plane. Equation 19 assumes that the frustum exhibits x and y symmetry such that the left and bottom bounding planes need not be specified. It was found to be more intuitive to parameterise the projection in terms of the frustum angle $\alpha$ and aspect ratio $\beta$ of the frustum z-planes alongside $f$ and $n$. To this end, $n/r$ and $n/t$ in Equation 19 are computed as

$$\frac{n}{r} = \frac{1}{\alpha \tan(\beta/2)}, \tag{20}$$

$$\frac{n}{r} = \frac{1}{\tan(\beta/2)}. \tag{21}$$



**(a) Cube face texture** **(b) Triangle components** **(c) Textured cube**

**Figure 10. Stages of the process to render a construction cube using OpenGL.**

## 3.5 Computer Vision System

The design of the overall system facilitates open-loop operation of the robotic subsystem when the computer vision subsystem is excluded. This means that the system is capable of arbitrary shape construction without visual feedback during the construction process, given that the initial cube positions are known. However, should a cube be dropped during this process, the system would not be capable of an intelligent response[8]. Furthermore, any significant disturbance of the shape under construction would be undetectable by the system. The computer vision system is required, and was designed, to address these two cases. In the first case, the dropped cube should be detected and localised with respect to the robot

[8]The system would be able to detect the dropped cube condition through the pressure sensor in the vacuum system detecting the unplanned pressure decrease.

coordinate system if it falls within the camera's field of view. Secondly, when damage to the structure is detected in the image input data, the computer vision system should signal a construction halt condition. This section begins with an overview of the cube detection approaches investigated. A description of the method used to map points detected in the image frame to the world frame follows in Section 3.5.2. These sections lay the foundation for the integrated computer vision algorithm and supporting functions discussed from Section 3.5.3 onward.

### *3.5.1 Cube Feature Investigation*

Detection of an object in a single-view image generally involves the identification of features in the image which can be compared to a template set of features for the desired object to determine its presence and location in the image. Furthermore, certain features may be used to generate further information about the object beyond feature matching. For example, edges and corners may be used to identify planes which are combined to identify the presence of a cuboid. The OpenCV C++ library was utilised as the source of various pre-written image processing and machine vision functions required to create a prototype of the feature detection process. Specifically, a set of cube images were captured and the Canny edge detector, Harris corner detector and SIFT feature descriptor implementations were utilised to detect edges, corners and features in the cube images respectively.

The primary challenge encountered in cube detection process was the extraction of useful feature information from individual cubes. The aluminium cubes used in this project are a singular shade of grey with an almost textureless surfaces which offers little unique information to detect the cubes. As a result, the prototype feature detection algorithms discussed above performed poorly and did not extract a sufficient number of features to uniquely detect the cubes. Specifically, in even lighting conditions, the Canny edge detector failed to identify internal edges within the outer bounding contour. Similarly, the Harris corner detector failed to consistently identify the corners of the cube while the SIFT algorithm failed to identify a sufficient number of features on the cube itself. The Hough transform parameterised for lines was also explored with limited success.

The process of image segmentation was also explored to isolate the cubes from the background. Histogram plots showing pixel intensity distributions for each of the RGB colour channels were generated from images containing cubes with various monochrome background colours. Each of the colour channels exhibited nearly identical distributions which indicated the primary information in these images was grey-scale intensity information. The grey-scale histogram information from the images with a matte black background exhibited a peak at greater pixel intensity values corresponding with the cube pixels. Based on this, the cubes were successfully segmented in the image through application of a binary threshold.

In even lighting conditions, the cube faces could not be distinguished from each other with a significant degree of reliability. This presents a problem as face segmentation is necessary for cube pose estimation. However, when the dominant lighting source was placed directly above the cubes, a peak in the pixel intensity histogram was observed which corresponded to the top face of the cube. This allowed the segmentation of the top face with a great degree of reliability. If the scene is constrained such that cubes are the only objects present and the

assumption is made that a cube face is always parallel to the base plane, then the segmented top cube face is sufficient information to detect a cube and uniquely determine its orientation about the z-axis. Therefore, this approach was selected as the basis of the computer vision cube detection component.

### *3.5.2 3D Localisation*

The location of the cube with respect to the robot needs to be determined from the location of the cube in the image for the system controller to make decisions based on the location of the cube and for the robot to interact with the cube. For the purposes of this project, the robot coordinate system was defined to be equivalent to the world coordinate system. This problem is referred to as object localisation and requires a solution to map an arbitrary point in image coordinate system, denoted by $\boldsymbol{p}^i \in \mathbb{R}^2$, to a corresponding point in the world coordinate system, denoted by $\boldsymbol{p}^w \in \mathbb{R}^3$. The 3D camera coordinate system is useful as an intermediate frame to relate the world frame to the image frame. Let a point in the camera coordinate system be denoted by $\boldsymbol{p}^c \in \mathbb{R}^3$.

The pinhole camera model was used as the foundation for the mapping methods discussed here. This model requires that several parameters about the camera and its orientation to be known. These parameters can be divided into two categories, namely intrinsic and extrinsic parameters. Intrinsic parameters describe internal properties inherent to the camera itself and include the principal point $(c_x, c_y)$ as well as the focal lengths ($f_x$ and $f_y$) of the camera. Extrinsic parameters describe the location and orientation of the camera with respect to the world coordinate system and include the rotation and translation transformations that are required to be performed to map arbitrary point $\boldsymbol{p}^w$ to $\boldsymbol{p}^c$. These transformations are captured by the rotation-translation matrix $[\boldsymbol{R}|\boldsymbol{t}]$.

The intrinsic parameters of the camera can be expressed as a calibration matrix $\boldsymbol{K}$ defined as

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{22}$$

The parameter $s$ in Equation 22 captures the skew of the sensor axes that occurs as a result of the optical axis not being exactly perpendicular to the sensor plane. However, for practical purposes $s$ can be set to 0. Since the extrinsic parameters are captured by the rotation-translation matrix $[\boldsymbol{R}|\boldsymbol{t}]$, both the intrinsic and extrinsic parameters are captured by the product of this matrix and the calibration matrix $\boldsymbol{K}$. This 3 x 4 matrix product is referred to the camera matrix $\boldsymbol{P}$ and expressed mathematically as

$$\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]. \tag{23}$$

The camera matrix is sufficient to map the world coordinate system to the the image coordinate plane provided that the pinhole camera model is used and no lens distortion effects are present. This mapping is defined as

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{P}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{24}$$

where $(X,Y,Z)$ are the coordinates of a given point in the world coordinate system $\boldsymbol{p}^w$, $(u,v)$ are the coordinates of the corresponding point in the image coordinate system $\boldsymbol{p}^i$ and $s$ is simply a scaling factor. It is noted there is a loss of depth information when mapping from the world frame to the image frame. This is observed mathematically in Equation 24 since $\boldsymbol{P}$ is not a square matrix and not invertible as a result. Therefore it is not possible to map from the image frame to the world frame without an additional piece of information. It was postulated that this piece of information could be obtained given that the length of the cube edge is known in the world frame. However, the cube side length differential was not found to be sufficiently large between vertical layers to make this distinction. Instead it was decided to provide the z coordinate of $\boldsymbol{p}^w$ based on the horizontal plane layer a cube was expected to be detected in. This is reasonable given that a cube dropped away from the source and structural cubes will always be found on the base plane.

In order to solve for the the world coordinates given the image coordinates, camera matrix $\boldsymbol{K}$ and world coordinate plane $Z$ using the pinhole camera model, Equation 24 is expanded and rearranged as

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{K}\left(\boldsymbol{R}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \boldsymbol{t}\right), \tag{25}$$

$$\boldsymbol{R}^{-1}\boldsymbol{M}^{-1}s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \boldsymbol{R}^{-1}\boldsymbol{t}. \tag{26}$$

In order to solve for the unknown scaling factor $s$, the intermediate vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are defined as

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \boldsymbol{R}^{-1}\boldsymbol{M}^{-1}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{27}$$

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \boldsymbol{R}^{-1}\boldsymbol{t}, \tag{28}$$

such that Equation 26 can be rewritten as

$$s\boldsymbol{x} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \boldsymbol{y}. \tag{29}$$

Since the camera matrix $\boldsymbol{P}$ is given, the rotation matrix $\boldsymbol{R}$, translation matrix $\boldsymbol{t}$ and intrinsic

matrix $\boldsymbol{M}$ in Equations 27 and 28 are known. Consequently, $s$ can be computed as

$$s = \frac{Z + y_3}{x_3}. \tag{30}$$

Finally, with $s$ known, it is trivial to make use of Equation 29 to obtain the world coordinates. For completeness, this is expressed as

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = s\boldsymbol{x} - \boldsymbol{y}. \tag{31}$$

In order to make use of the mathematical tools discussed above, the intrinsic parameters and extrinsic parameters of the camera needed to be determined. The intrinsic parameters were determined using a checkerboard calibration. A number of images of the checkerboard were captured at various poses in the robotic subsystem's workspace. The *findChessboardCorners*, *calibrateCamera* and *getOptimalNewCameraMatrix* OpenCV library functions were used in the camera calibration process. The camera's extrinsic parameters were obtained using the EP*n*P variation of the *solvePnP* function from the OpenCV library which serves as a solution to the P*n*P problem. This requires point correspondences which refers to points that have a known location in both the image frame and the world frame. Given at least four of these points, it is possible to estimate the rotation and translation matrices.

### *3.5.3 Top-Level Design*

The *System Controller* maintains a non-probabilistic belief state for the location and orientation of each cube with respect to the robot's coordinate system. Furthermore, four mutually exclusive states are used to distinguish between cubes. A *source cube* is a cube that is believed to be in its known initial position while a *structure cube* is believed to have been successfully placed in its designated position within the 3D shape under construction. If no unexpected events occur during the construction process, each cube will only exist in either of one these two states. The two unexpected events the system is expected to deal with are the dropped cube case and the structural damage case. If the vacuum system pressure sensor detects that a cube is dropped during manipulation by the robot, the cube is classified as a *missing cube*. A cube that is detected by the *Vision System* that is not in an expected *source cube* or *structure cube* location is classified as an *independent cube*. The *System Controller* is able to deal with both the dropped cube case and the structural damage case when the position and orientation of all *independent cubes* with respect to the robot's coordinate system are provided. As such, the purpose of the *Vision System* is to detect and localise all *independent cubes* with respect to the robot given the image input data of the robot's workspace. The expected location of the *source cubes* and the *structure cubes* with respect to the robot for a given time instance are also required as input to distinguish these cubes from the *independent cubes*.

Figure 11 shows the high level steps the *Vision System* performs each time it is instructed to process the image input data captured by the camera. The findings from the cube feature investigation performed in Section 3.5.1 guided the design of this process. The input image

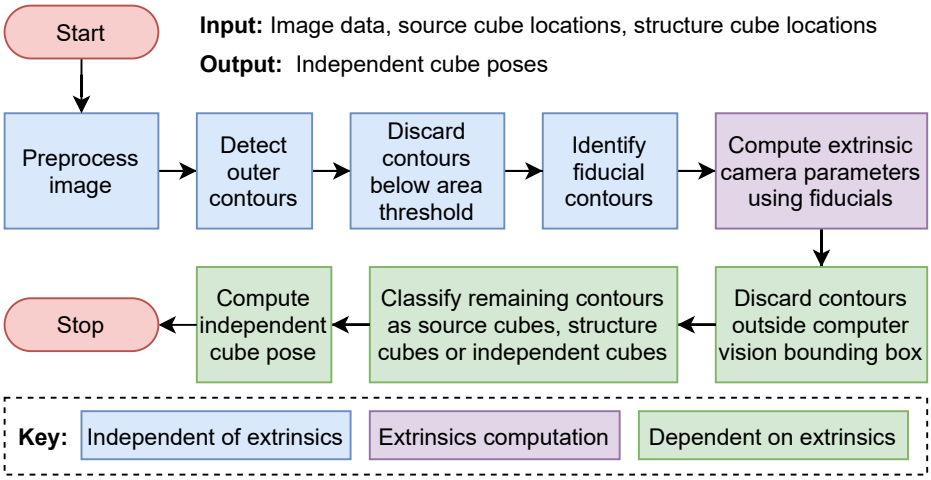



**Figure 11. Flow diagram showing the steps in the integrated computer vision process.**

is expected to be in RGB color format and may be of arbitrary size. A preprocessing step is applied to the image in which the image is first converted to grey-scale format. This is followed by the application of a Gaussian blur function to eliminate high-frequency noise within the image that appears as local outlier pixel intensities. Finally, a fixed binary threshold is applied to the image as the initial step to segment the top faces of the cube, as discussed in Section 3.5.1. This step also segments the fiducial markers, the design of which are discussed later in Section 3.5.4.

Following the preprocessing step, contour detection is applied to the binary image to convert the connected 1-components within the image to a discrete set of elements that can be processed individually. The cubes and fiducials are assumed to be surrounded by the plain black background of the robot's base plane and, as such, only the outer contours are detected to ensure the patterns within the fiducial are not identified as separate elements to the fiducial. Finally, a number of contours originating from image artifacts that are not cubes or fiducials are also detected as part of this step. In order to reduce the amount of processing required during later contour classification steps, the area enclosed by each contour is computed and all contours with an area significantly smaller than that of cubes and fiducials on the base plane are discarded.

With the set of fiducial and cube candidate contours compiled, the next phase of the algorithm is concerned with the classification of these contours. Firstly, the contours that enclose fiducial markers are identified as part of the fiducial identification step (see Section 3.5.4) which includes the extraction of the unique fiducial identifier values. The location of each fiducial in the world coordinate system is known and retrieved from a look-up table based on the fiducial identifier. The location of the fiducial within the image coordinate system is taken as the location of the fiducial contour centroid. With this information, a point correspondence between the world coordinate system and image coordinate system can be formed for each fiducial and used to compute the extrinsic camera parameters as discussed in Section 3.5.2. Furthermore, the methods to map between the image frame and world frame may now be used with the camera extrinsics calibrated. The base plane of the robot was controlled such that the workspace only contains cubes and fiducials. By restricting the remaining contours under

consideration to only those that are detected within the robot's workspace, these contours are guaranteed to be artifacts of construction cubes since the fiducial contours have already been identified.

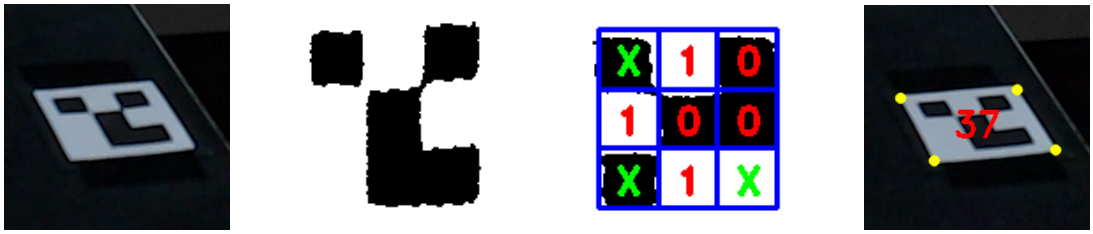
The rectangular bounding region of the robot's workspace is aligned with the world coordinate system. As such, it is trivial to determine if a point falls within this region if the point is also defined in the world frame. Therefore, to determine if the centroid point of a given contour in the image frame falls within the bounding region, the point is first projected to the world coordinate system using Equation 31. As noted in Section 3.5.2, when projecting from image space to world space, it is necessary to specify the $Z$ plane of projection in the world frame. Since the camera captured images from the vertical perspective, it was considered sufficient to project the contour centroids to the base plane, $Z = 0$. Following this step, all contours with centroids external to the bounding region are discarded.

With the remaining contours assumed to only be image artifacts originating from cubes, the contours need to be classified as either *source cube*, *structure cube* or *independent cube* contours. This classification is based on expected locations of the *source cubes* and *structure cubes* in the world frame as provided to the *Vision System* by the *System Controller*. For the *source cubes*, the proximity of the centroid of each contour to the centre of the top face of each *source cube* is considered. If the contour centroid is considered sufficiently close[9] to the top face centre of any of the *source cubes*, the contour is considered to be a *source cube* contour. The same applies for *structure cubes*. To facilitate the proximity calculations, the contour centroid is projected to the world coordinate system for each cube the contour is compared with using Equation 31. The $Z$ coordinate of the top face of the cube under consideration is used as the $Z$ plane of projection in the world frame. The world frame Euclidean distance between the projected centroid point and the cube top face centre is computed as the measure of their proximity. If a contour is classified as neither a *source cube* nor a *structure cube* contour, it is assumed to be an *independent cube* contour. Finally, The orientation and location of each detected *independent cube* is estimated as outlined in Section 3.5.6.

### *3.5.4 Fiducial Identification*

The location of the fiducials within the image input data provide the corresponding image frame coordinates to the known world coordinates of the fiducials to form point correspondences. It has been found that a greater number of point correspondences generally leads to an improved solution to the P*n*P problem. As such, eight fiducial markers were placed at eight known locations on the robot's base plane. Each fiducial was structured as a square with a white border with an imaginary internal grid of 3x3 squares each having a side length of 5mm. The black squares were defined to represent a binary zero and the white squares a binary one. In order to ensure the rotation of the fiducial can be uniquely determined, the squares at coordinates $(0,0)$, $(2,0)$ and $(2,2)$ were assigned the binary values of 0, 0 and 1 respectively. The six remaining binary squares facilitated the representation of $2^6 = 64$ unique identifiers. Figure 12a shows an example of a fiducial marker located on the robot's base plane.

[9] A centroid is considered sufficiently close to the centre of the top cube face if the Euclidean distance between these points is less than one cube side length.




**(a) Original** **(b) Isolated** **(c) Processed** **(d) Annotated**

**Figure 12. Various stages of fiducial detection and identification in the computer vision subsystem.**

The fiducial identification step takes place after the contour extraction phase shown in Figure 11. The purpose of this step is to identify which of the detected contours are an artifacts of fiducials, to acquire the image coordinates of the fiducials and to extract the fiducial identifiers. An overview of the fiducial identification algorithm developed for this project is shown in Figure 13. This algorithm follows the approach of initially assuming that each contour is a fiducial and discarding contours in latter steps if the contour does not fulfill certain fiducial requirements. Following this logic, the four corners of the fiducial candidate contour are extracted using the algorithm presented in later Section 3.5.5 based on the assumption that the contour is a square. Using the detected corners, the internal angles and side lengths of this quadrilateral are computed. If they do not approximate the properties of a square, the contour is discarded.

