

Entwicklerdokumentation

PackMate

SoSe 2025, Cross Innovation Class 2025

Lukas Krämer



Einführung

Diese Entwicklerdokumentation beschreibt unseren Prototypen PackMate, der im Rahmen der Cross Innovation Class 2025 als interdisziplinäres Projekt von sechs Studierenden der Leuphana Universität Lüneburg, Akademie für Mode und Design Hamburg, Brand University Hamburg und FH Wedel entstanden ist. PackMate erleichtert das sichere und umweltfreundliche Verpacken von Kleinteilen und ist anfänglich unter Betreuung des Unternehmens STILL entstanden, um ihre Lieferanten zu motivieren, auf nachhaltiges Verpackungsmaterial zu wechseln. Die tatsächliche Fragestellung lautete dabei wie folgt:

„How can STILL provide their European, German, and local suppliers with solutions that motivate them to use sustainable material in wrapping and packaging?“

PackMate kann dazu genutzt werden, ein bereits bestehendes Bauteil, welches versendet werden soll, mittels zwei Kameras einzuscannen und dann optimiertes Verpackungsmaterial dazu aus Pappe zu lasercutten, um einen sicheren Versand zu gewährleisten. Er substituiert dabei die bisher häufig eingesetzte Luftpolsterfolie, was langfristig nicht nur Materialkosten spart, sondern natürlich auch besser für die Umwelt ist. Selbstredend ist Pappe dabei ohnehin schon ein deutlich ökologischerer Werkstoff als Plastik mit vergleichsweise geringen Treibhausgasemissionen, allerdings kann PackMate darüber hinaus auch noch alte und somit ohnehin schon produzierte Pappkartons wiederverwenden, wodurch die Umweltbilanz weiter verbessert wird und zumindest bilanziell keine neuen Rohstoffe genutzt und Treibhausgase emittiert werden.

Der Entwicklungsfokus lag dabei neben der Umsetzung der bereits beschriebenen Kriterien hauptsächlich auf der leichten Bedienung, die möglichst selbsterklärend sein sollte und sich gut in bestehende Betriebsprozesse integrieren muss. Aufgrund von Sicherheitsbedenken und des engen Zeitplans wurde zudem zunächst auf den Einbau des eigentlichen Lasercutters verzichtet. Dieser könnte später allerdings aufgrund der modularen Bauweise mehr oder weniger einfach hinzugefügt werden.

Hardware

Der Großteil des Designkonzeptes von PackMate wurde in diesem Projekt von den beiden Industriedesignern der AMD entwickelt und soll sich nicht nur gut in ein industrielles Umfeld integrieren können, sondern zudem leicht bedienbar sein.

Grundsätzlich besteht das Produkt dabei aus vier verschiedenen Bauteilen. Dazu gehört neben der Scan-Kammer, Papplager und dem Lasercutting-Bereich noch das interaktive Bedienelement auf der rechten Seite. Dieses besteht im Wesentlichen aus einem 7 Zoll Touchscreen-Display, mit dem die gesamte Steuerung der Maschine erfolgt. Vom Einschaltknopf abgesehen, gibt es keine weiteren Eingabemöglichkeiten.

Links neben dem Bedienelement befindet sich zunächst die Scan-Kammer, in der zwei gewöhnliche USB-Kameras im 90 Grad Winkel zueinander befestigt sind. Diese sind nachher für den Scan-Prozess zuständig, der im Software Abschnitt näher erläutert wird.

Unterhalb der Kammer sind zu guter Letzt noch zwei Fächer, in denen die Pappe verstaut (Fach 2) bzw. das fertig zugeschnittene Verpackungsmaterial entnommen werden kann (Fach 1).

