Praxis der Softwareentwicklung: 3D Reconstruction Framework from Multi-View Images (3D-MuVi)

Pflichtenheft

Grigori Schapoval Nathanael Schneider Tim Brodbeck
Stefan Wolf Jens Manig Laurenz Thiel

WS 2015/16

Inhaltsverzeichnis

1 Einleitung **4**

2 Zielbestimmung **5**

3 Anforderungen **6**
3.1 Abgrenzungskriterien 6
3.2 Musskriterien 6
3.3 Kannkriterien 7

4 Produkteinsatz **9**
4.1 Einsatzgebiete 9
4.2 Zielgruppe 9
4.3 Betriebsbedingungen 9

5 Produktumgebung **10**
5.1 Software 10
5.2 Hardware 10
5.3 Schnittstellen 10

6 Produktfunktion **11**
6.1 Grundfunktionen 11
6.2 Funktional 11

7 Ablaufplan **13**
7.1 Ablaufplan 13

8 Systemmodell **15**
8.1 Systemübersicht 15
8.2 Module 16
8.3 Produktdaten 17
8.4 Schnittstellen 17
8.5 Datenflussübersicht 18
8.6 Anwendugsfälle mit Sequenzdiagramm 19
8.7 Typische Nutzungsabläufe 20

9 Produktleistungen **22**
9.1 Laufzeitverhalten 22
9.2 Speicherplatzbedarf 22

10 Benutzeroberfläche **23**
10.1 Hauptfenster 23
10.2 Erweitertes Hauptfenster zur Umsetzung von Kann-Kriterien 25

11 Testfälle und Testszenarien **27**
11.1 Testfälle 27
11.2 Testszenarien 31

12 Qualitätszielbestimmungen **34**
12.1 Übersicht 34
12.2 Zusammenfassung 37

13 Entwicklungsumgebung **38**
13.1 Allgemein 38
13.2 Implementierung 38
13.3 Hilfsbibliotheken 38
13.4 Validierung 39
13.5 Erstellung von Grafiken 39

14 Glossar **40**

1 Einleitung

Die Welt wird digital - und mit ihr viele Prozesse. Die Kommunikation läuft heutzutage größtenteils mit digitalen Hilfsmitteln. Auch Geschäftsprozesse werden immer mehr Digital abgewickelt, soweit möglich. Um dies zu bewerkstelligen, müssen die Daten aus der Realen Welt allerdings erstmal digitalisiert werden. Viel ist in dieser Richtung geschehen, wie z.B. Kameras, die Digitalisierung ganzer Bibliotheken und das erfassen von Echtzeitdaten über Sensornetze sind nur ein Bruchteil. Doch das einlesen komplexer Daten ist bis heute ein nicht-triviales Problem.

Viele Arbeitsbereiche erfordern eine Speicherung und Übertragung von Daten, wie sie im echten Leben vorzufinden sind - z.B. Dreidimensionale Bilder. In der Medizin werden sie benutzt, um Knochen und Skelette platzsparend verfügbar zu haben, in der Automobilindustrie benötigt man sie, um Prototypen schon vor dem ersten tatsächlichen Fertigungsschritt vor Augen zu haben. Besonders die Spielindustrie steigert ihre Anforderungen nach immer besseren und realistischeren Modellen.

Für viele dieser Anforderungen gibt es eine einfach erscheinende Lösung. Man nehme ein reales Objekt und digitalisiere seinen dreidimensionalen Aufbau. Doch leider ist das Problem nicht so einfach, wie es sich zuerst anhört. Es gibt viele verschiedene Verfahren um diese Aufgabenstellung und jedes einzelne bewährt sich besonders in einem spezifischen Problemfeld. Ein Beispiel ist Motion Capture, das besonders für die Rekonstruktion von Bewegungen im Dreidimensionalen Raum über die Zeit genutzt wird, Laserscanner können schnell und zuverlässig Personen vermessen. Dieses Projekt beschäftigt sich gezielt mit der Rekonstruktion von digitalen Repräsentationen der Objekte über zweidimensionale Aufnahmen des gewünschten Gegenstandes oder einer kompletten Szene.

Die für den Prozess verwendeten Algorithmen sind jedoch sehr spezialisiert und decken meist nur einen bestimmten Anwendungsfall ab. Zudem lassen sich für die einzelnen Schritte auch unterschiedliche Algorithmen kombinieren, was die Zusammenstellung für ein perfektes Ergebnis umso schwerer macht. So macht es z.B. einen Unterschied, ob man eine Szene aus einem Videoclip rekonstruieren möchte, ob man einzelne Fotos aus verschiedenen Blickrichtungen hat oder ob es sich um eine Luftaufnahme handelt.

Ziel des 3D-MuVi Projekts ist es, die Kombination von Algorithmen und das damit verbundene Suchen nach dem bestmöglichen Ergebnis zu vereinfachen. Der Benutzer bekommt die Möglichkeit, verschiedene Algorithmen für die einzelnen Schritte auszuwählen und sich die Ergebnisse anzusehen, um somit das bestmögliche Resultat zu erzielen.

2 Zielbestimmung

Die Zielbestimmung definiert die Kriterien des Programms. Zum einen die Musskriterien, also die Funktionen die das Programm mindestens bereit stellen muss und zum anderen die Kannkriterien also mit welchen Funktionen das Programm sinnvoll erweitert werden könnte. Zusätzlich wird in den Abgrenzungskriterien explizit ausgeschlossen was das Programm nicht leisten muss. Die Muss- und Kannkriterien untergliedern sich jeweils in sechs logische Module des Programms. Bei diesen Modulen handelt es sich um:

1. **Workflow** Abstrahiert die Anordnung der Algorithmen.
2. **Input/Output** Eingabe, Übergabe, Speichern und Laden von Daten.
3. **Visualisierung** Art der Darstellung der Ergebnisse.
4. **Logger** Aufzeichnen von relevanten Ereignissen und Meldungen der Algorithmen.
5. **Einstellungen** Konfigurationsmöglichkeiten der Software.
6. **Interaktion** Bedienung und Rückmeldung des Programms.

3 Anforderungen

Anforderungen
  Kriterien

3.0.1 Abgrenzungskriterien

AK110\ Videos liegen als sortierte Einzelbilder vor
AK120\ Algorithmen müssen nicht Implementiert werden (Bereitstellung von Schnittstellen zur Einbindung)
AK130\ Eine Änderung der Eingabe erfordert eine erneute Ausführung der Algorithmen

3.1 Abgrenzungskriterien

- Videos liegen als sortierte Einzelbilder vor.
- Algorithmen müssen nicht implementiert werden.
- Eine Änderung der Eingabe erfordert eine erneute Ausführung der Algorithmen.
- Die Benutzeroberfläche beschränkt sich auf englisch Sprache.

3.2 Musskriterien

3.2.1 Modul: Workflow

- Die Software soll einen standardisierten 4-Phase-Workflow bereitstellen, bestehend aus:
  1. Feature Extraktion / Matching
  2. Posenschätzung
  3. Tiefenschätzung
  4. 3D Fusion
- Für jede Phase dieses Workflows sollen verschiedene Algorithmen ausgewählt werden können.

3.2.2 Modul: Input/Output

- Das Programm soll eine Menge von Einzelbildern einlesen und diese an die Algorithmen zur Bearbeitung übergeben können.
- Ergebnisse bzw. Zwischenergebnisse sollen gespeichert werden können.

3.2.3 Modul: Visualisierung

Darstellung der Ergebnisse:

- Features die durch die Algorithmen erkannt wurden, sollen in Einzelbildern eingezeichnet werden.
- Kameraposen und -orientierungen sollen im 3D Modell als Kamerapyramide eingezeichnet werden .
- Anzeigen der Tiefenkarten
- Anzeige des 3D Modells als Piont Cloud.

3.2.4 Modul: Logger

Aufzeichnen, Anzeigen und Abspeichern von Informationen, Warnungen, Fehlern und Debugmeldungen die von den Algorithmen ausgeworfen werden.

3.2.5 Modul: Einstellungen

Einstellungen der einzelnen Verfahren können abgespeichert und geladen werden.

3.2.6 Modul: Interaktion

- Der Workflow soll gestartet und gestoppt werden können.
  Das Stoppen der Verarbeitung bewirkt, dass keine Daten weitergegeben werden und lediglich die restliche Berechnung der Workflowstufe ausgeführt wird.
- Ein globaler Arbeitsindikator soll die andauernde Verarbeitung der Verfahren anzeigen.

3.3 Kannkriterien

3.3.1 Modul: Workflow

- Es sollen mehrere fest implementierte Workflows zu Verfügung stehen.
- Workflows können variable erstellt und verändert werden.
- Abspeichern und laden der Workflow-Konfiguration.

