Flatbed 3D Scanning

Authors: Theo Cockrell, Sophia King, Yon Maor
Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—3D scanning is a developing technology that already sees plenty of use in commercial quality control, art and history research, and hobbyist projects. Current state of the art falls short of an accessible price point and at scanning small features. In this report we outline a 3D scanning process that utilizes off-the-shelf flatbed scanners with our open source hardware and software to offer a less expensive, detail-focused alternative to existing 3D scanning solutions.**

***Index Terms*—Computer Graphics, Hausdorff Distance, Open Source, Optics, 3D Scanning**

1 INTRODUCTION

3D scanning has become an extremely useful tool in many different fields and disciplines, from studying archaeological finds and modeling deformation to generating assets for video games. However, 3D scanners are expensive and bulky, and more affordable scanners generally have trouble scanning small objects. This leaves a market gap for cheap, high-resolution 3D scanning solutions, particularly for use on small objects. This project implements an open source hardware-software system that uses an existing flatbed scanner, along with our custom hardware and software, to generate high-resolution 3D scans. The 3D scans are computed using photometric stereo.

This project may be of interest to researchers working in the above fields and hobbyists interested in 3D modeling and 3D printing. As a result, we prioritize this project being open source and accessible, and integrate our control software with Blender, which is ubiquitous open software in the 3D modeling community.

2 USE-CASE REQUIREMENTS

Our primary use-case requirements are for the project to be cheap, accessible, and have the ability to scan small objects with high resolution. Accessibility can be quantified mainly by the cost of components, but other metrics can be applied to measure the difficulty of assembling the hardware and operating the software. Specifically, we aim to have the entire system cost less than $200 (not including the flatbed scanner, which we assume the user has access to), and for the system to take at most 20 minutes to assemble for a non-engineer. Additionally, we will ensure compatibility across all major operating systems (MacOS, Windows, Linux), and with a wide variety of flatbed scanners across the major brands (Canon, Brother, HP, etc.).

Quantifying resolution in 3D is more difficult. The resolution of our scans depends on the DPI of the scan, and different scanners have varying DPI capabilities. Instead of creating a resolution benchmark, we will compare a 3D model generated by our system at 900DPI, with those generated with common commercial 3D scanners (Afinia 3D and Artek 3D). We will model a test object and manufacture it in two different materials (3D printed plastic and machined aluminum), then scan it with our system and the two benchmark scanners. We will then compare each one with the ground truth via the Metro method. We aim to achieve accuracy at most 10% worse than the Afinia 3D Scanner.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

3.1 Overview

Flatbed 3D Scanning relies on a process called photometric stereo to build up the 3D models. This process illuminates an object from multiple different angles and compares the perceived brightness at each pixel across the different illumination angles to compute the orientation of the surface at that pixel. A flatbed scanner is ideal for this application because it captures images that are essentially orthographic and maintains a constant angle between the light source and the light sensor across all surface points.

Our system, when prompted by a blender plugin, will scan the same object with the flatbed scanner n times, rotating the object by a constant amount between each scan, then rotate each image in the opposite direction to create a set of n aligned images with different illumination angles. We can then set up a system of equations that relate the brightness value at a pixel to the normal vector at that pixel. Solving for the normal at each pixel will yield a normal map, which can be converted into a depth map (then, trivially, to any 3D model format) via existing tools.

3.2 Hardware

This project utilizes a manipulator that is designed to be placed on top of an opened flatbed scanner and rotate the object between scans. We intend to achieve this rotation with a stepper motor normal to the scanner surface that rotates a suction cup meant to hold the object with

both suction and friction. These parts will be installed on a 3D-printed mount with free vertical movement on top of a structure made of aluminum T-channel extrusions.

The electronics are implemented with an ESP32 that controls a stepper motor for rotation and a DC motor air pump for suction. This system is powered by 12 volts of wall power and controlled by Python software using USB serial communication .

3.3 Software

3.3.1 Control Software and UI

The control software subsystem is responsible for commanding the manipulator device and the flatbed scanner, as well as communicating with the UI and the image data processing software. The UI will be a Blender add-on that communicates directly with the scanner controller software.

