Team A5:
FLATBED 3D SCANNING

Theo Cockrell, Sophia King, Yon Maor

Proposal Recap

- **Gap :** 3D scanning expensive, not optimized for small objects
- **Solution :** produce 3D scans using existing flatbed scanner + cheap hardware and custom software

Use Case Requirements

- **Small** object scanning
  - Keep scanning area to max of 6 in. x 6 in.
- **Works across platforms**
  - Test on Windows, Mac, and Linux environments separately and ensure scanning outputs match, given using same flatbed scanner
- **Works for variety of scanners**
  - Test for results on Canon scanner as default and a variety of other scanners brands around campus to ensure scanning output match with < 10% error

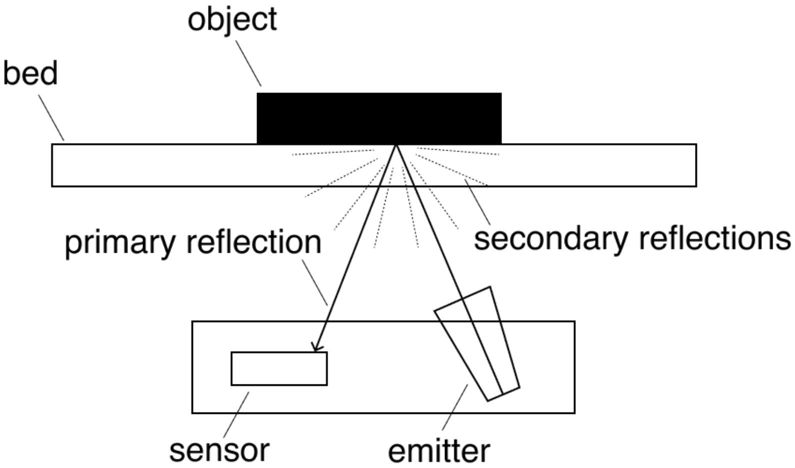
Design Requirements

- **Cheap** design
  - Track expensenses so prototype device is built within $200 limit
- **Simple** assembly procedure
  - Time how long it takes a new user to set up hardware and software, under 20 minutes
- Generates scans of **comparable quality** to commercial 3D scanners
  - 90% accuracy to commercial scanner output measured by Hausdorff distance
- **Automated** scanning and 3D model generation process
  - Takes < 6 minutes to create a scan for variety of objects of different detail level

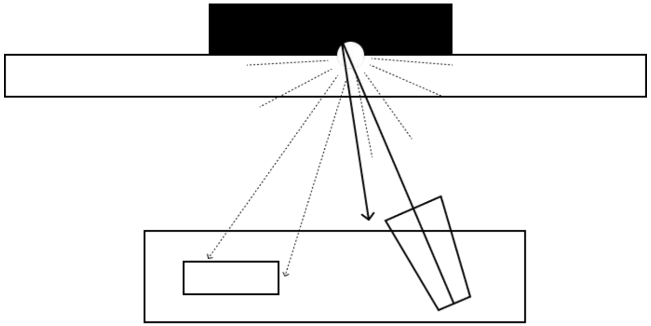
Solution Approach

- **Cheap and Simple**
  - How - Simple parts are cheaper, minimal assembly and software setup
  - Why - Easy to use for target user of non-engineer/hobbyist
- **Fast**
  - How - Focus on scanning small objects with a fast algorithm
  - Why - Users won’t want to wait a long time
- **Quality outputs**
  - How - Algorithm tested and tweaked for precision
  - Why - Deliver quality to compete, even as a budget alternative
- **Universal**
  - How - Universal library for OS and scanners
  - Why - To be compatible without extra effort or funds

