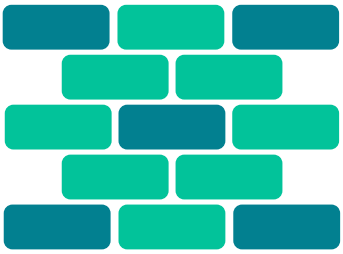


AutoArtisan

Mission Report

Prepared by:

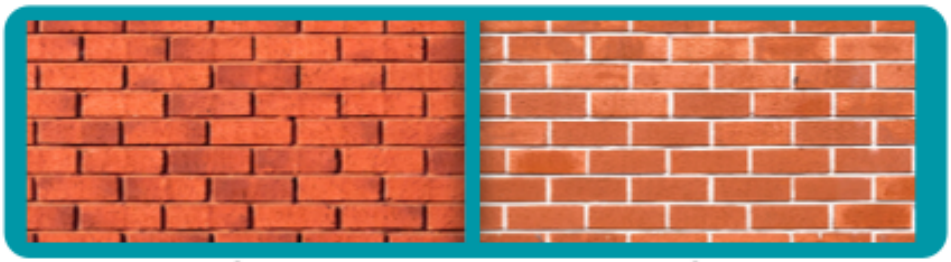
Max van 't Hart
Neel Lodha
Tristan Little
Melihcan Balasar
Lili Sitányi
Rob Krüger

Date: September 4, 2026

Design Brief

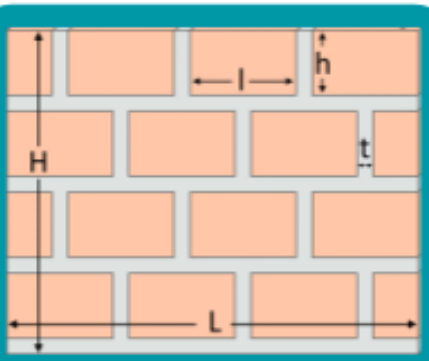
SMB Geveltechniek

AutoArtisan



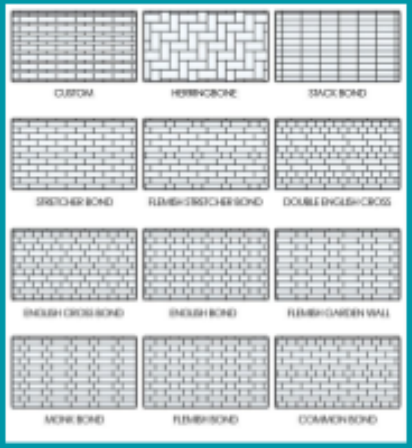
Before After

Autonomous jointing

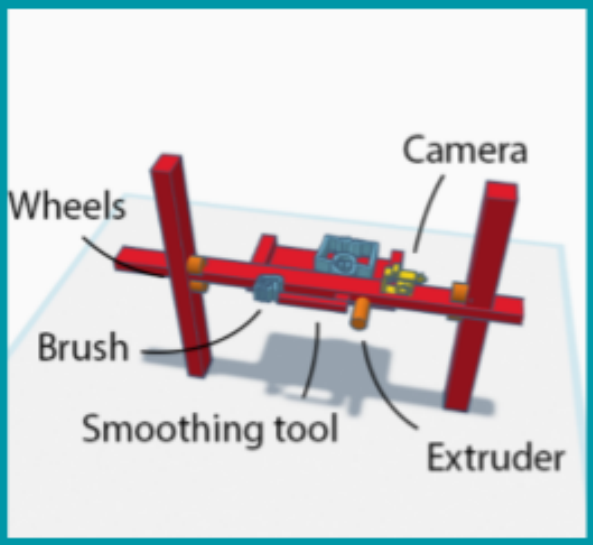


Varying wall and brick dimensions

Any wall pattern



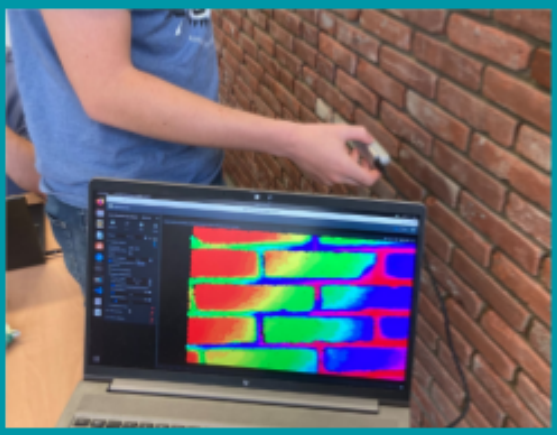
Robot concept design



Metal gantry
With multi tool
head

Continuous
motorised
extrusion

Binocular
depth sensor
for joint
identification



1. People and Process

1.1 Team

Our team consists of one electrical engineer; Rob Krüger, two software engineers; Lili Sitányi and Neel Lodha, and three mechanical engineers; Tristan Little, Melihcan Balasar and Max van 't Hart. Our electrical engineer has extensive experience with power supplies, inductive power transfer, working on robots including the Mirte Master and has a good theoretical foundation on signal processing and sensors. Our software engineers have experience with machine learning in pattern recognition, programming finite state machines and have web development expertise. Last, our team of mechanical engineers consists of Certified Solidworks Associates and a Certified Solidworks Professional. They all have lathe-, mill- and welding capabilities. All the mechanical engineers also have experience with Finite element analysis, building 3-axis movement systems, 3D-printing, and are all educated on the fundamental design process from prototype to final product.

1.2 Mission and Values

Our mission is to help lessen the workload on jointers improving their productivity while decreasing the risk of health complications caused by the strenuous work. We will be doing this by making a jointing robot that does the most of the monotonous jointing work almost autonomously. At the same time, it is also our desire to preserve the traditional masonry craft. Our robot should not replace jointers or their jobs, instead act as a tool to support and collaborate with them.

1.3 Roadmap

To ensure we accomplish the task ahead of us, we have partitioned our time into smaller segments with more tangible deliverables and objectives. The project spans a total of 20 working weeks. This mission report marks the end of week 6. Bellow in Figure 1.1 a simple graphic of this timeline is shown.

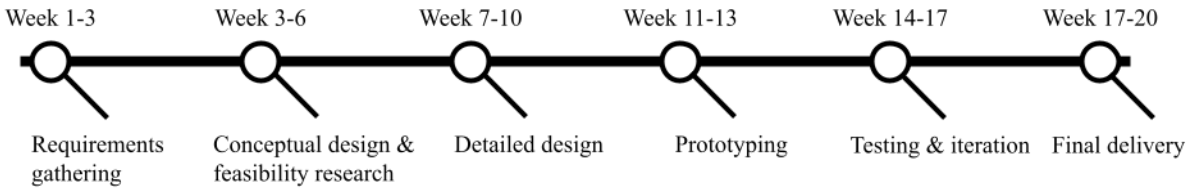


Figure 1.1: Timeline

For the requirement gathering, the Objective was to understand the client needs for jointing operations, materials, precision, and budget. The deliverable was the requirements document in MoSCoW format for all disciplines. During the conceptual design & feasibility research phase, we discovered existing solutions, generated multiple design

concepts, and assess technical and financial feasibility. Deliverables included all elements of this mission report. When we enter the detailed design stage, we must create detailed CAD models, electrical schematics, and software architecture of the prototype robot. The corresponding CAD drawings, circuit diagrams, software flowcharts, and simulations are to be delivered by the end if this stage. As part of prototyping we will build a working first prototype with mechanical, electrical, and software integration. Moving into testing & iteration our goal becomes to test prototype, gather client feedback, and refine the design. Here, the deliverables are Test reports and an improved prototype. Lastly for the final delivery phase, our objective becomes: finalizing the design, preparing handover documents and demo-day performance. The ultimate deliverables are a final prototype and the handover report.