Something about the overview flow of signals between UI, control software, manipulator device, and flatbed scanner. When the user presses the start button in the Blender add-on, it will send a signal to the control software that it should begin. The control software then sends a command to the flatbed scanner to take a scan. After the scan is saved, the scanner controller receives a return signal that the scan completed. The scanner controller then sends a command to the manipulator device to rotate the object. After the manipulator completes its rotation, the scanner controller then sends another command to the flatbed scanner to scan again. This repeats as many times as the user configures, with a minimum of three scans taken.

After all the scans required are saved, they are sent from the scanner controller to the image data processing software. Once the image processing is complete and has constructed a 3D model, the scanner controller will send it to the Blender add-on, where it will be directly imported into the Blender program for any further refinement needed from the user.

3.3.2 Image Data Processing

This software subsystem is responsible for aligning the scanned images, computing the object's normal map, converting the normal map to a height map / .obj, and passing the .obj back to the blender plugin caller.

Aligning the images should be trivial, as the stepper motor and object fixture mechanism will ensure the images are near-perfect rotations of each other. This means that we can simply rotate each image back by the amount the object was rotated when the scan was taken.

The normal map is computed via photometric stereo as described above. [add more details on the math]. This will be implemented in a low-level language such as Rust or C/C++, as this computation requires solving many systems of equations, which can be computationally intensive. That said, for testing purposes we may implement a python version first, to begin characterizing out the system early and not bottleneck the implementation of this subsystem.

Computing a height map from a normal map is far from trivial, but many implementations of this process exist in open source and commercial software, as both height maps and normal maps are common wars for communicating 3D objects. The general approach involves computing the gradients of the normal map in x and y and then finding the height at a given point as the indefinite surface integral of the gradients evaluated at (x, y). This can be approximated via a Reimann sum and averaged across several gradient orientations to reduce noise from the approximation.

4 DESIGN REQUIREMENTS

The primary design requirements are for the project to be cheap, have a simple setup procedure, generate a scan of at least comparable quality to commercial 3D scanners, and to automate the scanning process. Keeping all of these in mind will ensure that we meet our use-case requirements to make the product accessible and approachable for non-engineer users.

The final design should be cheap to compete effectively for its limited object scope. We aim for the final design to cost $200 or less to create, not including the cost of a flatbed scanner or computer which we assume the user already has can access. This also does not include cost of materials for scrapped prototypes. We are tracking the expenses incurred for each part of the design and carefully selecting parts that keep costs low while serving the needed purpose.

To complete the setup procedure of the final design should take under 20 minutes or less for a user who is not an engineer, but is relatively competent with computer technology such as installing programs and a command line. The setup process would include setting up the manipulator device, the flatbed scanner, and the software. For the scanner, it is the same as setting up the same scanner for average use, following the manufacturer's directions. For the manipulator, the user would need to plug it into their computer through USB-A. They would then need to put the manipulator on top of the object they want to scan on the flatbed scanner, ensuring the arm of the manipulator is aligned with the object. Finally, the software setup would involve installing our custom Blender add-on to a copy of the Blender software that is on their computer. They will need to install Blender if they have not already.

Even though it's a budget option, we'd like our project

to have comparable quality to more expensive commercial 3D scanners. To test this, we are going to design a test object and 3D print it to use as a benchmark for comparing the different 3D scanning options. Since we would then have the exact model measurements, we can then use Hausdorff distance to measure our benchmark model to the project scan of our object. Finding the Hausdorff distance between our benchmark model and the commercial 3D scan as well, we can compare how accurate our project scan is against the commercial scan.

Our project should allow scans to take place with minimal user labor. After setup and inputting some parameters, the user should be able to hit the start button and have a completed 3D scan. The whole process of scanning and model construction after pressing start should take less than 6 minutes, regardless of object size as long as it's in scope and regardless of detail level.

5 DESIGN TRADE STUDIES

This section describes the trade-offs involved in the design process, derived from design equations and/or plotted graphs to identify a design that satisfies the use-case requirements. The trade-offs are evaluated to ensure the design meets the related design specifications. Tests to evaluate the design implementation and compare it to theoretical trade-offs will be discussed in Section 7 (Test, Verification, and Validation).