Constructing the Model



scanning a flat object



scanning a contoured object

We can now model the observed intensity at a given pixel as

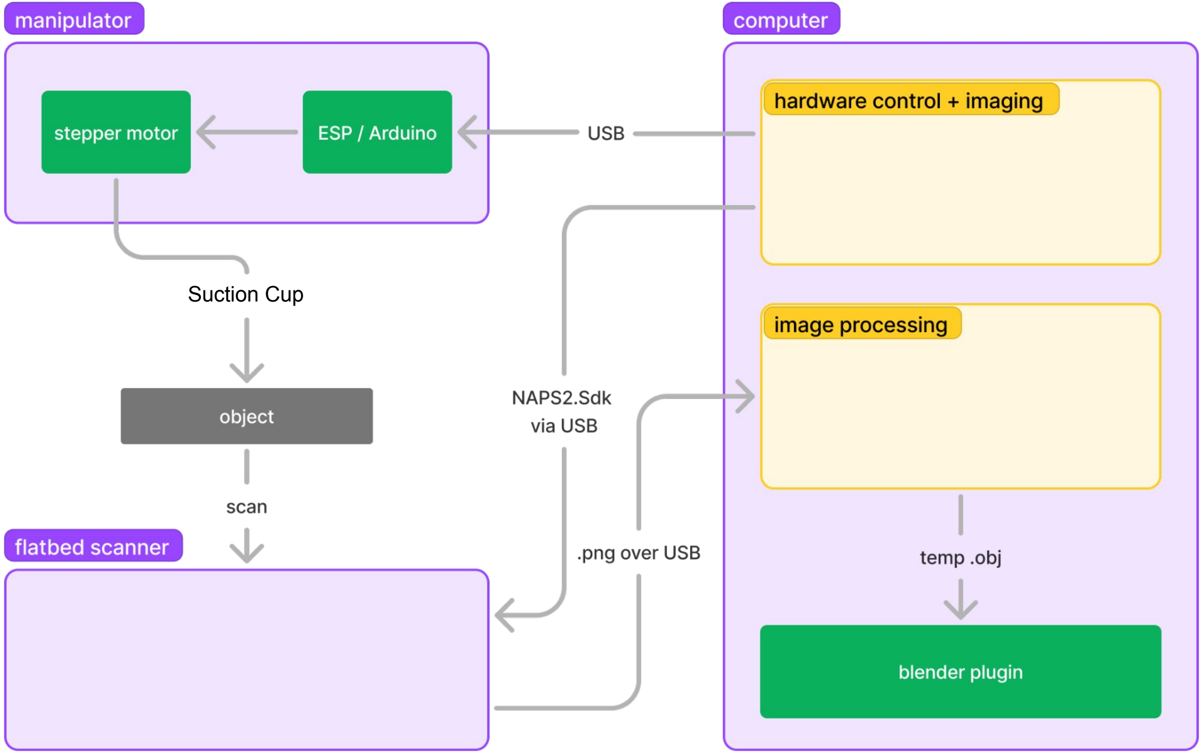
$$I_\alpha = f(n_x, n_y, n_z, \alpha)$$

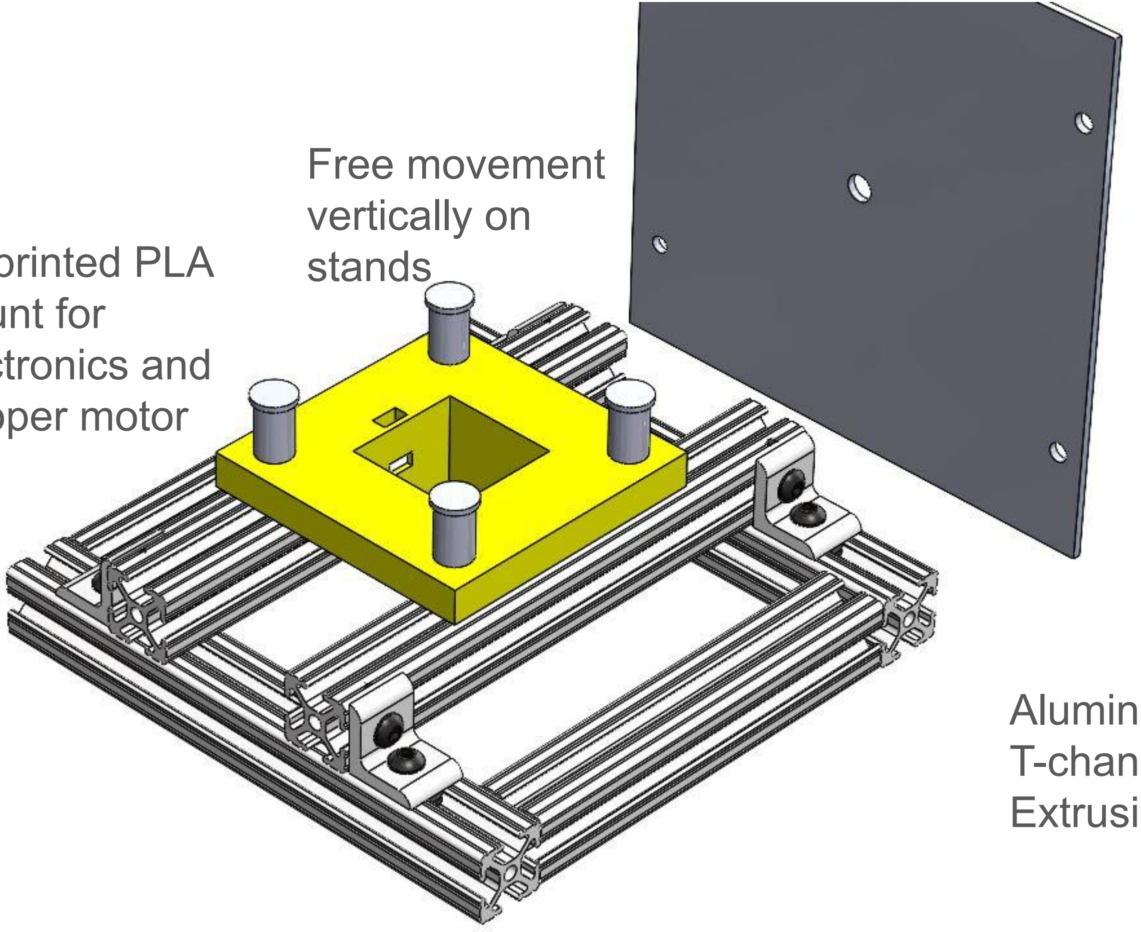
for some angle α. Repeating this for k unique angles α, β, γ, etc. we can construct a system of k equations—