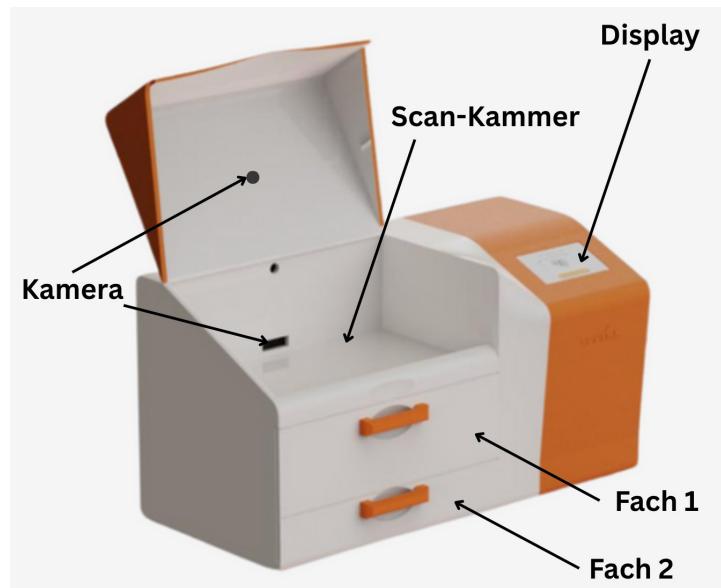


Bild 1: Übersicht PackMate

Elektronik

Um die beiden USB-Kameras auszulesen und per Touch-Display das User Interface bereitzustellen, kam ein Raspberry Pi 5 zum Einsatz. Dieser ist auch dafür verantwortlich, die Scan-Kammer mittels eines 1m langen LED Streifens (WS2816, 144 Stück) während des Scan-Prozesses ausreichend zu beleuchten. Zur Stromversorgung wurde ein 5V 15A Netzteil verwendet, um für zukünftige

Erweiterungen (Lasercutter) genügend Puffer zur Verfügung zu haben, aktuell beträgt die Stromaufnahme unter Volllast jedoch lediglich ca. 7A.

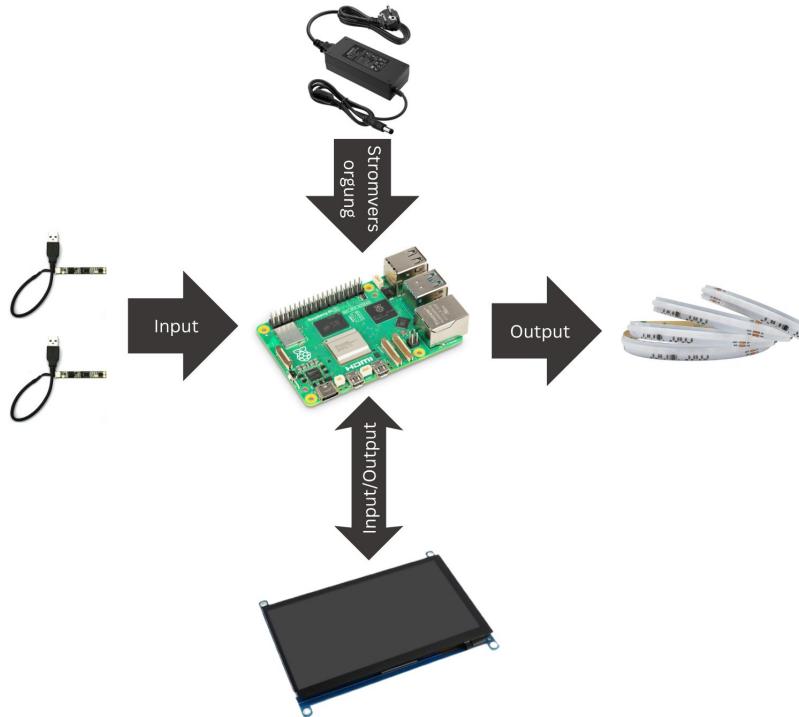


Bild 2: Schematischer Schaltplan

Software

Die Software für PackMate ist vollständig in Python implementiert, wobei die Kernfunktionalität das Erfassen von Bildern mit den beiden USB-Kameras, die Verarbeitung dieser Bilder zur Erstellung binärer Masken, die Generierung eines 3D-Modells und die Vorbereitung optimierter Verpackungsdesigns umfasst. Der Benutzer interagiert über das Touchdisplay mit dem Gerät, das eine mit Pygame entwickelte grafische Benutzeroberfläche bereitstellt. Das System steuert zudem den WS2816-LED-Streifen, um die Scankammer während der Bildaufnahme auszuleuchten, was nicht nur den Kontrast zwischen Objekt und Hintergrund erhöht, sondern auch Schattenwürfe auf ein Minimum reduziert. Beim Kauf der Kameras wurde zudem darauf geachtet, dass diese über automatische Kontrast- und Fokusanpassung verfügen, was den Softwareaufwand erheblich reduziert.