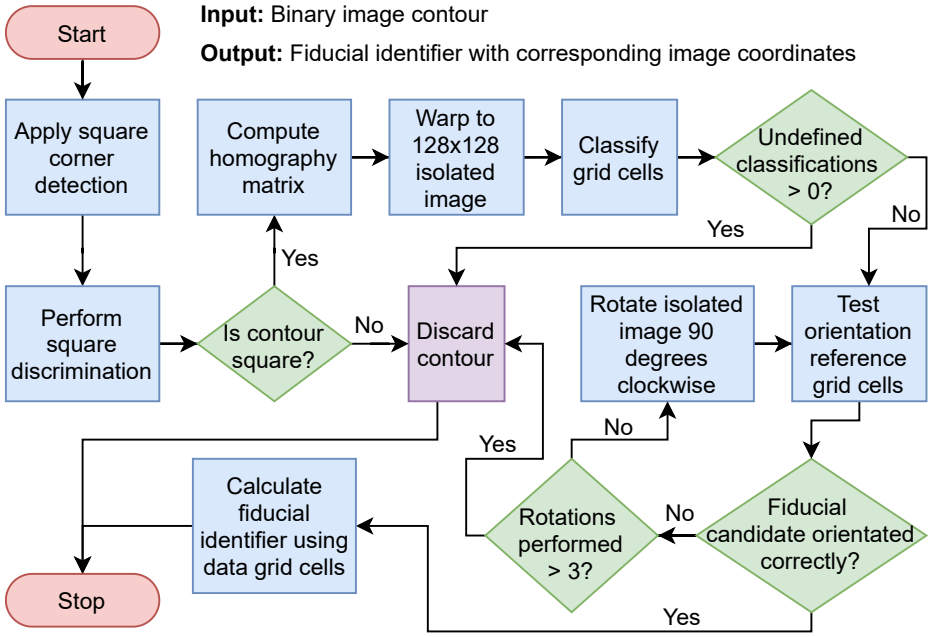



**Figure 13. Flow diagram showing the steps in the fiducial identification process.**

It was noted that the fiducial in the image captured by the camera has a degree of perspective warping compared to the reference digital fiducial image. In order to correct for this perspective

warp, a 3x3 homography matrix $\boldsymbol{H}$ is computed using the *findHomography* OpenCV library function. Four point correspondences are required to compute $\boldsymbol{H}$. These correspondences are obtained by arbitrarily pairing each of the fiducial candidate's detected corner coordinates in the original image to the corner coordinates of a 128 x 128 pixel destination image. The fiducial candidate region in the original image is warped to the destination image such that the fiducial pattern is isolated as shown in Figure 12b. The internal region of the isolated fiducial candidate image is divided into a pre-placed 3x3 grid of squares corresponding to the expected positions of the fiducial squares. This grid is shown in Figure 12c. For the grid cell to be classified as a binary one or zero, at least 75% of the inner 30 x 30 pixels of the square need to have a pixel intensity of 255 or 0 respectively. If the condition is not met for either value for any grid cell, the grid cell is considered unclassified and the contour is discarded.

Once all the grid cells have been classified, the orientation reference cells at $(0,0)$, $(2,0)$ and $(2,2)$ are compared with the expected binary values of 0, 0 and 1 respectively. If these match, the fiducial is considered to be correctly oriented. Otherwise, the isolated fiducial candidate is rotated 90° clockwise until the correct orientation is found. If the correct orientation is not found after three rotations have been performed, the contour is discarded. Finally the unique binary identifier encoded in the fiducial is extracted by reading the binary values of the cells from left to right and top to bottom excluding the orientation cells. The cells are ordered from the least significant bit (LSB) to the most significant bit (MSB). Figure 12c shows the result of applying the fiducial identification step to the isolated fiducial in Figure 12b. The orientation reference cells are indicated by green crosses while the fiducial has been rotated to align with these cells. The binary values detected in each cell are indicated in red. In this case the identifier is calculated as $100101_2 = 37$. The fiducial in the original image is annotated with the detected corners as well as the fiducial identifier as shown in Figure 12d.

### *3.5.5 Square Corner Detection*

Both the fiducial identification (see Section 3.5.4) and cube orientation computation (see Section 3.5.6) algorithms require the corners of the contours to be known in the image coordinate system. It was decided to extract these corners from a given contour based on the assumption that the underlying shape is a square. The first step in the corner detection process is the computation of the centroid of the contour. Following this, the Euclidean distances between the centroid and each of the contour points are computed. The contour point that has the greatest Euclidean distance from the centroid is taken to be the first corner. The remaining contour points are then segmented into four quadrants with the origin of the quadrant axes coincident with the centroid of the contour and the axes oriented such that the detected corner falls at the centre of the first quadrant. This is based on the fact that the four corners of a square can be separated into four quadrants. The contour points with the greatest Euclidean distance from the centroid in each of the remaining three quadrants are taken to be the remaining three corners.

### *3.5.6 Cube Pose Estimation*

From a black box perspective, the cube pose estimation step in this project takes the contour of the top face of a cube as input and produces an estimate of the orientation and location of the cube with respect to the world coordinate system as output. The location of the centre of the top face of the cube in the world coordinate system is obtained through the projection of the centroid of the cube contour from the image coordinate system using Equations 30 and 31. The $Z$ plane of projection in the world coordinate system needs to be specified as part of this step. Since only *independent cubes* need to be localised as discussed in Section 3.5.3, it is assumed the cube to be localised always be on the base plane. This is reasonable as *independent cubes* only exist as the result of an unexpected event such as the dropped cube event in which case the independent cube will always be on the base plane. Therefore, for projection purposes, the $Z$ plane of projection is defined as the $Z$ position of the top-face of a cube on the base plane. In other words, the $Z$ plane of projection is one cube side length above the base plane.

Under normal circumstances, every cube in the robot's workspace will always have a face parallel to the base plane. Therefore, the detection of the orientation of a cube with respect to the world frame is reduced to the detection of rotation of the cube about the z-axis. Specifically, this rotation is defined as the angle between the positive x-axis in the world frame and the perpendicular line from any cube edge to the centre of the top face of the cube. Furthermore, since the top face of the cube is a square which has rotational symmetry of order 4 and a 90° angle of rotational symmetry, the rotational angle range of the cube is mapped to the range $(45°, 45°]$. The centroid and the four square corners of the cube contour are used to estimate the orientation of the cube. Since the square corners are already computed for each salient contour in the fiducial identification step (see Section 3.5.4), this result is simply reused for the relevant contours in this step.

Since the corners are defined with respect to the image frame, they are also projected to the same $Z$ plane as the contour centroid projected initially. The coordinates of the centroid and four corners in the world $Z$ plane are used to compute the rotation of the cube. There are four lines, each between a corner and the centroid, that can be used to form an angle with the positive x-axis. By computing the angle using each of these lines separately, the average estimated angle can then be computed with reduced high-frequency noise. However, an average angle cannot be computed through the sum and normalisation of the angle data as this is incorrect mathematically. Instead, one of the lines is chosen as a reference while the three corners are rotated by either -90°, 90° or 180° to be in the same quadrant as the reference line. The average line position is computed by taking the average X and Y position of the rotated corners and this can be used to calculate the average estimated angle with respect to the positive x-axis. Finally, since the cube orientation is defined using a perpendicular line to the nearest edge, 22.5° is added to the estimated angle to account for the fact a corner line was used to compute the average angle. Finally, this angle is mapped to the range $(45°, 45°]$.

## 3.6 System Controller

The *System Controller* serves as the point of intersection between the *Vision System*, *Shape Definition* component and the *Robotic System*. Furthermore, the functional purpose of the *System Controller* is to integrate these components to allow the system to function as a whole. Since these components make up the majority of the PC-based software component, it was natural to structure the system controller as the base of this component. To this end, the design of the integrated system software is first presented in Section 3.6.1. Finally the design of the final two software components, namely the construction planner and the robotic motion planner, are discussed in Sections 3.6.2 and 3.6.3 respectively.

### *3.6.1 Integrated Software*

The first step in the design of the integrated system software was to identify the requirements of the software. Given the *Vision System* and *Shape Definition* components developed in Sections 3.5 and 3.4 respectively, the following core requirements were derived:

- The software needs integrate the *Vision System* and *Shape Definition* software components into a single unified piece of software.
- The software needs to use the shape model information generated by the *Shape Definition* component to generate a set of instructions for the *Robotic System* to build the shape.
- The software needs to facilitate communication with the *Robotic System* such that the robot can be controlled through the integrated software.
- The software needs to use the information generated by the *Vision System* to determine an intelligent response to unexpected events and direct the *Robotic System* accordingly.
- The software needs to provide an integrated user interface that allows the user to make use of the *Shape Definition* component as well as initiate and monitor the 3D shape construction process.

Based on these requirements, a number of modular components were identified and developed into the component-oriented solution shown in Figure 14. Each of the components were implemented as a class using C++ and the QT framework to form the base of the integrated software implementation. The *System Controller* class sits at the top level of this implementation and directly contains the user interface classes[10] as well as the *Logger* class through which all system event information, warnings and errors are recorded. The user interface classes partition the software based on functionality requirements. Specifically, the *Home View* serves as the entrance point to the software and its purpose is to ensure the *Robotic System* and camera hardware are present and connected. The *Design View* integrates the components related to the *Shape Definition* function. This includes the *OpenGL View* which uses OpenGL to create a 3D render of a model of cubes. This class is supported by the *Shader Program* class which manages the shaders used for the rendering task. The model itself is sourced from the *Cube World Model* class. The purpose of this class is to simply capture the

[10]Each of the user interface classes correspond to a distinct screen in the user interface.

arrangement of cubes, which are each represented by an instance of the *Cube* class, and relate this arrangement to the world coordinate system. The *Design View* interface facilitates the creation and manipulation of a *Cube World Model* instance which serves as output from the *Shape Definition* component.

The *Cube World Model* instance created in the *Design View* acts as input to the construction process around which the *Construction View* class is based. In order for construction to take place, the system needs to interact with the robot. The *Robot* class was created for this purpose. The class provides an abstract interface for the *Construction View* instance to send position and actuation control commands to the *Robotic System*. The units of communication which the *Robot* class uses to interact with embedded robotic controller are abstracted in the form of the *Packet* class. This class is based on the packet designed in Section 3.3.4. Similarly, the *Vision* class exists to offer an abstract interface to the *Vision System* developed in Section 3.5. The *Construction View* instance receives information from this component that is used to guide the control of the *Robot* instance during the construction process accordingly.

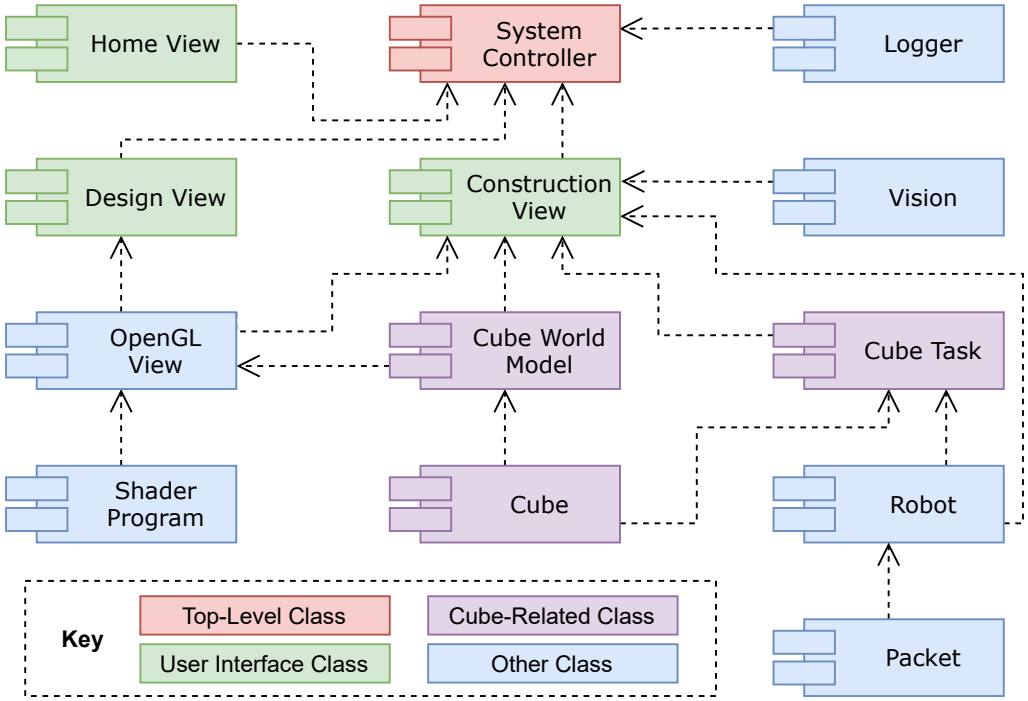



**Figure 14. Component diagram showing the structure of the integrated PC-based software solution.**

### *3.6.2 Construction Planner*

The modular component-based integrated software design developed in Section 3.6.1 greatly simplified the design of the construction planner functional unit to the extent where it was not necessary to create a separate class for this component. Rather, the construction planner was simply developed as an algorithm that was implemented as a function within the *Construction View* class. From an algorithmic perspective, the construction planner takes as input the *Cube*

*World Model* instance to be constructed and converts this to a list of cube translation and rotation tasks that need to occur in a given order. Specifically the physical cubes begin in a number of known *source cube* positions in the robot's workspace. The construction planner generates a number of tasks, which are represented by the *Cube Task* class, which each relate a *source cube* to a destination *structure cube* position. The construction planner determines the order in which these tasks are completed. The structure is build one layer at a time by scheduling the *Cube Task* instances with *structure cubes* in the lowest layer first followed by the tasks for the next layer and so forth. Should an unexpected event occur and a cube be dropped and lost, the *Cube Task* instance can simply be restarted with a different *source cube*.

### *3.6.3 Robotic Motion Planner*

The construction planner defines the start and end position for a cube associated with a particular *Cube Task* instance. In order for the robot to execute this task, a sequence of individual robotic positioning and actuation actions need to be performed in sequence. The sequence of steps performed by the robot to complete a cube tasks were defined as follows:

1. Move end-effector to just above the *source cube*.
2. Lower the end-effector onto the cube.
3. Actuate the end-effector vacuum mechanism.
4. Lift the cube up to one layer above the destination layer height.
5. Translate the cube to above the destination position.
6. Lower the cube to the destination layer.
7. Release the end-effector vacuum mechanism.
8. Lift the end-effector to just above the cube..
9. Move the end-effector to the image capture position such that the *Vision System* can process the scene.

To execute these steps within the *Cube Task* instance, an internal state is maintained. When the task is initiated or a previous task step is completed, the state is updated and the next step is performed. Each step is performed by calling the appropriate function in the *Robot* instance's interface. The *Robot* instance generates a signal when the step is complete to trigger the next state update. When all the steps within a *Cube Task* instance are complete, the next scheduled instance is initiated. This repeats until all *Cube Task* instances have been completed at which point the 3D shape should be complete in the physical world.

# 4. Results

## 4.1 Summary of results achieved

| Intended outcome | Actual outcome | Location in report |
|---|---|---|
| **Core mission requirements and specifications** | | |
| The system should construct novel and moderately complex 3D shapes using small cubes. The system should handle shapes up to at least 4 cubes in height containing up to at least 30 cubes where each cube has a face parallel to the base plane. The system should handle equal size cubes with a side length between 10mm and 15mm. | The system was able to construct a wide variety of shapes up to 6 cubes in height and containing 30 cubes where each cube has a face parallel to the base plane and a side length of 12.6mm ±0.05mm. | Section 4.2.1 |
| The GUI should allow the user to define a wide range of 3D shapes to be constructed. For each constituent cube in the shape, the GUI should allow the position of each cube to be specified along each Cartesian axis with at least 1 mm resolution as well as the rotation of each cube about the z-axis with at least 1° resolution. | The GUI allowed the user to define 3D shapes such that the position of each constituent cube could be specified along the x and y Cartesian axes with 0.2 mm resolution. The rotation of each cube about the z-axis could be specified with 0.9° resolution. | Section 4.2.2 |
| The end-effector should be able to grip a cube, maintain its grip during motion and release the cube. The end-effector should maintain the cube in its grip when the robotic manipulator is at maximum acceleration. The end-effector should be able to maintain the cube in its grip for at least 20 seconds continuously. | The end-effector was able to grip a cube, maintain its grip during motion and release the cube. The end-effector maintained the cube in its grip when the robotic manipulator was at maximum acceleration. The end-effector was able to maintain the cube in its grip for at least 30 seconds continuously. | Section 4.2.3 |

**Table 5. Summary of results achieved.**

| Intended outcome | Actual outcome | Location in report |
|---|---|---|
| **Core mission requirements and specifications** | | |
| The robotic manipulator should accurately translate the end-effector in 3D space and rotate it about its vertical axis. The robotic manipulator should have a repeatability of at least 2mm for each Cartesian axis and a repeatability of at least 5 degrees for the rotation about the z-axis. | The robotic manipulator was found to have a repeatability of less than 0.4532 mm along the x-axis, 0.3610 mm along the y-axis, 0.7666 mm along the z-axis and 0.8132° about the z-axis. | Section 4.2.4<br>Section 4.2.5 |
| The computer vision component should detect and localise the construction cubes in the workspace to facilitate re-gripping dropped cubes and identifying damage to the 3D shape under construction to signal a construction halt condition. Only the cubes whose faces are visible from a vertical perspective need to be detected and localised. Cubes that need to be gripped should be localised with a positional accuracy of 2mm and a rotational accuracy of 5 degrees about the z-axis. | The positional localisation accuracy of the *Vision System* was at worst 1.4 mm along the x-axis and 1.2 mm along the y-axis. The positional localisation accuracy was 0.4914 mm on average along the x-axis and 0.4971 mm along the y-axis. The rotational accuracy was at worst 3.5° about the z-axis and on average 0.9114°. | Section 4.2.6<br>Section 4.2.7 |
| The system should detect when a cube is unintentionally dropped by the end-effector. The system should detect a cube has been dropped before the end-effector grips the next cube to be placed. | The system was able detect when a cube was unintentionally dropped by the end-effector. The system was able to detect a cube was dropped before the end-effector gripped the next cube to be placed. | Section 4.2.7 |

**Table 6. Summary of results achieved continued (1).**

| Intended outcome | Actual outcome | Location in report |
|---|---|---|
| **Field condition requirements and specifications** | | |
| The system should work under laboratory conditions. The ambient lighting level should be approximately 500 lux. | The computer vision test was performed in laboratory lighting conditions of approximately 500 lux and worked correctly. | Section 4.2.6 |
| The image background should be controlled. The immediate background of the construction cubes in the captured images should be non-reflective and of a single hue. | The immediate image background of the construction cubes was matte black rubber which is non-reflective and is of a single hue. | Section 4.2.6 |

**Table 7. Summary of results achieved continued (2).**

## 4.2 Qualification Tests

This section presents the set of qualification tests that were performed to demonstrate conformance of the overall system, and various subsystems, to the system specifications. Prior to the commencement of a number of the qualification tests, the following setup steps, hereafter referred to as the *General System Initialisation* procedure, must have been completed:

1. Ensure robotic manipulator's workspace is completely empty.
2. Power on the PC that will run the system control software.
3. Ensure the system camera has a clear view of the system workspace and connect the camera to the PC.
4. Connect the *Robotic subsystem* to the PC by connecting the micro USB port on the embedded robot controller to a USB Type A port on the PC using a USB Type A to micro USB connector cable.
5. Power on the robotic subsystem by connecting the power supply to a main's electricity outlet.
6. Start the system control software on the PC.
7. On the home screen in the system control software, verify the correct camera feed is displayed in the camera view.
8. On the same screen, select the *USB-Serial CH340* port from the available ports list.
9. On the same screen, click the *Connect to Robot* button and verify the robot is connected.
10. Select the *Construction* view in the system control software.
11. Initialise calibration of the robot by clicking the *Calibrate* button and verify the calibration sequence completes successfully.

**Qualification Test 1: Test of the system's capability to build 3D shapes**

*Objectives of the test or experiment*

The aim of this test is to determine if the system is capable of constructing a variety of novel and moderately complex 3D shapes using small cubes each with a side length of between 10mm and 15mm. Novel and moderately complex 3D shapes constitute shapes containing up to at least 30 cubes where each cube has a face parallel to the base plane.