$$\begin{cases} I_\alpha = f(n_x, n_y, n_z, \alpha) \\ I_\beta = f(n_x, n_y, n_z, \beta) \\ I_\gamma = f(n_x, n_y, n_z, \gamma) \\ \dots \end{cases}$$

—from which we can solve for the normal vector. With 3 equations (from 3 images) we have a system with rank 3 and a unique solution. Adding more equations (taking more images) overparametrises the system with linearly dependent columns, possibly reducing noise (test this).

Use Case -> Solution -> Comms





3D printed PLA mount for electronics and stepper motor

Free movement vertically on stands

Matte Acrylic "Cover Sheet"

Aluminum T-channel Extrusion Frame

Dimensions:
8"x8"x3.75"

Testing, Verification, and Validation

- **Test Inputs**
  - Software - Injecting mock data or images depending on test
  - Hardware - Different sized objects, rotation commands
  - Signals Processing - Unit tests for intensity calculation, comparison
- **Test Outputs**
  - Software - Checking output signals and uncorrupted image output file
  - Hardware - Observed stability, measured rotation precision
  - Signals Processing - Passing unit tests, Hausdorff distance
- **Risk factors and Unknowns**
  - Objects not rotating exactly, margin of error for “real” conditions
    - How much error to allow and possible remedies?
    - Correction in software or hardware?
  - Limiting noise from the scanning process
    - Blocking outside light from interfering with flatbed scanner light
    - The math not accounting for real world conditions

Tasks and Division of Labor

Sophia -> Software

- Import 3D model directly into **Blender** through add-on
- Create **UI** for the scanning and model generation process
- **Automate** the scanning and file transfers through scripts