Die Software ist modular aufgebaut, mit einer klaren Trennung zwischen GUI-Handling, Bildverarbeitung, 3D-Modellgenerierung und Hardwaresteuerung, was die Wartbarkeit erhöhen soll und mögliche Erweiterungen in Zukunft erleichtert.

Die verwendeten Hauptbibliotheken sind dabei:

Pygame: Darstellung der GUI und die Verarbeitung von Benutzerinteraktionen auf dem Touchscreen

OpenCV: Bildaufnahme und -verarbeitung.

NumPy: Für effiziente Array-Operationen bei der Bildverarbeitung und 3D-Modellgenerierung

scikit-image (skimage): Generierung von 3D-Modellen mit dem Marching-Cubes-Algorithmus (wird später näher erläutert)

Matplotlib: Darstellung der STL-Vorschau

pi5neo: Steuerung des WS2816-LED-Streifens

Die Software ist in einem einzigen, monolithischen Hauptskript (main.py) strukturiert, das alle Funktionalitäten, einschließlich des GUI-Managements, Kameraoperationen und 3D-Modellgenerierung, umfasst.

Die Architektur folgt dabei einem zustandsbasierten Design, bei dem das System je nach Benutzerinteraktionen und Verarbeitungsphasen zwischen verschiedenen Bildschirmen (Zuständen) wechselt. Die Hauptbestandteile der Software sind wie folgt:

1. Benutzeroberfläche

Die GUI ist mit Pygame implementiert und läuft im Vollbildmodus auf dem 7 Zoll Touchscreen-Display (die gesamte Ansicht wird dabei um 180 Grad gedreht, da das Display falsch herum verbaut ist). Sie ist möglichst intuitiv und selbsterklärend gestaltet, um die Bedienung zu vereinfachen. Die einzelnen UI-Elemente stammen dabei auch von den Studierenden der AMD. Das System unterstützt die folgenden Bildschirme:

Hauptmenü: Zeigt Optionen zum Starten eines neuen Scans oder zum Zugriff auf ein Vorabmodell an

Scanner-Bildschirm: Zeigt Live-Feeds beider USB-Kameras an, sodass der Benutzer Bilder aufnehmen kann. Befindet sich der Nutzer auf diesem Screen, wird zudem der LED-Streifen eingeschaltet.

Maskenvorschau-Bildschirm: Zeigt die aus den aufgenommenen Bildern generierten binären Masken, damit der Nutzer diese Verifizieren und ggf. erneut aufnehmen kann.

Normalisierte Masken-Bildschirm: Zeigt die ausgerichteten und skalierten Masken für die 3D-Modellgenerierung. Dieser Schritt ist notwendig, da die beiden Kameras sich nicht immer gleich weit weg und genau gleich rotiert vom Objekt befinden.

STL-Generierung-Bildschirm: Zeigt einen Fortschrittsbalken während der Erstellung des 3D-Modells.

STL-Vorschau-Bildschirm: Rendert eine Vorschau des generierten 3D-Modells mit Optionen zum Wiederholen oder Fortfahren mit der Verpackung. Dieser Schritt ist wichtig, damit der Benutzer ggf. Artefakte im Scan erkennen kann und diesen wiederholt, bevor das Material zugeschnitten wurde.

Cutting-Bildschirm: Simuliert den Laserschneideprozess mit einem Fortschrittsbalken (derzeit ein Platzhalter, weil der Laser noch nicht verbaut ist).

Schneiden-abgeschlossen-Bildschirm: Bestätigt den Abschluss und fordert den Benutzer auf, das geschnittene Material zu entnehmen.