5.1 Affordable Design

The primary trade-off in achieving an affordable design is balancing cost with performance. To keep the total cost below $200 (excluding the flatbed scanner), we prioritized off-the-shelf components and open-source software. However, this introduces limitations in terms of hardware precision and computational efficiency. For example, using a low-cost stepper motor and suction cup mechanism may result in less precise rotations, which could affect the alignment of scanned images. Additionally, relying on photometric stereo without additional sensors (e.g., depth cameras) increases the computational complexity of the image processing pipeline. The trade-off here is between cost and the accuracy of the final 3D model.

5.2 Simple Setup Procedure

Ensuring a simple setup procedure for non-engineer users requires careful design of both the hardware and software. The trade-off lies in the complexity of the user interface (UI) and the time required for assembly. For instance, designing a Blender add-on for the UI simplifies the user experience but increases development time and effort. Similarly, the manipulator's design must balance ease of assembly with structural stability. Using modular components (e.g. aluminum T-channel extrusions) simplifies assembly but may require additional adjustments to ensure precise alignment during operation. The trade-off is between user accessibility and the engineering effort required to achieve it.

5.3 Scans of Quality

Achieving high-quality scans involves trade-offs in resolution, computational resources, and scanning time. Higher DPI scans provide more detail but require more storage and processing power. Additionally, the photometric stereo method relies on solving systems of equations for each pixel, which is computationally intensive. To balance quality and performance, we use a low-level language (e.g., Rust or C/C++) for the image processing software. However, this increases development complexity compared to using a higher-level language like Python. The trade-off is between scan quality, computational efficiency, and development time.

5.4 Automated Process

Automating the scanning process reduces user effort but increases the complexity of the control software and hardware integration. The trade-off involves ensuring seamless communication between the Blender add-on, the manipulator, and the flatbed scanner. For example, automating the rotation and scanning sequence requires precise timing and error handling, which adds complexity to the software. Additionally, automating the suction mechanism introduces challenges in ensuring consistent object fixation during rotation. The trade-off is between user convenience and the engineering effort required to achieve a fully automated system.

6 SYSTEM IMPLEMENTATION

6.1 Hardware

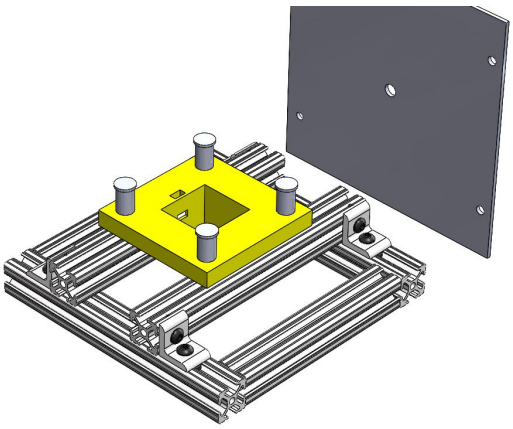

Fig 1. A CAD rendering of the manipulator. The 3D printed electronics mount is colored yellow.

The structure of the manipulator is built from aluminum T-channel extrusions and general hardware like screws and nuts. An 8 in. x 8 in. square is used as the base so that the manipulator can fit on most flatbed scanners, which are designed to scan on at least an 8.5 in. x 11 in. surface. A second layer of T-channel extrusions is used as support for the 3D printed electronics mount that is centered on the manipulator. This second layer also holds vertical screws with spacers that function as guide rails for the electronics mount to freely move on. This is meant to automatically fit objects up to 1 in. (the space under the second layer of T-channels) by allowing the mount to slide upwards in order to accommodate the object and then rest on top of it. The mount itself will contain the ESP32, motor controllers, stepper motor, DC motor air pump, and suction cup used to manipulate a scanned object. With the stepper motor and its suction cup extension resting on top of the object, the suction cup is activated before rotating the stepper motor. Between the friction of the mount resting on the object and the suction "fixing" the object to the suction cup, precise rotations of the stepper motor should rotate the object as well.

The ESP32 is programmed using the Arduino IDE (C++). The core functionality consists of commands that are encoded as individual characters over serial. Each command is characterized as a single-line command for either the stepper motor or DC motor air pump. For example, stepper motor commands are specified to rotate forward or backward over a given number of degrees; sending "a" over serial could command a clockwise rotation of 45 degrees. DC motor air pump controls are simply "Forward," "Reverse," and "Release." Each of these could also nbe encoded as a single-character serial command. Currently, the serial connection has been tested with pyserial (to be integrated into the blender plugin), but identical functionality could be achieved in a different language if necessary.