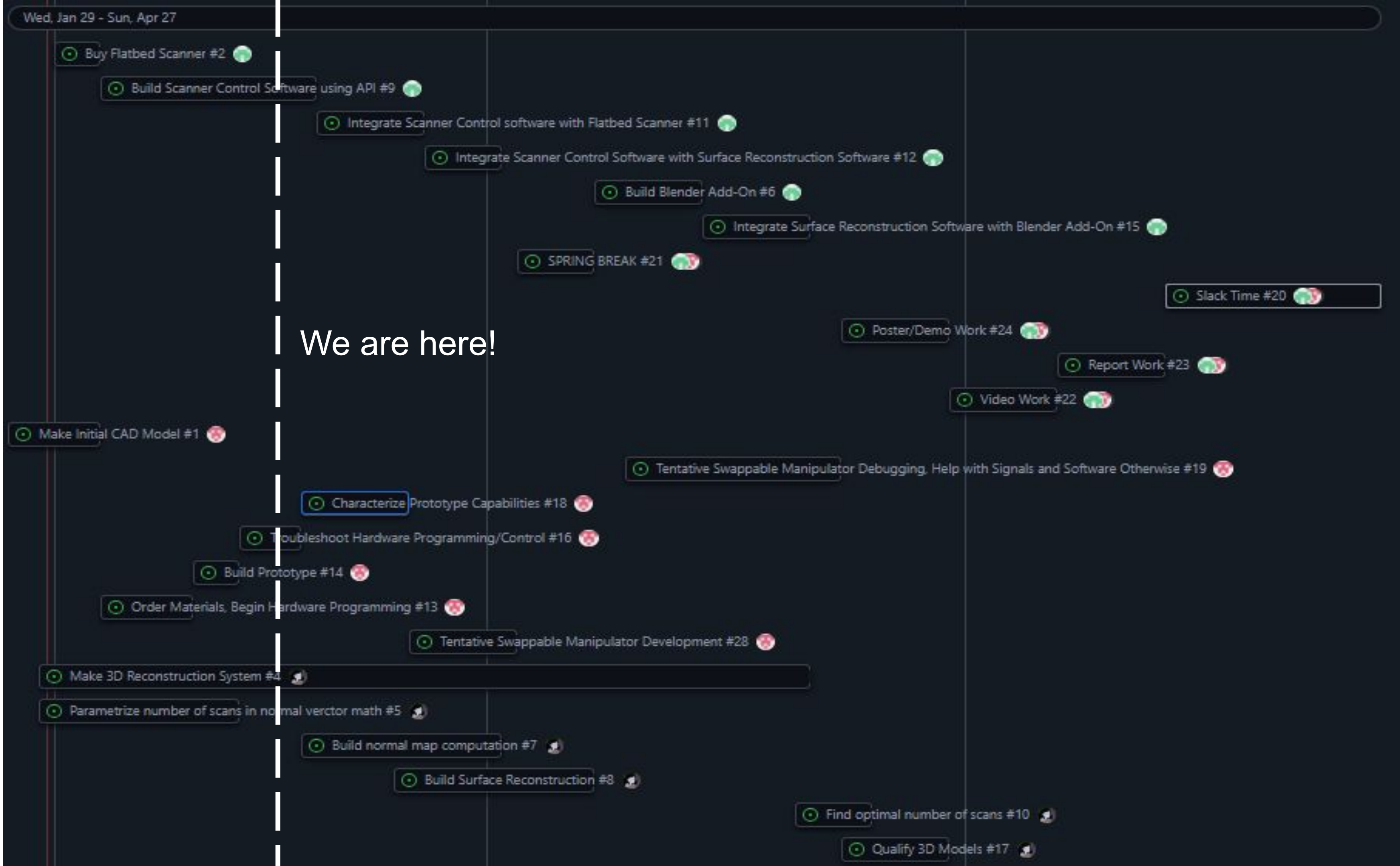
Theo -> Hardware

- Design **manipulator** for interacting with object
- Program manipulator to cooperate with software
- **Investigate limitations** of scannable objects and potential solutions

Yon -> Signal Processing

- Work out **optics math** for computing normal-map
- Implement-normal map to **height-map** algorithm in low-level language
- **Qualify** final 3D models against benchmark

Schedule



Wed, Jan 29 - Sun, Apr 27

We are here!

- Buy Flatbed Scanner #2
- Build Scanner Control Software using API #9
- Integrate Scanner Control software with Flatbed Scanner #11
- Integrate Scanner Control Software with Surface Reconstruction Software #12
- Build Blender Add-On #6
- Integrate Surface Reconstruction Software with Blender Add-On #15
- SPRING BREAK #21
- Slack Time #20
- Poster/Demo Work #24
- Report Work #23
- Video Work #22
- Make Initial CAD Model #1
- Tentative Swappable Manipulator Debugging, Help with Signals and Software Otherwise #19
- Characterize Prototype Capabilities #18
- Troubleshoot Hardware Programming/Control #16
- Build Prototype #14
- Order Materials, Begin Hardware Programming #13
- Tentative Swappable Manipulator Development #28
- Make 3D Reconstruction System #4
- Parametrize number of scans in normal verctor math #5
- Build normal map computation #7
- Build Surface Reconstruction #8
- Find optimal number of scans #10
- Qualify 3D Models #17

Current Work

Sophia - Software

- Downloading NAPS2 universal scanner library
- Use online tutorials for serial communication development
- Use online tutorials for Blender add-on development

Theo - Hardware

- Purchasing materials
- 3D printing custom parts/mounts for electronics

Yon - Signal Processing

- Iterating on existing algorithm
- Researching Hausdorff distance