IV Visualizer: Entwurfsheft – Änderungen

Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung IOSB

Josua Benjamin Eyl, Lukas Friedrich,
Max Bretschneider, Nathaniel Till Hartmann, Robin Köchel

Betreut von: Mickael Cormier M.Sc., Stefan Wolf M.Sc

Wintersemester 2023/2024

1 Einführung

Allgemein kam es zu diversen Namensänderungen und Refactorings, die hier nicht weiter betrachtet werden.

2 Klassenänderungen – Frontend

2.1 GUI

Einige Klassen, darunter Main Window und Export Dialog wurden in eigene Pakete ausgelagert.

2.2 Pipeline

Es wurde eine Klasse zum Mappen von Annotationsnamen auf Farben, zur unterschiedlichen Darstellung der Bounding Boxen hinzugefügt. Es wurde RegionOfInterestItem stellte die Funktionalität, ROIs zu zeichnen zur Verfügung.

2.3 Video player

Es wurde eine Klasse für die beiden Slider hinzugefügt Es wurde eine Klasse(AskingAfterStreamThread) für die Streamanfrage ans Backend hinzugefügt.

2.4 BackendConnector

Es wurde eine Klasse(IvBackendClient) erstellt, um die GRPC Services im Backend auszuführen. Es wurde eine Klasse(MessageConverter) erstellt, um die GRPC Messages in die Frontend/Backend Klassen zu konvertieren.

2.5 Frame data

Annotation und BoundingBox wurden entfernt und durch eine Library ersetzt.

2.6 BackendStub

Es wurden mehrere Klassen erstellt, um das Backend zu simulieren und erste Tests zu ermöglichen. Keine der Klassen ist von funktionaler Bedeutung für das Projekt.

2.7 DataStreams

Es wurden Klassen zum Versenden von Data Frames erstellt. Die Data Stream Klasse holt sich Data Frames aus dem Data Frame Ring Buffer. Diese Frames wurden zuvor vom DataFrameStream in den Buffer geschrieben.

2.8 ExportSettings

Es wurden Klassen zum Exportieren und importieren der Stream Objekte erstellt.

2.9 Main

Main Widget wurde hierher ausgelagert und auch die Konfigurationsdatei liegt in diesem Ordner

3 Klassenänderungen – Backend

3.1 MessageHandler

Generell wurde der MessageHandler durch mehrere Methoden erweitert. Diese wurden auch entfernt aus der Protodatei (Frontend-Backend). Klassen wie ThreadHandler sind nicht mehr explizit genannt und werden von gRPC Funktionalitäten abgedeckt. Zur einfacheren Handhabung wurden Klassen wie TimeRangeVector und TimeRange hinzugefuegt oder erweitert.
Zusätzlich wurde die DataFrameFactory zur Utility Klasse MessageConverter erweitert. In dieser sind Methoden enthalten zur Erleichterung der Kommunikation mit verschiedenen Proto Dateien, die im MessageHandler stattfindet. Insbesondere hilft diese Klasse bei der Konvertierung von verschiedenen Nachrichtentypen, aber auch bei der Erstellung der DataFrames.

3.2 DataManager

Da C++ keine return Werte in Konstruktoren erlaubt, wird das Multiton über die neue Funktion `getInstance` abgerufen.
`addMask` wurde mit dem Umstieg auf `iv-2d-annotation-handler` umbenannt in `addRegionOfInterest`.
`setDbConfig` wurde nicht mehr benötigt, weil die Konfiguration über ENV Variablen gesetzt werden. (Das ermöglicht eine Konfiguration ohne Frontend)
`getOutputQueue` nutzt die im Entwurfsheft nicht näher beschriebene `SafeQueue` Klasse.
Alternativ kann auch auf die `fillOutputQueue` zurück gegirffen werden, um einen blockierenden Aufruf zu tätigen.
`stopQueue` wurde nicht mehr benötigt, da diese Methode nur für Fehlerfälle benötigt wurde, die wir ausschließen können.

3.3 FSManager

`insertImages` wurde zu `addImage`, da der Aufruf auf den Encoder auch nur einzelne Bilder entgegen nimmt und ein Bulk-Insert hier keinerleit Vorteile geboten hätte.

3.4 DBManager

`DBManager` ist jetzt ein Singleton, da das von Scylla empfohlen wird. Daraus ergibt sich, dass die meisten Methoden statisch sind.
`insertMask` wurde umbenannt zu `insrtROI` und nutzt `iv-2d-annotation-handler`. In diesem Zuge wurde auch `getAnnoations` zu `getBoundingBoxes` und `insertAnnotation` zu `insertBoundingBoxes`.
`updateTTL` wurde entfernt, um eine versehentliche Überlastung der Datenbank zu verhindern. Stattdessen wurden Instruktionen geschrieben, wie man alte TTL abänder. TTL für neue Werte kann in ENV gesetzt wird.
Außerdem wurden hinzugefügt: `executeStaticQuery, isConnected, getRegionOfInterest`

3.5 Encoding (jetzt Encoder)

Es wurde keine Hardwarebeschleunigung implementiert, weil keine Hardware zur Verfügung stand.
Die ffmpeg Klassen wurde nicht direkt sonder über openCV eingebunden.
Der Konstruktor benötigt andere Werte. `add` wurde zu `add_image`. `getFrame` wurde zu `get_Frames`.
Außerdem wurden hinzugefügt: `finish_writing_mp4` sowie getter für fps, width, hight, path und timestamp.

3.6 SafeQueue

Um parallelität zu ermöglichen wurden normale Queues erweitert um Threadsafe zu sein.
Dadurch ist eine Klasse entstanden, die `produce` und `consume` Methoden anbietet. Diese sind blockierend und Thread-sicher.

3.7 Allgemeine Änderungen am Model

Die Architektur im Model ist gleich geblieben. Einige Threads die innerhalb von Klassen Aufgaben übernehmen sollten, sind entfallen, weil der Code auch ohne diese performant genug war.
Es kam zu einigen Namensänderungen die nur teilweise aufgeführt wurden.
Der Code nutzt größtenteils keine Methoden und Objekte aus der Standard Bibliothen, sondern aus der QT Bibliothek.

3.8 Allgemeine Änderungen am Controller

Es wurden Klassen entfernt. Hierzu zählen zum Beispiel die Streamreader Klasse und einige Klassen, die genutzt werden sollten zur Thread-Verarbeitung.

3.9 Dummy

Zu Demonstrations- und Testzwecken wurde eigens ein Dummy entwickelt, der die Funktionen der Processing Tools simulieren soll.

4 Protodateien

Besonders die Protodatei zwischen Frontend und Backend wurde erweitert, um die gRPC-Verbindung zu realisieren. An der Protodatei zwischen Backend und Processing Node hat sich nichts verändert.