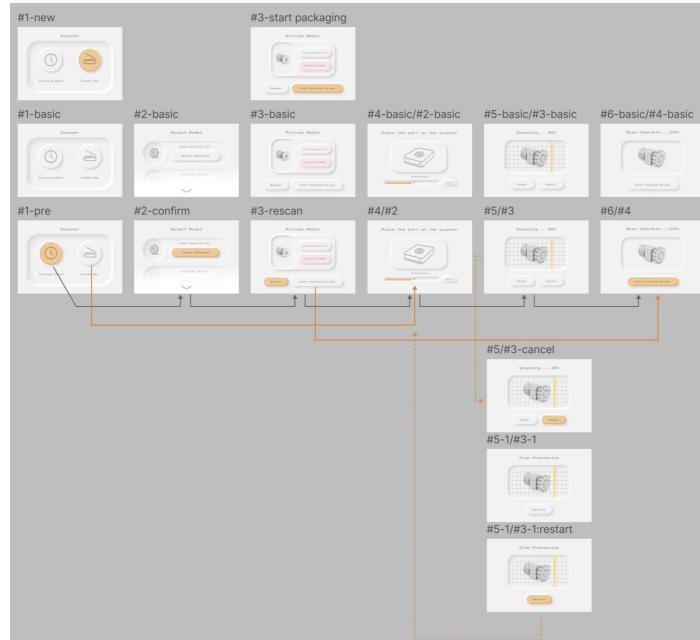


Bild 3: UX-Konzept

Jeder Bildschirm wird durch eine dedizierte Methode (z. B. draw_main_menu, draw_camera_preview) gerendert, Benutzerinteraktionen werden hingegen über Touch-Events in der Methode handle_events verarbeitet. Die GUI verwendet vorgefertigte Bilder aus dem Ordner „ui elements“ für Schaltflächen und Hintergründe, um ein konsistentes und professionelles Erscheinungsbild zu

gewährleisten. Wie bereits erwähnt sind die UI-Elemente dabei in Kooperation und in mehreren Iterationen mit den Designern der AMD entstanden.

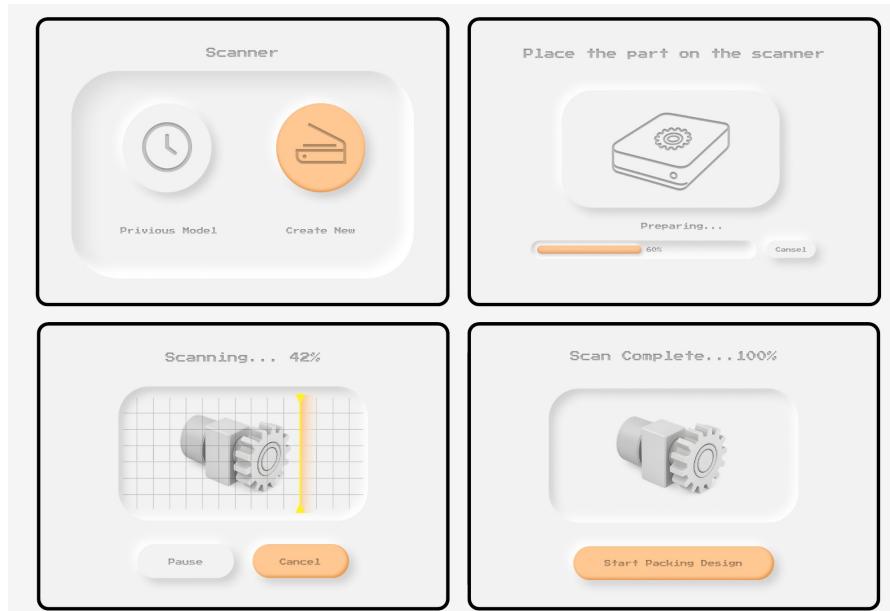


Bild 4: UI-Konzept

2. Kamera- und Bildverarbeitung

Das System nutzt zwei USB-Kameras, die im 90-Grad-Winkel zueinander positioniert sind, um Bilder des in der Scankammer platzierten Objekts aufzunehmen. Die Kameras werden mittels OpenCV (cv2.VideoCapture) auf den Geräten /dev/video0 und /dev/video2 ausgelesen. Der Bildverarbeitungsprozess umfasst die folgenden Schritte:

Aufnahme: Bilder werden durch Drücken der „Weiter“-Schaltfläche auf dem Scanner-Bildschirm aufgenommen (snap_image).