2. Scope of robot

2.1 Contextual analysis

As previously discussed, the heavy loads that construction workers regularly carry can negatively impact their long-term health and well-being. A wide variety of engineering solutions have been developed to improve working conditions on construction sites, such as exoskeletons and assistive robots. With regards to robots in construction, research has predominantly focused on robots designed to handle heavy lifting or perform repetitive tasks that would otherwise be physically demanding to workers (bricklaying for example). Our robot's function aligns more closely with the latter: an automated jointing robot would alleviate the burden on human labor. Currently only a few robots used in construction offer functionality similar to ours. Examples include Okibo's wall finishing robot or the mortar spraying MSP-robot among many more which can be found in the appendix. However none of these are specifically designed to perform jointing specifically. This makes our robot the first of its kind to address this critical need in the field of construction.

Critical aspects we have to account for when developing this robot include brick variance, proper finishing (mortar → smooth → brush → spray) and adaptability to various wall sizes and patterns. We strive to make our robot as proficient as possible on the following aspects: Efficiency, Precision, Autonomy, Durability, Cost, Portability, Usability, Analytics and Quality. Descriptions of these criteria are shown in Figure 2.1 With these criteria in mind we will develop a robot that performs the jointing of brick walls with speed and results matching or exceeding that of human jointers.

Efficiency: This measures how quickly the robot can perform tasks. A faster robot will save time and reduce operational costs. This is an especially important performance metric for us, as our client has a threshold for speed that he would like our robot to meet/exceed.

Precision: Precision refers to the robot's ability to perform tasks with accuracy and minimal errors, especially in repetitive or fine-detail tasks. This is especially important for us as jointing requires fine motor control, and increased precision would allow us to make the robot more efficient in terms of material usage.

Autonomy: Autonomy reflects the robot's ability to perform tasks without human intervention. High autonomy is important for our robot as our client mentioned that he wants to be able to focus on other tasks whilst the robot does the repetitive, time-consuming task of jointing.

Durability: Durability measures how well the robot withstands wear and tear over time, especially in demanding environments. A highly durable robot reduces maintenance costs and this is especially important for us, as the robot has to withstand dust, weather conditions and being transported to and from building sites.

Cost: The price of parts and materials is of significant importance since budgets are not endless. Costs from operation and repairs after the development of a robot must also be considered. The affordability of a robot's component can factor into how the robot is designed. This metric is relevant to us since we have an allotted budget of 3000 euros.

Portability: Portability assesses the ease with which the robot can be transported or deployed to various locations. This criterion is important for our robot, as it will need to be moved between different construction sites.

Adaptability: Adaptability measures the robot's ability to adjust to new tasks, environments, or conditions. This is important for our robot, as it will have to adapt to variations in building materials and wall surfaces without experiencing errors.

Usability: Usability pertains to how intuitive the user interface is and how clear the error messages are. This is important for our robot, as the client requested that the operator be able to use the robot without additional training (safety or otherwise).

Analytics: Analytics capabilities refer to the robot's ability to collect, process, and analyze data from its environment or operations. Advanced analytics will optimize our robot's performance and ensure that it is consistent with the client's expectations for productivity. Real-time data processing will help our robot respond to inconsistencies detected in the wall and monitor its material usage.

Quality: This criterion measures how competently the robot does it's job and whether the quality matches the work done by its human counterpart. This is important for our robot as (according to our client) scarcity of skilled human jointers is becoming more of a problem.

Figure 2.1: Evaluation criteria

2.2 Use case and procedure

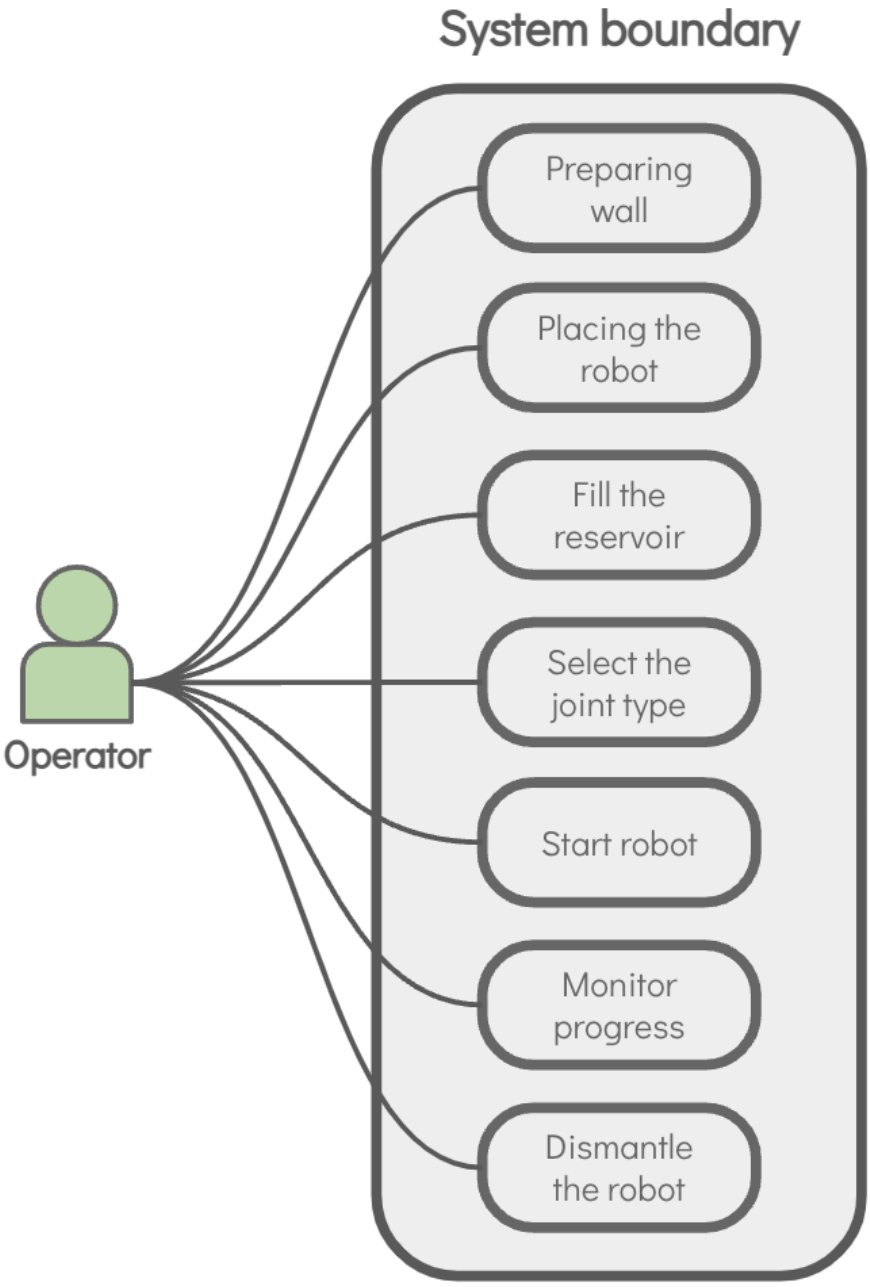


Figure 2.2: The use case diagram for the Jointing Robot

As can be seen from Figure 2.2, although the jointing robot has various stakeholders, the only stakeholder that will be directly interacting with the robot will be the operator. All the functions that the operator interacts with refer to different parts of the jointing process: manual set-up of the robot; choosing the settings; starting the robot and monitoring the robot. The 'Monitoring the robot' interaction refers to the operator responding to potential error messages that the robot alerts them of on the graphical user interface. Possible contents of error messages include: empty reservoir; mortar build-up in the nozzle; significant variation in joint size; etc.) The operator may also need to manage uncaught errors (i.e., the robot detects that it has finished filling a joint when it hasn't). The exact order of operation and the steps needed to accomplish the jointing task can be seen in

Figure 2.3.