*Equipment used*

The following equipment was used to execute this qualification test:

- PC,
- *PC System* software,
- *Robotic Subsystem*,
- USB Type A to micro USB connector cable,
- Logitech C920 HD Pro Webcam,
- Test set of ten 3D shape models [11] (*.cubeworld* files),
- and 30 aluminium cubes with side lengths of 12.6mm ±0.05mm.

*Test setup and experimental parameters*

The following steps were completed in preparation for this qualification test:

1. Ensure the *General System Initialisation* procedure has been completed.
2. Navigate to the *Construction* view in the system control GUI.
3. Click on the *Load Model* button and verify the *.cubeworld* test shape files are available for construction.

*Steps followed in the test or experiment*

The following steps were carried out to execute this qualification test:

1. Clear the *Robotic Subsystem's* workspace and place the 30 cubes at the pre-defined *source cube* locations.
2. Select and load a pre-defined test shape model in the *Construction* view of the system control GUI using the *Load Model* button.
3. Verify the correct model is loaded in the 3D shape display.
4. Start the construction process by clicking the *Start Construction* button in the system control GUI and wait for the robotic subsystem to construct the shape.
5. Compare the constructed shape to the selected test shape in the 3D shape display in the GUI and qualitatively classify the shape construction as either a success or failure.
6. Repeat all the steps 1 to 5 of the experimental protocol with a different pre-defined test shape selected in step 2.
7. Repeat step 6 until all the pre-defined 3D test shapes have been built.

*Results or measurements*

The full results from this qualification test can be found in Table 13 in the technical documentation appendix. The models and resulting shape constructions for the shapes with IDs 2 and 3 are shown in Figure 15. The following is a summary of these results:

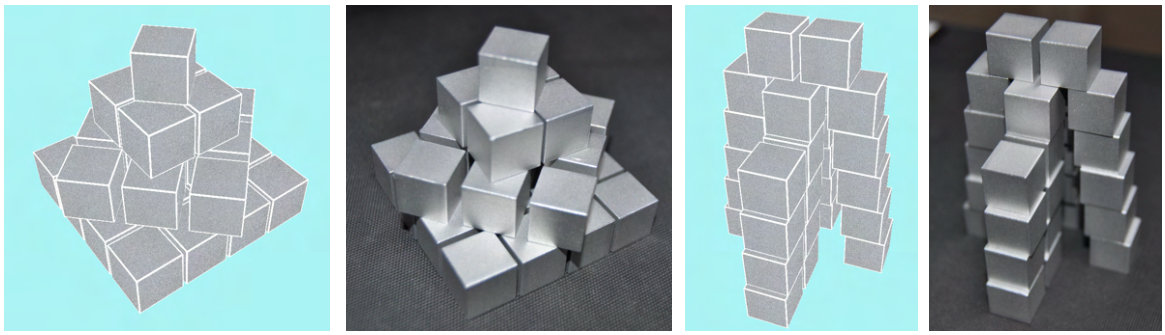
- Total shapes constructed = 50,

[11] The 3D shape models adhered to the properties outlined in Table 12 in the technical documentation appendix.

- Total construction successes = 50,
- Total construction failures = 0,
- Construction success rate = 100 %.

*Observations*

All construction processes resulted in successfully constructed shapes. No cubes were dropped and no construction failure conditions occurred during any of the construction iterations.



**(a) Shape ID 2** **(b) Shape ID 2** **(c) Shape ID 3** **(d) Shape ID 3**

**Figure 15. 3D shape models and corresponding shapes constructed by the system for qualification test 1.**

**Qualification Test 2: Test of system's capability to facilitate the definition of 3D shapes**

*Objectives of the test or experiment*

The aim of this test is to determine if the system is capable of capturing and representing a user-specified 3D shape where the position of each constituent cube is specified along each Cartesian axis as well as the orientation about the z-axis.

*Equipment used*

The following steps were carried out to execute this qualification test:

- PC,
- and *PC System* software.

*Test setup and experimental parameters*

The following steps were completed in preparation for this qualification test:

1. Start the system control software on the PC.
2. Navigate to the *Shape Design* view in the system control GUI.

*Steps followed in the test or experiment*

The following steps were carried out to execute this qualification test:

1. Click on the *Insert Cube* button in the *Shape Design* view in the system control GUI.
2. Verify a cube is displayed in the 3D shape display.
3. Record the displayed x-axis position value for the cube.

4. Translate the cube one step in the positive x-axis direction.
5. Record the displayed x-axis position value for the cube.
6. Translate the cube one step in the negative x-axis direction
7. Record the displayed x-axis position value for the cube.
8. Repeat steps 2 to 6 for translation along the y-axis.
9. Repeat steps 2 to 6 for rotation about the z-axis.
10. Click on the *Insert Cube* button
11. Verify an additional cube is added to the 3D shape display.
12. Verify the cube can be translated along the x-, y-, and z-axis and rotated about the z-axis.
13. Repeat steps 10 to 12 until 30 cubes are displayed in the 3D shape display.

*Results or measurements*

The cube linear step size on both the x- and y-axis in both the positive and negative directions was 0.1 mm. The cube rotational step size about the z-axis in both the positive and negative directions was 0.9°. 30 cubes were successfully displayed in the shape display. Each of the 30 cubes could be translated along each axis and rotated about the z-axis.

**Qualification Test 3: Test of end-effector's capability to manipulate cubes**

*Objectives of the test or experiment*

The aim of this test is to determine if the end-effector is capable of maintaining a cube in its grip under motion when the robotic manipulator is at maximum acceleration. The test also aims to determine if the end-effector is able to maintain the cube in its grip for at least 20 seconds continuously and if it is able to grip and ungrip the cube.

*Equipment used*

The following equipment was used to execute this qualification test:

- PC,
- *PC System* software,
- *Robotic Subsystem*,
- USB Type A to micro USB connector cable,
- Digital stopwatch,
- and an aluminium cube with a side length of 12.6mm ±0.05mm.

*Test setup and experimental parameters*

The following steps were completed in preparation for this qualification test:

1. Ensure the *General System Initialisation* procedure has been completed.
2. Navigate to the *Construction* view in the system control GUI.

*Steps followed in the test or experiment*

The following steps were carried out to execute this qualification test:

1. Place the cube in first position of the pre-defined *source cube* locations.
2. Click on the *Execute QTP 3* button to initiate the robot's routine for this qualification test.

3. Verify the robotic subsystem proceeds to grip the cube placed during the test setup.
4. Start the stopwatch as the cube is lifted off the base plane by the robotic manipulator.
5. Verify, the robot continuously between the extreme locations on each axis.
6. Stop the stopwatch and record the elapsed time if the cube is dropped by the robot at any point during the robot's movement sequence up to 30 seconds.
7. After 30 seconds halt the robot's movement sequence and record if the cube is still gripped by the robot.
8. If the cube is still gripped by the robot, press the *Release Actuator* button and record if the cube is released.
9. Repeat steps 1 to 7 until a total of 10 iterations have been completed.

*Results or measurements*

The full results from this qualification test can be found in Table 14 in the technical documentation appendix. The following is a summary of these results:

- Iterations performed = 10,
- Number of iterations where cube was successfully gripped = 10,
- Number of iterations where the cube was dropped before 30 seconds = 0,
- Number of iterations where the move sequence was completed with the cube still gripped = 10,
- Number of iterations where the cube was successfully released on command = 10.

*Observations*

No unexpected events occurred during any of the movement iterations which were all completed successfully.

**Qualification Test 4: Measurement of robotic manipulator linear accuracy**

*Objectives of the test or experiment*

The aim of this test is to determine the linear repeatability of the robotic manipulator's positioning along each Cartesian axis.

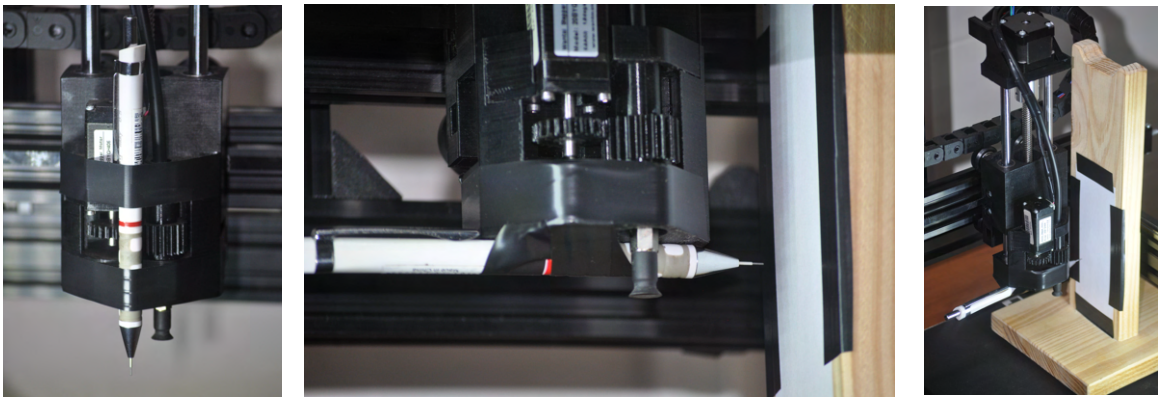
*Equipment used*

The following equipment was used to execute this qualification test:

- PC,
- *PC System* software,
- ImageJ image processing and analysis software,
- *Robotic Subsystem*,
- USB Type A to micro USB connector cable,
- Vertical flat-faced stand (see Figure 16c),
- 2 sheets of plain white A4 paper,
- Digital caliper,
- 0.5mm Mechanical pencil,
- Electrical tape,
- and a Digital camera.

*Test setup and experimental parameters*

1. Ensure the *General System Initialisation* procedure has been completed.
2. Navigate to the *Construction* view in the system control GUI.
3. Attach the mechanical pencil vertically and securely to the robot's *End-Effector Assembly* using electrical tape as shown in Figure 16a.
4. Place the two sheets of plain white A4 paper on the base plane of the robot's workspace such that entire plane accessible by mechanical pencil tip is covered.
5. Secure the paper sheets in place to the base plane using electrical tape.



**(a) Vertical pencil** **(b) Horizontal pencil attachment** **(c) Stand setup**

**Figure 16. Experimental setup configuration used during qualification test 4.**

*Steps followed in the test or experiment*

The following steps were carried out to execute this qualification test:

1. Using the position controls in the system GUI's *Construction* view, iteratively reduce the z-position of the robot until the tip of the mechanical pencil touches the sheet of paper on the base plane firmly enough to leave a mark on the paper. Record this z-position in steps as $z_{mark}$.
2. Let the x and y coordinates of the position where the robot's repeatability is being tested be denoted by $x_{test}$ and $y_{test}$ respectively. Let the x- and y- coordinates of the reference position to where the robot moves after making a mark be denoted by $x_{ref}$ and $y_{ref}$. Select these values as $x_{test} = 0$, $y_{test} = 0$, $x_{ref} = 1015$ and $y_{ref} = 1125$ steps.
3. Move the robotic end-effector to position $(x_{test}, y_{test}, z_{mark} + 50)$ where the elements of this tuple refer to the x-, y- and z-position of the robot respectively.
4. Decrease the z-position of the robotic end-effector to $z_{mark}$ to place a mark on the paper on the base plane.
5. Increase the z-position of the robotic end-effector to $z_{mark} + 50$ to remove the pencil tip from the paper.
6. Move the robotic end-effector to the reference position $(x_{ref}, y_{ref}, 2200)$ where the state tuple contains the target x-, y- and z-position of the robotic end-effector respectively.
7. Repeat steps 3 to 6 until a total of 10 iterations have been completed for the given test position.
8. Repeat steps 2 to 7 with $x_{test} = 1015$, $y_{test} = 1125$, $x_{ref} = 0$ and $y_{ref} = 0$ steps.
9. Repeat steps 2 to 7 with $x_{test} = 1015$, $y_{test} = 0$, $x_{ref} = 0$ and $y_{ref} = 1125$ steps.

10. Repeat steps 2 to 7 with $x_{test}$ = 0, $y_{test}$ = 0, $x_{ref}$ = 1015 and $y_{ref}$ = 0 steps.
11. Repeat steps 2 to 7 with $x_{test}$ = 507, $y_{test}$ = 562, $x_{ref}$ = 0 and $y_{ref}$ = 0 steps to test the repeatability in the middle of the robot's workspace.
12. Remove the and reattach the mechanical pencil horizontally and securely to the robot's *End-Effector Assembly* using electrical tape as shown in Figure 16b.
13. Attach a piece of plain white paper to the vertical surface of the flat-faced stand as shown in Figure 16c and place the stand in the back right corner of the robot's workspace with the vertical face aligned with the zy plane.
14. The z-repeatability for two z-planes are tested simultaneously by moving between the two planes and marking a point on the paper for each plane. Let the lower plane be denoted by $z_{lower}$ and the upper plane be denoted by $z_{upper}$. Select these values as $z_{lower}$ = 500 and $z_{upper}$ = 2300 steps.
15. Position the robotic end-effector such that the tip of the mechanical pencil touches the sheet of paper on the vertical stand surface firmly enough to leave a mark on the paper. Record the x- and y-position in steps as $x_{mark}$ and $y_{mark}$ respectively.
16. Move the robotic end effector to ($x_{mark}$ - *offset*, $y_{mark}$, $z_{lower}$) where *offset*=50 steps when the vertical stand is facing left and *offset*=-50 steps when facing right.
17. Set the x-position of the robotic end-effector to $x_{mark}$ to mark the point on the paper.
18. Set the x-position of the robotic end-effector to $x_{mark}$ - *offset* to remove the pencil tip from the paper.
19. Set the z-position of the robotic end-effector to $z_{upper}$.
20. Set the x-position of the robotic end-effector to $x_{mark}$ to mark the point on the paper.
21. Set the x-position of the robotic end-effector to $x_{mark}$ - *offset* to remove the pencil tip from the paper.
22. Repeat steps 16 to 21 until 10 iterations have been performed.
23. Repeat steps 15 to 22 with the vertical stand in the back left, front right, front left and centre of the robot's workspace.
24. For each cluster of point markings on each sheet of paper, set the digital caliper to 2.00 mm and press the tips of the caliper into the sheet of paper near the point such that the paper is indented with the 2.00 mm reference mark.
25. Take a photo of each cluster of point markings using the digital camera.
26. Use the ImageJ image processing software to isolate the cluster of point markings.
27. Using ImageJ, calibrate for length using the 2.00mm reference indents and measure the spread of the markings in the x, y and z directions depending on the sample. Record these measurements.

*Results or measurements*

The full results from this qualification test can be found in Tables 15, 16 and 17 in the technical documentation appendix. Tables 8 and 9 show a reduced set of these results:

| Sample Set | X Position (steps) | Y Position (steps) | Z Position (steps) | X Points Range (mm) | Y Points Range (mm) |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0,2463 | 0,2028 |
| 2 | 0 | 1125 | 0 | 0,4532 | 0,3610 |
| 3 | 507 | 562 | 0 | 0,4530 | 0,3473 |
| 4 | 1015 | 0 | 0 | 0,4211 | 0,3256 |
| 5 | 1015 | 1125 | 0 | 0,3064 | 0,2323 |

**Table 8. Range of points along the x- and y-axis measured from the point cluster images for qualification test 4.**

| Sample Set | X Position (steps) | Y Position (steps) | Z Position (steps) | Reference Length (pixels) | Z Points Range (pixels) | Z Points Range (mm) |
|---|---|---|---|---|---|---|
| 1 | 852 | 70 | 500 | 60,001 | 23 | 0,7666 |
| 2 | 852 | 70 | 2300 | 61,26 | 19,96 | 0,6516 |
| 3 | 502 | 400 | 500 | 60,962 | 18,506 | 0,6071 |
| 4 | 502 | 400 | 2300 | 60,705 | 15,173 | 0,4998 |
| 5 | 194 | 700 | 500 | 60,397 | 17,557 | 0,5813 |
| 6 | 194 | 700 | 2300 | 60,27 | 19 | 0,6304 |

**Table 9. Range of points along the z-axis measured from the point cluster images for qualification test 4.**

*Observations*

The main observation that was made during the executing of this qualification test was that the size of the pencil lead used to mark the points was similar to the range of the points distribution along all axes. No deviation among the 10 points per sample was observable with the naked eye.

*Statistical analysis*

The mean and maximum range of the points along each axis were computed across the sample sets and are shown in Table 10.

| Statistic | X Axis | Y Axis | Z Axis |
|---|---|---|---|
| *Mean (mm)* | 0.3760 | 0.2938 | 0.6229 |
| *Maximum (mm)* | 0.4532 | 0.3610 | 0.7666 |

**Table 10. Mean and maximum range for each axis across all the sample sets.**

### Qualification Test 5: Measurement of robotic manipulator's rotational accuracy

*Objectives of the test or experiment*

The aim of this test is to determine the rotational repeatability of the robotic manipulator rotational positioning about the z-axis.

*Equipment used*

The following equipment was used to execute this qualification test:

- PC,
- *PC System* software
- *Robotic Subsystem*,
- USB Type A to micro USB connector cable,
- an aluminium cube with a side length of 12.6mm ±0.05mm,
- 25 x 20 grid of 10mm squares printed on a sheet of white A4 paper,
- Digital caliper,
- Electrical tape,
- and a 30 cm ruler.

*Test setup and experimental parameters*

The following steps were completed in preparation for this qualification test:

1. Ensure the *General System Initialisation* procedure has been completed.
2. Navigate to the *Construction* view in the system control GUI.
3. Place the grid paper approximately in the centre of the robot's workspace. The grid does not need to be aligned with the robot's coordinate system.
4. Secure the grid paper in place to the base plane using electrical tape as shown in Figure 17.

*Steps followed in the test or experiment*

The following steps were carried out to execute this qualification test:

1. Choose any one of the 250 mm long grid lines and align the ruler with the line. Press down on the ruler so that the ruler does not shift from this position.
2. Place the cube on the base plane and use the ruler to align the cube with the 250mm long grid line.
3. Remove the ruler without disturbing the position or the orientation of the cube.
4. Use the robot position controls in the *Construction* view of the system control GUI to align the end-effector suction cup with the top of the cube.