6.2 Control Software and UI

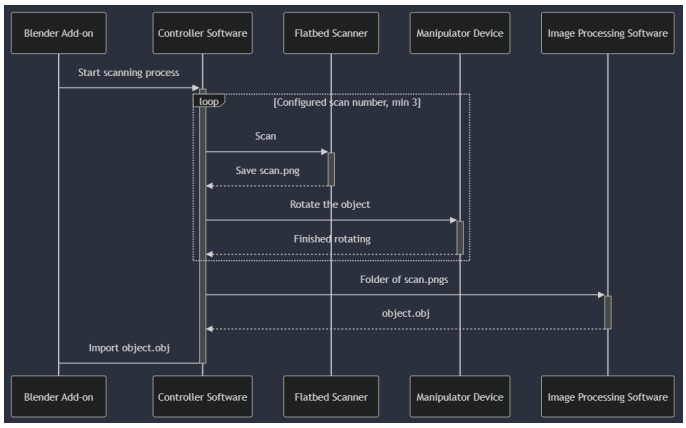

Fig 2. A sequence diagram of communication between software and hardware elements.

The Blender add-on that acts as the UI will be in Python, as that's the language that Blender itself is written in and requires. However, using a module such as ctypes or a library like cffi will allow the Python file to call C functions and files.

The controller software will be in C++, but that is subject to change. Originally, we were planning on using the NAPS2 scanning library to have one library available for any of the three main OS, but ran into complications with trying to run the project on different OS. NAPS2 requires a dotnet project to be used, which was difficult to set up and NAPS2 required the dotnet 4.8 framework, which is worrying with the latest release being dotnet 9.0.

As a result, we may pivot to having the controller file call one of 3 different files depending on user's OS in order to use an individual scanning library that suits them instead of using a universal library in one file. Possible libraries include: the built-in WIA library with C for Windows, either the built-in ImageCaptureCore or the SANE open-source library with Python for Mac, and SANE with Python also is compatible with Linux.

6.3 Image Processing Software

To find the normal vector at each pixel, we set up a system of equations that relate the observed intensity to the normal vector at the given pixel. For $n = 4$ rotations it is relatively simple to derive a closed form:

The observed intensity I_0 of a surface point is given by:

$$I_0 = \rho \int_{-l}^{l} \langle \mathbf{n}, \mathbf{l} \rangle \, dx = \rho \int_{-l}^{l} \frac{n_x x + n_y a + n_z b}{\sqrt{x^2 + a^2 + b^2}} \, dx$$

where:

- ρ is the surface albedo,
- $\mathbf{n} = (n_x, n_y, n_z)$ is the normal vector at the surface point,
- $\mathbf{l} = (x, a, b)$ is the light source direction,
- l is the integration limit.

The integral simplifies to:

$$I_0 = \rho(n_y a + n_z b) \int_{-l}^{l} \frac{1}{\sqrt{x^2 + a^2 + b^2}}\, dx = \rho s(n_y a + n_z b)$$

where:

$$s = \int_{-l}^{l} \frac{1}{\sqrt{x^2 + a^2 + b^2}}\, dx = 2 \ln \frac{l + \sqrt{l^2 + a^2 + b^2}}{\sqrt{a^2 + b^2}}$$

The integral of the term involving $n_x x$ vanishes because it is an odd (anti-symmetrical) function over the interval $(-l, l)$:

$$\int_{-l}^{l} \frac{n_x x}{\sqrt{x^2 + a^2 + b^2}}\, dx = 0$$

By scanning the same surface point with the object rotated by 90°, 180°, and 270°, we obtain additional intensity measurements:

$$I_{90} = \rho s(-n_x a + n_z b)$$

$$I_{180} = \rho s(-n_y a + n_z b)$$

$$I_{270} = \rho s(n_x a + n_z b)$$

These four equations (including I_0) can be arranged into a matrix equation and solved using linear least-squares. The solution for the components of the normal vector is:

$$\rho s b n_x = \frac{I_{270} - I_{90}}{2 \tan a}$$

$$\rho s b n_y = \frac{I_0 - I_{180}}{2 \tan a}$$

$$\rho s b n_z = \frac{I_0 + I_{90} + I_{180} + I_{270}}{4}$$

ρ, s, b cancel out, leaving a closed form for the normal vector.