3.3.2 Modul: Visualisierung

- Manipulation und Interaktion mit (Zwischen-) Ergebnisse.
  (z.B. Ein-, Ausblenden und Entfernen einzelner Punkte, Kameras, Matches etc.).
- Darstellung des 3D Modells in Form von Point Cloud, Mesh und Texturiert.

3.3.3 Modul: Einstellungen

Abspeichern und laden von globalen Einstellungen.

3.3.4 Modul: Interaktion

- Einzelne Schritte sollen gestartet werden können.
- Laden vorhergegangener (Zwischen-) Ergebnisse.
- Automatisierte wiederholte Ausführung von einzelnen Algorithmen auf verschiedenen Datensätze.
- Auswahl der Workflow-Konfiguration und Verzeichnis mit Eingangsdaten durch Commandline-Optionen beim Programmstart.
- Daten neu Sortieren / Ausführen auf Untergruppen der Daten.
- Arbeitsindikatoren für jeden einzelnen Verarbeitungsschritt.

4 Produkteinsatz

4.1 Einsatzgebiete

Die Software dient zum Konfigurieren von Algorithmen, welche anschließend in einem Workflow kombiniert werden. Diese ermitteln aus einer Menge von Bildern ein 3D Modell, das von der Software dargestellt wird. Somit kann die Software in allen Gebieten eingesetzt werden, in denen Algorithmen und deren Konfigurationen auf eine gegebene Problemstellung angewendet und getestet werden sollen.

4.2 Zielgruppe

Die Software richtet sich an Entwickler von Algorithmen für Feature Extraktion / Matching, Posenschätzung, Tiefenschätzung und 3D Fusion. Diese können mithilfe der Software ihre Algorithmen testen und validieren. Im Allgemeinen kann jeder die Software nutzen, der sich bereits eingehend mit der Thematik befasst hat und ein Objekt mithilfe verschiedener Kombinationen von Algorithmen digitalisieren möchte.

4.3 Betriebsbedingungen

Die Software soll unter Bürobindungen laufen. Vorgesehen ist der Einsatz auf Linux, kann aber theoretisch auch auf Windows und Mac OS genutzt werden.

5 Produktumgebung

In diesem Kapitel werden die Schnittstellen, Hardware- und Softwarevoraussetzungen beschrieben.

5.1 Software

Das Programm soll auf linuxbasierten Betriebssystemen laufen, da es mit QT entwickelt und möglichst Plattform unabhängig entwickelt wird, kann es sein, dass es auch unter anderen Betriebssystemen läuft.

5.2 Hardware

Als Hardware wird ein einfacher PC benötigt, welcher die Sofwtarebedingungen erfüllt.

5.3 Schnittstellen

Das Programm soll mithilfe der Kommandozeile aufrufbar sein.

6 Produktfunktion

Die Beschreibung der Funktionalität gliedert sich in Grundfunktionen und optionale Funktionen.

6.1 Grundfunktionen

6.2 Funktional

6.2.1 Obligatorische Funktionen

Modul: Workflow

\FM110 Standardisierter 4-Phase-Workflow bestehend aus:
Einlesen von Bildern
1 Auswahl verschiedener Algorithmen für Feature Extraktion / Matching
2 Auswahl verschiedener Algorithmen für Posenschätzung
3 Auswahl verschiedener Algorithmen für Tiefenschätzung
4 3D Fusion
Fertiges Modell
sollte ausgewertet werden.

\FM120 Auswahl verschiedener Algorithmen für jede Phase

Modul: I/O

\FM210 Einlesen und Übergabe von Einzelbildern

\FM220 Abspeicherung der (Zwischen-) Ergebnisse

\FM230 Speicherort in verschiedenen Verzeichnissen: Ein Hauptverzeichnis, darunter für jede Phase ein Unterverzeichnis (Auswählbar)

Modul: Visualisierung

\FM310 Visualisierung der (Zwischen-) Ergebnisse

\FM320 Einzeichnen der Features in Einzelbildern

\FM330 Einzeichnen der Kamerposen und -orientierungen im 3D Modell (Kamerapyramide)

\FM340 Anzeigen der Tiefenkarten

\FM350 Anzeigen des 3D-Modells (PCL)

Modul: Logger

\FM410 Aufzeichnung, Anzeige und Abspeichern eines Logs zur Nachverfolgung von
1 Allgemeine Informationen
2 Warnungen
3 Fehlermeldungen
4 Debugging

Modul: Einstellung
\FM510 Abspeichern und Laden von Einstellungen der einzelnen Verfahren
Modul: Interaktion
\FM610 Starten und Stoppen der Verarbeitung
\FM620 Stopp => keine Weitergabe der Daten, aktuelle Verarbeitung läuft zu Ende.
\FM630 Globaler Arbeitsindikator

6.2.2 Optionale Funktionen

Modul: Visualisierung
\FK110 Manipulation und Interaktion mit (Zwischen-) Ergebnissen, wie beispielsweise Ein-, Ausblenden und Entfernen einzelner Punkte/Kameras/Matches, sollte gegeben sein
\FK120 Schalter zur Auswahl der Darstellung des 3D-Modells Point Cloud, Mesh, Texturiert Auch hier: Daten werden geliefert, lediglich Auswahl welche Daten angezeigt werden
Modul: Workflows
\FK210 Weitere (feste) Workflows
\FK220 Variable Workflow-Definition als Plugin
\FK230 Abspeichern und Laden der Workflow-Konfiguration
Modul: Interaktion
\FK310 Starten einzelner Schritte, Laden vorhergegangener (Zwischen-) Ergebnisse
\FK320 Automatisierte, wiederholte Ausführung von einzelnen Algorithmen auf verschiedene Datensätze
\FK330 Auswahl der Workflow-Konfiguration und Verzeichnis mit Eingangsdaten durch Command-Line-Optionen beim Programmstart (versteckte Ausführung)
\FK340 Daten neu Sortieren / Ausführung auf Untergruppe der Daten
\FK350 Arbeitsindikator für jeden Schritt
Modul: Einstellung
\FK410 Abspeichern und Laden von globale Einstellungen

6.2.3 Nichtunktionalität

\NF110 Entwicklung für Linux aber möglichst Plattform-unabhängig (Qt)
\NF120 Responsive Oberfläche während der Ausführung
\NF130 Läuft auf einzelnen Rechnern
\NF140 Zielgruppe sind durchschnittliche PC-Benutzer, die sich mit den Algorithmen und den damit verbundenen Workflows auskennen
\NF150 Folgen der Coding Conventions der Abteilung VID (siehe Anhang)
\NF160 Alle Namen und Kommentare im Quellcode in Englisch
\NF170 Quellcode Dokumentation in Doxygen
\NF180 Methoden und Klassen eine Dokumentation
\NF190 Gui-Sprache: Englisch

7 Ablaufplan

7.1 Ablaufplan

In diesem Kapitel wird der Ablaufplan des Projekts beschrieben

7.1.1 Phase 1: Pflichtenheft

Ziel: Erstellung des Pflichtenhefts
Phasenverantwortlicher: Laurenz Thiel
Abgabe: 29 November 2015
Artefakt: Pflichtenheft
Im Pflichtenheft werden die Anforderungen, Funktionen, die grobe Benutzerschnittstellen, Systemmodelle und Testszenarios für das Programm erstellt. Des weiteren werden Sachen wie die Entwicklungsumgebung und mögliche Hardware- und Softwarevoraussetzungen festgehalten. Das Pflichtenheft soll präzise, vollständig und konsistent sein.