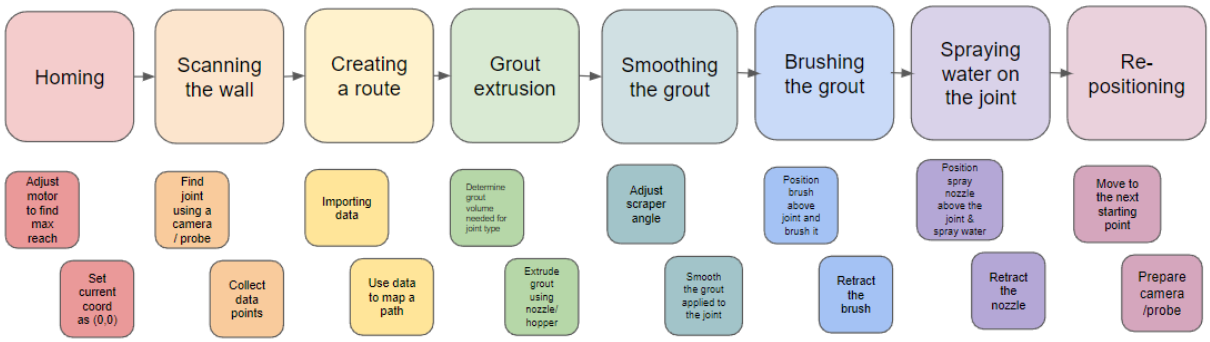


Figure 2.3: Steps and substeps

2.3 Prioritized list of Design Challenges

The main challenge of this project can be divided into a myriad of smaller questions. These questions are not equally important, meaning we can prioritise and focus on the most critical challenges first. The following prioritised list of design challenges is ordered from most to least important and colour-coded by discipline according to the following legend: Electrical, Mechanical, Software, Design.

1. How will the robot adapt itself to different wall heights? 2
2. How to find the optimal route for the best efficiency? 3
3. How to power the robot? 1
4. How to differentiate between brick and joint? 3
5. How do we move mortar continuously? 4
6. How to detect the wall? 1
7. How do we move the tools far and accurately? 2
8. How to control motors? 1
9. How will the different parts of the robot connect (electronically)? 1
10. How to communicate effectively with the electronics? 3
11. How will the robot mount itself into the wall? 4
12. How do the different components of the robot attach and detach? 4
13. How do we give the motors sufficient torque? 2
14. How do we pivot for different angles? 2
15. How do we prevent lockups? 2

16. How to adapt the amount of mortar extruded to the density? 3
17. How to detect when the task is finished? 3
18. How to collect important data and store it? 3
19. Will the robot only use local or cloud computing? 1
20. How to keep track of position? 1
21. How to find optimal start and end point? 3
22. How to detect mortar reservoir levels? 1
23. How will the robot know the amount of mortar extruded? 1
24. How to keep track of the mortar used? 3
25. How to stow tools when not in use? 2
26. How to monitor the robot's own speed? 3
27. How will the operator be able to monitor the robot's statistics? 1
28. How do we move to compensate for variance in pattern? 2
29. How to re-align to adapt to imperfections on the wall surface? 3
30. How do we prevent blockage? 2
31. How do we supply water at any height? 2
32. How do we make it rigid enough? 2
33. How do we brush off the excess mortar enough? 2
34. How to adapt speed based on scraper angle and mortar density? 3
35. How to handle interrupts and unexpected events? 3
36. How to handle significant variation in joint size? 3
37. How do we execute different joint types? 2
38. How do we limit wear? 2
39. How to detect mortar consistency errors? 3
40. How to detect mortar build up that may cause error? 3
41. How to communicate that no joint/wall is detected? 3
42. How do we communicate failure to the operator? 4
43. How many safety buttons and sensors will the robot have and where will they be placed? 1

44. How to give safe control of the robot to the operator (wireless controller?) 1
45. How do we make the design usable? 4
46. How to make the robot waterproof? 4
47. How to make the robot dust-proof? 4
48. How can we improve the design of the robot w.r.t safety? 4
49. How do we repair or replace parts? 4
50. How do we maintain accuracy at larger scales? 2
51. How will the operator transport the robot? 4
52. How to make the robot more compact when components are too long? 4

Five important design challenges are elaborated on as working principles in chapter 4.

3. Concept outlines of Robot

3.1 Frame: Description of Robot concept

The design in mind for the robot is an robot based on an extendable rigid frame with wheels to move along the rigid frame. The tool-head of the robot will feature a nozzle for extruding mortar, a tool to make flush joints and a motorized brush for cleaning up the bricks and joints after the flush joints are performed. The robot will use a short range depth camera to be able to distinguish between bricks and joints and to find faults in the wall.

3.2 Critical Challenges

3.2.1 Continuous Mortar Movement

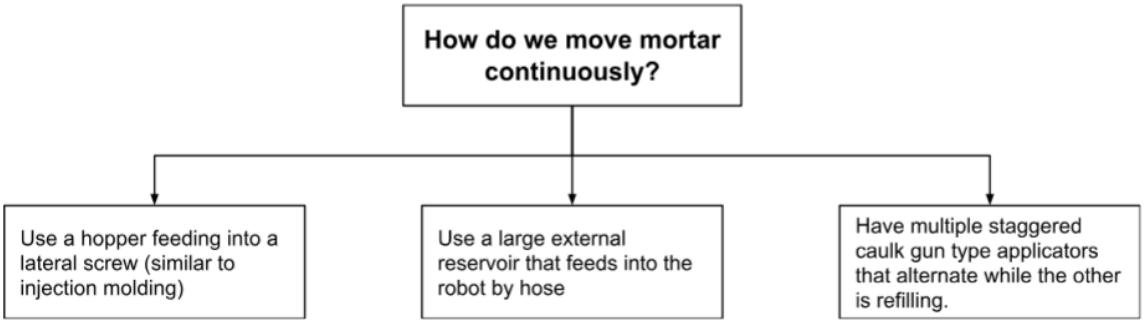


Figure 3.1: Design Option Tree for Moving Mortar Continuously

Challenge: Efficiently and accurately dispense mortar into the joint gap autonomously. **Possible Solutions:** Mortar must be continuously injected into the gap in order to prevent stoppages and an uneven joint finish. Viscous flow pumps could be used to route mortar from a large external tank to the tool tip of the robot. One issue this poses is if the mortar dries in the pipe and can't easily be removed. Instead, a hopper might be used to store smaller amounts of mortar at a time and feed the mortar into an extrusion mechanism. Such an extrusion mechanism could take many forms, but a promising one is an Archimedes style screw which can turn perpendicular rotary motion into continuous linear motion. Such a mechanism would be more simple to clean in case of blockage. Lastly, there is the option of multiple alternating batched dispensing systems in a style similar to that of a caulk gun. This offers the simplest clean up as the mortar storage and flow system is entirely replaced, but the downside of this is the associated cost with prepackaged mortar canisters instead of mixing your own.

