Lastenheft

Synchronisation und Auswertung von Roboterfußballvideos

tj18b

Mitglieder
Dan Häßler
Robert Wagner
Sirk Petzold
Alex Eichhorn
Erik Diener
Jonas Wrubel
Tomas Daetz Chacon

Betreuer
Hans-Gert Gräbe
Tobias Wieprich
Tobias Jagla
André Köhler

Inhaltsverzeichnis

1 Visionen und Ziele

/LV10/ Das System soll für das NAO-Team HTWK die Bearbeitung und Auswertung von Videos einzelner Roboterfußballspiele vereinfachen und diverse Bearbeitungsmöglichkeiten bereitstellen.

/LZ10/ Das System soll eine beliebige Anzahl von Videos und Logs mit festem Dateiformat und FPS-Zahl automatisch synchronisieren. Dabei sollen möglichst viele Schritte editierbar und transparent gehalten werden.

2 Rahmenbedingungen und Produkteinsatz

2.1 Anwendungsbereich & Zielgruppe

Die Software wird während oder nach einem Turnier Verwendung finden. Anwendungsbereich sind somit vom Turnierveranstalter bereitgestellte Räume oder die Örtlichkeiten des Teams. Die Zielgruppe ist ein Roboterfußballteam bzw. dessen Mitglieder. Mitglieder des Teams sind Studenten, universitäre Mitarbeiter, beziehungsweise deren Professoren.

2.2 Betriebsbedingungen

Das Produkt muss im mobilen sowie stationären Betrieb laufen. Die Software unterliegt eher unregelmäßigen Nutzungszeiträumen.

2.3 Technische Produktumgebung

Die Software ist eine Desktopanwendung.

Software	**Hardware**	**Orgware**
Windows/Linux, Linux	Aktueller Laptop	Zugriff auf Videodaten

2.4 Anforderungen an die Entwicklungsumgebung

Software	**Hardware**	**Orgware**
Windows/Linux/Mac, Java & IDE	PC	Zugriff auf Videodaten, Git & Logs

3 Kontext und Überblick

/LK10/ Das System ist eine Standalone-Software, die auf Windows- und Linux-Rechnern lauffähig sein muss.
/LK20/ Das System liest Log-Files einer gegebenen GameController-Instanz im txt-Format ein.

4 Funktionale Anforderungen

4.1 Pflicht

Die grundlegenden funktionalen Anforderungen werden im Folgenden aufgelistet

/F10/ Abspielen von bis zu vier Videos gleichzeitg und Synchronisierung der Videos anhand ihres Offsets. Dafür soll jedes Video ein eigenes Anzeigelement erhalten, sowie eine zentrale Steuereinheit implementiert werden. Diese soll Sprünge zu beliebigen Punkten anhand eines Sliders, Vorwärts- und Rückwärts-Sprünge um beliebige Frames oder Sekunden sowie eine Pausefunktion beinhalten.

/F11/ Die zum Videoset gehörenden Logfiles sollen geparst und mit der Timeline im Video synchronisiert werden. Durch Klicken auf die spezifische Logeinträge, soll im Video an die entsprechende Stelle gesprungen werden. Ebenso soll ein Sprung im Video auch den entsprechenden Logeintrag markieren.

/F12/ Zu jedem Set geladener Videos, soll bestimmt werden können, welche Tonspur global verwendet werden soll.

/F13/ In den Videos sollen für beliebig viele Kamerawechsel Markierungspunkte gesetzt werden können. Diese sollen in einem MLT-file gespeichert werden können. Darüber hinaus sollen auch png-files eingelesen und mit entsprechenden Markierungspunkten versehen werden können.

4.2 Optional

Eine Vielzahl von Zusatzfeatures sind optional implemententbar. Dazu zählen Folgende:

/F20/ Automatische Synchronisation der Audiospur

/F21/ Automatischer Kamerawechsel

/F22/ Umgang mit gesplitteten Logfiles oder fehlenden Audiospuren

/F23/ das Automatisieren der Offset-Brechnung

/F24/ Umgang mit unterschiedlichen Videoformaten und FPS-Konvertierung

/F25/ Erzeugung eines Overlays aus einer Bilddatei.