7.1.2 Phase 2: Entwurf

Ziel: Erstellung des Entwurfs
Phasenverantwortlicher: Tim Brodbeck
Abgabe: 10 Januar 2016
Artefakt: UML-Diagramm des Entwurfs
Im Entwurf wird das Design der Software festgelegt. Es wird die Klassenstruktur, die Beziehungen zwischen den Klassen und die Klassenschnittstellen definiert. Des weiteren werden Klassendiagramme erstellt und zusätzlich können noch Sequenz- und Zustandsdiagramme erstellt werden. Es sollten Entwurfsmuster aus der Softwaretechnik zum Einsatz kommen. Es soll auf das Geheimnisprinzip, schwache Kopplung, hohe Kohäsion, Lokalitätsprinzip und Wiederverwendbarkeit von Klassen/Subsystemen geachtet werden

7.1.3 Phase 3: Implementierung

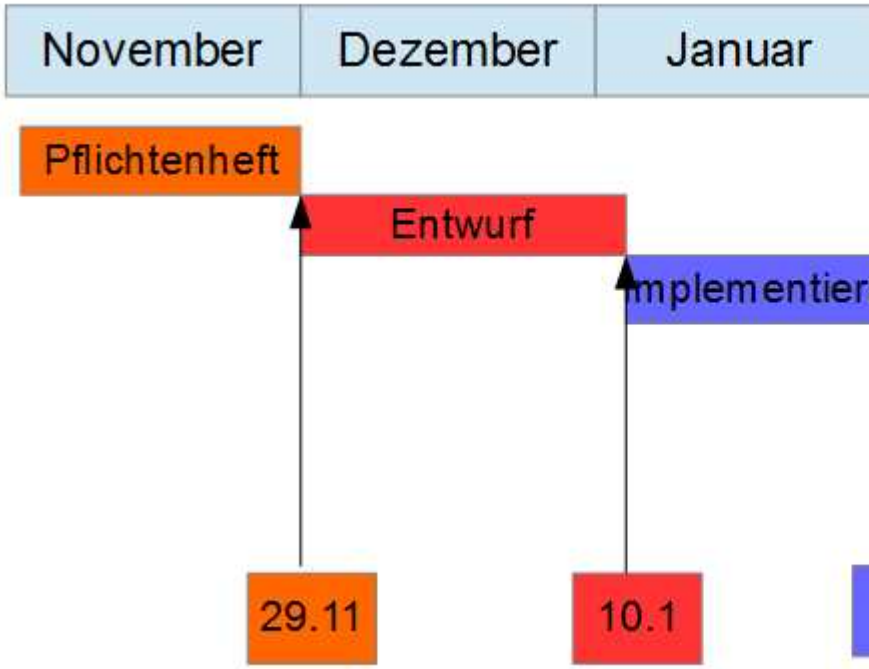
Ziel: Schreiben des Quellcodes
Phasenverantwortlicher: Jens Manig, Nathanael Schneider
Abgabe: 7 Februar 2016
Artefakt: Quellcode
In dieser Phase wird das Programm implementiert. Es werden die Klassen und deren Methoden aus der Entwurfsphase implementiert.

7.1.4 Phase 4: Qualitätssicherung

Ziel: Testen des Systems und schreiben von Testberichten
Phasenverantwortlicher: Stefan Wolf
Abgabe: 6 März 2016
Artefakt: Tests
Es werden verschiedene Tests(z.B. funktionale Komponententests, Überdeckungstests) durchgeführt und Testberichte geschrieben. Es werden ebenfalls die, im Pflichtenheft beschriebenen, Testfälle getestet. Ebenfalls wird die Robustheit des Systems überprüft.

7.1.5 Phase 5: Abschlusspräsentation

Ziel: Vorbereiten der Abschlusspräsentation
Phasenverantwortlicher: Grigori Schapoval
Abgabe: 20 März 2016
Artefakt: Präsentation
Es wird die Abschlusspräsentation vorbereitet, in der dass Produkt vorgestellt wird.



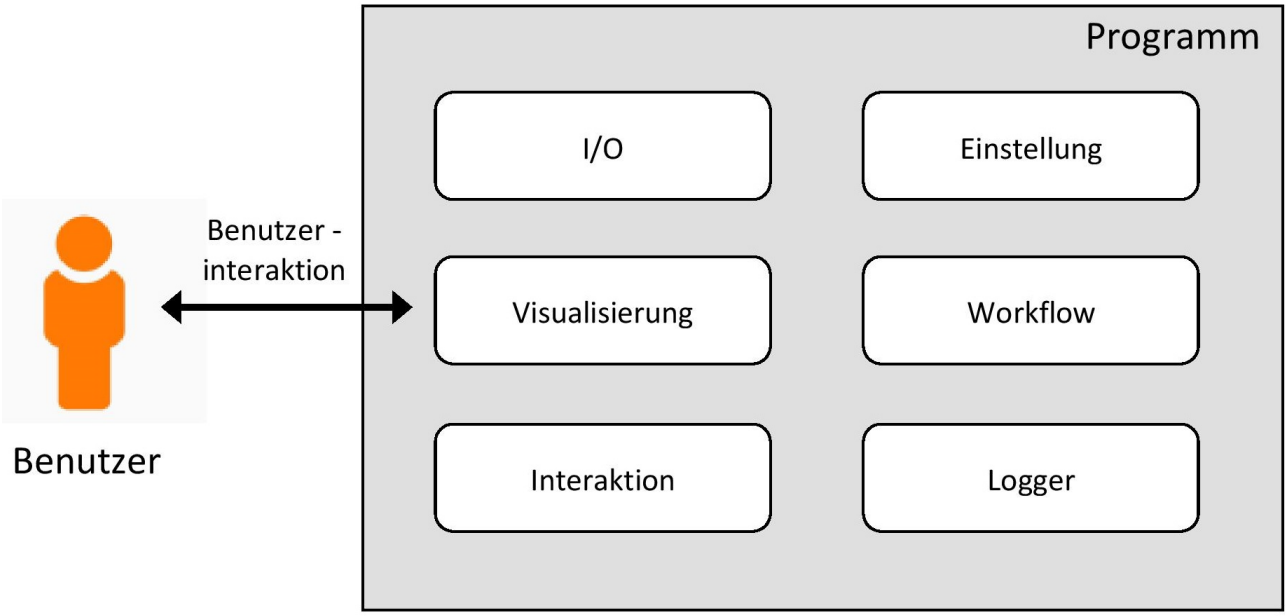
Hier der Ablauf als Diagramm

8 Systemmodell

8.1 Systemübersicht

Die Systemarchitektur wird durch einen Modularen Ansatz umgesetzt. Die einzelnen Module sind durch ihre Aufgabenbereiche getrennt. Bei den einzelnen Modulen wird intern zwischen Steuerung, Anzeige und Daten aufgeteilt. Es wird in folgende 6 Module aufgeteilt:

- Workflow
- I/O
- Visualisierung
- Logger
- Einstellungen
- Interaktion



8.2 Module

Im weiteren folgt eine kurze Definition der Module.

8.2.1 Workflow

Das Modul Workflow bietet mindestens einen Workflow an und bündelt alle workflowspezifischen Einstellungen und Auswahlmöglichkeiten, inklusive:

- Workflows wie den 4-Phasen-Workflow
- Speichern und laden von Wokflowkonfigurationen
- Algorithmenauswahl für jede Phase

8.2.2 I/O

Das Modul I/O bietet Funktionen für die Ein- und Ausgabedaten an, inklusive:

- Einlesen und übergeben von Einzelbilder
- Speichern von (Zwischen-) Ergebnissen

8.2.3 Visualisierung

Das Modul Visualisierung bietet Anzeigen der (Zwischen-) Endergebnissen an, inklusive:

- Anzeigen der Ausgabedaten der (Zwischen-) Ergebnisse
- optional, Manipulation der Ausgangsdaten der (Zwischen-) Ergebnisse

8.2.4 Logger

Das Modul Logger stellt Logs bereit, inklusive:

- Info, Warning, Error, Debug
- Anzeige und Abspeichern dieser Logs

8.2.5 Einstellung

Das Modul Einstellungen behandelt (globale) lokale Einstellungen, inklusive:

- Speichern und Laden von Einstellungen einzelner Verfahren

8.2.6 Interaktion

Das Modul Interaktion behandelt Benutzereingaben innerhalb des Programms, inklusive:

- Starten und Stoppen, Arbeitsindikator

8.3 Produktdaten

Im folgenden werden alle Daten gelistet, welche abgespeichert werden.

8.3.1 Daten der obligatorischen Funktionen

\PD1010 Zwischenergebnisse
\PD1020 Endergebnis
\PD1030 Parameter der einzelnen Algorithmen

8.3.2 Daten der optionalen Funktionen

\PD2010 Globale Einstellungen
\PD2020 Konfiguration des Workflows
\PD2030 Speichern weiterer Workflows