3.2.2 Robot movement

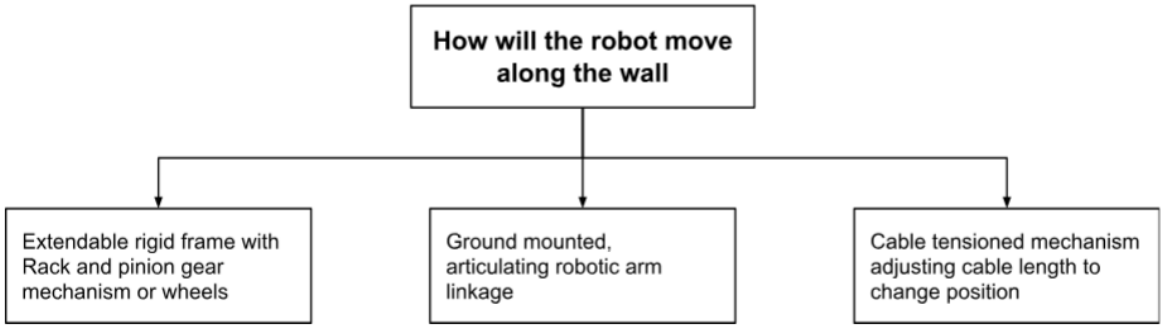


Figure 3.2: Design Option Tree for Moving Robot along the wall

Challenge: Precisely moving across tall and wide brick walls while being resistant to strong weather and uneven ground.
Possible Solutions: One solution involves a metal frame being bolted into the wall. The robot could then move up and down, left and right in a manner similar to that of a 3D-printer. This would be very rigid and precise, but would not be so convenient to relocate to other sections of wall. A common staple of other similar construction robots is an articulated robotic arm. These offer high precision and are very adaptable to wall variance. On the other hand, they can be prohibitively expensive, hard to control, they have a hard limit on the height they can reach and would struggle with uneven terrain as they must stay on the ground. Lastly we can consider a cable tensioned mechanism where the motors can be mounted into single spots on the wall and change the cable length to adjust the position of the tool tips. This would be much simpler to install and could be highly accurate in the 2D plane. The drawbacks of such a system are rigidity and a smaller effective working area.

3.2.3 Power source

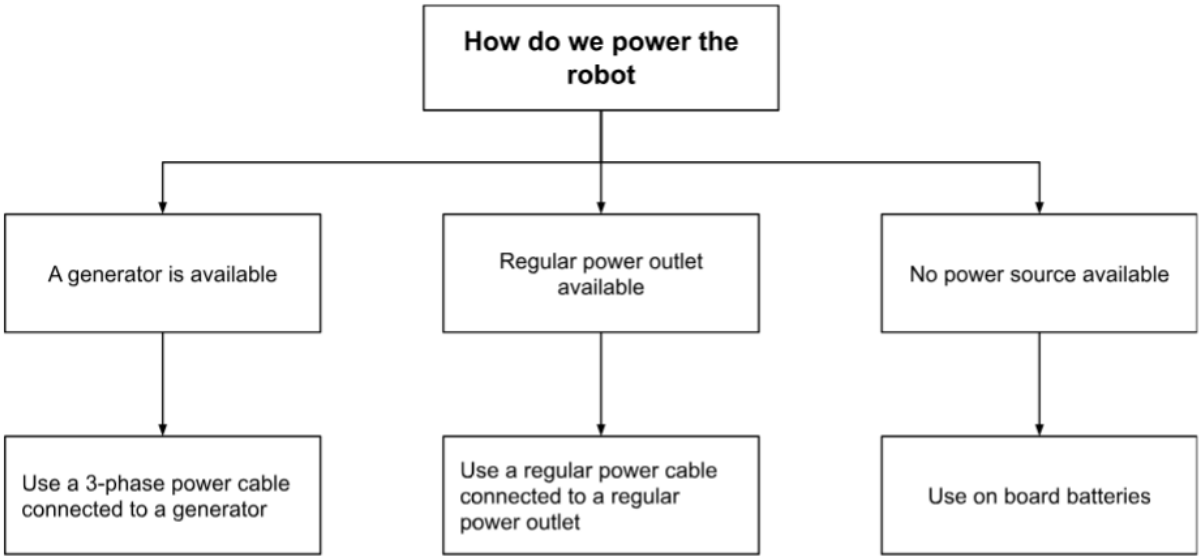


Figure 3.3: Design Option Tree for Powering the Robot

3.2.4 Differentiating Bricks and Joints

Challenge: Efficiently and accurately distinguish between a brick and a joint in the image captured by the depth sensor.
Possible Solutions: We have identified 3 possible solutions to this challenge: Finite State Machine (FSM), Image processing Algorithms and Machine learning Model. All three of them have their pros and cons. FSM would likely be the most efficient in terms of performance. However, its accuracy in identifying bricks and joints is debatable, as FSMs are typically rule-based and may struggle with more complex or ambiguous cases. Image processing algorithms will involve comparing the depth values of pixels to differentiate between bricks and joints. It is reliable and relatively fast but may fail in edge cases, such as when dealing with broken or irregular bricks. A machine learning model, with sufficient training data, is expected to deliver the highest accuracy among the proposed solutions. However, this approach may incur significant computational costs, and its resource requirements could be excessive for execution on hardware such as a Raspberry Pi.

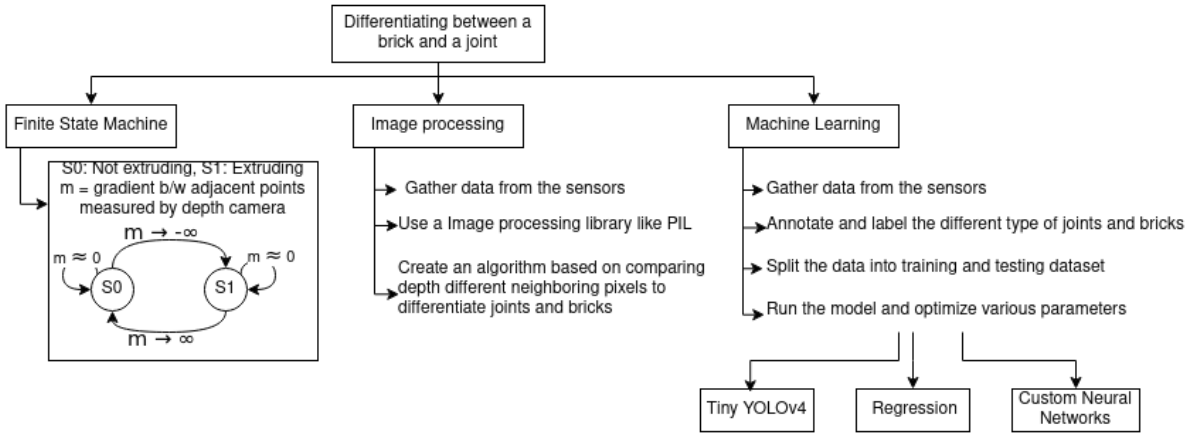


Figure 3.4: Design Option Tree for Software Engineering challenges

3.2.5 Optimal Routing

Challenge: Find an optimal route for the robot to joint.
Possible Solutions: All proposed solutions rely on distinguishing between bricks and joints, followed by creating a virtual map or layout using a coordinate system or a graph structure with nodes and edges. One solution involves constructing a graph where nodes represent the intersections of joints, and edges represent the joints themselves. A routing algorithm, such as Breadth-First Search (BFS) or Depth-First Search (DFS), can then be applied to find an optimal path. A simpler approach would involve predefined movement patterns, such as covering all the horizontal joints (rows) first, followed by the vertical joints (columns). Alternatively, the robot could complete two rows at a time, then move to the vertical joints connecting those rows. All these solutions are fairly similar in terms of functionality, and none offer a distinct advantage over the others.