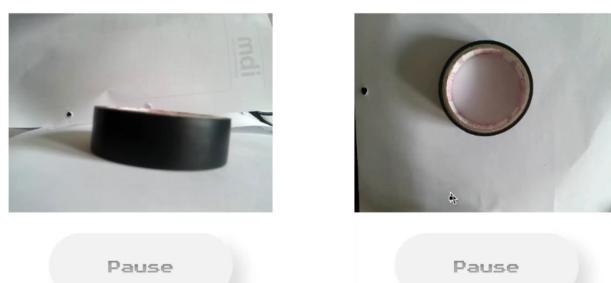


Bild 5: Aufnahmebildschirm

Generierung binärer Masken: Die Methode process_mask wandelt die Bilder in Graustufen um und wendet eine Gaußsche Weichzeichnung an, um den Einfluss von Rauschen und kleinen Verschmutzungen auf das Scan-Ergebnis zu verringern. Anschließend werden zwei binäre Masken durch Schwellenwertbildung erstellt, morphologische Operationen (Erosion und Dilatation) reinigen dabei die Masken und die größte Kontur wird gefüllt, um das Objekt zu isolieren.

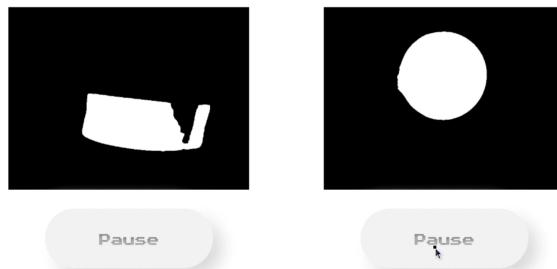


Bild 6: Masken des Objektes

Normalisierung: Die Methode normalize_mask rotiert, schneidet und skaliert die Masken auf eine Standardgröße (400x300 Pixel), um die unterschiedlichen Positionierungen zwischen Kameränen und Objekt auszugleichen.

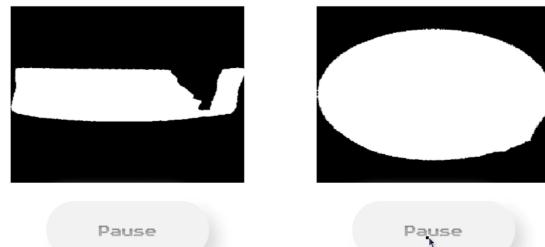


Bild 7: Normalisierte Masken

3D-Modellgenerierung: Die Methode generate_stl_and_preview kombiniert die normalisierten Masken zu einem 3D-Volumen und speichert das Ergebnis als STL-Datei (model.stl). Eine Vorschau des 3D-Modells wird mit Matplotlib gerendert und auf dem STL-Vorschau-Bildschirm angezeigt.

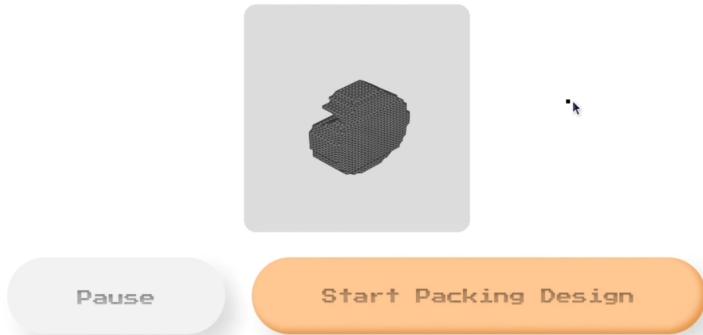


Bild 8: Vorschau des 3D-Modells

Die 3D-Modellgenerierung basiert auf dem Ansatz der Volumenschnittmenge-Methode (Volume Intersection), der aus zwei orthogonalen 2D-Silhouetten ein dreidimensionales Volumenmodell generiert. Der Prozess beginnt mit der Skalierung der normalisierten Masken auf eine einheitliche Voxel-Auflösung von 30x30x30 Elementen, wobei die Seitenansicht auf die X-Y-Ebene und die Draufsicht auf die X-Z-Ebene projiziert wird. Diese bewusst niedrig gewählte Auflösung stellt einen Kompromiss zwischen Rechenzeit und Detailgenauigkeit dar, der für die Anwendung völlig ausreichend ist und dafür sorgt, dass die Berechnung in wenigen Sekunden durchgeführt werden kann. Da der Zusammenhang zwischen Voxel-Größe und Rechenzeit kubisch ist, würde bereits eine leicht bessere Auflösung zu erheblich verlängerter Rechenzeit führen.