5 Produktdaten und nicht-funktionale Anforderungen

5.1 Nicht-funktionale Anforderungen

Pflicht

/NF10/ Wartbarkeit: Javadoc muss vollständig sein. Dokumentation und Kommentare vollständig auf englisch. Code conventions einheitlich in der gesamten Software.

/NF11/ Die Software auf Linux anwendbar sein.

/NF12/ Bedienbarkeit: Einfache Bedienung des Programms, dazu gehört ein vollständiges Manual mit Funktionsbeschreibungen und eine nutzerfreundliche bzw. selbstbeschreibende GUI.

/NF13/ Modularität: Modulare Programmierung als Programmierparadigma.

/NF13/ Java als Programmiersprache.

Optional

/NF20/ Interoperabilität: Die Software soll plattformunabhängig sein, d.h. auf Windows, Linux und Mac anwendbar sein.

/NF21/ Benutzung von Logging Framework

5.2 Produktdaten

/PD10/ Inputdaten: Alle Daten, die vom Benutzer ins Programm geladen werden können. Dazu gehören bis zu 4 **Videodaten**, sowie ein **GameController (GC) Log**.

/PD11/ Videodaten: Videodateien im Format MP4 mit 30 FPS und dem zu den Video gehörenden Offset.

/PD12/ GC LOG: Log-Datei als Textdokument, die Daten zum Spielablauf enthält.

/PD13/ Konfigurationsdaten: Alle die vom Benutzer bestimmbare Daten, die den Videobearbeitungsprozess beeinflussen. Alle automatisierten Prozesse sollen vom Benutzer konfigurierbar sein, dazu gehört unter anderem zu einem bestimmten Videoevent davor springen um die Kamera auszuwählen. Weiterhin soll das Outputformat einstellbar sein.

/PD14/ Outputdaten: Die aus dem Videosynchronisationsprozess hervorgehenden Daten. Darunter fallen Projektdateien, sowie MLT Dateien.

Wichtigkeit	Funktionale Tauglichkeit	Effizienz	Kompatibilität	Nutzbarkeit	Zuverlässigkeit	Sicherheit	Wartbarkeit	Übertragbarkeit
Hoch	X						X	
Mittel		X	X	X	X			X
Niedrig								
N.A.						X		

6 Qualitätsmatrix nach ISO 25010

6.1 Zu den Qualitätsparametern:

Funktionale Tauglichkeit: (Funktionale Vollständigkeit/Korrektheit/Eignung)
Vollständig funktionierendes, den funktionalen Anforderungen entsprechendes Endprodukt ⇒ hohe Priorität
Effizienz: (Zeitverhalten, Ressourcennutzung, Kapazität)
Tolerierbare Lauf-/Antwortzeiten und Ressourcenauslastung ⇒ mittlere Priorität
Kompatibilität: (Koexistenz, Interoperabilität)
Unhinderliche Koexistenz mit anderer Software; Informationsaustausch zwischen Modulen ⇒ mittlere Priorität
Nutzbarkeit: (Lernbarkeit, Bedienbarkeit, Vorbeugung von Nutzerfehlern, Gestaltung des Nutzerinterfaces, Zugänglichkeit)
Gestaltung des Produkts für (Hoch-)Spezialisierte Anwendergruppe; Übersichtliche, intuitive Bedienung ⇒ mittlere Priorität
Zuverlässigkeit: (Ausgereiftheit, Verfügbarkeit, Fehlertoleranz, Wiederherstellbarkeit)
Stabile Laufzeit/Fehlerbehandlung; gehobene Wiederherstellbarkeit als nice-to-have ⇒ mittlere Priorität
Sicherheit: (Diskretion, Integrität, Unverfolgbarkeit, Verantwortlichkeit, Authentizität)
Irrelevant/Nicht gefordert ⇒ nicht anwendbar
Wartbarkeit: (Modularität, Wiederverwendbarkeit, Analysierbarkeit, Modifizierbarkeit, Testbarkeit)
Modularität/Modifizierbarkeit gefordert ⇒ hohe Priorität
Übertragbarkeit: (Adaptivität, Installierbarkeit, Ersetzbarkeit)
Plattformunabhängigkeit gefordert ⇒ mittlere Priorität