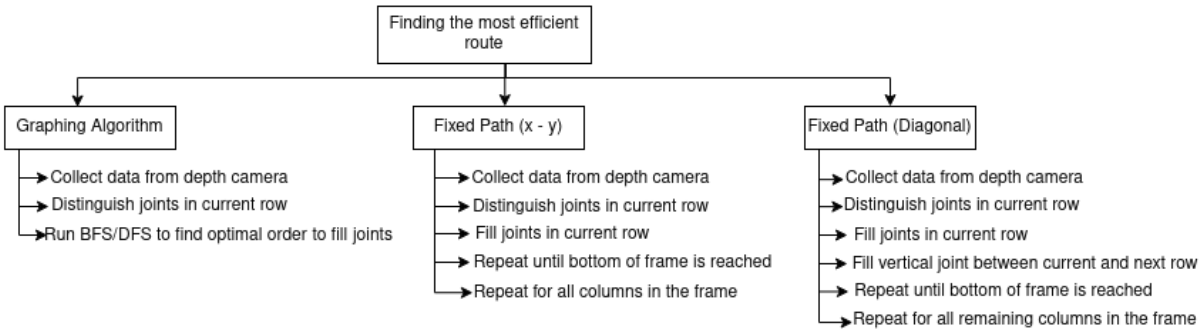


Figure 3.5: Design Option Tree for Software Engineering challenges

3.3 Requirements (Moscow)

3.3.1 Mechanical Requirements

Functional Requirements

Must Haves

- Robot must be able to dispense mortar
- Robot must execute a flush joint
- Robot must brush the mortar after it has been smoothed out
- Robot must smooth out the joints in horizontal & vertical direction
- Robot must be able to home itself
- Robot must be able to move along the wall
- Robot must be mounted into solidified mortar

Should Haves

- Robot should be adaptable to variance in bricks
- Tools should be replaceable in case of wear
- Tools should last at least 100m^2 before significant wear

Could Haves

- Robot could be able to moisten the mortar after smoothing
- Robot could moisten the wall within 5 minutes of smoothing the mortar.
- Robot could execute other types of joints

Won't Haves

- Robot won't chisel joints
- Robot won't execute the jointing of corners

3.3.2 Software Requirements

Functional Requirements

Must Haves

- System must issue warning messages when mortar in the reservoir is running low
- System must be able to control the filling of flush joints
- System must be able to control the filling of joints for various brick patterns
- System must issue operator alerts when an error occurs or maintenance is required

- System must keep track of how many square metres of wall it has already jointed
- System must be able to control the usage of multiple tools (mortar nozzle, water nozzle, voegspijker, joint-roller, etc.)
- User must be able to select a joint type for the system to coordinate the filling of joints
- System must be able to keep track of its current position/establish a coordinate system
- System must show different states and errors on a digital/interactive interface.

Should Haves

- System should be able to autonomously complete routes without operator interference
- System should calculate the most efficient path

Could Haves

- System could be able to coordinate the filling of various types of joints (i.e., raked joint, weathered joint, etc.)
- System could be able to coordinate the filling of bricks arranged decoratively around a window frame
- System could monitor the quantity of mortar left in the reservoir

Won't Haves

- System won't be able to coordinate the jointing of bricks across concave or convex corners
- System won't coordinate chiselling of old joints
- System won't coordinate jointing for horizontal surfaces
- System won't be able to coordinate the jointing of a brick casement that surrounds a window

Non-functional Requirements

- System will be programmed using software libraries from ROS2 using Python3
- Error/maintenance messages should be as precise as possible to improve the GUI's usability

3.3.3 Electrical Requirements

Functional Requirements

Must Haves

- Robot must have a motor strong enough to extrude mortar out of its nozzle
- Robot must have an emergency stop button
- Robot must have a waterproof power supply that provides enough power to every component
- Robot must have a reliable AC to DC converter
- Robot must have a sensor to detect the joint pattern on the wall (One or more of: camera, depth sensor, LiDAR, colour sensor, ultrasound)
- Robot must have motors to control movement (stepper motors/servo motors)

Should Haves

- Robot should have a sensor to detect the mortar flow
- Robot should have a controller, preferably wireless
- Robot should have a pinch point sensor to prevent accidents with trapped limbs and objects
- Robot should have a digital/interactive interface to show different states like errors

Could Haves

- Robot could have a sensor to detect water flow
- Robot could have an IMU to improve positioning
- Robot could have a sensor to keep track of reservoir contents

Won't Haves

- Robot won't have 120 Volt & 60 Hz compatibility

3.3.4 Design Requirements

Functional Requirements

Must Haves

- Robot must be transportable by hand
- Robot must be applicable to varying wall dimensions
- The robot must fit within a 'standard' van
- The components of the robot must individually weigh a maximum of 20 kgs
- System must extrude mortar in a way that results in a consistently uniform finish

Should Haves

- Robot should be waterproof
- Robot should be dust-proof
- Robot should be easy to disassemble for transport purposes
- Robot should have built in cushioning against heavy handling

Could Haves

- Robot could have coloured lights, sounds or screen displays for communication with users
- Robot could display its next steps along the planned path

Won't Haves

- Robot won't lay bricks
- Robot won't remove old joints
- Robot won't operate in locations inaccessible to the operator

Non-functional Requirements

- System shall operate at a speed of at least $30m^2$ per day
- Robot shall have a fail-safe for if the power is cut
- Robot should be developed within a budget of €3000
- System must operate consistently irrespective of a work-site's layout (i.e., whether it is jointing a small patch of wall or a wall spanning multiple floors)

3.4 Prototype: Learnings and Insight

3.4.1 Prototype and Experiments

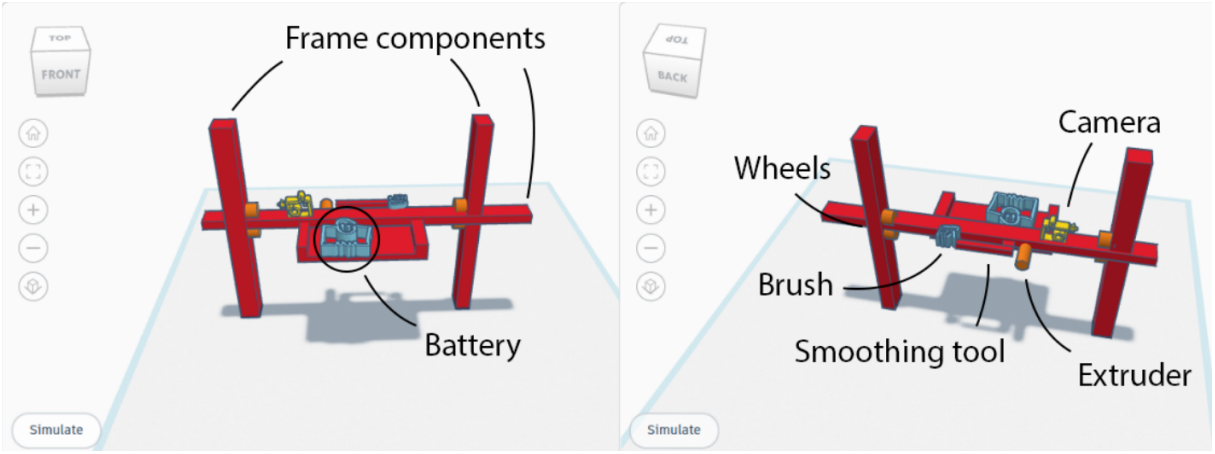


Figure 3.6: Labelled 3D model of prototype

As shown in Figure 3.6, the current design of the jointing robot is a frame structure with the smoothing tool, the mortar nozzle and the brush fixed to the horizontal part of the frame. With the prototype shown in Figure 3.7, we were unable to test the most significant design challenges presented by our jointing robot - i.e., mortar extrusion, wall mounting, frame movement - so the design problems we experimented with were brush mechanics, creating a custom smoothing tool and depth camera testing.

Figure 3.7: Actual prototype

Brush Mechanics

The movements required to dust off a wall with a standard brush would be difficult for the jointing robot to execute. The use of a rotary brush would not require such complex movement and would therefore be easier to control. Thus, the aim of this experiment was to test the efficacy of a rotary brush for cleaning dust from a surface.

Custom Smoothing Tool

The team would like to make a tool that is thinner than a standard jointing trowel, and just as effective at smoothing mortar (so that the robot is less prone to errors caused by deviations in joint width). This experiment consisted of testing the efficacy of a wooden stick, a scraper and scribing needle for smoothing re-hydrated cement over a brick. The tool was judged on precision and number of movements required.

Depth Camera Viability

The jointing robot needs some way to distinguish between joint and brick. The team could opt for a physical probe or an electronic solution such as an ultrasonic sensor or a depth camera. The electronic solution would be more ideal as that is easier to protect from physical damage than a probe. Thus, the aim of this experiment is to test whether the depth camera data is precise enough to distinguish between a brick and a joint.