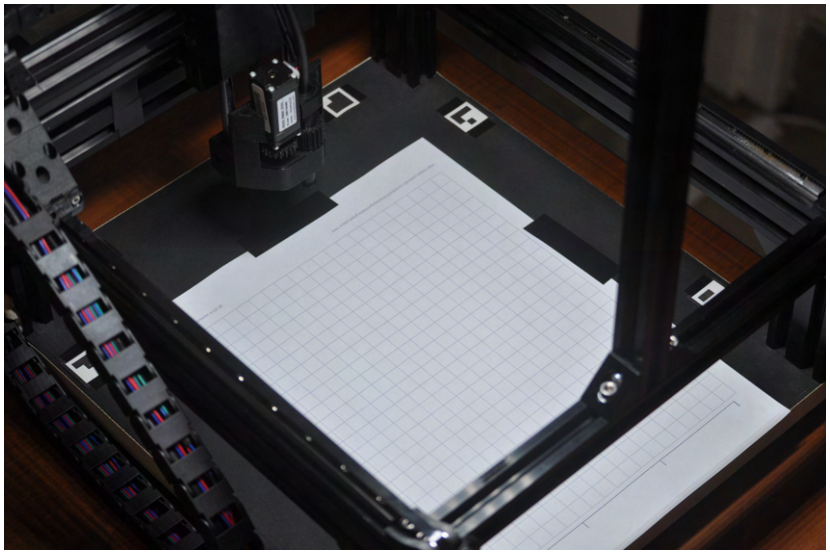


**Figure 17. Placement of grid in the robot's workspace to facilitate the measurement of the z-axis rotational repeatability.**

5. Set the rotational step position of the end-effector to 0 steps.
6. Use the robot position and actuation controls in the *Construction* view of the system control GUI to pick up the cube slightly above the base plane.
7. Set the robotic end-effector's rotational step position to either 78 steps (odd iterations) or -78 steps (even iterations).
8. Set the robotic end-effector's rotational step position back to 0 steps.
9. Use the robot controls to move the cube vertically downwards, place the cube on the base plane and release the cube.
10. Move the robotic end-effector vertically upwards and to a position that does not restrict access to the robot's workspace.
11. Press down on the top face of the cube to preserve its orientation and position.
12. Use the face nearest to the 250 mm grid line to align the ruler against the cube. Press down on the ruler to preserve the ruler's orientation and position.
13. Remove the cube from the robot's workspace.
14. Let the first and last 200 mm long grid lines be denoted by $y_0$ and $y_1$ respectively. Use the placed ruler to draw a line that extends the full length of the grid and intersects with $y_0$ and $y_1$.
15. Repeat steps to 1 to 14 until a total of 18 iterations have been completed using a different 250 mm grid line in each instance.
16. Using the digital caliper, measure the deviation of the intersection of each drawn line with $y_0$ from the intersection of the corresponding 250 mm grid line with $y_0$. Repeat this with $y_1$.
17. Calculate $\phi$, the angle of each drawn line with respect to the corresponding 250 mm grid line as

$$\phi = \arctan \frac{\delta_0 - \delta_1}{\Delta x}, \tag{32}$$

where $\delta_0$ and $\delta_1$ are the intersection deviations on $y_0$ and $y_1$ respectively while $\Delta x$ is the

length of the 250 mm grid line
18. Record these results.

*Results or measurements*

The full results from this qualification test can be found in Table 18 in the technical documentation appendix. The following is a summary of these results:

*Observations*

Notable position offsets of the axis of rotation were observed during the rotation of the cube as a result from the end-effector gear surface being uneven.

*Statistical analysis*

The mean angular deviation magnitude, maximum deviation and deviation bias was computed for all 18 angular deviation samples. These results are as follows:

- Mean angular deviation = 0.3153°,
- Maximum angular deviation = 0.8132°,
- Angular deviation bias = 0.08404°.

**Qualification Test 6: Measurement of computer vision cube detection accuracy**

*Objectives of the test or experiment*

The aim of this test is to determine the accuracy of the computer vision system in detecting and localising cubes whose faces are visible from a vertical perspective. Specifically, the test aims to determine the accuracy of the linear localisation along the x-axis and y-axis as well as the rotational pose estimation about the z-axis.

*Equipment used*

The following equipment was used to execute this qualification test:

- PC,
- *PC System* software
- *Robotic Subsystem*,
- USB Type A to micro USB connector cable,
- Logitech C920 HD Pro Webcam,
- 19 Aluminium cubes with side lengths of 12.6mm ±0.05mm,
- Mechanical pencil,
- Matte black paper,
- Electrical tape,
- and a 30 cm ruler.

*Test setup and experimental parameters*

The following steps were completed in preparation for this qualification test:

1. Place the matte black paper on the base plane of the robot's workspace and secure it in place using the electrical tape.
2. Navigate to the *Construction* view in the system control GUI.
3. Move the robotic end-effector to the four extreme points on the base plane of the

workspace and mark these point on the black paper using the mechanical pencil.
4. Draw four lines to join these points to form a rectangle on the black paper using the mechanical pencil and ruler. Using the ruler, mark every 10 mm on each of the four lines.

*Steps followed in the test or experiment*

The following steps were carried out to execute this qualification test:

1. Using the 10 mm markings as reference, place the ruler across the robot's workspace at an angle of 0° with the x-axis of the robot's coordinate system. Do this at a number of positions. In each instance, press the ruler down firmly to preserve it's orientation and position.
2. For each ruler placement, place a number of cubes by using the ruler to ensure the angle of each cube with with respect to the x-axis is 0°. Continue this process until 16 cubes have been placed.
3. Using the robot position control in the system control GUI, align the suction cup of the robotic end-effector with the centre of the top face of each cube. Record the robotic end-effector's x and y coordinates when aligned with each cube. These are taken as the known world coordinate's of each cube.
4. Click the *Process Scene* button in the *Construction* view of the GUI to trigger a capture and process action from the *Vision System*.
5. Record the position and orientation of each cube as estimated by the *Vision System*.
6. Repeat steps 1 to 5 using a ruler angle of 45 ° and a total of 19 cubes.

*Results or measurements*

The full results from this qualification test can be found in Tables 19 and 20 in the technical documentation appendix.

*Observations*

A degree of misalignment was observed between the position of the cube as estimated by the computer vision component and the actual position of the cube based on the position of the end-effector. Furthermore, a degree of high-frequency noise was observed in the computer vision's estimate of the corner positions of cubes.

*Statistical analysis*

The mean error magnitude, error bias and maximum error for the linear x-axis, linear y-axis and rotational z-axis were computed across all the cube samples and are shown in Table 11. Furthermore, the linear error for the x- and y- axis as is shown in Figures 18 and 19 using colour on a per cube point basis with the points plotted as a scatter plot of the robot's workspace.

| Statistic | X Axis (mm) | Y Axis (mm) | Z Axis Rotation (°) |
|---|---|---|---|
| *Mean Error Magnitude* | 0.4914 | 0.4971 | 0.9114 |
| *Error Bias* | -0.32 | 0.1314 | 0.3783 |
| *Maximum Error* | 1.4 | 1.2 | 3.5 |

**Table 11. *Vision System* cube detection mean error magnitude, error bias and maximum error for the linear x-axis, linear y-axis and rotational z-axis.**

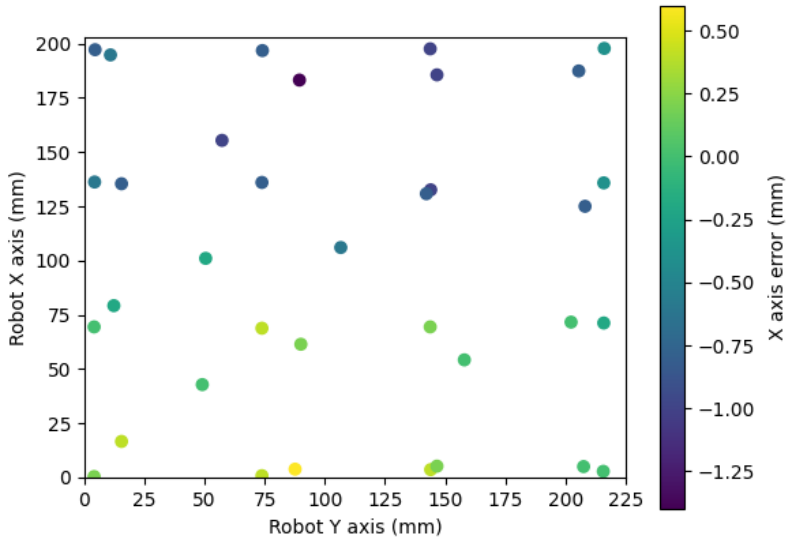



**Figure 18. *Vision System* cube position detection error along the x-axis.**

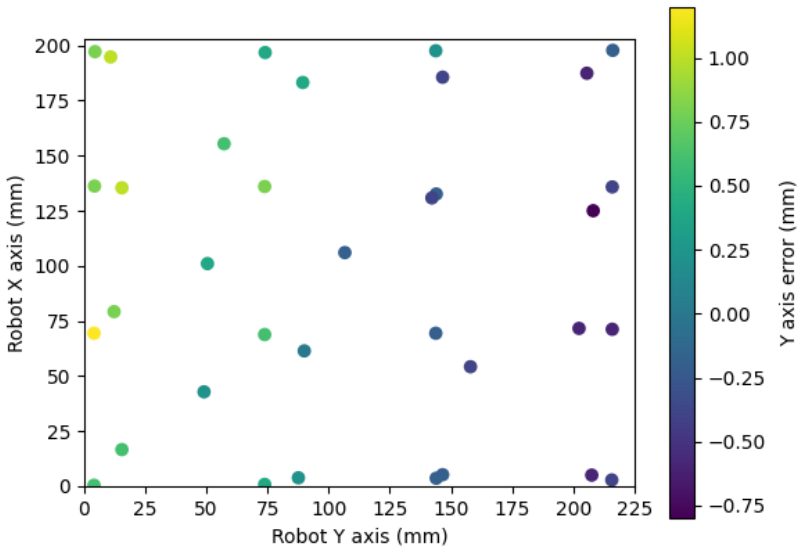



**Figure 19. *Vision System* cube position detection error along the y-axis.**

**Qualification Test 7: Test of system's capability to detect a dropped cube and and shape construction failure**

*Objectives of the test or experiment*

The aim of this test is to determine if the system is capable of detecting when a cube has been dropped by the end-effector.

*Equipment used*

The following equipment was used to execute this qualification test:

- PC,
- *PC System* software
- *Robotic Subsystem*,
- USB Type A to micro USB connector cable,
- Logitech C920 HD Pro Webcam,
- and 30 Aluminium cubes with side lengths of 12.6mm ±0.05mm.

*Test setup and experimental parameters*

The following steps were completed in preparation for this qualification test:

1. Ensure the *General System Initialisation* procedure has been completed.
2. Navigate to the *Construction* view in the system control GUI.

*Steps followed in the test or experiment*

The following steps were carried out to execute this qualification test:

1. Click on the *Load Model* button and select an arbitrary 30 cube shape model.
2. Clear the *Robotic Subsystem's* workspace and place the 30 cubes at the pre-defined *source cube* locations.
3. Click on the *Start Construction* button in the system control software to initiate the construction process.
4. For each of the first 5 cubes, click the *Release Actuator* button in the system control software to force the robot to drop the cube while carrying the cube to the structure. Note whether the robot detects the dropped cube condition before in the GUI info log before proceeding with construction.
5. For cubes 6 to 10, manually remove the cube from the grip of the end-effector during the robot's final downward movement to place the cube and place the cube in the robot's workspace.
6. Note whether the robot detects the dropped cube condition before in the GUI info log before proceeding with construction.
7. Repeat steps 5 to 6 with cubes 11 to 15, but remove the cube from the end-effector during the robot's horizontal motion while moving the cube to the structure.
8. Repeat steps 5 to 6 with cubes 16 to 20, but remove the cube from the end-effector during the robot's upward motion after gripping the cube.
9. Repeat steps 5 to 6 with cubes 21 to 25, but remove the cube from the end-effector while the robot is moving to grip the cube.
10. After the 25th cube has been placed, push the structure under construction until at least

1 cube falls from the structure.
11. Note whether the system detects a construction failure condition using the GUI info log.
12. Repeat steps 1 to 11 with a different test shape structure until 5 iterations have been completed.

*Results or measurements*

The full results from this qualification test can be found in Table 21 in the technical documentation appendix. The following is a summary of these results:

- Iterations performed = 5,
- Number of iterations where dropped cubes 1-5 were successfully detected = 5,
- Number of iterations where dropped cubes 6-10 were successfully detected = 5,
- Number of iterations where dropped cubes 11-15 were successfully detected = 5,
- Number of iterations where dropped cubes 16-21 were successfully detected = 5,
- Number of iterations where dropped cubes 20-25 were successfully detected = 5,
- Number of iterations where the final construction failure was successfully detected = 5.

*Observations*

No unexpected events occurred during any of the movement iterations which were all completed successfully. Furthermore, the robot was successfully able to re-grip each dropped cube and continue with construction in each instance.

# 5. Discussion

## 5.1 Interpretation of results

The integrated system developed in this project, which comprises of the *PC System* and the *Robotic System*, has one overarching mission. That goal is the successful construction of 3D shapes using small construction cubes. The system can be considered to have successfully attained this goal if it satisfies the first two system specifications. These are the system's capability to build 3D shapes and the system's capability to facilitate the definition of these shapes. With these two specifications met, the system is capable of capturing an arbitrary 3D shape concept and producing a physical realisation thereof. The remaining four specifications are primarily concerned with how well and how robustly the various subsystems support this task.

### *5.1.1 Shape Construction*

Considering the cube construction specification, it was observed the system had a success rate of 100% when constructing the predefined set of 3D test shapes. In order to evaluate the weight of this result, the nature of the 3D test shape set first needs to be discussed. Firstly, all of the test shapes were chosen to comply with the shape construction specification. This means that all of the 3D test shape models consisted of at least 30 cubes[12] and were at least 4 cubes in height. Furthermore, recall that the fourth system specification requires at least 2mm of linear repeatability. Based on this, the 3D test shape models were assigned a spacing tolerance of 2 mm between cubes. However, within these constraints, there are an infinite number of shapes that can be constructed.

To the student's knowledge, there is no known set of shapes that, if constructed, definitively proves all shapes within these constraints are constructable by the system. As such, the test shapes were chosen to exhibit as wide a range of properties across the set as possible that may arise from arbitrarily defined shapes. These properties included arbitrary cube rotation about the z-axis, partially supported cubes, structures up to the maximum height of 6 cubes, single cube tower stacks, slanted single cube tower stacks and finally structures that span the majority of the robot's workspace. Lastly, the tolerance between cubes was reduced to 1 mm between cubes for some structures to test the envelope of the system's capability. With the nature of the 3D test shapes considered, the 100% construction success rate indicates with a strong degree of confidence that the system is capable of successfully constructing arbitrary 3D shapes.

[12] In fact, each test shape consisted of exactly 30 cubes as, at the time of writing, only 30 cubes were available for testing purposes.

### *5.1.2 Shape Definition*

All of the test shape's defined using the *Shape Definition* component of the system which is indicative that the system is capable of facilitating the definition of arbitrary 3D shapes. It was shown that the *Shape Definition* input controls allow the linear x- and y-axis position of each individual cube to be specified with a resolution of 0.2 mm and the rotational position of each cube to be specified with a resolution of 0.9 degrees. These values originate from the resolution of the stepper motors in the *Robotic Subsystem*. Specifically, the use of a 20 tooth timing belt pulley on a stepper motor with 1.8 degrees of resolution in full-step mode results in a linear step size of 0.2mm. Furthermore, the 1.8 degree end-effector rotation stepper motor has a step size of 0.9 degrees in half-step mode. Therefore, the use of higher resolutions in the *Shape Definition* component is not practical as this is not physically realisable. Based on this, the system meets the shape definition specification which requires 1 mm linear step resolution and 1 degree rotational resolution.

The *Shape Definition* component relies predominantly on OpenGL to create a 3D model of the shape under construction that can be interacted with to create the structure. The use of a 3D model to facilitate shape construction provides a far more intuitive experience for the user than simply specifying the coordinates of each cube or working with 2D slices of the shape. Furthermore, the lower-level graphics API nature of OpenGL also offered further control over the scene being rendered than higher level solutions which allowed the *Shape Definition* component to be better tuned to the task at hand. However, these benefits are offset by the greater time investment that is required when developing software with OpenGL. Due to the simplistic nature of the rendering task in the *Shape Definition* component, this was not considered a major drawback.

### *5.1.3 Robotic System*

The system depends on the end-effector as interface to enable the system to interact with the cube. Therefore, in order, to construct 3D shapes, the system relies heavily on this component during the manipulation of the pose of each cube. As such, the success of the system observed when testing the system's shape construction ability implied that the end-effector was capable of manipulating cubes. However, there is no guarantee that the most strenuous conditions possible for the end-effector were arbitrarily encountered during this process. Therefore, due to the critical nature of this component, a test was required to verify its capability in isolation. Based on the acceleration and movement speed of similar existing systems as well as the approximate dimensions of the workspace, it was estimated that the maximum duration the end-effector would be required to grip the cube for would never exceed 20 seconds. The third system specification is based on this estimate.

The test results showed that for 10 iterations, the end-effector achieved a 100 % success rate in maintaining the cube in its grip for 30 seconds. The test was halted at 30 seconds which indicates the maximum possible duration is likely longer, yet still clearly demonstrates the end-effector complies with the specification. Furthermore, the nature of the test means the end-effector was exposed to the maximum force it was expected to endure during the period of

motor acceleration. This is backed by early tests of the vacuum system which yielded holding times of up to 8 hours. However, there may be trade-off with the lifetime of the vacuum system servo motor and the suction force. The first servo motor installed in the vacuum system failed and ceased during operation. It is suspected that the large forces endured by the motor when inducing a comparatively low pressure in the vacuum system shortened the lifespan of the motor. However, despite reducing the suction force of the vacuum system with the new servo motor, the end-effector still performed within the desired specification. A number of other approaches were considered for the end-effector mechanism from electromagnetic solutions to finger grippers. However, only the suction cup based vacuum approach met all the functional requirements of the end-effector and was selected as a result.

The tests concerning the capability of the system to construct arbitrary 3D shapes generally only offer information at a qualitative level. This is due to the difficulty that exists in measuring the accuracy of the position and orientation of the cubes internal to the structure under construction. Therefore, to attain a quantitative measurement of this process, the assumption was made that the accuracy of the constructed 3D structure can be assessed, at least in part, from the accuracy of the mechanism responsible for placing the cubes. The repeatability of the robotic manipulator is the quantitative indicator that was assessed during the testing process of this component. The linear repeatability of robotic manipulator along each Cartesian axis and its rotational repeatability about the z-axis is highlighted in Table 6.

In comparison to the specification for the robot's repeatability, the repeatability along each linear axis is very good and comfortably within the required 2mm. The rotational repeatability is excellent in comparison to to the 5° requirement. The rotational repeatability remains impressive when compared to the rotational step size. However, the linear step size of 0.2mm is notably less than the linear repeatability. This would indicate that the stepper motors are potentially losing steps. However, due to the nature of the repeatability test, the linear repeatability performance is likely underrated. Specifically, a mechanical pencil with a lead size of 0.5 mm was used to mark the position of the end-effector during the test. This size is comparable to the magnitude of the robot's repeatability measurement which was based on the spread of pencil marks. This indicates the width of the pencil lead likely inflated the repeatability readings.

One way to correct for this would be to simply subtract the width of a single mark left by the pencil from the repeatability measurement. However, there are a number of variables that determine the size of a mark and therefore an exact measurement of this is difficult to ascertain. However, for the purposes of the repeatability test in this project, it was considered sufficient to ignore this step and simply use the measurement as an upper bound that is highly unlikely to be an underestimation. Therefore, the repeatability was judged to have met the specification with a very high degree of confidence. The accuracy of the *Robotic System* is largely based on the robust design of the mechanical manipulator.

Previous iterations of this project suffered with precision control issues due to vibration. Therefore, a large emphasis was placed on the mechanical design aspect of this project to eliminate these issues. A gantry robot approach was selected with these factors in mind as this design generally exhibits better accuracy than the articulated robot and SCARA approach. The gantry approach offered a much greater opportunity to ensure stability through its frame. Finally, the accuracy of gantry robots are constant across their workspace and are better suited

to Cartesian based problems as a result. In order to introduce stability to the *Robotic System*, 20x40 aluminium extrusions were used instead of the linear rail seen in previous projects. Furthermore, the robot was designed such that the *Y-Axis Assembly* moves on top of the frame to allow the frame's centre of gravity to be lowered. This design approach resulted in a very stable *Robotic System* that mostly eliminated vibration issues and facilitated precision control. Overall, the mechanical component of the *Robotic System* was highly successful in fulfilling its functions and provided a solid foundation for the other subsystems within this project.