8.4 Schnittstellen

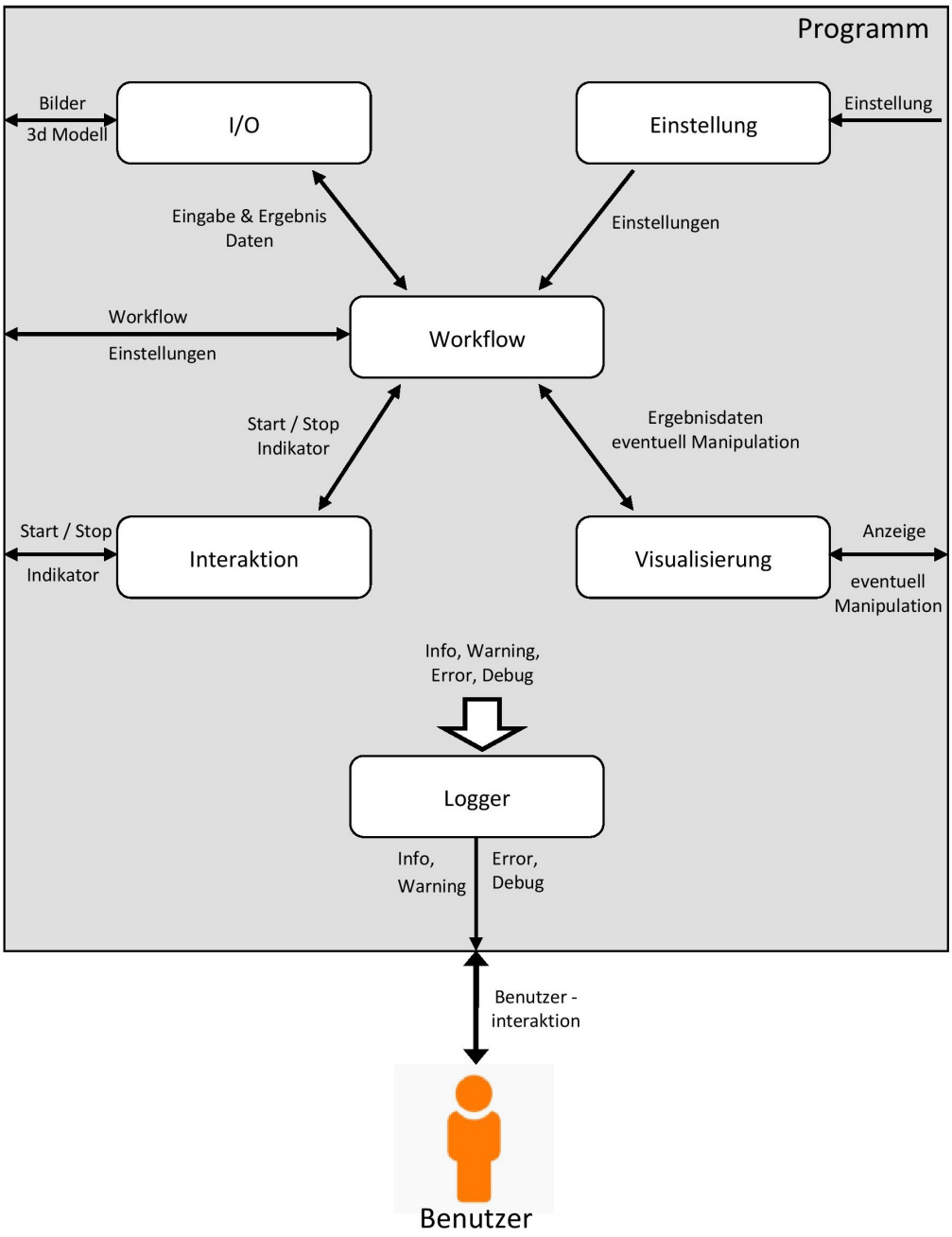
Hier werden die vom Programm bereitgestellten Schnittstellen beschrieben

\S110 (Funktionale-) Schnittstellen zum Aufrufen und Datenaustausch der Algorithmen

\S120 Schnittstelle zum Einstellen der Algorithmen Angabe der Config Datei und direkte Angabe eines Parameter-Trees

8.5 Datenflussübersicht

Dieses Diagramm stellt eine Grobübersicht über den Datenfluss zwischen den einzelnen Modulen dar. Diese sind nur als grobe Orientierung anzusehen und werden im Entwurf und in der Implementierung möglicherweise auf andere Weise umgesetzt.



8.6 Anwendugsfälle mit Sequenzdiagramm

8.6.1 Anwendugsfall: Rekonstruktion aus unzusammenhängenden Einzelbilder

Gegeben: Vielzahl an Einzelbilder einer Szene die nicht von der selben Kamera oder Position aufgenommen wurden. Ziel: Rekonstruktion der Szene oder eines einzelnen Objektes. Vorgehen:

- 1. Feature Detektor: In den Bildern wird automatisch nach „markanten“ Punkten gesucht, die hohen Wiedererkennungswert haben.
- 2. Feature Matching: Die gefundenen „markanten Punkte“ in zwei Bildern werden verglichen und einander zugeordnet.
- 3. Posenschätzung: Aus den Punktzuordnungen von zwei oder mehr Bildern werden die Positionen der Kameras ermittelt, von denen aus die Bilder aufgenommen wurden.
- 4. Tiefenschätzung: Mit Hilfe der Kamerapositionen werden die Tiefen in den Bildern trianguliert. Dadurch werden die Entfernungen der Objekte auf den Bildern bestimmt.
- 5. Modellerzeugung: Die Einzelansichten werden zu einem kompletten 3D Modell zusammengesetzt.

Stellschrauben:

- 1. Der Feature Detektor kann ausgewechselt werden (z. B. SIFT oder SURF)
- 2. Das Feature Matching kann auf verschiedene Arten durchgeführt werden (z. B. RANSAC oder MLESAC, jeweils mit verschiedener Initialisierungen)

8.6.2 Anwendugsfall: Structure-from-Motion (SfM)

Gegeben: Videosequenz einer Szene, mit eigenbewegung der Kamera. Ziel: Rekonstruktion der Szene oder eines einzelnen Objektes.

Structure-from-Motion (SfM) ist ein Verfahren zur 3D-Rekonstruktion mit nur einer Kamera. Dabei wird bei SfM das 3D-Modell aus einem Video, also einer Reihe von Einzelbildern rekonstruiert. Dabei ist es wichtig, dass das Video nicht von einer statischen Position aus aufgenommen wird, vielmehr sollte die Kamera sich dabei mit Blick auf die zu rekonstruierende Szene bewegen. Bei der Rekonstruktion mittels SfM werden zwei oder mehr Einzelbilder der Eingangssequenz herangezogen, für die angenommen wird, dass sie dieselbe Szene aus verschiedenen Blickrichtungen betrachten und die Szene zwischen den Einzelbildern statisch (unverändert) geblieben ist. Vorgehen:

- 1. Feature Detection: Ermitteln leicht verfolgbarer markanter Bildpunkte
- 2. Feature Tracking: Verfolgen der markanten Bildpunkte
- 3. Posenschätzung: Bestimmen der Aufnahmepunkte, Verfolgen der Kamerafahrt
- 4. Tiefenschätzung: Bestimmen der Tiefe mit lokalen Optimierungsverfahren
- 5. Modellerzeugung: Die Einzelansichten werden zu einem kompletten 3D Modell zusammengesetzt

8.7 Typische Nutzungsabläufe

In diesem Abschnitt werden typische Abläufe von Aktionen dargestellt, die ein Anwender mit der Software durchführt.

8.7.1 Laden von Einzelbildern

Vorbedingung:
Das Programm wurde gestartet
Ablauf:
1. Auswählen der "Lade Bilder" im Menü
2. Suchen und Öffnen eines oder mehrerer Bilder

8.7.2 Auswählen von Algorithmen

Vorbedingung:
Das Programm wurde gestartet

Ablauf:
1. Auswahl der Algorithmen für die einzelnen Schritte im rechten Bereich der GUI

8.7.3 Ändern von Einstellungen:

Vorbedingung:
Es wurden für die einzelnen Schritte entsprechende Algorithmen ausgewählt.
Ablauf:
1. Im linken Bereich der GUI können die einzelnen Einstellungen für die Algorithmen geändert werden.

8.7.4 Speichern der Einstellungen

Vorbedingung:
Es wurden Algorithmen für die einzelnen Schritte ausgewählt
Ablauf:
1. Der Benutzer wählt im Menü die Option "Speichern"
2. Es wird eine Zieldatei gewählt, in die das Setup gespeichert werden soll
3. Durch Klick auf "Speichern" werden die selektierten Algorithmen und deren Einstellungen gespeichert.