3.4.2 Conclusions

Brush Mechanics

The rotary brush was attached to a drill and was tested on the dust on top of a pillar drill. This brush was specifically selected for its similar stiffness to the brush used by our client. It took approximately 12 seconds to clear the dust off the top of the pillar drill entirely. This test is not entirely representative of how difficult it would be to clean up a smoothed joint. Though, this does demonstrate that the selected brush will have a sufficient bristle density to leave a finish without streaks. The greater resistance the brush would encounter from polishing a smoothed joint can be compensated for by applying more pressure behind the brush and using more torque from the motor and slower rotation speeds. A rotating brush moved laterally across a joint also ensures that the joint is brushed in opposite directions from the leading and trailing edge. This even further mimics the motion of manual brushing. Therefore, a rotary brush attached to a motor of similar torque and speed of a drill will be sufficient for our robot.

Custom Smoothing Tool

Out of all the tools tested, the tool that was most effective at spreading mortar between the bricks was the wooden stick. The wooden stick was just malleable enough to transfer pressure effectively - therefore, if the team wants to create a custom smoothing tool, the flexibility would ideally be similar to that of a popsicle stick.

Depth Camera Viability

For this experiment, the Intel RealSense D435i and the Orbbec Astra Pro Plus were tested. Both of the depth sensors have clear resolution and show a distinct difference in depth between the brick surface and the inset joints between them. Although either sensor would be suitable for data collection, the Intel RealSense can resolve at a closer distance which might be preferable for a compact robot. Therefore, a depth camera - particularly the Intel RealSense - would be a better solution for distinguishing between bricks and joints than an Orbbec Astra Pro Plus.

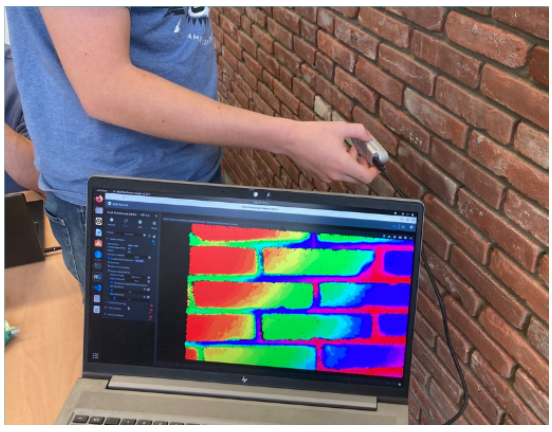


Figure 3.8: Intel RealSense depth camera test

Figure 3.9: Orbbec Astra Pro Plus depth camera test

4. Five Working Principles

4.1 5 Working Principles with 3 concepts each

4.1.1 Mechanical

How will the robot adapt itself to different wall heights?

The robot will should be able to adapt to all kinds of wall types, which means; different brick-types and wall heights. This can be accomplished in different ways depending on the design of the robot. If the robot will move along the wall using an extendable rigid frame with either a rack and pinion gears mechanism or wheels, the frame will be extendable in the vertical direction by using modular vertical frame elements. If the robot will move along the wall using a ground mounted articulating robotic arm, the platform on which the robot arm will be mounted on should be adjustable in height. This can be achieved by mounting the robot on a scissor lift, this way the robot can be lifted up, while the robot remains stationary on the platform. Lastly, if the robot is going to be using a cable tensioned mechanism which will adjust the cable length to change position of the toolhead the cable tension mechanism will already integrate the adjustment of height into the robot. Different brick-types will be a challenge for the robots that need to be mounted directly on the wall. To make it possible for these robots to be mounted on a large variety of brick-types a mounting system needs to be designed. The wall mounted robots will be mounted onto the wall with large screws directly screwed into the mortar.

4.1.2 Electrical

How to power the robot?

The robot will have a lot of electrical components that all need power, at varying voltages and currents. Therefore, the robot will need a power supply with the proper converters to ensure correct and safe power to every component. This power supply in turn needs its power coming from somewhere and there are three main methods to deliver this power: a 3-phase power generator, a regular 230V-outlet or batteries. Batteries might seem as a practical solution, as it does not need any cables dangling around but to power a rather large robot as this robot will likely be it would need hefty batteries, adding a lot of weight and generally being unsafe. Therefore, this is not a solution. The other methods do not differ all too much from each other, a 3-phase power generator would just be able to provide more power. Dutch outlets are limited at 16A, so the maximum power they can deliver is about 3700 Watt. If this appears to be sufficient, a regular outlet will be used to power the robot, aiding in the usability of the robot since you can find an outlet almost everywhere but if the robot will require more power, it will need to be powered by a 3-phase power generator.

4.1.3 Software

Differentiating between brick and joint

Finite State Machine

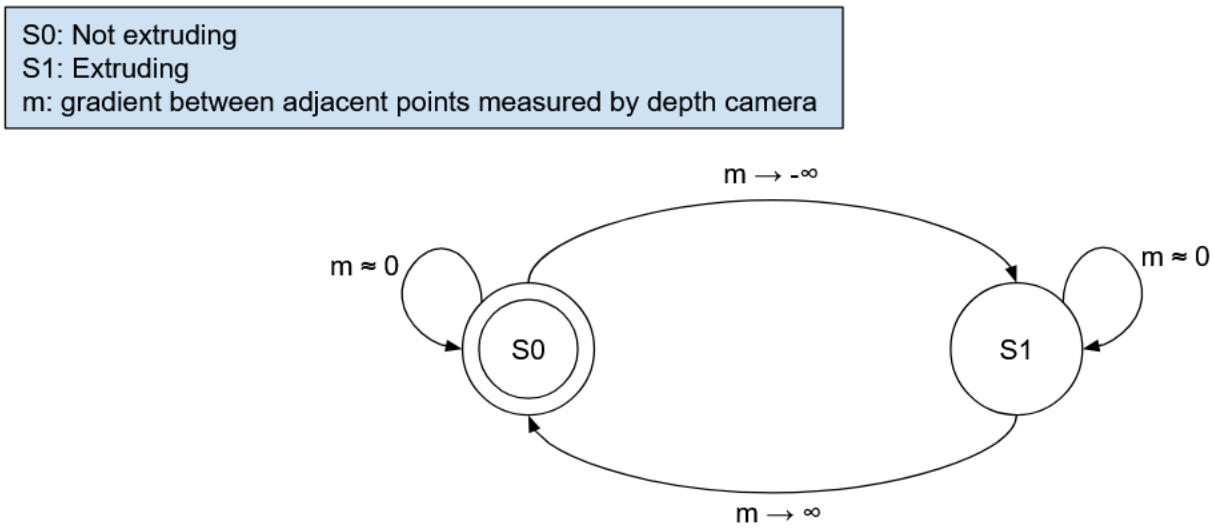


Figure 4.1: FSM for mortar extrusion

Figure 4.1 shows the finite state machine for determining when to extrude mortar. One consideration with this solution is that it would require fast communication between nodes to avoid mortar being extruded at the wrong time.

The node layout shown in Figure 4.2 was devised to make communication between the robot's subsystems as efficient as possible. As nodes should be responsible for a single modular purpose, there will be separate nodes for mortar extrusion, navigation, GUI, statistics, material monitoring and the depth camera. In the current layout, the mortar extrusion node publishes to an extruder status topic; the navigation node publishes to motor status and frame boundary topics; the statistics node provides the 'compute stats' service; material monitoring publishes to the mortar fill status topic and the depth camera has the 'extrude' action and publishes to the camera status topic.
The GUI node is subscribed to all the status topics because it needs to display error and maintenance messages to the operator. The navigation node is subscribed to the mortar fill status and extruder status topics to prevent the robot from moving along joints when there is no mortar left (i.e., no unwanted gaps will be created). The reason the statistics node is subscribed to the mortar fill status topic is that it may be useful to measure the percentage of time spent refilling the mortar - such insight could help the operator determine how much of the time is being spent productively. The mortar extrusion node is subscribed to the motor status topic because if there is an error with the motor, the extruder should stop dispensing mortar. These communications between nodes were made to cover as many edge cases as possible to make the FSM more dependable in complex scenarios.