### *5.1.4 Computer Vision System*

As mentioned in Section 3.5, the *Vision System* is not required for open-loop shape construction but is required to make the system closed-loop so that it can handle unexpected events. The *Vision System* is centred around the detection of cubes and the classification of cubes into *source cubes*, *structure cubes* and *independent cubes*. The reliability of this classification as well as the accuracy with which the robot is able to re-grip a dropped cube depends on the accuracy with which the *Vision System* is able to estimate the pose of a cube in the world coordinate system. The accuracy of cube localisation of the *Vision System* for the base plane is is highlighted in Table 6.

These results indicate that the *Vision System* cube localisation process meets the linear accuracy specification of 2mm as well as the rotational accuracy specification of 5°. When gripping a cube, the distance from the edge of the cube to the outer perimeter of the suction cup centred on the top face of the cube is greater than the cube localisation linear deviation. This implies that the localisation inaccuracies should not prevent the system from re-gripping a cube. However, there is still a degree of inaccuracy introduced nonetheless which limits how far the inter-cube tolerance can be reduced when the system constructs a shape in closed-loop mode. However, with an inter-cube tolerance of greater or equal to 2 mm should almost always ensure successful construction in closed-loop mode.

The main source of inaccuracy is introduced by the pin-hole camera based approach used to map between the world coordinate system and the image coordinate system. This approach is dependent on the accuracy of the known positions of the fiducials in the world coordinate system. However, due to the obstructed nature of the robot's workspace and the form of the end-effector which defines the robot's workspace, it is challenging to measure the exact world coordinates of the each fiducial. Furthermore, the centre point of each fiducial is identified by the *Vision System* as the centre of the fiducial's contour. However, lighting variations and noise in the square corner detection processes can cause misalignment between the centroid and actual fiducial centre. These sources of uncertainty contribute to the inaccuracy exhibited in the cube localisation process.

Since the dropped cubes to be re-gripped are almost exclusively found on the base plane, the use of a homography to perform the mapping was considered. However, the cube classification step requires the classification of cubes on all planes which cannot be solved using only a homography. Overall the cube localisation approach was considered successful, however, the system would benefit from accuracy improvements in this process.

The system makes use of the vacuum system pressure sensor to detect a dropped cube case

and the *Vision System* to deal with the dropped cube case and detect the construction failure case. The test of the system's response to unexpected events showed that the system was able to respond to all dropped cube cases and detect all construction failures. The dropped cube test involved forcing a dropped cube response by removing the cube during every possible phase of the robot's cube handling movement during construction. This revealed that the system was also successfully able to deal with the missing cube case where a cube is removed from its *source cube* position before the end-effector reaches this position. Supplementing the *Vision System* with the pressure sensor made the dropped cube detection mechanism robust as indicated by the test results. Finally, each construction failure case was also successfully detected. These results demonstrate compliance of the system with the sixth and final system specification.

## 5.2 Critical evaluation of the design

### *5.2.1 Aspects to be improved in the present design*

Due to the number of subsystems within this project and the depth to which each subsystem was explored, it is expected that there would be many facets which could not be explored sufficiently or issues which could not be corrected within the given time frame. As a result, there are many aspects that could be improved in the current design. With respect to the mechanical component of the *Robotic Subsystem*, the z-axis motor mount exhibited issues in securely gripping the *Z-Axis Assembly* linear rods. The z-axis motor mount relied on the tightness of the hole around the linear rod in the 3D printed part to create a connection. However, this connection became looser over time and should be replaced with a clamping mechanism. Secondly, the robotic manipulator was far more rigid than anticipated. This is a desirable characteristic, however, the height of the system could be increased significantly to take advantage of this and facilitate taller structure construction. Lastly, the 3D printed rotational end-effector rod has slightly too much play which reduces the cube placement accuracy. This part should be re-printed with tighter tolerances.

The design of the robotic controller had an error that was not uncovered until the PCB was received and assembled where two of the I/O pins were not powered correctly. These pins corresponded to the y-axis motor driver microstepping mode selection inputs. This meant the motor could not be configured in 1/32 microstepping mode but rather only 1/16 microstepping mode which introduced additional vibrations and noise. This design error should be corrected and the PCB re-manufactured. In addition, the global semi-conductor shortage resulted in the selection of a microcontroller with a slightly lower clock speed than desired. A faster microcontroller would allow a higher stepper motor pulse rate and faster system operation.

The first aspect of the *Vision System* that needs to be improved is the system's tolerance to lighting variability. The system uses a fixed camera exposure level and binary threshold level which need to be adjusted when operating in notably different lighting conditions. This should be upgraded to an adaptive threshold mechanism. Secondly, the mapping between the world and image coordinate system only considers the camera intrinsics and not the distortion coefficients. Including these in the mapping computation should improve cube

localisation accuracy. In addition, the fiducial pattern should be updated to facilitate more accurate identification of a world reference point as discussed in 5.1.4.

The *Shape Definition* component also has a number of elements that can be improved. The primary form of cube manipulation is through a number of keyboard controls to alter the position and orientation of the cube. However, when a cube needs to be moved to a position far from its initial insertion point, it can take a while due to the small step resolution of the system. The implementation of controls to specify the coordinates of the cube directly as well as the option to change the step resolution of the translation action would improve the usability of the *Shape Definition* component.

### *5.2.2 Strong points of the current design*

The primary strong point of the design is the rigidity of the mechanical component of the *Robotic System*. This was discussed in depth in Section 5.1.3. The rigidity provides a very stable foundation for the motion of the *Y-Axis Assembly* and *X-Axis Assembly* which ultimately results in a high degree of accuracy in the positioning of the robotic end-effector. This in turn allows shapes to be constructed with a small tolerance which facilitates the construction of a wide range of shapes. Furthermore, this rigidity, combined with the low centre of mass of the robotic manipulator reduces the vibrations that reach the robot's workspace and end-effector significantly.

The embedded robotic controller has also proved to be a very reliable component of the design. The controller has only required minor firmware tweaks since its completion and allowed the design focus to be on the *PC System* software for the latter phases of the project. The controller is also robust and highly compact as it was manufactured as a PCB. Lastly, the *Vision System* approach, despite having the potential for accuracy improvements, has provided a very solid foundation for mapping between the image frame and the world frame. Once the mathematical challenges in projecting back from the image frame to the world frame were resolved, the solid theoretical foundation of the approach allowed the relatively fast and seamless introduction of fiducials, cube classification based on location and vision detection region bounding in the world plane into the *Vision System*.

### *5.2.3 Under which circumstances is the system expected to fail?*

The *Shape Definition* component has a number of ways in which a failure state could be induced. Firstly, the component does not include any intersection or collision logic between cubes in the design. Therefore, a design with intersecting cubes could be produced which would lead to a failed construction in the physical world. Secondly, the component does not perform any analysis of the physics of the cubes. Therefore, it is possible to design a shape where the cube is not correctly supported which would also lead to a construction failure in the physical world. It is also possible for a system failure to occur during a dropped cube event. If a cube is dropped near a structure, it is possible for the *Z-Axis Assembly* to collide with the structure when moving down to re-grip the cube.

## 5.3 Design ergonomics

During the development of the system, a number of design decisions were made that took the ergonomics of the design into consideration. Firstly, the robotic manipulator was designed to be as compact as possible. The design of the custom *X-Axis Assembly* belt clamps that do not protrude from the assembly to improve compactness along the x-axis is an example of this. The culmination of these decisions resulted in a robot that fits on a desktop with all the cables and tubes routed very cleanly to the back left of the robot. The aesthetics of a system can also be considered as an ergonomics factor. The robot exhibits a sleek plain black design with white highlights that gives the the robot a professional feel. Lastly, the robot was designed with the intention of being viewed from the front perspective. This facilitates a view of the shape under construction with minimal occlusions.

The *System Control* software also exhibits a number of ergonomic features. The *Shape Definition* component allows the intuitive 3D design and inspection of the shape to be constructed. Furthermore, the *Construction* view interface offers a real-time 3D visualisation of the cube belief states during the construction process. Furthermore, this can be viewed in isolation or in conjunction with the computer vision feed which can also be viewed in isolation. The computer vision display also has a number of controls that allow the computer vision stages and information displayed to be customised.

## 5.4 Health, safety and environmental impact

The primary safety concern in this project is electrical safety. The greatest danger is posed through the main's electricity wires connecting to the power supply. In order to protect against this, a cover for the power supply connection pins was designed and 3D printed. It is not possible to touch the electrical connection points without removing the cover. The main PCB board was designed with mounting holes to facilitate mounting in an electrical cover box which had not yet been created at the time of writing. A number of wires were spliced together in the design. Heat shrink tubing was used to cover these connections. Lastly, the moving mechanical parts of the robot pose an injury threat. Limit switches are included in the design which prevents the robot from exceeding the limits of its axes. However, it cannot sense obstructions in its path so care must be taken not to get body parts caught in the robot during operation.

In terms of environmental impact, the stepper motors of the robot produce acoustic noise during operation. Microstepping was implemented which reduces the vibrations generated by the motors which in turn reduces the acoustic noise generated. Furthermore, by ensuring the motors only rest in full step positions, the ringing noise that is emitted in microstep positions is eliminated. The PCB was designed in such a manner as to reduce the size of the current loops formed which minimises electromagnetic noise. In addition, the use of a ground plane on the PCB also helps to shield electromagnetic noise.

## 5.5 Social and legal impact of the design

The technology developed and utilised in this project relates directly to the field of industrial robotics and automation. If social issues are considered from the perspective that the technology developed in this project supports the rate of development in these domains, then there may be an increase in the number of jobs lost to automation as a result. However, such efficiency improvements will likely lead to the creation of wealth, and therefore jobs, in other domains. The project can also be viewed as a general robotic platform which could be purchased and adapted to various applications, such as 3D printing or laser engraving. In this sense, the project has the potential to generate wealth and improve the economic standing of one or more individuals. Lastly, in the same vein of thought, the project as a product could serve as a educational platform as an entry point to using computer vision with robotics in a highly constrained environment. However, due to the unavoidable potential for injury resulting from the mechanical component of the robot, the legal issues relating to this need to be taken into consideration.

# 6. Conclusion

## 6.1 Summary of the work completed

This report details the work that was performed during the design and development of a robotic system with the overarching goal of constructing arbitrary 3d shapes using small construction cubes.

A literature study was undertaken into computer vision approaches to object detection, with a focus on traditional techniques, as well 3D object localisation methods and their application to the robotics domain. Firstly, a gantry robot was designed from first principles and manufactured using a combination of 3D printing and metal machining technologies. Following this, the hardware of embedded robot control circuit was designed from first principles and a prototype was created on a breadboard. The software was implemented on the embedded controller using C. A PCB was designed for the circuit and sent for manufacturing overseas.

A 3D render based GUI was developed using a low-level graphics API to facilitate the definition of 3D shapes. A computer vision system was developed to detect and localise the cubes within the robot's workspace. Finally, PC-based software was developed to integrate the shape definition GUI and computer vision components as well as to control the robot. A number of test shapes were defined using the shape definition GUI and constructed closed loop by the gantry robot supported by the computer vision system.

## 6.2 Summary of the observations and findings

The system developed, which had a PC-based software component and a robotic system as its primary two constituents, was successful in fulfilling the overarching goal of constructing arbitrary 3D shapes using small construction cubes. The system was capable of constructing all test shapes that met the minimum specifications of containing 30 cubes with at least four cubes in height. In addition, the system was capable of constructing shapes up to six cubes in height that contained arbitrary cube rotations about the z-axis, partially supported cubes, small inter-cube tolerances and leaning cube stacks.

The system was able to perform the construction closed loop by using the computer vision system to assist in handling unexpected events. Specifically, the system was able to successfully detect, re-grip and re-orient the cube in the dropped cube case and issue a construction failure signal in the structural damage case. A gantry robot approach with a design focus on the rigidity of the mechanical component was found to be a successful approach to the cube construction task. Furthermore, the pin-hole camera model based approach to 3D cube localisation component of the through mapping image coordinate system to the world coordinate system was found to be a robust solution for the computer vision component of this task.

## 6.3 Contribution

The domain of mechanical design for robots and the construction needed to be explored to complete this project. In particular, the domain of CAD software, and specifically the Fusion 360 CAD software package, needed to be mastered to assist in the creation of the mechanical component of the *Robotic System*. Furthermore, an understanding of the functionality and applicability of a number of mechanical components, including linear drive and linear motion systems, needed to be acquired. In particular, this included the integration and control of servo and stepper motors. The mechanical construction required the attainment of knowledge to facilitate the direct use of a 3D printer as well as metal machining tools such as a lathe and milling machine. All of the aforementioned components are common in a Mechanical Engineering undergraduate course but are all non-existent in a Computer Engineering undergraduate course. The study leader provided helpful guidance in terms of highlighting the challenging aspects of the mechanical design which should be focused on as well as favorable characteristics that should form part of the design.

A combination of new theory and the approaches arising from this theory needed to be mastered in the computer vision domain. Specifically, traditional computer vision techniques used for object detection needed to be understood and implemented as well as 3D localisation approaches. In service of the latter aspect, knowledge of the pin-hole camera model needed to be acquired and used in mapping between the image coordinate system and world coordinate system. This approach followed from the study leader's suggestion to use and explanation of camera intrinsics and extrinsics. Furthermore, knowledge of the computer vision library OpenCV was acquired to support the computer vision system development at various stages. None of this computer vision knowledge is covered by undergraduate modules.

In a number of undergraduate modules, first principles 8-bit microcontroller development and 32-bit microcontroller development boards with hardware abstraction libraries were explored. The first principles development of embedded software for a 32-bit microcontroller as well s the complete first principles design of the controller circuit required the attainment of new knowledge. Furthermore, the development of a PCB for the controller required the PCB design software KiCAD to be mastered. Lastly, for the shape definition component, new knowledge about the theory relating to the graphics pipeline and transformation matrices used in 3D graphics rendering needed to be acquired. The use of the low-level graphics API OpenGL needed to be understood for this purpose.

There were no novel software algorithms or hardware circuits developed in this project. However, the design of the mechanical component of the *Robotic System*, the hardware of the embedded controller circuit, the embedded software, the 3D rendering software, the computer vision software and system controller was completely from first principles which resulted in unique designs and implementations for each of these facets. During the course of the development of these components, the study leader highlighted the challenging facets of each which should receive the requisite attention.

Libraries were relied on heavily during the initial design and prototyping phase for the computer vision system and embedded controller. The embedded controller implementation

was converted completely to a first principles implementation. The core aspects of the computer vision system were developed from first principles while basic image processing functions were retained from the OpenCV library. The calibration aspects of the computer vision component were also considered as not a core aspect of the computer vision system and OpenCV was used for this purpose. Lastly, the high-level idea for the approach to the custom square corner detection algorithm was inspired by a student in the same research group in their approach to detecting puzzle-piece corners. However, the design and implementation of this algorithm was from first principles and only loosely related.

## 6.4 Future work

The success of the design of the robotic subsystem provides a solid foundation for further development going forward. The first aspect should be investigated further is the improvement of the cube localisation accuracy of the computer vision system to improve the tolerance used for shapes constructed in a closed loop manner. Secondly, the use of a stereo vision computer vision approach or an ToF camera over the monocular vision approach used in this project are possible future avenues of exploration that would eliminate the need to assume the z-plane in which a cube is detected. In addition, relatively simplistic path planning approaches were used in this project. Therefore, a possible avenue of improvement would involve an investigation into the use of more sophisticated path planning approaches.

For a given construction sequence, the cubes are made available to the robot by placing the cubes in pre-defined locations. An alternative approach that should be explored involves the arbitrary initial placement of the cubes within the robot's workspace followed by the implementation of an algorithm that would allow the robot to detect the cubes and construct the shape from this initial state. Finally, on a related note, it was observed that one of the main sources of inaccuracy in the system was the deviations introduced when the robot gripped the *source cube* for construction. Therefore, further work into improving the accuracy of the *source cube* attainment mechanism should be done.

# 7. References

[1] G. C. Burdea and H. J. Wolfson, "Solving jigsaw puzzles by a robot," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 752–764, Dec. 1989.

[2] G. S. Sharath, N. Hiremath, and G. Manjunatha, "Design and analysis of gantry robot for pick and place mechanism with Arduino Mega 2560 microcontroller and processed using pythons," *Materials Today: Proceedings*, vol. 45, pp. 377–384, Jan. 2021.

[3] G. Lundstrom, "Industrial robot grippers," *Industrial Robot*, vol. 1, pp. 72–82, Feb. 1973.

[4] R. Miller, *Robots and Robotics Principles, Systems, and Industrial Applications*. New York: McGraw-Hill Education, 2017.

[5] The MathWorks, Inc., "Object recognition," Accessed Oct. 30, 2021. [Online]. Available: https://www.mathworks.com/solutions/image-video-processing/object-recognition.html

[6] A. Kumar, "An overview of visual servoing for robot manipulators," Control Automation, May 15 2020. [Online]. Available: https://control.com/technical-articles/an-overview-of-visual-servoing-for-robot-manipulators/

[7] J. Xiao, B. C. Russell, and A. Torralba, "Localizing 3D cuboids in single-view images," *Advances in Neural Information Processing Systems*, vol. 1, 2012.

[8] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 679–698, Nov. 1986.

[9] C. G. Harris and M. J. Stephens, "A combined corner and edge detector," in *Alvey Vision Conference*, vol. 15, 1988.

[10] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[11] P. Guruprasad, "Overview of different thresholding methods in image processing," in *Proc. TEQIP Sponsored 3rd Nat. Conf. ETACC*, Jun. 2020, pp. 1–4.

[12] X. Y. Gong, H. Su, D. Xu, Z. T. Zhang, F. Shen, and H. B. Yang, "An overview of contour detection approaches," *International Journal of Automation and Computing*, vol. 15, pp. 656–672, Jun. 2018.

[13] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[14] S. Yokoi, J. ichiro Toriwaki, and T. Fukumura, "An analysis of topological properties of digitized binary pictures using local features," *Computer Graphics and Image Processing*, vol. 4, no. 1, pp. 63–73, 1975.

[15] H. Wei and B. Y. Chen, "Robotic object recognition and grasping with a natural background," *International Journal of Advanced Robotic Systems*, vol. 17, pp. 1–17, Mar. 2020.

[16] N. Aggarwal and W. C. Karl, "Line detection in images through regularized Hough transform," *IEEE Transactions on Image Processing*, vol. 15, pp. 582–591, Feb. 2006.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, p. 91–110, Nov. 2004.

[18] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[19] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443.

[20] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571.

[22] C.-Y. Lin and C.-L. Hsueh, "Recognition technique for character cube stacking robot," in *2008 10th International Conference on Control, Automation, Robotics and Vision*, Dec. 2008, pp. 791–796.

[23] K. Liu, W. Shang, S. Du, and S. Cong, "6-DOF object localization by combining monocular vision and robot arm kinematics," in *2017 36th Chinese Control Conference (CCC)*, 2017, pp. 6575–6580.

[24] P. Azad, T. Asfour, and R. Dillmann, "Stereo-based 6D object localization for grasping with humanoid robot systems," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 919–924.

[25] R. Szeliski, *Computer Vision : Algorithms and Applications*. London: Springer, 2011.