8.7.5 Laden von Einstellungen

Vorbedingung:
Das Programm wurde gestartet
Ablauf:
1. Der Benutzer wählt im Menü die Option "Laden"

2. Im folgenden Dialog sucht der Nutzer eine passende Datei
3. Durch Klick auf "Öffnen" werden die Algorithmen und ihre Einstellungen geladen

8.7.6 Öffnen der Hilfe

Vorbedingung:
Das Programm wurde gestartet
Ablauf:
1. Der Benutzer wählt im Menü "Hilfe" aus
2. Es öffnet sich ein neues Fenster, welches die Hilfe anzeigt

8.7.7 Filtern des Logs

Vorbedingung:
Das Logfenster zeigt bereits ein paar Nachrichten an
Ablauf:
1. Der Benutzer wählt ein oder mehrere Loglevel aus
2. Es werden nur noch Nachrichten vom entsprechenden Level angezeigt

8.7.8 Ausführen von Algorithmen

Vorbedingung:
Es wurden Algorithmen und Bilder ausgewählt und geladen.
Ablauf:
1. Der Benutzer wählt "Start"
2. Die Abarbeitung der Algorithmen auf den Eingabedaten beginnt nun

8.7.9 Unterbrechen der Verarbeitung

Vorbedingung:
Die Verarbeitung wurde erfolgreich gestartet
Ablauf:
1. Durch die Anwahl von "Stop" wird das weiterreichen der Daten angehalten
2. Nach dem Abschluss des aktuell laufenden Algorithmus kommt die Abarbeitung zum Stillstand

8.7.10 Vorschau der Ergebnisse

Vorbedingung:
Die Algorithmen wurden erfolgreich ausgeführt
Ablauf:
1. Durch Auswählen der einzelnen Vorschaumodi in der Mitte der GUI wählt der Benutzer die Daten, die er sehen möchte.
2. Der Platz darunter wird zu einer Visualisierung der Daten, die ggf. auch manipuliert werden können.

9 Produktleistungen

9.1 Laufzeitverhalten

Das Konfigurieren und Vorbereiten der Software auf das Ausführen der Algorithmen und das Anzeigen der Ergebnisse erfolgt in Echtzeit. Verzögerungen entstehen lediglich beim Ausführen des Workflows und werden dabei maßgeblich von den verwendeten Algorithmen vorgegeben.

9.2 Speicherplatzbedarf

Für die Anwendung wird kaum Festplattenspeicher benötigt, da nur die Einstellungen der Verfahren lokale gespeichert werden. Der Hauptspeicher wird der Anwendung selbst kaum benötigt und hauptsächlich von den als Plugins implementierten Algorithmen in Anspruch genommen.

10 Benutzeroberfläche

In diesem Kapitel werden die Anforderungen an die Benutzeroberfläche genannt und beschrieben. Außerdem wird ein möglicher Entwurf dargestellt, der die Anforderungen erfüllt. Gegliedert ist das Kapitel dabei in die Anforderungen, die in jedem Fall erfüllt werden müssen, und jene, die je nach Möglichkeit umgesetzt werden.

10.1 Hauptfenster

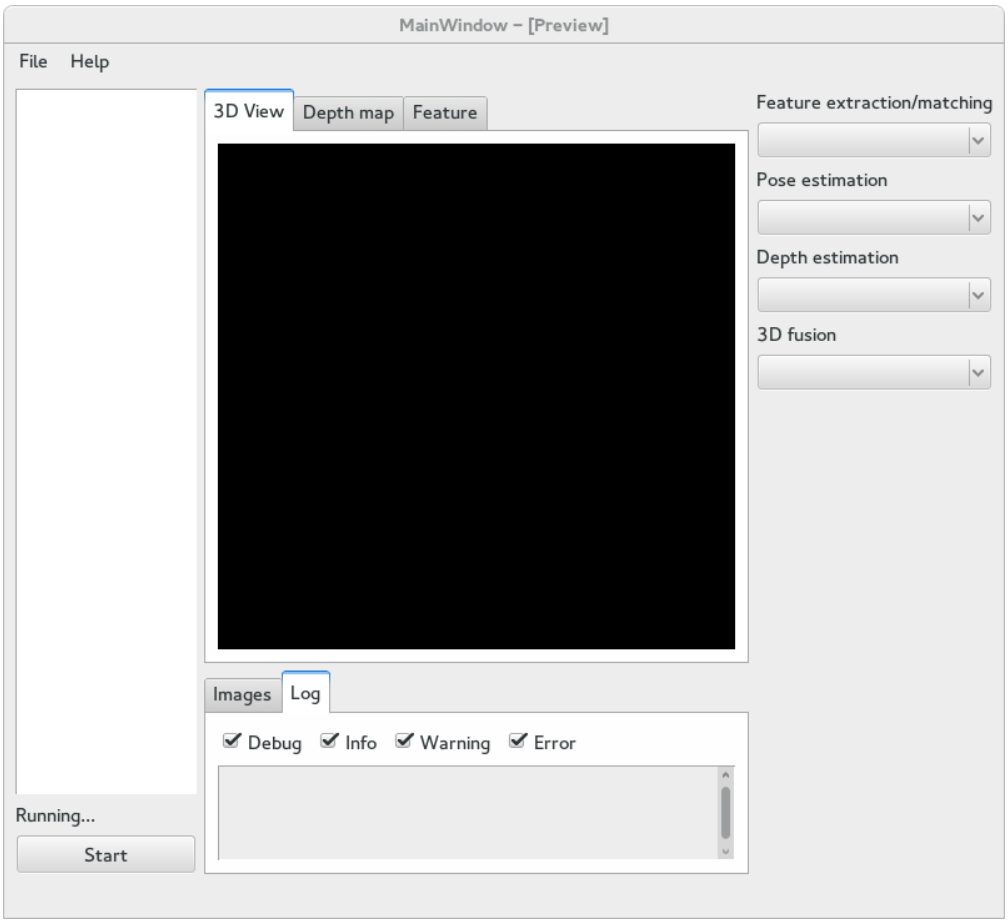


Abbildung 10.1: Hauptfenster

\B1010\ Algorithmenauswahl Algorithmen können dem Workflow entsprechend aus ComboBoxen gewählt werden.

\B1020\ Bilderinput Über den Eintrag „File → Load images...“ in der Menüleiste können Bilder als Eingabe der Algorithmen ausgewählt werden.

\B1030\ Bildervorschau Die Bilder, die als Eingabe für die Algorithmen gewählt worden sind, werden in einer Vorschau als Thumbnails angezeigt. In der Vorschau können einzelne oder mehrere Bilder ausgewählt werden, so dass sie in **\B1080\c** dargestellt werden.

\B1040\ Starten Ein Button soll den Start der Ausführung der ausgewählten Algorithmen ermöglichen.

\B1050\ Arbeitsindikator Ein globaler Arbeitsindikator in der Form eines Symbols oder Texts zeigt dem Nutzer an, ob gerade Algorithmen in Ausführung sind.

\B1060\ Stoppen Über ein Button kann der Nutzer die Ausführung der Algorithmen anhalten.

\B1070\ Ergebnisanzeige Die Ergebnisse werden nach Möglichkeit zentral und groß dargestellt.

\B1080\ Ansichtsauswahl Der Nutzer soll zwischen folgenden Ansichten wählen können:

- **a** 3D-Ansicht
- **b** Tiefenkarte
- **c** Features auf einem Bild, falls in **\B1030** ein Bild gewählt ist bzw. des Matchings von Features auf mehreren Bildern, falls mehrere Bilder gewählt sind

\B1090\ Protokoll Dem Benutzer soll ein Protokoll über die Ausführung der Algorithmen bereit gestellt werden. Die verschiedenen Arten vom Protokollmeldungen (Fehler, Warnung, Info, Debug) sollen einzeln ein- und ausgeblendet werden können.

\B10100\ Speichern des Workflows Die Algorithmenauswahl und der gewählte Workflows, falls vorhanden, sollen gespeichert werden können. Dafür stehen in der Menüleiste die Einträge „File → Save workflow“ und „File → Save worflow As...“ bereit.

\B10110\ Speichern der Ergebnisse Die Ausgabedaten der Algorithmen können über die Einträge „File → Save results“ und „File → Save results As...“ in der Menüleiste gespeichert werden.