Im nächsten Schritt werden die beiden 2D-Masken in dreidimensionale Volumen extrudiert: Die Seitensilhouette wird entlang der Z-Achse ausgedehnt, während die Draufsicht-Silhouette entlang der Y-Achse projiziert wird. Diese Extrusion erfolgt mittels NumPy's broadcast_to-Funktion, die effizient die 2D-Information auf alle Schichten des entsprechenden 3D-Volumens überträgt. Das resultierende Objekt entsteht schließlich durch die logische UND-Verknüpfung beider Volumen, wodurch nur die Voxel als "solid" markiert werden, die in beiden Projektionen als Objekt erkannt wurden. Dieser Ansatz eliminiert effektiv Bereiche, die nur in einer der beiden Ansichten sichtbar sind, und rekonstruiert dadurch eine plausible 3D-Approximation des ursprünglichen Objekts.

Der Marching Cubes Algorithmus kommt anschließend zum Einsatz, um aus diesem binären Voxel-Grid eine glatte Oberflächendarstellung zu generieren. Dieser Algorithmus analysiert systematisch jeden 2x2x2-Würfel im Voxel-Grid und bestimmt, wie die Iso-Oberfläche (hier bei Level 0.5, also der Grenze zwischen

"solid" und "leer") durch diesen Würfel verläuft. Basierend auf den acht Eckpunkten jedes Würfels und deren Zuordnung zu "innerhalb" oder "außerhalb" des Objekts, wählt der Algorithmus aus einer vordefinierten Lookup-Table die entsprechende Dreieckskonfiguration aus. Diese Dreiecke werden dann durch lineare Interpolation exakt an der Objektgrenze positioniert, wodurch eine glatte, kontinuierliche Oberflächendarstellung entsteht.

Die durch Marching Cubes generierten Vertices und Faces werden anschließend in das gängige STL-Format konvertiert und auf der Micro-SD Karte abgespeichert, um später für weitere Verarbeitungsschritte genutzt werden zu können(Berechnung der optimierten Verpackung und Lasercutten)

3. Hardwaresteuerung

Die Software steuert einen 1 Meter langen WS2816-LED-Streifen (144 LEDs), um die Scankammer während der Bildaufnahme auszuleuchten. Die Bibliothek pi5neo kommuniziert über die SPI-Schnittstelle des Raspberry Pi (/dev/spidev0.0), um die Helligkeit des LED-Streifens zu setzen und diesen je nach Bildschirm ein- oder auszuschalten. Die LEDs werden nur während des Scanner-Bildschirms aktiviert, um Strom zu sparen und eine optimale Beleuchtung für die Bildaufnahme zu gewährleisten.

4. Zustandsmanagement

Um den zustandsbasierten Ansatz verwalten zu können, wird über das Attribut current_screen der aktiven Bildschirm bestimmt. Übergänge zwischen Bildschirmen werden entweder durch Benutzerinteraktionen (z. B. Berühren von Schaltflächen) oder automatisierte Prozesse (z. B. Abschluss der STL-Generierung) ausgelöst. Die Zustandsmaschine ist in der Methode handle_events implementiert, die Touch-Events überprüft und current_screen entsprechend aktualisiert.

5. Konfiguration

Die Software enthält mehrere konfigurierbare Parameter, die am Anfang des Skripts definiert sind und so auch leicht angepasst werden können, wenn man selbst vielleicht nur Benutzer der Maschine ist und den Code versteht:

ROTATE_SCREEN: Eine Boolean-Flag, um das Display um 180 Grad zu drehen. Das ist für den echten Prototyp notwendig, kann für das Debuggen über VNC

allerdings dadurch schnell geändert werden, damit man das Interface auf seinem eigenen Monitor richtig gedreht angezeigt bekommt.