[26] OpenCV team, "Camera calibration and 3D reconstruction," Accessed Oct. 31, 2021. [Online]. Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

[27] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[28] The MathWorks, Inc., "What is camera calibration?" Accessed Nov. 01, 2021. [Online]. Available: https://www.mathworks.com/help/vision/ug/camera-calibration.html

[29] OpenCV team, "Real time pose estimation of a textured object," Accessed Nov. 6, 2021. [Online]. Available: https://docs.opencv.org/3.4.15/dc/d2c/tutorial_real_time_pose.html

[30] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *International Journal of Computer Vision*, vol. 81, pp. 155–166, 02 2009.

[31] Z. Zhang, Y. Hu, G. Yu, and J. Dai, "DeepTag: A general framework for fiducial marker design and detection," *ArXiv*, vol. abs/2105.13731, 2021.

[32] M. Kostak and A. Slaby, "Designing a simple fiducial marker for localization in spatial scenes using neural networks," *Sensors*, vol. 21, no. 16, p. 5407, 2021.

[33] D. Wagner and D. Schmalstieg, "ARToolKitPlus for pose tracking on mobile devices," in *Proceedings of 12th Computer Vision Winter Workshop (CVWW'07)*, Feb. 2007.

[34] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3400–3407.

[35] M. Hirzer, "Marker detection for augmented reality applications," Inst. for Computer Graphics and Vision, Graz University of Technology, Austria, Tech. Rep. ICG–TR–08/05, Oct. 2008.

# Part 4. Appendix: technical documentation

# HARDWARE part of the project

## Record 1. System block diagram

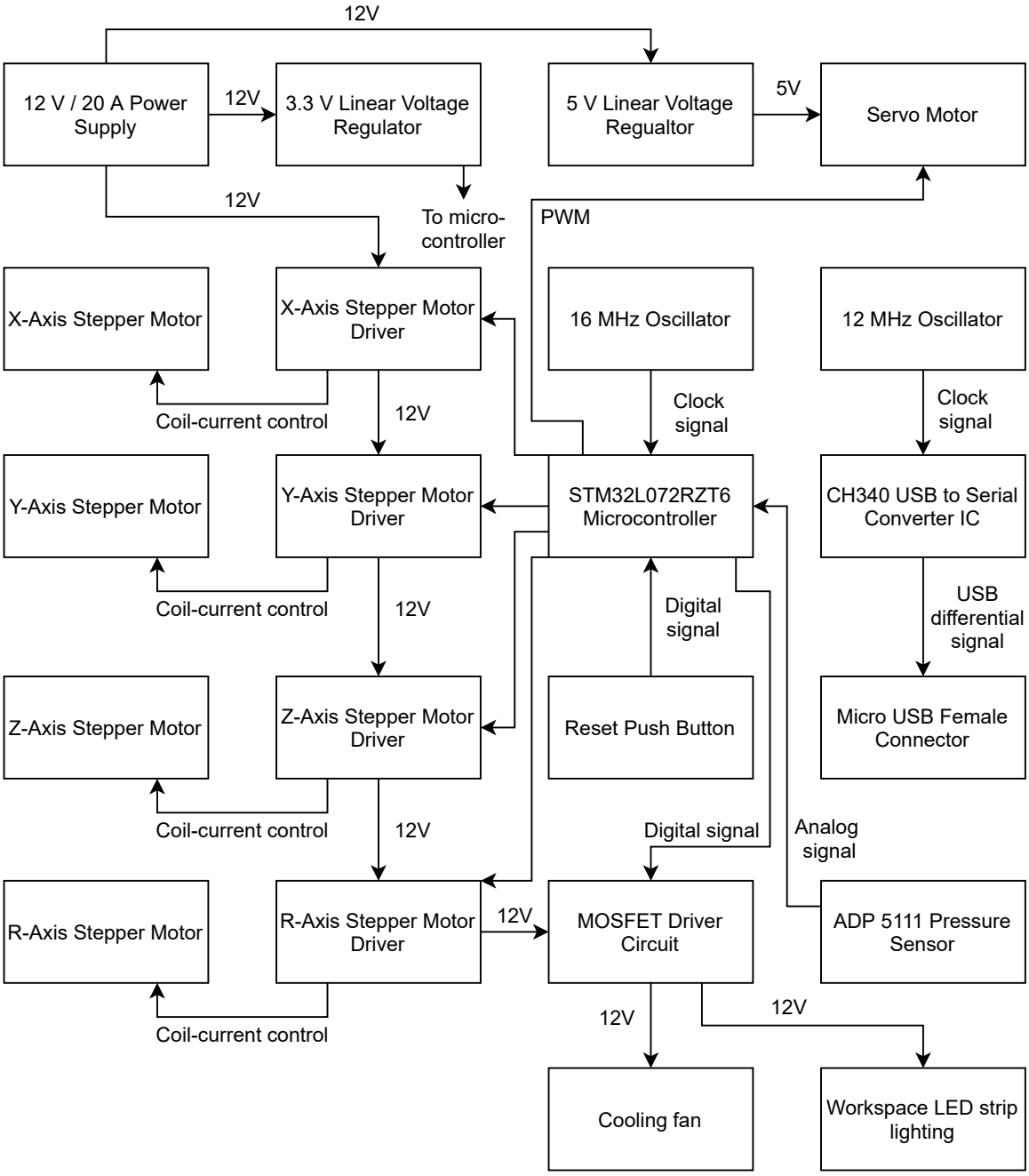


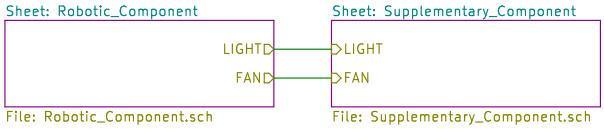

**Figure 20. System block diagram of the hardware component of the system.**

## Record 2. Systems level description of the design

The hardware component of the system that is captured by the system block diagram shown in Figure 20 consists of three primary components. The first component is the hardware that forms part of the embedded robotic controller itself. The second is the servo motor and pressure sensor that form part of the vacuum generation mechanism while the third is the set of stepper motors used to drive the robotic manipulator. The entire hardware component is centred around the STM32L072RZT6 microcontroller. In order to power the system, a 12 V and 20 A power supply was used. Since components operate at different voltage levels within the system, there is also a 3.3 V and a 5 V linear voltage regulator. The microcontroller itself uses the 3.3 V linear voltage regulator and is clocked by a 16 MHz oscillator. The microcontroller controls each of the robotic manipulator stepper motors through the use of DRV8825 stepper motor drivers. These take a 3.3 V signal as input and pass the 12 V power through to the stepper motors in a controlled fashion. The vacuum system servo motor is powered by the 5 V linear regulator and controlled by the microcontroller with a 3.3 V PWM signal. Similarly, the microcontroller also uses MOSFET circuits to control a cooling fan and the system workspace lighting. Lastly, the microcontroller takes an analog signal from the vacuum system pressure sensor as input to an ADC.

## Record 3. Complete circuit diagrams and description

See the next three pages for complete schematic of the embedded robotic subsystem controller circuit schematic that was created as part of the PCB design process.

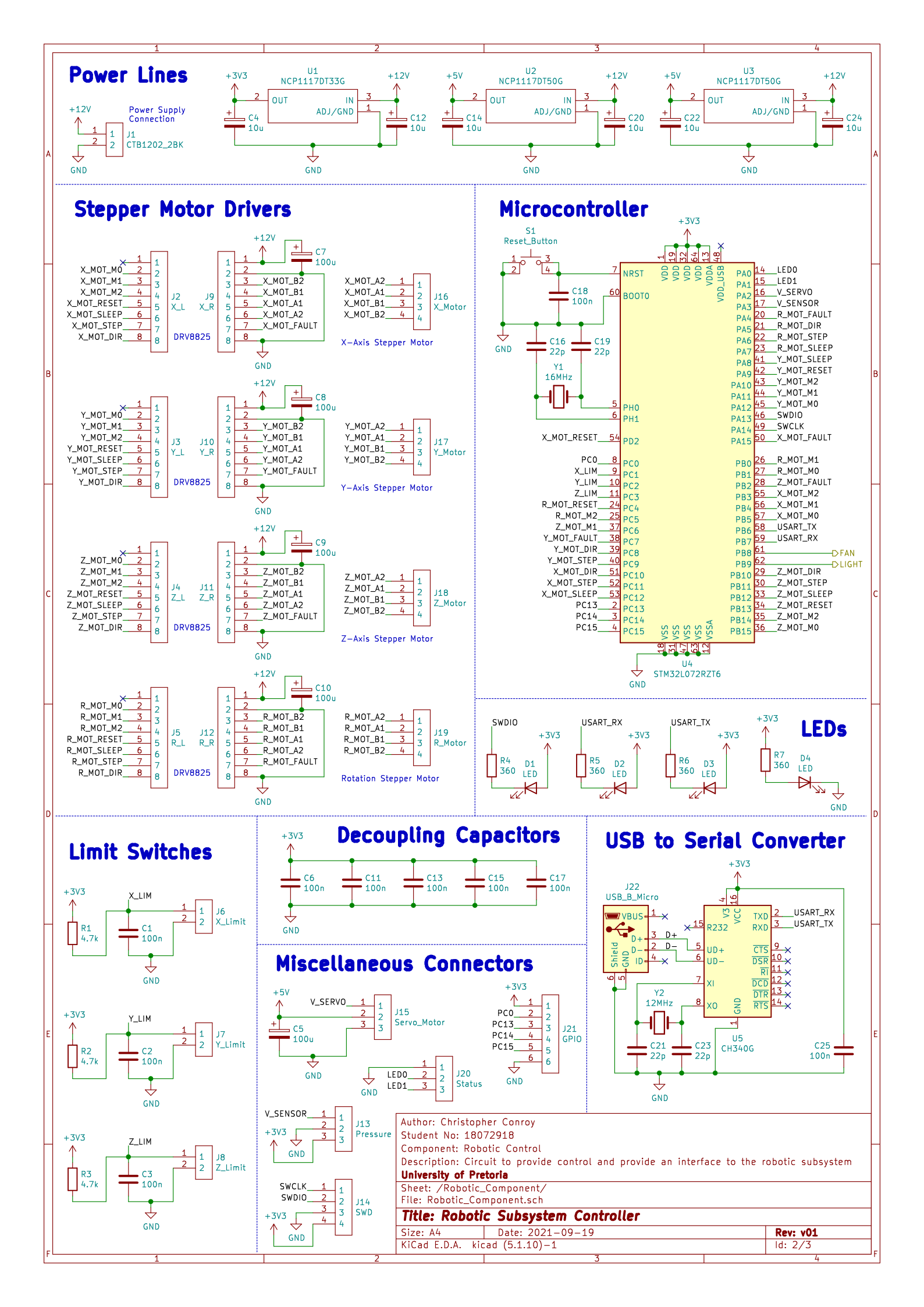



Author: Christopher Conroy
Student No: 18072918
Component: Integrated
Description: High level structure of the controller
**University of Pretoria**

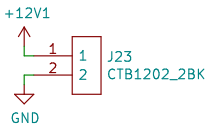
Sheet: /
File: Robot_Controller.sch

***Title: Robotic Subsystem Controller***

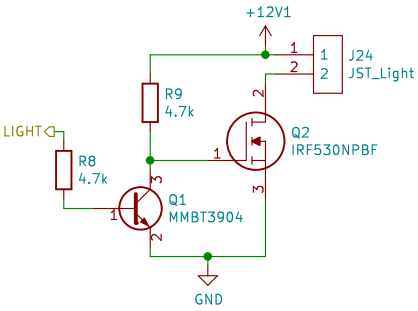
| Size: A4 | Date: 2021-09-19 | **Rev: v01** |
|---|---|---|
| KiCad E.D.A. kicad (5.1.10)-1 | | Id: 1/3 |


Power Lines
+12V
Power Supply Connection
J1
CTB1202_2BK
GND
+3V3
U1
NCP1117DT33G
OUT
IN
ADJ/GND
C4
10u
C12
10u
+5V
U2
NCP1117DT50G
C14
10u
C20
10u
U3
NCP1117DT50G
C22
10u
C24
10u
Stepper Motor Drivers
X_MOT_M0
X_MOT_M1
X_MOT_M2
X_MOT_RESET
X_MOT_SLEEP
X_MOT_STEP
X_MOT_DIR
J2
X_L
J9
X_R
DRV8825
C7
100u
X_MOT_B2
X_MOT_B1
X_MOT_A1
X_MOT_A2
X_MOT_FAULT
X_MOT_A2
X_MOT_A1
X_MOT_B1
X_MOT_B2
J16
X_Motor
X-Axis Stepper Motor
Y_MOT_M0
Y_MOT_M1
Y_MOT_M2
Y_MOT_RESET
Y_MOT_SLEEP
Y_MOT_STEP
Y_MOT_DIR
J3
Y_L
J10
Y_R
C8
100u
Y_MOT_B2
Y_MOT_B1
Y_MOT_A1
Y_MOT_A2
Y_MOT_FAULT
Y_MOT_A2
Y_MOT_A1
Y_MOT_B1
Y_MOT_B2
J17
Y_Motor
Y-Axis Stepper Motor
Z_MOT_M0
Z_MOT_M1
Z_MOT_M2
Z_MOT_RESET
Z_MOT_SLEEP
Z_MOT_STEP
Z_MOT_DIR
J4
Z_L
J11
Z_R
C9
100u
Z_MOT_B2
Z_MOT_B1
Z_MOT_A1
Z_MOT_A2
Z_MOT_FAULT
Z_MOT_A2
Z_MOT_A1
Z_MOT_B1
Z_MOT_B2
J18
Z_Motor
Z-Axis Stepper Motor
R_MOT_M0
R_MOT_M1
R_MOT_M2
R_MOT_RESET
R_MOT_SLEEP
R_MOT_STEP
R_MOT_DIR
J5
R_L
J12
R_R
C10
100u
R_MOT_B2
R_MOT_B1
R_MOT_A1
R_MOT_A2
R_MOT_FAULT
R_MOT_A2
R_MOT_A1
R_MOT_B1
R_MOT_B2
J19
R_Motor
Rotation Stepper Motor
GND
Microcontroller
S1
Reset_Button
C18
100n
C16
22p
C19
22p
Y1
16MHz
NRST
BOOT0
PH0
PH1
PD2
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PC8
PC9
PC10
PC11
PC12
PC13
PC14
PC15
X_LIM
Y_LIM
Z_LIM
R_MOT_RESET
R_MOT_M2
Z_MOT_M1
Y_MOT_FAULT
Y_MOT_DIR
Y_MOT_STEP
X_MOT_DIR
X_MOT_STEP
X_MOT_SLEEP
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PB8
PB9
PB10
PB11
PB12
PB13
PB14
PB15
LED0
LED1
V_SERVO
V_SENSOR
Y_MOT_M2
Y_MOT_M1
Y_MOT_M0
SWDIO
SWCLK
X_MOT_FAULT
R_MOT_M1
R_MOT_M0
Z_MOT_FAULT
Z_MOT_M2
Z_MOT_M0
USART_TX
USART_RX
FAN
LIGHT
Z_MOT_DIR
Z_MOT_STEP
Z_MOT_SLEEP
Z_MOT_RESET
U4
STM32L072RZT6
LEDs
SWDIO
USART_RX
USART_TX
R4
360
D1
LED
R5
360
D2
LED
R6
360
D3
LED
R7
360
D4
LED
Limit Switches
X_LIM
R1
4.7k
C1
100n
J6
X_Limit
Y_LIM
R2
4.7k
C2
100n
J7
Y_Limit
Z_LIM
R3
4.7k
C3
100n
J8
Z_Limit
Decoupling Capacitors
C6
100n
C11
100n
C13
100n
C15
100n
C17
100n
Miscellaneous Connectors
C5
100u
J15
Servo_Motor
LED0
LED1
J20
Status
PC0
PC13
PC14
PC15
J21
GPIO
V_SENSOR
J13
Pressure
SWCLK
SWDIO
J14
SWD
USB to Serial Converter
J22
USB_B_Micro
VBUS
D+
D-
ID
Shield
GND
R232
V3
VCC
TXD
RXD
CTS
DSR
RI
DCD
DTR
RTS
UD+
UD-
XI
XO
Y2
12MHz
C21
22p
C23
22p
U5
CH340G
C25
100n
USART_RX
USART_TX
Author: Christopher Conroy
Student No: 18072918
Component: Robotic Control
Description: Circuit to provide control and provide an interface to the robotic subsystem
University of Pretoria
Sheet: /Robotic_Component/
File: Robotic_Component.sch
Title: Robotic Subsystem Controller
Size: A4
Date: 2021-09-19
Rev: v01
KiCad E.D.A. kicad (5.1.10)-1
Id: 2/3

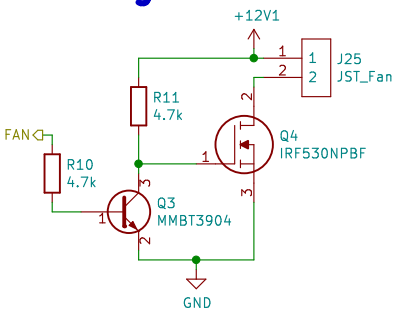
## Power Supply Connection




## LED Strip Driver




## Cooling Fan Driver




Author: Christopher Conroy
Student No: 18072918
Component: Supplementary
Description: Circuit to drive fan and LED strip
**University of Pretoria**

Sheet: /Supplementary_Component/
File: Supplementary_Component.sch

***Title: Robotic Subsystem Controller***

| Size: A4 | Date: 2021-09-19 | **Rev: v01** |
|---|---|---|
| KiCad E.D.A. kicad (5.1.10)-1 | | Id: 3/3 |

## Record 4. Hardware acceptance test procedure

The functionality of the hardware can be verified using the following procedure:

1. Connect the power supply to main's electricity using the three point plug.
2. Verify that the green light power light on the PCB turns on to indicate that the hardware is receiving power.
3. Connect the micro USB port on the embedded controller to the USB A port on a PC.
4. Send a pressure sensor request packet to the hardware as detailed in the serial communication design section.
5. Verify the receive and transmit lights both flash to indicate the hardware is responsive.

## Record 5. User guide

The hardware can be set up for use by performing the following steps:

1. Ensure that all the stepper motor cables, servo motor cable, pressure sensor cable, lighting cable and cooling fan cable are connected to the marked connectors on the board.
2. Connect the power supply to main's electricity using the three point plug.
3. Verify that the green light power light on the PCB turns on to indicate that the hardware is receiving power.
4. Connect the micro USB port on the embedded controller to the USB A port on a PC.
5. Start the system control software.
6. Select the robot's serial port and connect the embedded controller using the *Connect* button.
7. If a connection is successfully established, the system is ready to use.