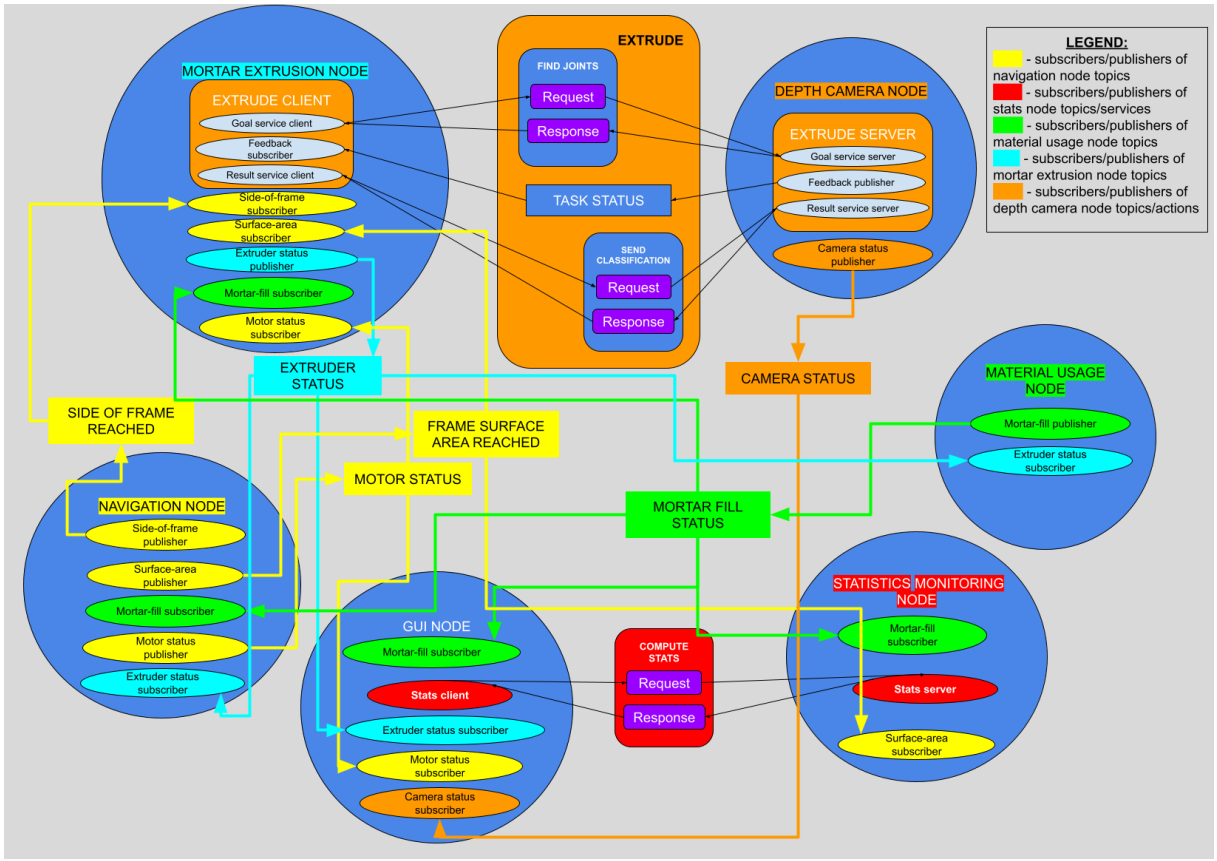


Figure 4.2: Planned layout for node communication

Machine Learning

This solution involves the following steps:

1. Read depth data from .bag file generated by the Intel realsense depth camera
2. Annotate/label the joints and the bricks
3. Split the data into training and testing data sets
4. Run the model and optimize various parameters

The model used varies depending on whether our solution uses (Linear) Regression, Tiny YOLO or custom neural networks, etc. If the machine learning approach is used, then the solution we choose will depend on how fast the model can be run.

Image Processing

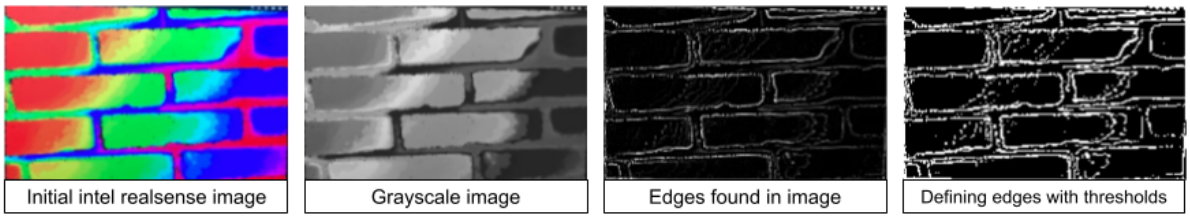


Figure 4.3: Stages of image processing (example)

A variety of image processing techniques may be used to distinguish between bricks and joints in the image captured by a depth camera. The examples of the stages of image processing shown in Figure 4.3 were produced with the Python code below.

```
from PIL import Image, ImageFilter

# img: Stage 1 in Fig. 4.3
with Image.open("intel_realsense_depth_data.png") as img:
    img.load()

# grayscale: Stage 2 in Fig. 4.3
grayscale = img.convert("L")

# edges: Stage 3 in Fig. 4.3
edges = grayscale.filter(ImageFilter.FIND_EDGES)

threshold = 30

# img_brick_thr: Stage 4 in Fig. 4.3
img_brick_thr = edges.point(
    lambda x: 255 if x > threshold else 0
)
```

Finding the optimal route

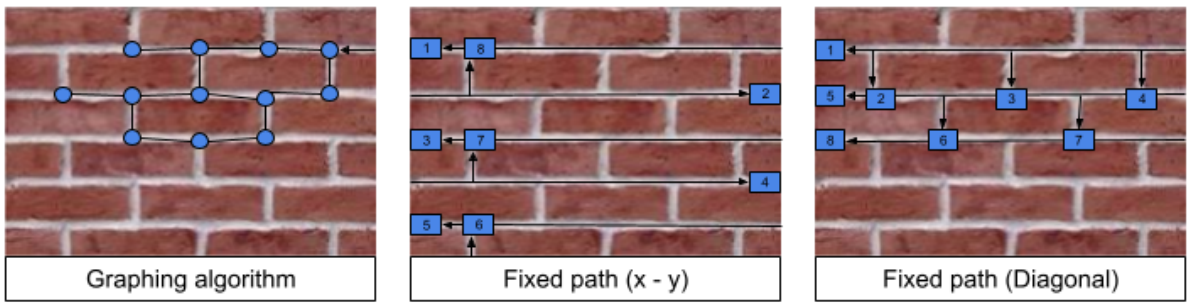


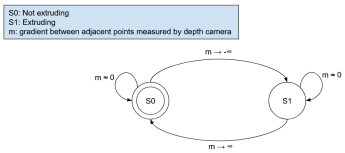
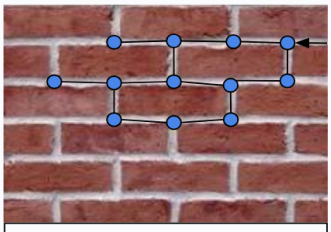
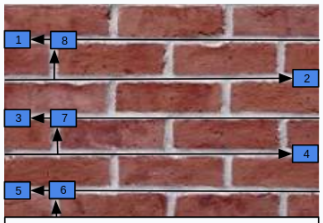
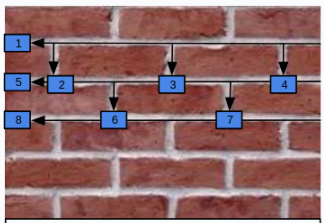
Figure 4.4: Potential pathing algorithm solutions

The potential paths produced by solutions are shown in Figure 4.5. For the graphing algorithm solution, the edges between nodes could be assigned the same weight, or have a weight assigned to them based on the distance between the two adjacent nodes, and

either BFS or DFS would determine an optimal path for the extrusion nozzle to follow. The predetermined routes would follow paths similar to those shown in Figure 4.5. As mentioned before, none of these solutions has a clear advantage over the others and the time taken to fill joints depends more on the solution chosen for differentiating between bricks and joints.