The math is less trivial for different numbers of rotations, so one approach is to solve the system numerically for each n during testing, then compute the closed form for whichever n shows the best results.

7 TEST & VALIDATION

7.1 Affordable Design

Testing this metric is trivial, as we only have to meet our self-imposed cost limit.

7.2 Simple Setup Procedure

We plan on performing user tests on a sample group with a diverse background in relevant fields, to determine whether we meet our assembly time benchmark. We can also use the same user tests to quantify the ease of use of our UI, although most Blender plugins have similar UI schemes.

7.3 Scans of Quality

Ensuring the scans are good quality will require testing and a way to compare our scans to other commercial scans. It will include unit testing for intensity calculation. For creating comparisons, we will create a benchmark 3d model object and 3D print it. We'll scan the benchmark object with our project and with other 3D commercial scanners available to us. After finding the Hausdorff distance, we can compare exactly which is closer to the benchmark and determine the quality of our scans vs a commercial scanner.

7.4 Automated Process

To make it all an automated process, there must be seamless communication between all the subsystem components. To ensure this, we're doing unit testing for each subsystem. After they're working individually, we slowly start to integrate them and do practical unit tests. For example, after the controller software passes unit tests checking its command signals, we can connect it with the manipulator and test individual commands to see if they work before testing the whole file and string of commands. A tradeoff of making it all automatic means that there's a lot of labor from the software to communicate with everything, which is more engineering work to create. However, this avoids a poor user experience of having to press a button to start another process frequently given how many subprocesses there are and simplifies the user experience, going back to the principles of being easily approachable without technical skills.

8 PROJECT MANAGEMENT

8.1 Schedule

See the Gantt Chart in Figure 2.

8.2 Team Member Responsibilities

Brief overview of what each team member is responsible for. This subsection should be no more than half a column.

Theo's primary responsibilities are the design, development, and characterization of the manipulator. His secondary responsibilities are to aid his teammates in hardware/software integration and image processing math.

Sophia's primary responsibilities are to code the controller software and UI, ensuring their integration with the hardware components and image processing software successfully. Her secondary responsibilities are to help her teammates with any difficulties along with ensuring quality of documentation including presentations, website updates, and status reports.

Yon's primary responsibilities are in the image data processing subunit. This includes aligning the images from the scanner, working out the math for computing the normal map and implementing the system, and implementing a normals-to-height-map system. Yon's secondary responsibilities include qualifying the final scans and generating benchmarks using other commercial 3D scanners. He is also responsible for most of the 3D printing.

8.3 Bill of Materials and Budget

See Table 1 for a table of materials and components purchased.

8.4 TechSpark Useage Plans

We plan to use TechSpark to laser cut the acrylic cover sheet that we will test to block ambient light during scanning.

8.5 Risk Mitigation Plans

The most critical risk factor we've identified is the possibility of an abrasive object scratching the scanning surface during rotation. The best solution we've identified so far would be to install solenoids where the mount rests on the T-channel extrusions. This would enable us to lift the mount slightly when rotating, picking up the object as well due to the suction cup. This introduces limiting factors and design changes: the lifting force of the suction cup becomes a bottleneck (ours can hold approximately 4 pounds), and either the mount or stepper-suction cup connection would need to be redesigned in order to keep the suction cup resting on the scanner surface if no object is being scanned (keeping the full 1 in. clearance previously specified).

9 RELATED WORK

Photometric stereo was first introduced by R. J. Woodham in Photometric Methods for Determining Surface Orientation from Multiple Images. This process was later applied by V. Skala and R. Pan using a flatbed scanner, but this project did not implement a solution for image alignment, making it extremely difficult to apply in practice.

Other related projects include W. W. Chi's and D. Byrne's several projects on 3D Microscopic Texture Interface in CAD, which are also low-cost scanning systems used in 3D modeling.