COLOR_THRESHOLD: Der Schwellenwert für die Generierung binärer Masken. Falls sehr helle (z.B. weiße) Objekte gescannt werden möchte, kann dieser (zu Lasten der Zuverlässigkeit) verringert werden.

LED_BRIGHTNESS: Die Helligkeit für den LED-Streifen. Diese sollte so gering wie möglich (Strom sparen, Vermeidung von Reflexionen und Überbelichtung), aber so hoch wie nötig (Eliminierung von Schatten, Rauschreduzierung) eingestellt werden.

UI_FOLDER: Das Verzeichnis (ui elements), das die GUI-Bilder enthält.

Verbesserungspotential

Abgesehen davon, dass PackMate ohne die Ergänzung des Lasercutters nicht wirklich funktionsfähig ist, gibt es auch bei der bestehenden Technik an mancher Stelle Verbesserungspotential.

Hardwareseitig sind so insbesondere die Scharniere der Fächer und Klappen eine große Schwachstelle, da diese zum Teil zu klein (und damit instabil) oder maßlos zu groß (und damit schwergängig zu bedienen) gewählt wurden. Diese sollten daher idealerweise durch eine adäquate Alternative getauscht werden.

Außerdem ist das 15A Netzteil zwar in der Lage, das gesamte Gerät mit Strom zu versorgen, allerdings erkennt der Raspberry Pi, dass es sich nicht um das originale Netzteil handelt und schaltet daher ab und zu un reproduzierbar, das per USB versorgte, Display aus. Durch einen Neustart kann das Problem zwar (temporär) behoben werden, dauerhaft sollte am Raspberry Pi jedoch die entsprechende Konfiguration vorgenommen werden, um ihm trotz nicht zertifizierter Hardware maximale Ausgangsleistung an den USB Ports zu erlauben. Erste Versuche diese Konfiguration vorzunehmen blieben bisher erfolglos.

Auch der Scan-Prozess könnte noch weiter verbessert werden. Aktuell würde aufgrund des verwendeten Verfahrens nämlich beispielsweise eine Rolle Isolierband im 3D-Modell wie ein durchgehender „Teller“ aussehen (die eigentlich hohle Mitte wird also ausgefüllt). Für den Anwendungsfall Verpackungsmaterial zu erstellen ist dies zwar meist irrelevant, weil die äußere Hülle des Objektes am Wichtigsten ist, besser wäre die erläuterte Alternative aber dennoch. Zudem sollte

in Erwägung gezogen werden, eine dritte Kamera zu ergänzen, damit auch „echte“ 3D-Scans möglich sind.

Wartung

Softwareseitig ist keine Wartung notwendig, insbesondere regelmäßige (Sicherheits-) Updates werden aufgrund des offline Betriebs nicht zwangsläufig benötigt.

Hardwareseitig sollten im Falle von unerwartetem Verhalten zunächst alle Kabelverbindungen (insbesondere die der Kameras und des Displays) überprüft werden, außerdem ist nicht ausgeschlossen, dass aufgrund der gewählten Scharniere langfristig Defekte auftreten.

Weiterentwicklung

Entsprechende Weiterentwicklungen können gerne als Pull-Request im folgenden GitHub Repository hinzugefügt werden:

<https://github.com/kraemerlukas314/Cross-Innovation-Class-2025>

Auch Feature-Requests oder Bug-Reports können gerne auf gleiche Weise eingereicht werden.

Teileliste

- [Raspberry Pi 5 \(4GB RAM\)](#)
- [Micro SD Karte \(32 GB\)](#)
- [4kg PLA \(weiß\)](#)
- [Elecrow 7-Inch 1024 x 600 HD TFT LCD](#)
- [2x USB Kamera](#)
- [LED Streifen \(WS2816, 1m, 144 LEDs\)](#)
- [Schalter](#)
- [5V 15A Netzteil](#)
- 4mm Sperrholzplatten (insgesamt ca. 1,5m²)

Werkzeug

- Lötkolben
- Lötzinn
- 3D Drucker
- Lasercutter