4.2 Morphological Chart

In the morphological chart we have highlighted the chosen solutions for each working principle in blue.

	Concept 1	Concept 2	Concept 3
Power the robot			
Differentiating between Bricks and Joints			
Optimal Routing	 Graphing algorithm	 Fixed path (x - y)	 Fixed path (Diagonal)

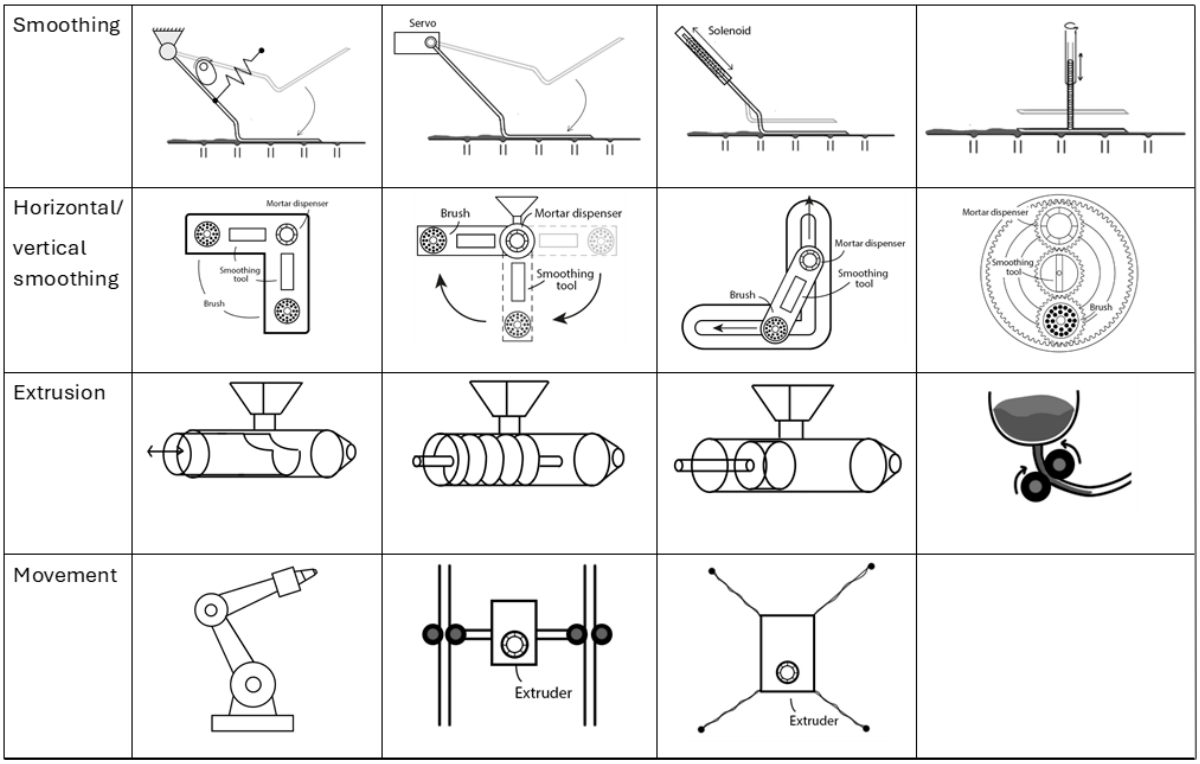


Figure 4.5: Proposed solutions to four mechanical problems

5. Appendix

5.1 Competitor robots

Platform.AS bricklaying robot - https://platform.as/en/bricklaying-robot/
This bricklaying robot lays 240 bricks an hour. It requires an operator solely for entering information about the wall being built (i.e., the measurement of brick bond, placement of windows, etc.). After the 'start' button is pressed, it requires no intervention. This bricklaying robot continuously corrects its movements so it is unaffected by any misalignments that may occur

Okibo - https://okibo.com/
Okibo is an autonomous wall finisher robot which doesn't joint walls but plasters them. It can also paint walls and works alongside the construction workers, requiring no operator. It is battery operated and works at 3x the pace of humans.

SAM - Construction robotics - https://www.construction-robotics.com/sam-2/
https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1275&context=cmsp
The SAM100 and automated brick laying and grouting robot that can lay up to 3,000 bricks per day. It can cost as much as 500,000 dollars and has an incorporated propane generator, sensors, conveyor belts, a concrete pup, and an articulating arm. The robot can pick up the bricks, apply the cement and place them on the wall, but does not actually automate the grout smoothing process. The SAM100 is still reliant on human jointers to finish a facade. In that sense the SAM100 is less competition and more a potential synergetic partner, seeing as AutoArtisan is tasked with making specifically a jointing robot.

Baubot MRS12/MRS5 - https://www.baubot.com/products
The MRS12 and MRS5 robots developed by Baubot both include a robotic arm. MRS12 is equipped with an end-effector to perform various tasks on ceilings and walls up to 4 m, whereas MRS5 is designed to mainly work on floors and walls. They have a payload capacity of at least 10 kg and are both designed to withstand all kinds of weather conditions. Moreover, MRS12 and the MRS5 both have 3D-printing capabilities.

Hadrian X - https://www.fbr.com.au/view/hadrian-x
Hadrian X is a bricklaying/construction robot that is capable of converting 3D CAD model designs into a building. It uses DST (Dynamic Stabilization Technology) to continuously adjust the robot's end effectors (i.e., the peripheral device used by the robot to interact with its environment and is attached to the robot's wrist) so that the arm is always at the correct point in 3D space. Hadrian X can lay around 1000 bricks per hour and is completely autonomous

MSP-robot - Mortar spraying and plastering integrated robot for wall construction
The MSP-robot consists of four modules: the horizontal, vertical, walking and measurement and control modules. The horizontal module is responsible for spraying mortar horizontally, the vertical module lifts/lowers the horizontal module. The measurement and control module uses LiDAR collect data about the surface of the wall and 5 laser-ranging sensors are used to get information about the distance between the robot and the wall. The speed is adjusted depending on scraper angle and plaster thickness.

Craftsmac Construction Robot - https://craftsmaclabs.com/construction-robotics/
The Craftsmac construction robot can build for multi-storey buildings using various standard building materials. It works at the speed 10x that of traditional masons. The robot

has a compact design intended for both internal and external construction. It has a high payload to weight ratio (i.e., can carry large masses of building material despite being lightweight).

ABLR - https://constructionautomation.co.uk/ablr/

The ABLR lays bricks and mortar by reading from digitized versions of an architect's plans. It operates from a track that is fixed around the perimeter of the house being built. The ABLR lays 240 bricks per hour, and a skilled tradesman follows it along the track for final quality inspection. The ABLR also provides real-time analytics to the operators such as: mortar level in each of the hoppers (shown as a percentage value), air temperature, wind speed, bricks laid on the current course, bricks laid during the current session, lost time, etc.

FLX Bot - https://www.flxsolutions.com/

This robot acts as an extension of construction workers' arms. The list of end effectors that can be interchanged include: a 360 degree 4K camera, an aerosolized sprayer, a 3D mapping camera, a thermal camera, a microphone, a caulk dispenser, gripper and gas sensors. This robot is comprised of 3 - 5 individual links which makes this robot especially versatile.

PaceRobotics Wall Finishing robot - https://www.pacerobotics.net/

This robot is able to finish walls at 10x the standard speed, whilst requiring little to no oversight (spraying and rendering is autonomous, navigation is semi-autonomous). This robot also collects execution data in real time