10 SUMMARY

Our project is a device and software package that utilizes the existing technology of a flatbed scanner to produce quality 3D object scans for a fraction of the price of existing commercial 3D scanners. It scans objects from different angles using the consistent light source of the flatbed scanner to be able to assess the shape of the object. The hardware utilizes a manipulator arm to rotate the object to get multiple angles and communicated directly with the user's computer and software. It will be a challenge to ensure that particularly abrasive objects are not given too much force, or else they will scratch the glass of the flatbed scanner. The controller software has a UI in the form of a Blender add-on for easy access and editing of the model after it's completed. The controller software will have a challenge in trying to be universal in OS's and flatbed scanner models. The controller software communicated with the flatbed scanner, the manipulator device, the UI, and the image processing software. The image processing software uses a set of equations and pixel-to-pixel comparisons of the scans to create a surface normal map that is converted into a 3D model, but may encounter difficulties accounting for noise and precise image alignment. The result is an accurately detailed .obj file 3D model in an approachable and affordable way, perfect for research departments with tight budgets or hobbyists.

References

[1] Woodham, R. J. (1980). *Photometric method for determining surface orientation from multiple images.* Optical Engineering, 19(1), 191139. Retrieved from `https://www.cs.ubc.ca/~woodham/papers/Woodham80c.pdf`

[2] Skala, V., & Pan, R. (2015). *Making 3D replicas using a flatbed scanner and a 3D printer.* Journal of WSCG, 23(1), 1–8. Retrieved from `https://www.researchgate.net/publication/`

281105714_Making_3D_Replicas_Using_a_Flatbed_Scanner_and_a_3D_Printer

[3] Chi, W.-W., & Byrne, D. (2020). *3D Microscopic Texture Interface in CAD: Cultivating Material Knowledge for the Practiced Digital Hand*. In Proceedings of the 2020 ACM Designing Interactive Systems Conference (pp. 1–13). Retrieved from https://dl.acm.org/doi/abs/10.1145/3357236.3395579

[4] Chi, W.-W. (2019). *3D Microscopic Texture Interface in CAD: Cultivating Material Knowledge for the Practiced Digital Hand*. Master's thesis, Carnegie Mellon University. Retrieved from https://kilthub.cmu.edu/articles/thesis/3D_Microscopic_Texture_Interface_in_CAD_Cultivating_Material_Knowledge_for_the_Practiced_Digital_Hand/8234276?file=15351137

[5] Skala, V. (1999). *Surface reconstruction from photometric stereo images*. Computer Graphics Forum, 18(3), C1–C8. Retrieved from https://onlinelibrary.wiley.com/doi/epdf/10.1111/1467-8659.00236

Table 1: Bill of Materials

Description	Model #	Manufacturer	Quantity	Cost @	Total
12V DC Power Supply with adapters	B0D7L6942H	Amazon	1	$9.98	$9.98
Aluminum Spacers 3/4" unthreaded for 1/4-20 bolts	B07K34NZ2K	Amazon	2	$4.16	$8.32
SAE Stainless Steel Assorted Bolts, Washers, Nuts	B0CL4PK2KY	Amazon	1	$14.99	$14.99
8x8 inch 1/8" opaque white acrylic sheet	B0DD6VJ4FB	Amazon	1	$8.99	$8.99
4mm silicone tubing t-connector	B083WNPXQ7	Amazon	1	$4.99	$4.99
5mm shaft coupler + mount	B08334MFVT	Amazon	1	$7.99	$7.99
M3 Screws Hardware package	B0D1457XQ3	Amazon	1	$9.99	$9.99
Adafruit Huzzah (ESP32)	3619	Adafruit	1	$21.95	$21.95
Adafruit DC Motor/Stepper Featherwing	2927	Adafruit	1	$21.20	$21.20
Adafruit NEMA-17 Stepper Motor	324	Adafruit	1	$14.00	$14.00
Adafruit DC Air Pump	4700	Adafruit	1	$6.95	$6.95
Adafruit 3mm silicone tubing for air pump	4661	Adafruit	1	$2.50	$2.50
Adafruit Double-sided Foam Tape	5019	Adafruit	4	$0.75	$3.00
6 in. 1010 series	1010	80/20	2	$4.89	$9.78
8 in. 1010 series	1010	80/20	4	$5.59	$22.36
Inside-inside 1010 connector	33440	80/20	4	$4.36	$17.44
Right-angle gussets	4119	80/20	4	$6.38	$25.52
Slip-in fasteners	3382	80/20	20	$0.32	$6.40
0.5" bolts	3061	80/20	10	$0.34	$3.40
0.75" bolts	3065	80/20	10	$0.42	$4.20
1 ft. Rubber Footing	2828	80/20	4	$1.66	$6.64
Slide-in full-sized T-nuts	3204	80/20	4	$1.16	$4.64
0.286" Washer	3258	80/20	4	$0.12	$0.48
13mm single bellows FDA Silicone suction cup	SB14-SIT-G18M	VacMotion	1	$6.67	$6.67
CanoScan Lide 400 Slim Flatbed Scanner	013803306521	Canon	1	$96.29	$96.29
					$338.67