10.2 Erweitertes Hauptfenster zur Umsetzung von Kann-Kriterien

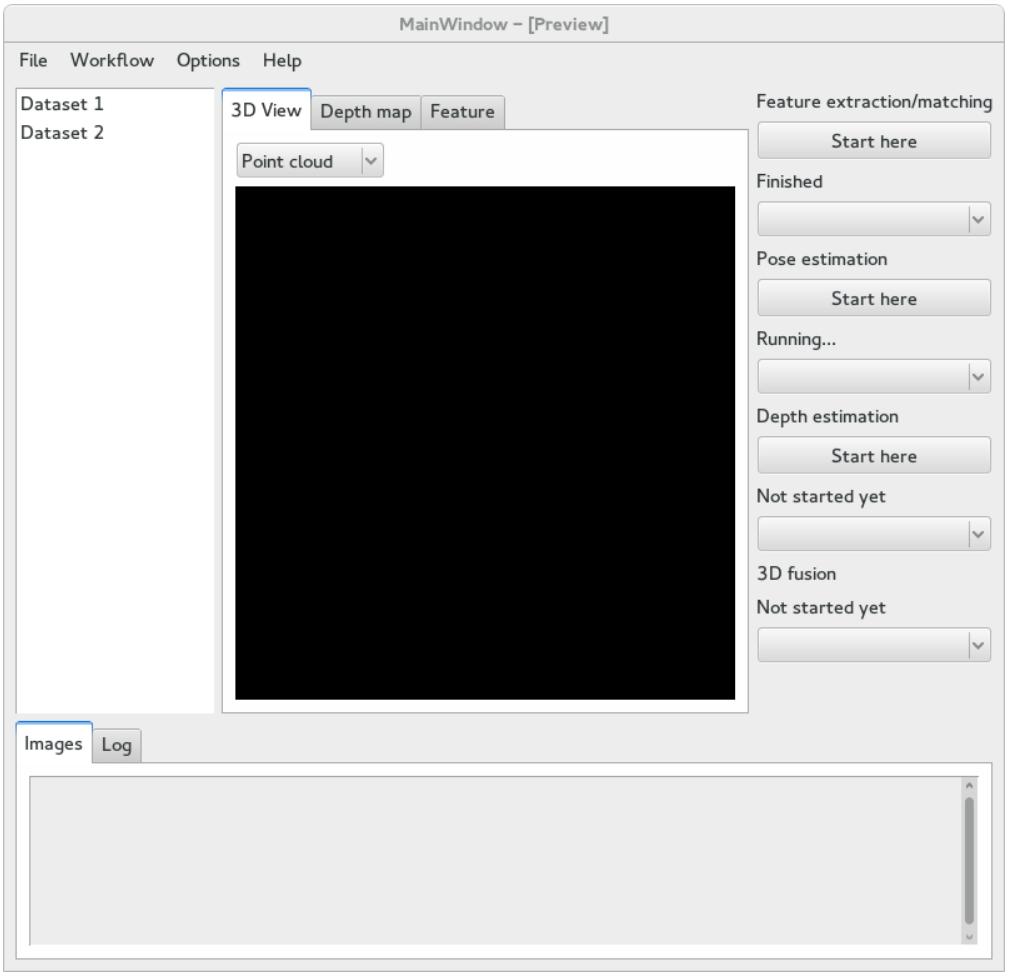


Abbildung 10.2: Erweitertes Hauptfenster

Hinzugekommen ist:

\B2010\ **Workflowauswahl** Dem Anwender wird eine Auswahl an Workflows in der Menüleiste unter dem Punkt „Workflow“ angeboten. Ein Workflow besteht aus verschiedenen Schritten, die festlegen, welche Algorithmen für einen Schritt ausgewählt werden kann.

\B2020\ **Laden von Ausgabedaten der Algorithmen** Über den Eintrag „File → Advanced load files...“ können die Ausgabedaten von vorherigen Läufen der Algorithmen geladen werden.

\B2030\ **Starten ab einem bestimmten Algorithmus** Für jeden Algorithmus, der die Vorbedingungen erfüllt, gibt es einen Button, der es dem Nutzer ermöglicht die-

sen Algorithmus und alle nachfolgenden Algorithmen auszuführen. Vorbedingung heißt, dass dem Algorithmus alle Daten zur Ausführung vorliegen. Diese Daten stammen entweder aus vergangen Läufen der vorhergehenden Algorithmen in der gleichen Sitzung oder wurden vom Nutzer manuell durch **\B2020** geladen. Die Implementierung dieses Punktes ersetzt **\B1040**.

\B2040\ Auswahl der 3D-Darstellung Mittels einer ComboBox kann der Nutzer die 3D-Darstellung zwischen folgenden Ansichten umschalten:

- **a Point cloud** zeigt die einzelnen Eckpunkte des 3D-Modells an.
- **b Mesh** stellt die Oberfläche des 3D-Modells dar.
- **c Textured** stellt das Mesh [**\B2040\b**] inklusive der auf die Oberfläche gezeichneten Texturen da.

\B2050\ Mehrere Datensätze Über eine Liste können mehrere Datensätze innerhalb einer Sitzung genutzt und verwaltet werden. Der Nutzer wird so in die Lage versetzt einen Workflow auf verschiedenen Eingabedaten zu evaluieren, ohne für jeden Test die Daten neu laden zu müssen.

\B2060\ Arbeitsindikator pro Algorithmus Pro Schritt eines Workflows stellt ein Symbol oder ein Text dar, ob der für den Schritt gewählte Algorithmus gerade ausgeführt wird. Die Implementierung dieses Punktes ersetzt **\B1050**.

\B2070\ Globale Einstellungen Über den Menüpunkt „Options → Settings...“ können globale Einstellungen gesetzt werden, falls sich solche im Laufe des Projektes ergeben. Globale Einstellungen sind Einstellungen, die für die gesamte Anwendung gelten und über das Ende der Sitzung gültig bleiben.

11 Testfälle und Testszenarien

Die Testfälle und Testszenarien dienen der Überprüfung der Software bezüglich ihrer Funktionalität und Konsistenz. Testfälle unterscheiden sich von den Testszenarien darin, dass diese automatisierte, somit atomare Funktionstests darstellen, während Testszenarien manuelle Testanweisungen sind, die durch eine ungeschulte Testperson durchgeführt werden sollen.

11.1 Testfälle

Die Testfälle sind alle Automatisiert und werden mit der in Qt integrierten Test-Suite durchgeführt.

11.1.1 Funktionstests

Zuerst wird jede Klasse einzeln bearbeitet. Jede Öffentliche Methode wird mit zwei Testfällen getestet. Der erste Testfall testet das Verhalten einer Funktion im normalen Umfeld mit gewöhnlichen Parametern. Dieser Test dient dazu, Fehler in Berechnungsvorschriften und Implementierungen zu finden. Der zweite Testfall ruft die Funktion mit ungewöhnlichen Parametern auf, eventuell auch mit falschen Datentypen. Diese Tests dienen dazu, die Robustheit der Funktionen zu garantieren. Die Argumente, die an eine Funktion übergeben werden, sind so weit wie möglich Zufällig verteilt zu wählen. Für das nachvollziehen von Fehlern werden die getesteten Parameter zusätzlich zu der Fehlerbeschreibung ausgegeben.

11.1.2 Komponententests

Komponententests testen das Verhalten eines isolierten Moduls der Anwendung. Viele der Tests lassen sich allerdings erst nach der Entwurfsphase festlegen, weshalb dieser Abschnitt bis zum Vollständigen Entwurf der Anwendung noch nicht vollständig sein wird.

\T2.10

Modul	Funktion
I\O	FM210

Ziel:

Testen der Funktionalität Laden und Weitergeben von Eingabebildern.

Dummies:

- Algorithmus, der die empfangenen Bilder an den Test meldet.
- Interaktion, die das Modul mit vordefinierten Bildern versorgt.

Erwartetes Verhalten:

Die vom Algorithmus gemeldeten Bilder entsprechen den Bildern, die dem Modul übergeben wurden.

\T2.20

Modul	Funktion
I\O	FM220 & FM230

Ziel:

Testen der Fähigkeit, Zwischenergebnisse strukturiert abzuspeichern.

Dummies:

- Algorithmen, die Zufällig verteilte Zwischenergebnisse liefern, deren Art und Typ nicht weiter Definiert ist.

Erwartetes Verhalten:

Es wird eine Ordnerstruktur in dem vom Test vorgegebenen Verzeichnis angelegt mit den gespeicherten Daten.

\T2.30

Modul	Funktion
Logger	FM410

Ziel:

Testen der Logfunktionalität

Dummies:

- Generator von Lognachrichten

Erwartetes Verhalten:

Nach dem Anlegen der Logs und dem auswählen der einzelnen Loglevel werden nur Nachrichten des gewählten Loglevels zurückgegeben.

\T2.40

Modul	Funktion
Logger	FM410

Ziel:

Testen der Speicherfunktionalität

Dummies:

- Generator von Lognachrichten

Erwartetes Verhalten:

Nach dem Anlegen der Logs und dem Abspeichern wird eine Datei an dem spezifizierten Pfad vorgefunden.

\T2.50

Modul	Funktion
Einstellungen	FM510

Ziel:

Test der Einstellungen für Algorithmen

Dummies:

- Algorithmus, der eine Liste von Einstellungen bereitstellt, welche alle Datentypen abdeckt und bei Variationsmöglichkeiten der Typen bzw. Wertebereiche diese Stichprobenartig vorhanden sind.

Erwartetes Verhalten:

Nach dem Abspeichern findet sich eine Datei an dem spezifizierten Pfad. Nach dem Laden dieser Datei enthält ein neu Instantiierter Algorithmus dieselben

Einstellungen wie der erste Algorithmus, welcher zur Speicherung verwendet wurde.

