# Gedanken

* Lebensmittel Nährstoffbestimmung 🡪 Volumenbestimmung 🡪 Hauptfokus = Bestimmung der Tiefeninformationen zu einem Bild
* Nährstoffbestimmung ist stark abhängig von:
  + Dichte
  + Volumen
  + Formel zur Masseberechnung: m = p \* V
* Volumenbestimmung ist stark abhängig von:
  + Bildauflösung und -rauschen
  + Genauigkeit der Tiefeninformationen
  + Aktive & passive Verfahren zur Bestimmung der Tiefeninformationen

Forschungslücke aufweisen 🡪 Fragestellungen ableiten 🡪 Ziel bestimmen

Fragen Kickoff Gespräch

1. Quellcode in Anhang oder als Zip (wie Internetquellen)? **Beides möglich**
2. Welchen Leitfaden benutzen? **IT-Management nutzen 02/2022**
3. Form der Einreichung der Arbeit? Ausdruck notwendig? **Nur PDF Upload OC**
4. Muster-Deckblatt Arbeit verwenden in Thesis (von Mail)? **Aus Mail verwenden**
5. Angewandte Methoden? **Theorie + Ausprägung + Begründen warum gewählt!**
6. Design Science Research?
   * Ziel: Untersuchung der Messgenauigkeit und praktischen Nutzbarkeit in Trackinglösungen
   * Neben der Messgenauigkeit werden weitere Faktoren zur Bewertung der Eignung ermittelt und betrachtet 🡪 subjektive Bewertung?
     + Andere Entwickler könnten die Bewertung anders sehen…

* **5 bis maximal 7 Hauptkapitel!**

Nährstoffbestimmung überall entfernen (Code + Text)

1. Wie kann mit Object Capture das Volumen von Lebensmitteln bestimmt werden?
2. Wie sieht eine mögliche technische Lösung aus?
3. Welche Messgenauigkeit liefert diese Lösung?
4. Welche Faktoren erhöhen bzw. senken die Messgenauigkeit?

Das zu entwickelnde Artefakt wird nur erstellt, um erste Messdaten mittels Object Capture zu liefern und zu zeigen, dass Object Capture als zentrale Komponente zur 3D Rekonstruktion in der Lage ist eine darauf aufbauende Volumenbestimmung zu ermöglichen.

Auf Basis der durch das Artefakt gesammelten Messdaten (Ergebnisse) kann weiterführend deren Genauigkeit durch den Vergleich der Messergebnisse mit dem tatsächlichen Zielwert bewertet werden.

Funktionale Anforderungen:

* Die Lösung muss in der Lage sein das vorliegende Lebensmittel zu erfassen und sein tatsächliches Volumen zu messen.
* Die Lösung muss in eine mobile Anwendung integrierbar sein, um die geplanten Messungen komfortabel durchführen zu können.
* Die Lösung muss für einen Endanwender bzgl. der Bedienung verständlich und benutzerfreundlich gestaltet sein.
* Die Lösung muss für die komfortable Durchführung der Messungen eine Möglichkeit bieten alle verfügbaren Konfigurationsprofile von Object Capture ansteuern zu können.
* Die Lösung muss in der Lage sein, das Messergebnis zur Dokumentation und weiterführenden Auswertung zu speichern.

Nicht-Funktionale Anforderungen:

* Benutzbarkeit
* Effizienzanforderungen
* Korrektheit
* Installierbarkeit
* Fehlertoleranz
* Verfügbarkeit

Entwicklungsumgebung:

**Xcode**

Für die Programmierung dieser App wurde die integrierte Entwicklungsumgebung Xcode

8.2 von Apple eingesetzt. Für die Entwicklung mit Xcode 8.2 wird ein PC/Notebook

mit macOS (Sierra 10.12 oder El Capitan 10.11.5) vorausgesetzt. Xcode unterstützt im

Wesentlichen zwei Programmiersprachen, Swift und Objective-C, unter Verwendung der

Cocoa-Frameworks (siehe 6.1).