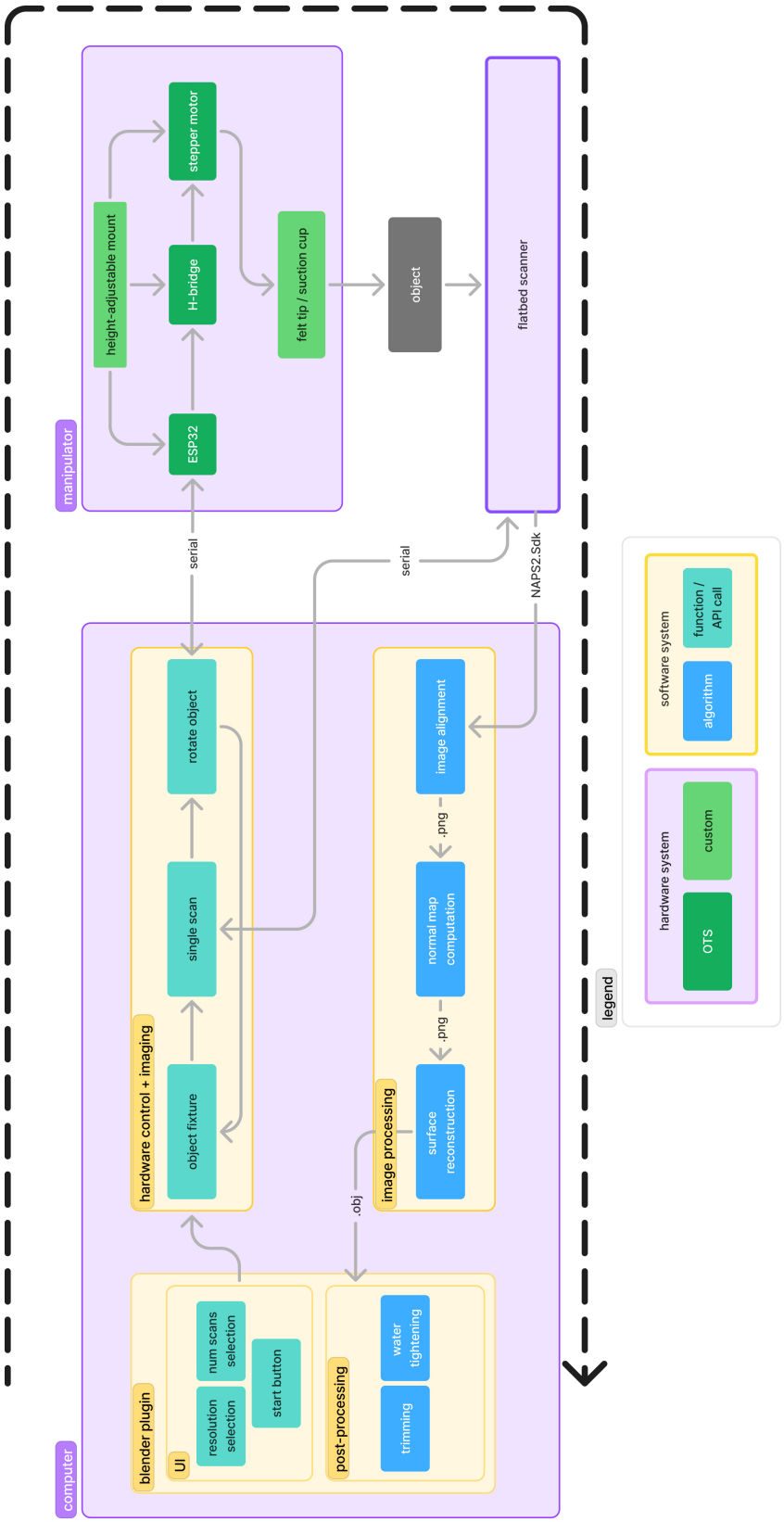

Figure 1: A full-page version of the