\T2.60

Modul	Funktion
Einstellungen	FK410

Ziel:

Test der Einstellungen für Globale Parameter

Dummies:

- Keine

Erwartetes Verhalten:

Nach dem Sichern globaler Einstellungen findet sich eine Datei an dem spezifizierten Pfad. Nach dem Lesen dieser Datei in eine neue Einstellungs-Instanz enthält diese dieselben Daten wie die Instanz, die zur Speicherung verwendet wurde.

\T2.70

Modul	Funktion
Workflows	FK230

Ziel:

Testen des Abspeicherns von Workflow-Konfigurationen

Dummies:

- Keine

Erwartetes Verhalten:

Nach dem zufälligen Zusammenbau eines Workflows kann dieser in eine Datei gesichert werden und entspricht nach dem laden dem gesicherten Workflow.

\T2.80

Modul	Funktion
Interaktion	FK310

Ziel:

Testen des Ladens von Zwischenergebnissen

Hinweis:

Wenn die Funktionalität implementiert wird, muss dieser Test mit dem Test \T2.20\ zusammengeführt werden.

Dummies (Ergänzung):

- Ein Algorithmus, der die Ausgabe des ersten Algorithmus als Eingabe nimmt und ihre Validität prüft.

Erwartetes Verhalten (Ergänzung):

Nach dem Laden der Zwischenergebnisse kann der zweite Algorithmus die Daten erfolgreich Validieren.

\T2.90

Modul	Funktion
Foo	Bar

Platzhalter für weitere Komponententests

11.1.3 Oberflächentests

Die Oberflächentests lassen sich mit der Qt Test-Suite ebenfalls automatisieren, eventuell muss allerdings bei der Designphase daran gedacht werden, gerade was das Laden von Dateien angeht.

\T3.10\ Tab Log

Dummies:

- Nachrichtengenerator, der für jedes Loglevel eine Nachricht in den Log schreibt

Ablauf:

Der Nachrichtendummy schreibt für jeden Loglevel eine Nachricht in den Log. Danach wird jeder Loglevel einzeln ausgewählt und die Textbox auf die erwartete Lognachricht überprüft.

\T3.20\ Tab Images

Ablauf:

Der Test lädt über die GUI Bilder in das Programm und prüft dann im Tab Images, ob die erwartete Anzahl an Vorschaubildern vorhanden ist.

Hinweis:

Das Menü "Load Images" kann nicht direkt getestet werden, da ein Dateidialog geöffnet wird. Stadtessen übergeht der Test den Dialog und ruft direkt die Lade-Funktion auf. Die GUI-Klasse muss entsprechend geplant werden.

\T3.30\ Algorithmenwahl

Dummies:

- Algorithmus, der als auffindbarer Platzhalter dient

Ablauf:

Es wird für jeden Schritt ein Algorithmus aus der ComboBox ausgewählt. Danach wird im Backend überprüft, ob der Algorithmus auch an die Kontrollschicht weitergegeben wurde und ob es der richtige ist.

\T3.40\ Ausführung der Algorithmen

Dummies:

- Ein Algorithmus für jeden Verarbeitungsschritt, der etwa zwei Sekunden wartet. Daten sind egal.

Ablauf:

Im Backend werden die Dummy-Algorithmen aktiviert. In der GUI wird dann der Button "Start" gedrückt. Nun müssen alle Interaktiven Elemente, die Einfluss auf die Verarbeitung nehmen, deaktiviert sein. Bekannt sind die Einstellungen für den Algorithmus, die Bilderliste und die Auswahl der Algorithmen. Als nächstes wird geprüft, ob der Arbeitsindikator des ersten und zweiten Algorithmus funktionieren. Ist die Ausführung beim dritten Algorithmus angelangt, wird geprüft, ob die ersten beiden Schritte ihren Zustand in "Finished" geändert

haben. Zudem wird die Ausführung angehalten. Sobald auch der dritte Algorithmus den Zustand "Finished" eingenommen hat, wird geprüft, ob der vierte Algorithmus wie gewünscht nicht gestartet wurde. Zuletzt wird noch geprüft, ob die Oberfläche wieder Interaktiv geschaltet wurde.

\T3.50\ Einstellungen

Dummies:

- Ein Algorithmus mit verschiedenen Einstellungen und Validierungen, die dem Test bekannt sind.

Ablauf:

Der Algorithmus wird im Backend aktiviert. Der Test setzt nun in der Oberfläche Werte für die verschiedenen Einstellungen. Es wird geprüft, ob die Werte in den Algorithmus übernommen werden. Danach werden Werte gewählt, die zu Validierungsfehlern führen. Es wird geprüft, ob die Werte im Algorithmus fälschlicherweise geändert wurden. Zudem wird das entsprechende Feld auf eine Visuelle Signalisierung des Fehlers geprüft.

11.2 Testszenarien

Die Testszenarien orientieren sich an den in Kapitel 8.7: Typische Nutzungsabläufe definierten Abläufen. Die Testszenarien sind von einem Benutzer durchzuführen, der idealerweise wenig bis gar keine Kenntnis über das Programm hat.

- **8.7.1: Laden von Einzelbildern**
  Vorbedingung:
  Das Programm wurde gestartet
  Ablauf:
  1. Auswählen der "Lade Bilder" im Menü
  2. Suchen und Öffnen eines oder mehrerer Bilder

  Nachbedingung:
  Im Feld "Images" erscheint eine Vorschau der gewählten Bilder
- **8.7.2: Auswählen von Algorithmen**
  Vorbedingung:
  Das Programm wurde gestartet

  Ablauf:
  1. Auswahl der Algorithmen für die einzelnen Schritte im rechten Bereich der GUI

  Nachbedingung:
  Die Ansicht in der GUI repräsentiert die ausgewählten Algorithmen.
- **8.7.3: Ändern von Einstellungen:**
  Vorbedingung:
  Es wurden für die einzelnen Schritte entsprechende Algorithmen ausgewählt.
  Ablauf:

1. Im linken Bereich der GUI können die einzelnen Einstellungen für die Algorithmen geändert werden.

Nachbedingung:
Die GUI repräsentiert die geänderten Einstellungen und die Algorithmen übernehmen die Änderungen. Letzteres ist nur prüfbar, wenn die Änderungen erhebliche Auswirkungen in der Visualisierung nach sich ziehen.

- **8.7.4: Speichern der Einstellungen**
  Vorbedingung:
  Es wurden Algorithmen für die einzelnen Schritte ausgewählt
  Ablauf:
  1. Der Benutzer wählt im Menü die Option "Speichern"
  2. Es wird eine Zieldatei gewählt, in die das Setup gespeichert werden soll
  3. Durch Klick auf "Speichern" werden die selektierten Algorithmen und deren Einstellungen gespeichert.

  Nachbedingung:
  Es wird eine Datei an dem gewählten Ort angelegt und nach dem Laden der Einstellungen werden Algorithmen und Einstellungen so übernommen, wie sie abgespeichert wurden.
- **8.7.5: Laden von Einstellungen**
  Vorbedingung:
  Das Programm wurde gestartet
  Ablauf:
  1. Der Benutzer wählt im Menü die Option "Laden"
  2. Im folgenden Dialog sucht der Nutzer eine passende Datei
  3. Durch Klick auf "Öffnen" werden die Algorithmen und ihre Einstellungen geladen

  Nachbedingung:
  Nach dem Laden der Datei sind die Algorithmen und Einstellungen so gesetzt, wie sie beim Abspeichern waren.
- **8.7.6: Öffnen der Hilfe**
  Vorbedingung:
  Das Programm wurde gestartet
  Ablauf:
  1. Der Benutzer wählt im Menü "Hilfe" aus
  2. Es öffnet sich ein neues Fenster, welches die Hilfe anzeigt

  Nachbedingung:
  Der Benutzer sieht ein Fenster, in dem er Anleitung und Hilfe zu dem Programm und seiner Benutzung findet
- **8.7.7: Filtern des Logs**
  Vorbedingung:
  Das Logfenster zeigt bereits ein paar Nachrichten an
  Ablauf:
  1. Der Benutzer wählt ein oder mehrere Loglevel aus
  2. Es werden nur noch Nachrichten vom entsprechenden Level angezeigt

Nachbedingung:
Die Nachrichten im Logfenster sind nur noch von den ausgewählten Schweregraden.