# SOFTWARE part of the project

## Record 6. Software process flow diagrams

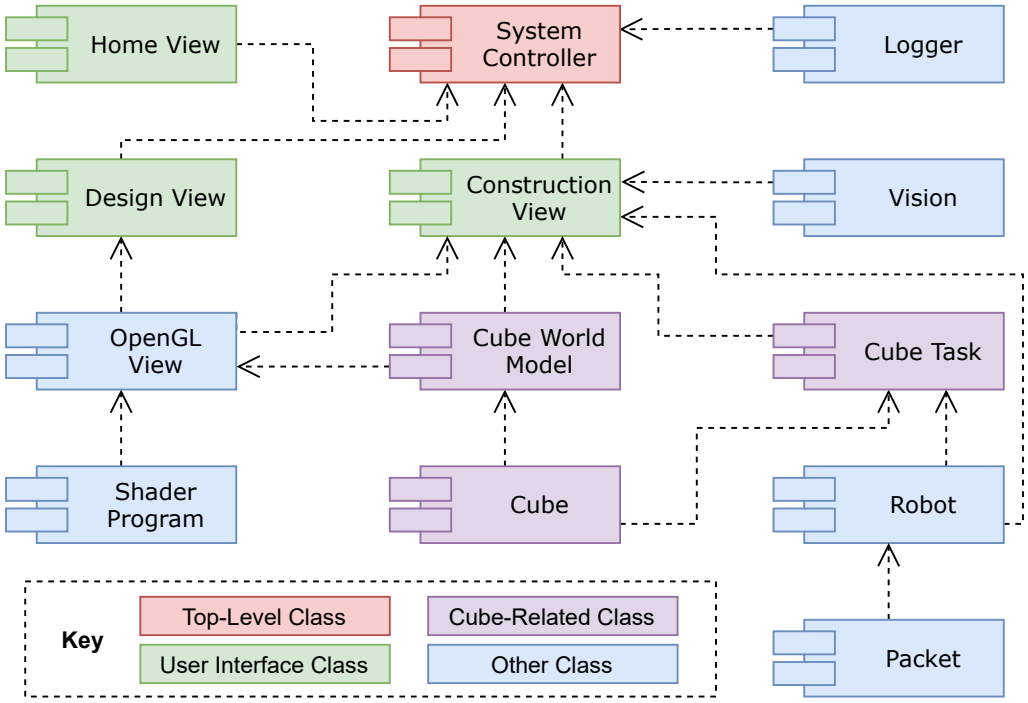



**Figure 21. Component diagram showing the structure of the integrated PC-based software solution.**

## Record 7. Explanation of software modules

The *System Controller* class shown in Figure 21 sits at the top level of this implementation and directly contains the user interface classes[13] as well as the *Logger* class through which all system event information, warnings and errors are recorded. The user interface classes partition the software based on functionality requirements. Specifically, the *Home View* serves as the entrance point to the software and its purpose is to ensure the *Robotic System* and camera hardware are present and connected. The *Design View* integrates the components related to the *Shape Definition* function. This includes the *OpenGL View* which uses OpenGL to create a 3D render of a model of cubes. This class is supported by the *Shader Program* class which manages the shaders used for the rendering task. The model itself is sourced from the *Cube World Model* class. The purpose of this class is to simply capture the arrangement of cubes, which are each represented by an instance of the *Cube* class, and relate this arrangement to the world coordinate system. The *Design View* interface facilitates the creation and manipulation of a *Cube World Model* instance which serves as output from the *Shape Definition* component.

The *Cube World Model* instance created in the *Design View* acts as input to the construction process around which the *Construction View* class is based. In order for construction to take place, the system needs to interact with the robot. The *Robot* class was created for this purpose. The class provides an abstract interface for the *Construction View* instance to send position and actuation control commands to the *Robotic System*. The units of communication which the *Robot* class uses to interact with embedded robotic controller are abstracted in the form of the *Packet* class. This class is based on the packet designed in Section 3.3.4. Similarly, the *Vision* class exists to offer an abstract interface to the *Vision System* developed in Section 3.5. The *Construction View* instance receives information from this component that is used to guide the control of the *Robot* instance during the construction process accordingly.

[13] Each of the user interface classes correspond to a distinct screen in the user interface.

## Record 8. Complete source code

Complete code has been submitted separately on the AMS.

## Record 9. Software acceptance test procedure

In order to make use of the PC-based software's full functionality, a connection to the *Robotic System* is required. Therefore, the user guide for the hardware setup should first be followed. After completing this process, the software should have an established connection with the *Robotic System*. Following this, navigate to the *Construction* view in the software. On this screen, click the *Calibrate* button. The robot calibration sequence should begin and carry out to completion. This verifies that the software is functioning correctly.

## Record 10. Software user guide

This section assumes that the software has been verified to be functioning correctly with the software acceptance test procedure. The system control software is started by running the PC-based software executable file. The initial screen displayed is the home screen as shown in Figure 22. A feed from the camera of the robot's workspace should be displayed to indicate the camera's presence. The system message log for informational messages, warnings and errors is displayed at the bottom of the screen. The messages can be filtered as needed with the filter checkboxes. Connect to the robot by selecting the CH340 serial port from the serial port list and clicking the *Connect* button.

The *Shape Definition* component can be accessed by clicking on the *Shape Design* button. This screen is shown in Figure 23. The set of buttons in the top left can be used to insert and remove cubes from the build as well as to load and save model designs. Individual cubes can be selected in the left cubes list. Use the arrow keys to translate the cube horizontally. Use the shift button with the arrow keys to move the cube vertically and to rotate the cube. Use the mouse buttons to move the camera around the structure.

The *Construction* view can be accessed by clicking on the *Construction* button. This view is shown in Figure 24. The first group of buttons can be used to calibrate the robot, apply the computer vision system to the current scene, load a model for construction and start a construction sequence. The vision view and model views shown in Figures 25 and 28 respectively. The next set of buttons are used to set the active state of the stepper motors and control the vacuum actuator state. Finally the last set of controls allows the control of the position of the robot.

The vision view shown in Figure 25 is used to set the annotations and stage displayed for the image display. Examples of these stages are shown in Figures 26 and 27. Finally, the model view in Figure 28 can be used to select the 3D render displayed. Either the shape to be built is displayed, or the live location of the cubes during construction is displayed.

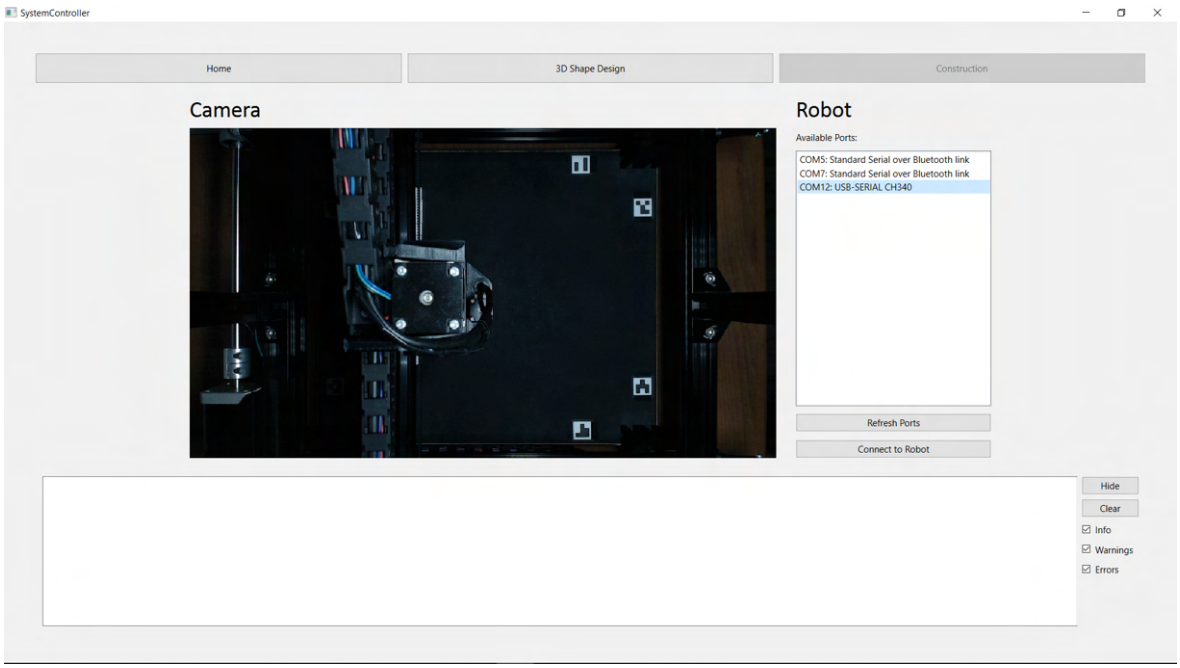



**Figure 22. System control software home screen.**

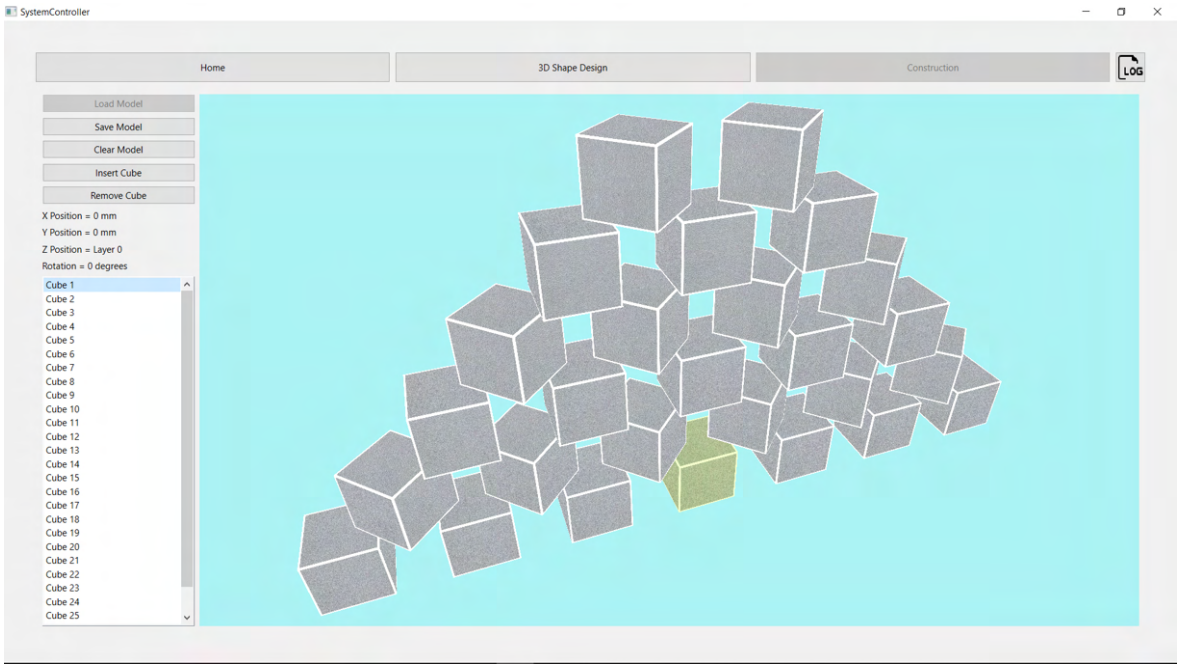



**Figure 23. System control software 3D shape design screen.**

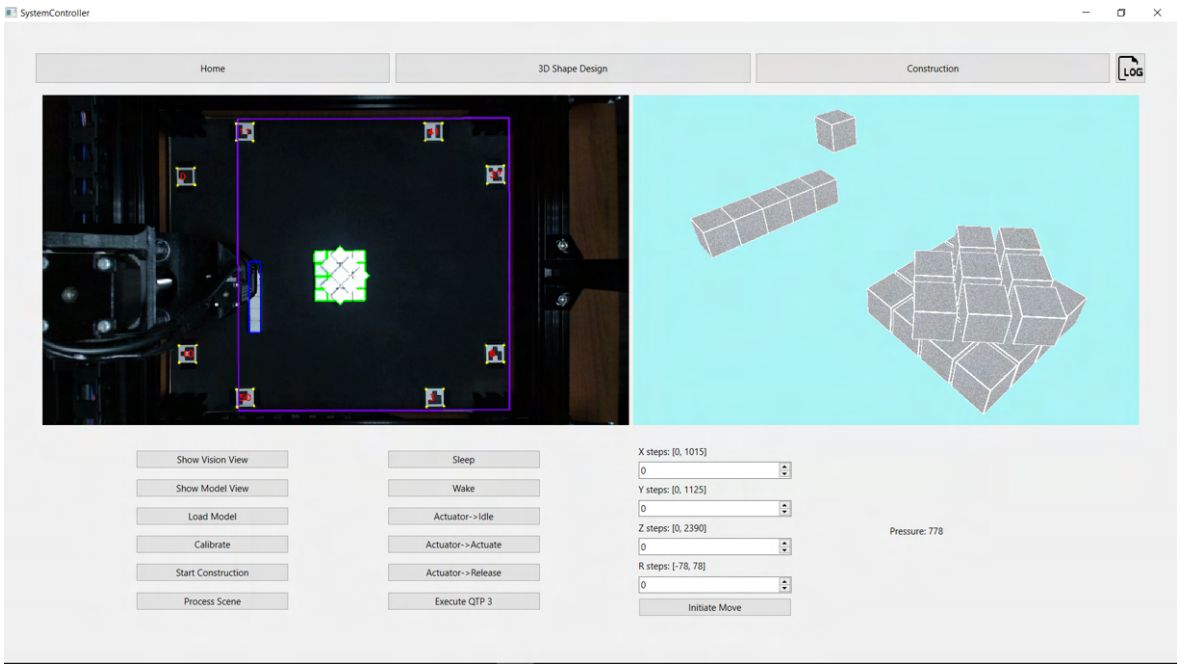



**Figure 24. System control software construction screen.**

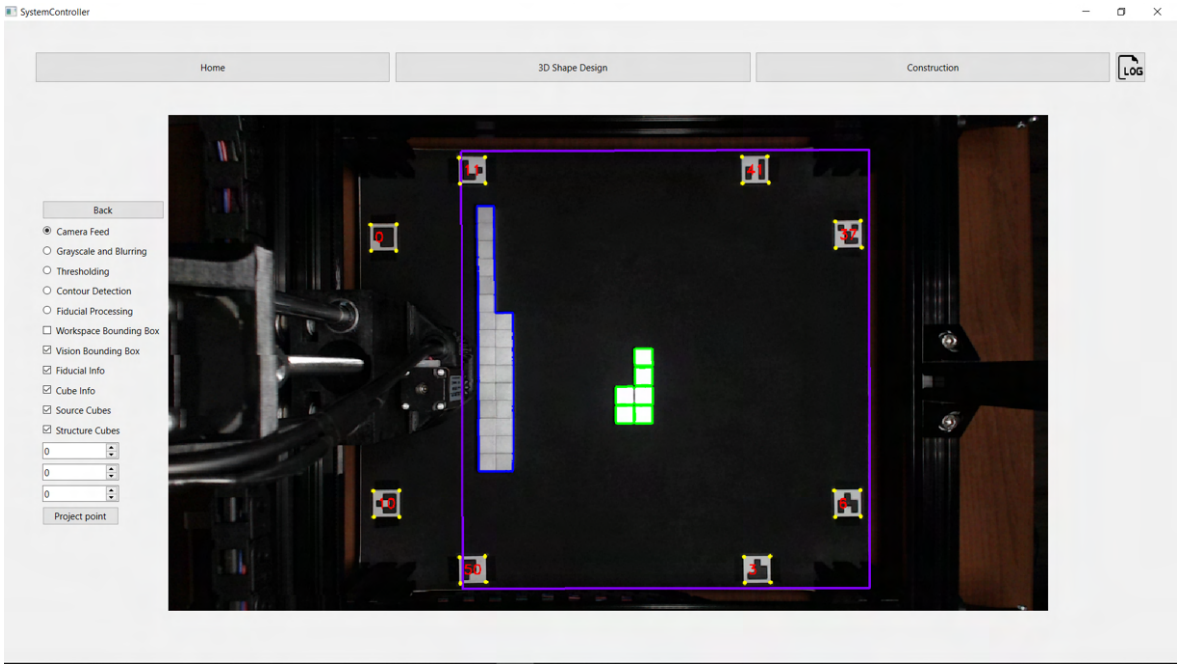



**Figure 25. System control software vision view screen.**

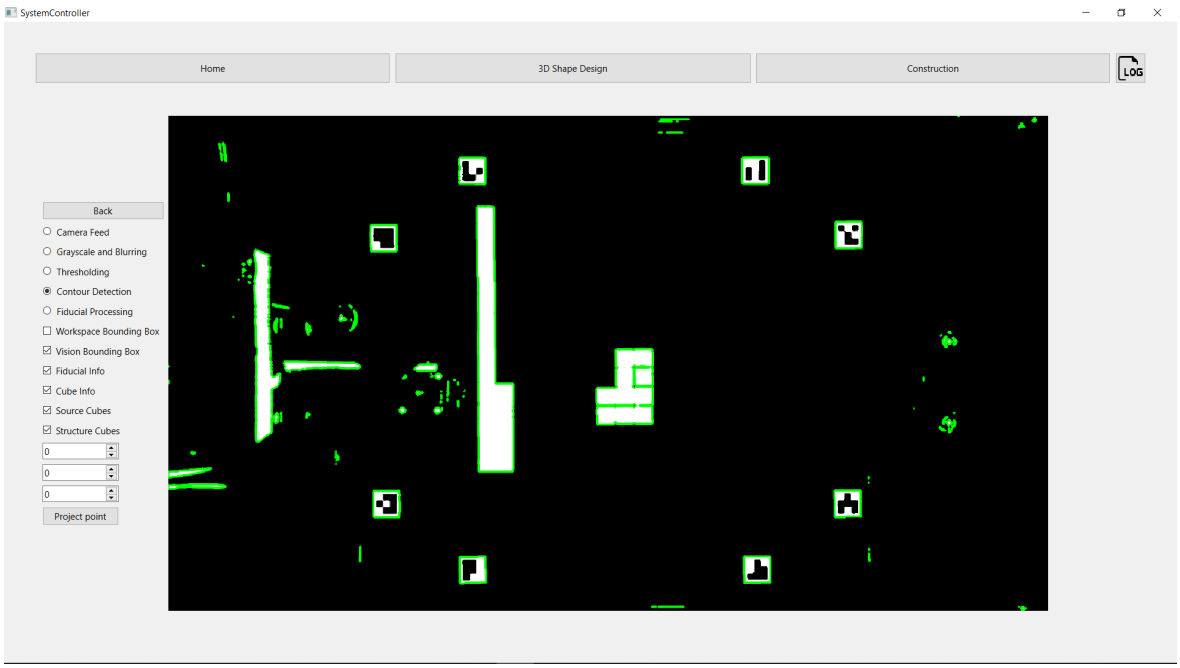



**Figure 26. System control software vision view screen with contour step displayed.**


**Figure 27. System control software vision view screen with fiducial step displayed.**

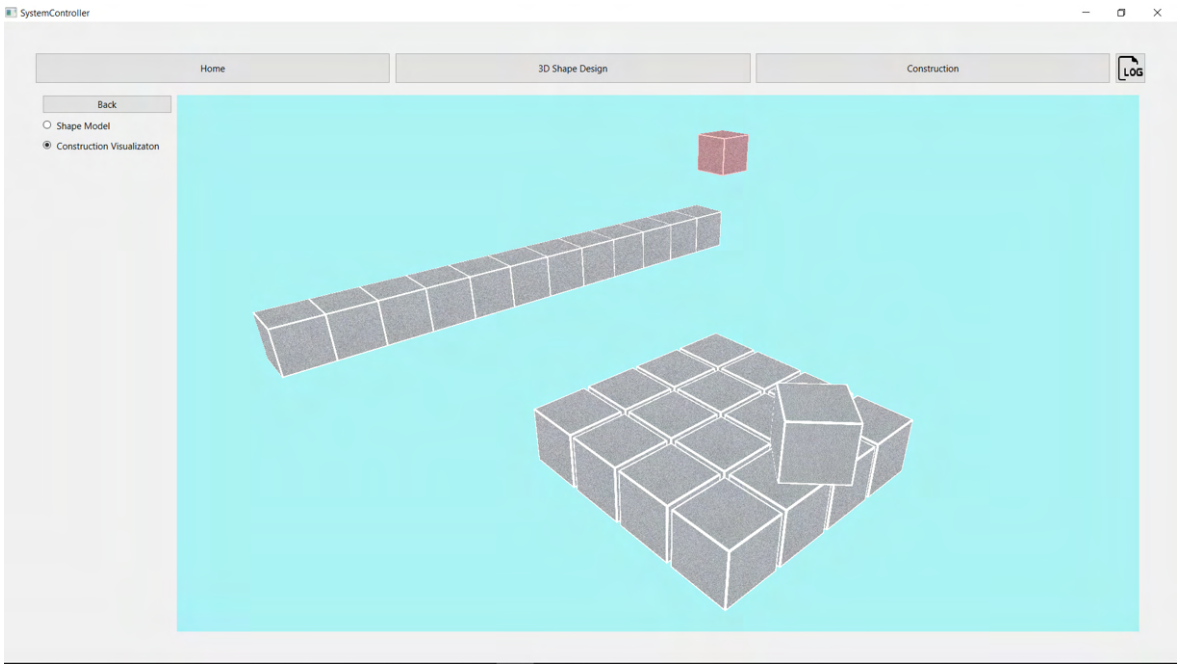



**Figure 28. System control software model screen.**

# EXPERIMENTAL DATA

## Record 11. Experimental data

### Qualification Test 1 Results

A test set of 10 3D shape models was defined using the *Shape Definition* component for use in qualification test 1. Table 12 summarises the important properties of each of these shapes

| Shape ID | Number of Cubes | Rotated Cubes | Partially Supported Cubes[14] | Slanted Cube Stack | Shape Height |
|---|---|---|---|---|---|
| 1 | 30 | No | No | No | 4 |
| 2 | 30 | Yes | No | No | 4 |
| 3 | 30 | No | Yes | Yes | 6 |
| 4 | 30 | Yes | No | No | 6 |
| 5 | 30 | Yes | Yes | No | 5 |
| 6 | 30 | No | No | No | 6 |
| 7 | 30 | Yes | Yes | Yes | 6 |
| 8 | 30 | Yes | Yes | No | 6 |
| 9 | 30 | No | No | Yes | 6 |
| 10 | 30 | No | Yes | No | 4 |