system block diagram

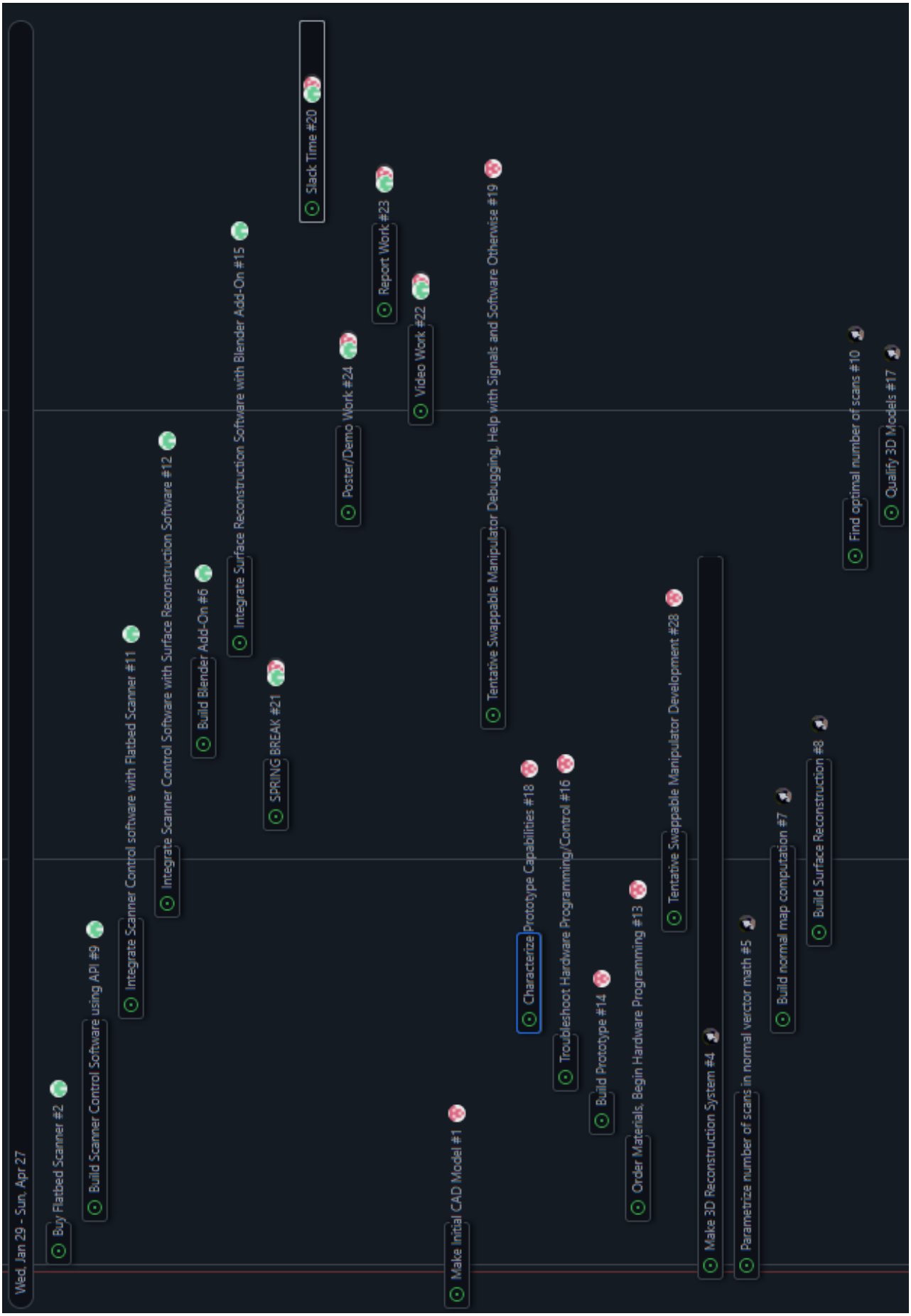

Figure 2: Roadmap