- **8.7.8: Ausführen von Algorithmen**
  Vorbedingung:
  Es wurden Algorithmen und Bilder ausgewählt und geladen.
  Ablauf:
  1. Der Benutzer wählt "Start"
  2. Die Abarbeitung der Algorithmen auf den Eingabedaten beginnt nun

  Nachbedingung:
  Die Ausführung der Algorithmen wurde erfolgreich beendet und die Ergebnisse können angesehen und manipuliert werden. Es wird in dem eingestellten Verzeichnis eine Ordnerstruktur mit den Zwischenergebnissen angelegt.
- **8.7.9: Unterbrechen der Verarbeitung**
  Vorbedingung:
  Die Verarbeitung wurde erfolgreich gestartet
  Ablauf:
  1. Durch die Anwahl von "Stop" wird das weiterreichen der Daten angehalten
  2. Nach dem Abschluss des aktuell laufenden Algorithmus kommt die Abarbeitung zum Stillstand

  Nachbedingung:
  Es liegen Anzeigedaten bis zum letzten ausgeführten Algorithmus vor. Das eingestellte Verzeichnis enthält Zwischenergebnisse bis zum letzten ausgeführten Algorithmus.
- **8.7.10: Vorschau der Ergebnisse**
  Vorbedingung:
  Die Algorithmen wurden erfolgreich ausgeführt
  Ablauf:
  1. Durch Auswählen der einzelnen Vorschaumodi in der Mitte der GUI wählt der Benutzer die Daten, die er sehen möchte.
  2. Der Platz darunter wird zu einer Visualisierung der Daten, die ggf. auch manipuliert werden können.

  Nachbedingung:
  Nach Anwahl einer Ansicht werden die entsprechenden Ergebnisse dort Visualisiert.

12 Qualitätszielbestimmungen

Durch begrenzte zeitliche und widersprüchliche Anforderungen der Qualitätsziele können nicht alle Qualitätsziele vollständig erfüllt werden. Im Folgenden Kapitel wird die Priorisierung der Qualitätsziele vorgestellt. Die Hauptqualitätsmerkmale sind Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit.

12.1 Übersicht

Es folgt eine detaillierte Tabelle der Gewichtung aller Qualitätsziele. Zur Erläuterung dieser werden alle Qualitätsziele und deren Bedeutungen im Kontext des Programmes vorgestellt.

Funktionalität

- Richtigkeit - Korrektheit der Zwischen und Endergebnisse bei korrekten Algorithmen
- Interoperabilität - Direkte Zusammenarbeit des Programms mit anderen Programmen
- Ordnungsgenmäßigkeit - Plangemäße Ausführung der Funktionen ohne Fehler
- Sicherheit - Kryptische Sicherheit des Programms

Zuverlässigkeit

- Reife - Bugfreiheit der Software
- Fehlertoleranz - Zuverlässigkeit bei Fehlern in Plugins
- Robustheit - Zuverlässigkeit bei falschen Benutzereingaben und Daten
- Wiederherstellbarkeit - Wiederherstellung des Programms nach starken Fehlern

Benutzbarkeit

- Verständlichkeit - Verständnis und Anschaulichkeit für Erstbenutzer
- Erlernbarkeit - Geringer zeitlicher Aufwand zur Einarbeitung
- Bedienbarkeit - Anzahl an Interaktionen um eine gewünschte Funktion durchzuführen

Effizienz

- Zeitverhalten - Zeitdauer um Funktionen auszuführen (maßgeblich von Algorithmenplugins abhängig)
- Verbrauchsverhalten - Ressourcenverbrauch um Funktionen auszuführen (maßgeblich von Algorithmenplugins abhängig)

Änderbarkeit

Änderbarkeit wird im Kontext zum 3dMuVi-Programm hauptsächlich auf die Pluginstruktur bezogen.

- Analysierbarkeit - Verständlichkeit und Dokumentation der Plugin-Schnittstellen
- Modifizierbarkeit - Modfizierbarkeit mit Hilfe von Plugins zu den gebotenen Schnittstellen
- Stabilität - Stabilität nach Anpassungen mittels funktionstüchtigen Plugins
- Prüfbarkeit - Testbarkeit nach Anpassungen mittels funktionstüchtigen Plugins

Übertragbarkeit

- Installierbarkeit - Installieraufwand für Systeme, die Hardware- und Softwareanforderungen erfüllen
- Anpassbarkeit - Mögliche spätere Anpassung auf andere Systeme
- Austauschbarkeit - Austauschbarkeit der Software durch andere Software

Priorisierung	sehr wichtig	wichtig	normal	weniger wichtig
Funktionalität				
Richtigkeit	**X**			
Interoperabilität				**X**
Ordnungsgemäßigkeit		**X**		
Sicherheit				**X**
Zuverlässigkeit				
Reife		**X**		
Fehlertoleranz			**X**	
Robustheit		**X**		
Wiederherstellbarkeit			**X**	
Benutzbarkeit				
Verständlichkeit			**X**	
Erlernbarkeit			**X**	
Bedienbarkeit		**X**		
Effizienz				
Zeitverhalten			**X**	
Verbrauchsverhalten			**X**	
Änderbarkeit				
Analysierbarkeit	**X**			
Modifizierbarkeit	**X**			
Stabilität		**X**		
Prüfbarkeit		**X**		
Übertragbarkeit				
Installierbarkeit		**X**		
Anpassbarkeit			**X**	
Austauschbarkeit				**X**

12.2 Zusammenfassung

- Der Hautfokus der Software liegt auf korrekter Funktionsweise und Erweiterungen mit Hilfe von Plugins (Hauptsächlich weitere Algorithmen und Workflows).
- wichtige Qualitätsmerkmale sind gute Bedienbarkeit bei fachlichem Verständnis und Zuverlässigkeit, solange funktionstüchtige Algorithmen verwendet werden. Außerdem soll das Programm ohne großen Aufwand auf einem Linux-System installierbar sein.
- Die Effizienz des Programms ist kaum beeinflussbar, da sie hauptsächlich von den verwendeten Algorithmen und Parametern dieser abhängt.
- Sicherheit, Interoperabilität und Austauschbarkeit werden als weniger wichtig erachtet, da das Programm eigenständig und auf einzelnen Computern ausgeführt wird.

13 Entwicklungsumgebung

Die Entwicklungsumgebung beschreibt die Werkzeuge und Hilfsmittel, die im Verlauf des Projektes verwendet werden. Dabei erfolgt die Gliederung entweder nach Verwendung der Werkzeuge in einzelnen Phasen oder dem Verwendungszweck bei Werkzeugen, die über mehrere Phasen hinweg verwendet werden.

13.1 Allgemein

- LaTeX zur Erstellung von Dokumenten
- TeXstudio zum Schreiben der LaTeX-Dokumente (Teammitglieder können, falls gewünscht, davon abweichen, da die LaTeX-Datei unabhängig von dem Editor ist.)
- Git als Versionskontrolle
- Der aktuelle Entwicklungsstand ist auf github abrufbar (https://github.com/boitumeloruf/3DMuVi)
- Das Fraunhofer-Institut stellt uns einen FTP-Server bereit, auf dem wir nicht-öffentliche Dateien austauschen können.

13.2 Implementierung

Anwendungen zur Implementierung des Projektes:

- Die Anwendung wird in der Sprache C++ umgesetzt. Wenn sinnvoll, sind Funktionen aus C++11 und C++14 zu verwenden.
- Als Compiler wird Clang verwendet um die Entwicklung durch bessere Fehlermeldungen des Compilers zu erleichtern.
- Als integrierte Entwicklungsumgebung ist Qt Creator zu verwenden. Auch darin integrierte Tools, wie bspw. Qt Designer zur grafischen Erstellung von Oberflächen, sollen verwendet werden.
- Doxygen wird zur Dokumentation des Quelltextes verwendet. Aus entsprechenden Markierungen im Quelltext kann damit automatisch eine grafische Übersicht erzeugt werden, die in einem Webbrowser betrachtet werden kann.

13.3 Hilfsbibliotheken

Hilfsbibliotheken dienen der einfacheren Implementierung durch die Nutzung von bereits durch Dritte implementierte Funktionen.

- Qt soll zur Umsetzung der grafischen Benutzeroberfläche verwendet werden. Dabei soll Qt Widgets verwendet werden, um eine native Darstellung der Oberfläche zu

erreichen. Außerdem kann Qt für Funktionen genutzt werden, die dort aber nicht in C++ enthalten sind. Qt ist in der Version 5.5 zu verwenden.
- PCL enthält Funktionen zur 3D Rekonstruktion von Bildern und Anzeige der Ergebnisse. Außerdem sind in PCL bereits Datenstrukturen definiert wie sie für den Austausch von Daten mit Plug-Ins benötigt werden.

13.4 Validierung

Tools zur Überprüfung auf vorhersehbare Fehler:
- Qt Test für automatisierte Tests von einzelnen Einheiten der Anwendung.

13.5 Erstellung von Grafiken

Anwendungen um Grafiken erstellen zu können werden individuell von dem erstellenden Teammitglied gewählt. Einzige Anforderung dabei ist, dass es eine Datei exportieren kann, die in LaTeX eingebunden werden kann.

14 Glossar

Point cloud Menge von Punkten eines Vektorraums.
Algorithmus
Visualisierung