**Table 12. Properties of each 3D shape model in the test set for qualification test 1.**

[14] A cube is considered to be partially supported if less than half of the bottom face of the cube is touching the top faces of the cubes in the layer below it.

| Shape ID | Back Left | Back Right | Centre | Front Left | Front Right |
|---|---|---|---|---|---|
| 1 | Success | Success | Success | Success | Success |
| 2 | Success | Success | Success | Success | Success |
| 3 | Success | Success | Success | Success | Success |
| 4 | Success | Success | Success | Success | Success |
| 5 | Success | Success | Success | Success | Success |
| 6 | Success | Success | Success | Success | Success |
| 7 | Success | Success | Success | Success | Success |
| 8 | Success | Success | Success | Success | Success |
| 9 | Success | Success | Success | Success | Success |
| 10 | Success | Success | Success | Success | Success |

**Table 13. Results of each shape construction sequence at 5 different locations in the robot's workspace.**

**Qualification Test 3 Results**

| Iteration | Cube Gripped (Y/N)? | Time Dropped (s) | Move Sequence Completed (Y/N)? | Cube Released (Y/N)? |
|---|---|---|---|---|
| 1 | Yes | Not Dropped | Yes | Yes |
| 2 | Yes | Not Dropped | Yes | Yes |
| 3 | Yes | Not Dropped | Yes | Yes |
| 4 | Yes | Not Dropped | Yes | Yes |
| 5 | Yes | Not Dropped | Yes | Yes |
| 6 | Yes | Not Dropped | Yes | Yes |
| 7 | Yes | Not Dropped | Yes | Yes |
| 8 | Yes | Not Dropped | Yes | Yes |
| 9 | Yes | Not Dropped | Yes | Yes |
| 10 | Yes | Not Dropped | Yes | Yes |

**Table 14. Results of the movement sequence iterations for qualification test 3.**

**Qualification Test 4 Results**

| Sample Set | X Position (steps) | Y Position (steps) | Z Position (steps) | Reference Length (pixels) | X Points Range (pixels) | Y Points Range (pixels) |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 79,168 | 9,751 | 8,03 |
| 2 | 0 | 1125 | 0 | 61,265 | 13,883 | 11,059 |
| 3 | 507 | 562 | 0 | 70,112 | 15,882 | 12,176 |
| 4 | 1015 | 0 | 0 | 75,279 | 15,853 | 12,257 |
| 5 | 1015 | 1125 | 0 | 72,198 | 11,063 | 8,388 |

**Table 15. Pixel measurements using the point cluster images for the x- and y-axis repeatability test.**

| Sample Set | X Points Range (mm) | Y Points Range (mm) |
|---|---|---|
| 1 | 0,2463 | 0,2028 |
| 2 | 0,4532 | 0,3610 |
| 3 | 0,4530 | 0,3473 |
| 4 | 0,4211 | 0,3256 |
| 5 | 0,3064 | 0,2323 |

**Table 16. X- and y-axis repeatability test measurements after conversion from pixel units in Table 15.**

| Sample Set | X Position (steps) | Y Position (steps) | Z Position (steps) | Reference Length (pixels) | Z Points Range (pixels) | Z Points Range (mm) |
|---|---|---|---|---|---|---|
| 1 | 852 | 70 | 500 | 60,001 | 23 | 0,7666 |
| 2 | 852 | 70 | 2300 | 61,26 | 19,96 | 0,6516 |
| 3 | 502 | 400 | 500 | 60,962 | 18,506 | 0,6071 |
| 4 | 502 | 400 | 2300 | 60,705 | 15,173 | 0,4998 |
| 5 | 194 | 700 | 500 | 60,397 | 17,557 | 0,5813 |
| 6 | 194 | 700 | 2300 | 60,27 | 19 | 0,6304 |

**Table 17. Pixel measurements using the point cluster images for the z-axis repeatability test.**

**Qualification Test 5 Results**

| **Sample** | $y_0$ **Deviation** $\delta_0$ **(mm)** | $y_1$ **Deviation** $\delta_1$ **(mm)** | **Deviation Angle** $\phi$ (°) |
|---|---|---|---|
| 1 | 0,64 | -0,66 | 0,3146 |
| 2 | -0,36 | 0,53 | -0,2154 |
| 3 | -0,11 | 0,2 | -0,0750 |
| 4 | 0,32 | -0,22 | 0,1307 |
| 5 | 0,12 | -0,08 | 0,0484 |
| 6 | 1,51 | -1,85 | 0,8132 |
| 7 | -0,16 | 0,32 | -0,1161 |
| 8 | 0,7 | -0,47 | 0,2831 |
| 9 | -1,24 | 1,01 | -0,5445 |
| 10 | -1,43 | 1,22 | -0,6414 |
| 11 | 1,12 | -0,31 | 0,3461 |
| 12 | 0,66 | -0,84 | 0,3630 |
| 13 | 0,98 | -1,18 | 0,5228 |
| 14 | -0,36 | 0,53 | -0,2154 |
| 15 | 0,23 | -0,44 | 0,1621 |
| 16 | 0,6 | -0,95 | 0,3751 |
| 17 | -0,47 | 0,66 | -0,2735 |
| 18 | 0,57 | -0,4 | 0,2347 |

**Table 18. Deviation measurements of drawn cube orientation lines and the calculated angle of deviation of the line cube.**

**Qualification Test 6 Results**

| Set Sample | True X Position (steps) | True Y Position (steps) | True Angle (°) | Detected X Position (steps) | Detected Y Position (steps) | Detected Angle (°) |
|---|---|---|---|---|---|---|
| 1 | 2 | 20 | 0 | 3 | 23 | -0,26 |
| 2 | 347 | 20 | 0 | 347 | 26 | -1,23 |
| 3 | 681 | 21 | 0 | 678 | 25 | 0,46 |
| 4 | 986 | 22 | 0 | 982 | 26 | 0,5 |
| 5 | 4 | 369 | 0 | 6 | 371 | 2,39 |
| 6 | 344 | 369 | 0 | 346 | 372 | 0,45 |
| 7 | 680 | 369 | 0 | 676 | 373 | 1,2 |
| 8 | 984 | 370 | 0 | 980 | 372 | 0 |
| 9 | 18 | 720 | 0 | 20 | 719 | 1,4 |
| 10 | 347 | 719 | 0 | 348 | 718 | 0,49 |
| 11 | 663 | 720 | 0 | 658 | 719 | 2,35 |
| 12 | 988 | 719 | 0 | 983 | 720 | 0,51 |
| 13 | 14 | 1079 | 0 | 14 | 1077 | 0,72 |
| 14 | 356 | 1080 | 0 | 355 | 1077 | 1,2 |
| 15 | 679 | 1080 | 0 | 677 | 1078 | 0,49 |
| 16 | 989 | 1081 | 0 | 987 | 1080 | 1,22 |

**Table 19. True poses of the first test set of cubes for qualification test 6 along with the corresponding estimated poses detected by the *Vision System*.**

| Set Sample | True X Position (steps) | True Y Position (steps) | True Angle (°) | Detected X Position (steps) | Detected Y Position (steps) | Detected Angle (°) |
|---|---|---|---|---|---|---|
| 1 | 83 | 77 | 45 | 85 | 80 | -44,98 |
| 2 | 396 | 61 | 45 | 395 | 65 | 44,31 |
| 3 | 677 | 77 | 45 | 673 | 82 | 44,66 |
| 4 | 974 | 54 | 45 | 971 | 59 | 41,5 |
| 5 | 214 | 245 | 45 | 214 | 246 | -44,68 |
| 6 | 505 | 252 | 45 | 504 | 524 | 44,65 |
| 7 | 777 | 286 | 45 | 772 | 289 | -44,62 |
| 8 | 19 | 438 | 45 | 22 | 439 | -44,66 |
| 9 | 307 | 450 | 45 | 308 | 450 | -43,96 |
| 10 | 530 | 533 | 45 | 527 | 532 | 44,07 |
| 11 | 916 | 447 | 45 | 909 | 449 | -43 |
| 12 | 26 | 733 | 45 | 27 | 732 | -44,65 |
| 13 | 271 | 790 | 45 | 271 | 788 | 43,31 |
| 14 | 654 | 711 | 45 | 650 | 709 | 45 |
| 15 | 928 | 733 | 45 | 923 | 731 | -43,66 |
| 16 | 25 | 1038 | 45 | 25 | 1035 | 44,66 |
| 17 | 358 | 1012 | 45 | 358 | 1009 | -44,67 |
| 18 | 625 | 1041 | 45 | 621 | 1037 | -44,66 |
| 19 | 937 | 1028 | 45 | 933 | 1025 | -42,27 |

**Table 20. True poses of the second test set of cubes for qualification test 6 along with the corresponding estimated poses detected by the *Vision System*.**

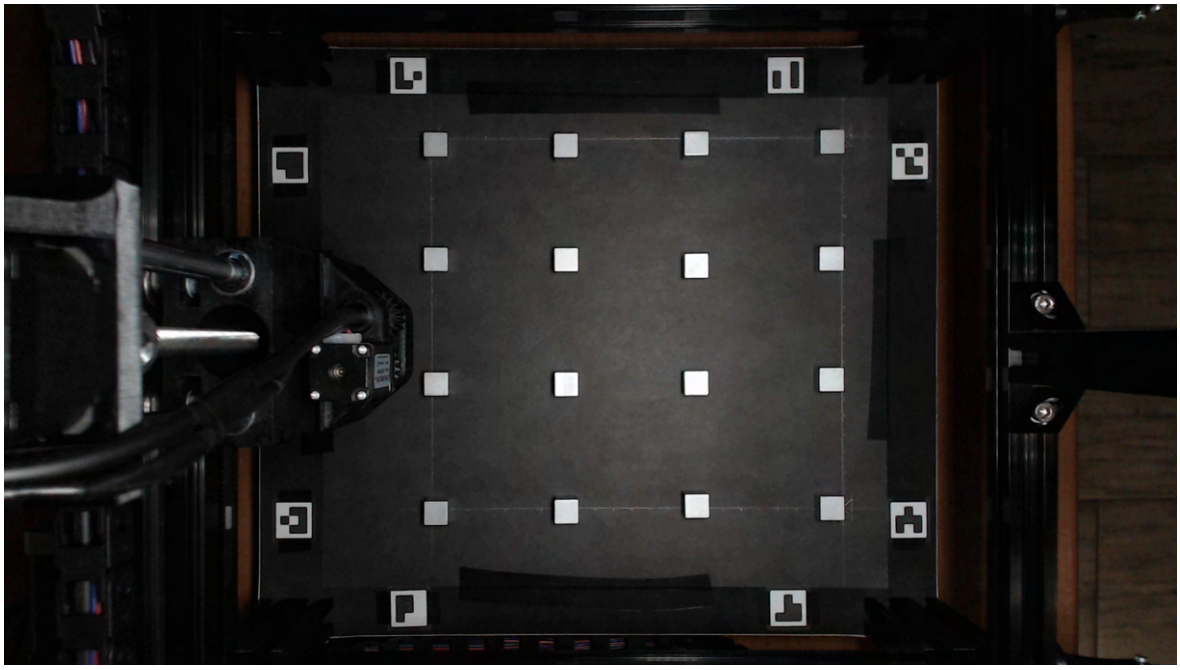


**Figure 29. First test set of known cube positions and orientations (0°) in the robot's workspace for qualification test 6.**

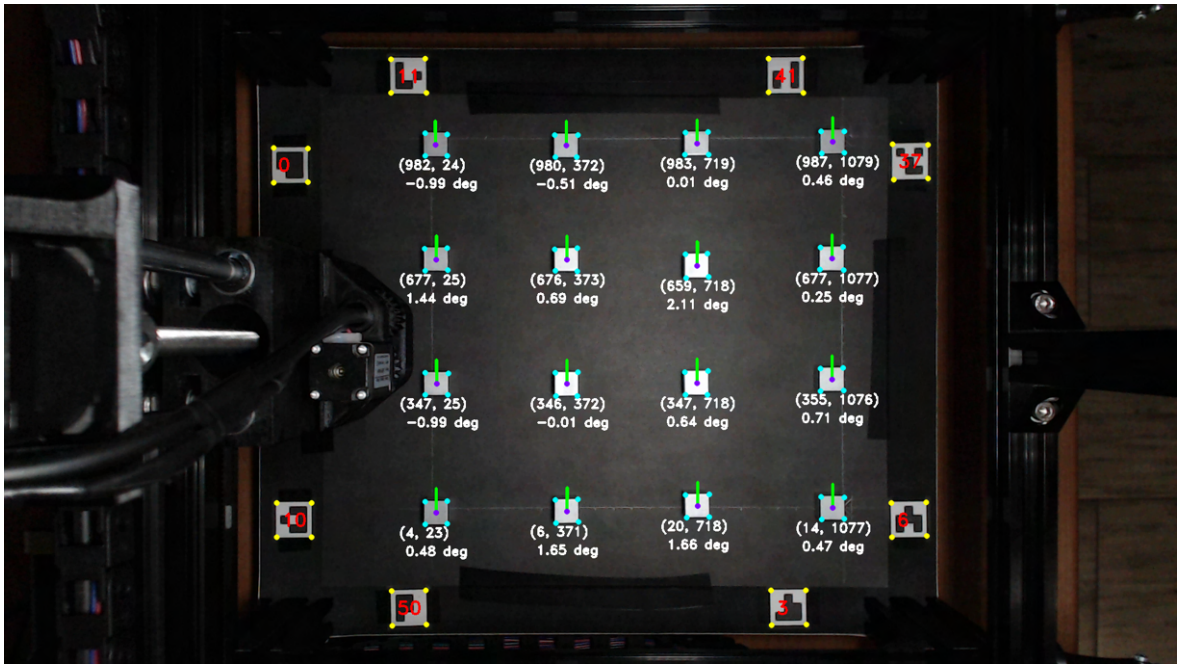



**Figure 30. First test set of cubes after pose estimation by the *Vision System* for qualification test 6.**

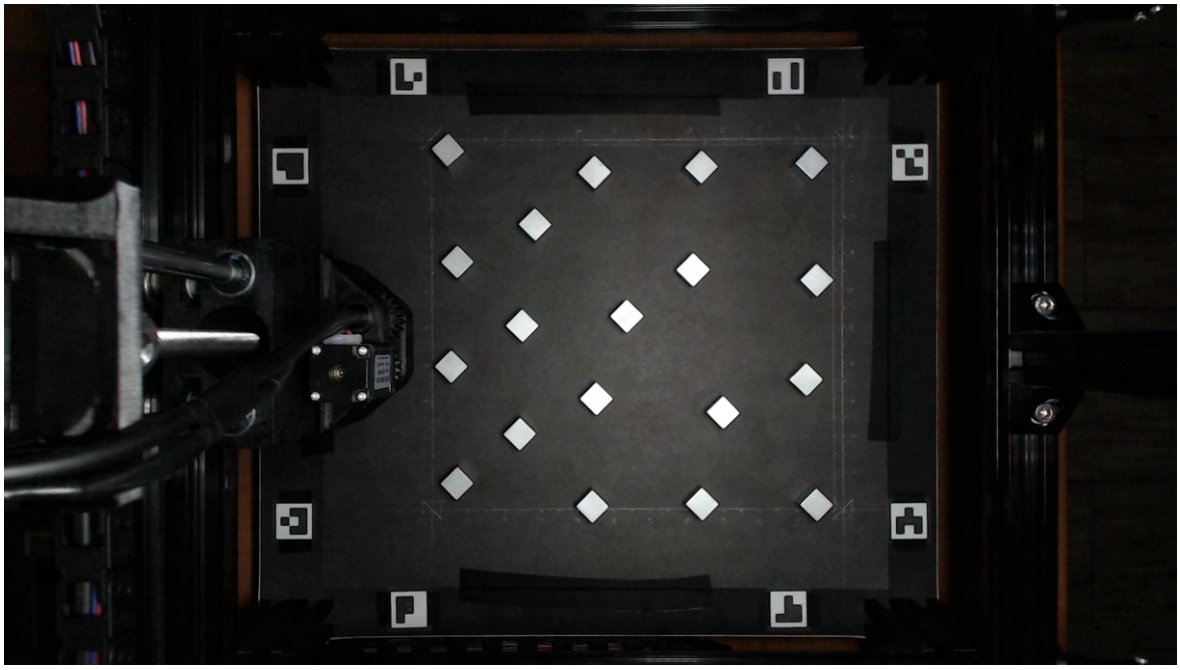


**Figure 31. Second test set of known cube positions and orientations (45°) in the robot's workspace for qualification test 6.**


**Figure 32. Second test set of cubes after pose estimation by the *Vision System* for qualification test 6.**

**Qualification Test 7 Results**

| Iteration | Dropped Cubes Detected? | | | | | Failure Detected (Y/N)? |
|---|---|---|---|---|---|---|
| | 1-5 | 6-10 | 11-15 | 16-20 | 21-25 | |
| 1 | Yes | Yes | Yes | Yes | Yes | Yes |
| 2 | Yes | Yes | Yes | Yes | Yes | Yes |
| 3 | Yes | Yes | Yes | Yes | Yes | Yes |
| 4 | Yes | Yes | Yes | Yes | Yes | Yes |
| 5 | Yes | Yes | Yes | Yes | Yes | Yes |

**Table 21. Results of qualification test 7 to assess capability of system to detect dropped cubes and construction failures.**