**Swift**

Bei Swift handelt es sich um die neuere Programmiersprache, da sie zur Worldwide

Developers Conference (WWDC) 2014 vorgestellt wurde. Hierbei handelt es sich um eine

objektorientierte Programmiersprache, welche unter anderem von Objective-C, Ruby

und Python beeinflusst wurde und bei iOS, OS X, tvOS, watchOS und Linux zum

Einsatz kommt. Swift ist eine Alternative zu Objective-C, welche vorher die primäre Programmiersprache zur Entwicklung von Anwendungen darstellte. 2016 hat Apple bereits

die dritte Version von Swift vorgestellt und auf lange Sicht wird diese wohl Objective-C

völlig ablösen, daher kam in dieser Arbeit Swift als primäre Programmiersprache zum

Einsatz.

**Interface-Builder**

Um die grafische Benutzeroberfläche zu designen, wurde der ebenfalls in Xcode integrierte

Interface-Builder verwendet. Dieser ermöglicht es, die komplette Benutzeroberfläche zu

erstellen, ohne eine Zeile Code schreiben zu müssen.

1. **Einleitung**
   1. Problemstellung
   2. Zielsetzung (Forschungsfragen einbauen)
   3. Themenabgrenzung
   4. Struktur der Arbeit
2. **Angewandte Methoden** (Design Science Research Methodologie)
   1. Relevance Cycle (Environment)

Systematische Literatur Review (Verweis auf K. 2.2)

Anforderungsanalyse (Verweis auf K. 4.1)

* 1. Rigor Cycle (Knowledge Base)

Systematische Literatur Review (Verweis auf K. 2.1 + 2.2)

* 1. Design Cycle (Iterative Implementierung + Evaluation)

Agile Softwareentw. (Erläuterung + geeignet, wegen DSR Iterationszyklen)

Systemmodellierung (Erläuterung + Verweis auf K. 4.2 – 4.3)

Prototyping (Erläuterung + Verweis auf K. 4.4 – 4.5)

Laborexperiment (Erläuterung, auch Gütekriterien + Verweis auf K. 4.6)

1. **Grundlagen und Forschungsansätze**
   1. Computer Vision und 3D Rekonstruktion
   2. Aktuelle Forschungsansätze
   3. RealityKit Object Capture
2. **Konzeption**
   1. Anforderungsanalyse (Kurze Zf. der Problemdefinition, Verweis auf K. 1.1. + 2.2.)
      1. Funktionale Anforderungen
      2. Nichtfunktionale Anforderungen
   2. Anwendungsfälle (UML-Anwendungsfalldiagramm + User Stories Karten)
   3. Systemarchitektur (Client-/ Serverarchitektur + UML-Sequenzdiagramm)
   4. Applikationsentwurf (Prototyping mithilfe von Mockups 🡪 Anwendungsfluss)
   5. Fachliches Datenmodell (siehe ios-pentest Beispiel + Klassendiagramm)
      1. Models
      2. View und Controller
   6. Laborexperiment (Beschreibung gilt für alle folgenden Experimente/Evaluationen)
      1. Versuchsaufbau (Transparente Beschreibung des Setups + Endgeräte)
      2. Versuchs- und Kontrollgruppen (alle Lebensmittel)

🡪 Kontrollgruppe = Zielwerte mittels Differenzmethode erhoben

1. **Implementierung** (Beschreibung iterative Umsetzung des Prototyps im Sinne des DSR + agiler Softwareentwicklung 🡪 nach jedem Iterationszyklus wird Prototyp gegen die in Kapitel 4.1 definierten Anforderungen evaluiert)
   1. Entwicklungsumgebung und Bibliotheken (siehe oben)

Xcode + Swift, Visual Studio Code, Python Version, Pyvista, Trimesh etc.

* 1. Erster Iteration
     1. Clientseitige Implementierung (iOS App)
        1. Benutzeroberfläche (Abbildung zum Application Storyboard)
        2. Kamerafunktionalität
        3. Serverkommunikation (nur Aufruf der eigenen Server APIs)
     2. Serverseitige Implementierung (FastAPI)
        1. Aufruf Object Capture API
        2. Volumenberechnung (mit PyVista und Trimesh)
     3. Evaluation (Beschreibung der Durchführung des Testdurchlaufs zum Testen des Prototyps 🡪 Ergebnisse sammeln 🡪 Bewertung, ob aktuelle Implementierungslösung die Anforderungen in K. 4.1 erfüllt 🡪 Aktuelles Problem darstellen, Lösung erst in zweiter Iteration beschreiben)
  2. Zweiter Iteration
     1. Notwendige Anpassungen (Depth einschalten)
     2. Evaluation (alle Anforderungen erfolgreich umgesetzt)

1. **Auswertung und Diskussion**
   1. Auswertung (Verweis auf Anhang 🡪 wichtigsten Messergebnisse beschreiben 🡪 Schlussfolgerungen)
   2. Diskussion (kritische Betrachtung: Laborbedingungen, nur ein Lebensmittel betrachtet etc.)
2. **Zusammenfassung und Ausblick**
   1. Zusammenfassung
   2. Ausblick

Laborexperiment

<https://www.scribbr.de/methodik/abhaengige-und-unabhaengige-variable/>

Versuchsgruppe:

* 6 verschiedene Lebensmittel

Kontrollgruppe:

Unabhängige Variable

* Eingabeparameter von Object Capture (Konfigurationsprofil)
* Aufnahmegerät für die Messung (iPhone 12 Pro + OS Version) 🡪 bleibt unverändert
* Bildauflösung (maximale Bildauflösung des iPhones) 🡪 bleibt unverändert
* Beleuchtungskonfiguration 🡪 bleibt unverändert
* Positionierung des Lebensmittels 🡪 bleibt unverändert
* Versuchsaufbau (Stehtisch, Drehteller, Stativposition, -höhe, -ausrichtung) 🡪 bleibt unverändert

Abhängige Variablen

* gemessene Volumen des Lebensmittels (Messvolumen)
* gemessene Zeit zur Bereitstellung des Volumens (Messzeit)
* Abweichung des Messvolumens vom tatsächlichen Zielvolumen in %

Visualisierung für Auswertung:

* Unabhängige Variable = x-Achse
* Abhängige Variable = y-Achse
* Säulendiagramm ist geeignet, wenn deine unabhängige Variable eine kategoriale Variable ist (z. B. eine nominalskalierte oder ordinalskalierte Variable)

# Pipeline 3D Rekonstruktion (image-based 3D reconstruction)

🡪 Image-based 3D reconstruction = passive Methode im Bereich der 3D Rekonstruktion

1. **Image acquisition** **and selection** = Fotografieren des Objects aus verschiedenen Blickwinkeln (= Serien von Bildern) + Selektion der besten Bilder zur Verarbeitung
2. **Feature point extraction and matching**
   1. **SIFT** (Extraktion von Merkmalspunkten (features) in ausgewählten Bildern
   2. **RANSAC** = Abgleich + Übereinstimmung zwischen mehreren Bildern finden
3. **Calculation of camera parameters + 3D coordinates of scene**
   1. **Structure from motion (SfM)** 🡪 SfM nutzt Korrespondenzen zur Schätzung der genauen Kameraposen und 3D-Koordinaten der Szene / des Objekts 🡪 spärliche 3D-Punktwolke des Objekts (sparse point cloud)
   2. **Bundle adjustment**
4. **Production of dense 3D scene model** = multi-view-stereo (MVS) 🡪 nimmt Bilder mit Kameraposen Information 🡪 dichte 3D-Punktwolke des Objekts
   1. **CMVS** (Clustering-views for Multi-View Stereo) 🡪 Preprocessor zur Kachelung großer Bilddaten
   2. **PMVS** (Patch-based Multi-view Stereo) 🡪 zur Verdichtung der Punktwolke
5. **Calculation of absolute object scale** (Helmert-Transformation)
6. **Mesh creation** = **Delaunay Triangulation** = Konstruktion von Triangulationsmeshes
7. **Fill holes in created 3D mesh**
   1. volume-based methods
   2. surface-based methods
8. **Calculate volume from 3D triangulated mesh**
   1. **Convex hull method** (1. **Konvexe Hülle** von 3D Mesh bilden 🡪 **Graham scan** algorithm; 2. Z. Xu and H. Xu, “Fast algorithm of computing volume based

on convex hull,” Computer Engineering and Applications,

vol. 29, no. 4, 2013.)

* 1. Slicing method (fällt raus, da als Eingabe eine Punktwolke erwartet wird)
  2. Projection method (fällt raus, da als Eingabe eine Punktwolke erwartet wird)