7 Lieferumfang und Abnahmekriterien

7.1 Lieferumfang

Zum Lieferumfang unseres Teams gehört zum einen das GUI basierte Tool zum Einlesen, Anzeigen und Bearbeiten von Roboterfußball-Videos, sowie deren synchronisierte Darstellung im GUI. Dabei sollen auch die dazugehörigen Log-Files eingelesen werden können und mit den Videos synchronisiert werden.
Zum anderen enthält der Lieferumfang auch die ausführliche Dokumentation unseres Projektes. So wird zur Abgabe ein Javadoc zur Codedokumentation abgegeben. Des weiteren wird ein Handbuch eingereicht, welches eine detaillierte Anleitung zur Handhabung der gelieferten Software, sowie eine Entwurfsbeschreibung, unter anderem in Form eines UML-Diagrammes, enthält. Schließlich gehöhrt auch ein Installationshandbuch zum Lieferumfang.

7.2 Abnahmekriterien

Das wichtigste Abnahmekriterium stellt die Einhaltung des Lieferumfanges dar. Dabei sollen alle Pflichtanforderungen erfüllt und eingehalten werden. Zusätzlich ist vorgesehen, soviele optionale Anforderungen wie möglich umzusetzen, sich dabei vorerst aber vor allem auf die Kerninhalte, unter anderem das Schneiden der Videos, zu konzentrieren.
Die Videos müssen synchronisiert abspielbar sein, ein framegenaues agieren und modifizieren soll gewährleistet sein. Des weiteren muss eine Audiospur auswählbar sein, um einen durchgängigen Ton zu gewährleisten. Diese wird ebenfalls mit den Videos synchronisiert.

8 Vorprojekt

Im Rahmen des Vorprojektes soll das Grundgerüst für das effiziente weitere Vorgehen geschaffen werden. So ist es unter anderem vorgesehen einen ersten Entwurf des GUI, als auch ein Klassengerüst zu implementieren. Des weiteren wird ein Gerüst für die Dokumentation und die Bedienungsanleitung geschaffen.

9 Glossar

Java Java ist eine objektoriente Programmiersprache, die sich durch ihre Plattformunabhängigkeit auszeichnet, der Quellcode wird in bytecode kompiliert, der dann auf jeder JVM unabhängig von System und Architektur laufen kann.

XML (engl.: Extensible Markup Language) ist eine Auszeichnungssprache, die hierarchisch strukturierte Daten darstellen kann. Es wird auch eingesetzt für den plattform- und implementationsunabhängigen Datenaustausch.

Parser Ein Parser ist ein Programm, das eine Eingabe derart zerlegt und umwandelt, dass es in einem, für die Weiterverarbeitung, (optimalen) Format ist.

API Eine Schnittstelle zur Anwendungsprogrammierung, häufig kurz API genannt (engl.: application prgramming interface, wörtl.: Anwendungsprogrammierschnittstelle), über das andere Programme sich an das Softwaresystem anbinden kann.

Library Als Library (dt.: Programmierbibliothek; kurz: Bibliothek) bezeichnet man eine Sammlung von Unterprogrammen/-Routinen inklusive, falls vorhanden, Dokumentation, templates, pre-written code, [...]. Eine Library bietet Lösungswege/-ansätze für thematisch zusammenhängende Problemstellungen. Der Zugriff auf eine Bibliothek erfolgt über eine API, die sich aus der Gesamtheit der öffentlichen Funktionen und Klassen zusammensetzt.

GUI (engl.: graphical user interface) ist eine grafisches Benutzeroberfläche, die als Benutzerschnittstelle dient, indem einerseits der Mensch mit dem Programm kommunizieren kann, in Form von Buttons, Slidern, o.Ä., andererseits aber auch das Programm mit dem Menschen kommunizieren kann, indem Daten auf der GUI visualisiert werden.

Frame Einzelbild in Filmen, Animationen und Computerspielen

FPS Frames pro Sekunde

Synchronisation bedeutet in diesem Zusammenhang mehrere Elemente zeitlich auf dem Wiedergabegerät an eine Datenquelle anzupassen.

Software ist ein Computerprogramm, einschließlich ihrer dazugehörigen Dokumentation und Konfigurationen, die erforderlich sind um das Programm auszuführen.

Plattform Eine Plattform ist die Umgebung, in welcher ein Stück Software ausgeführt wird, dabei kann es die Hardware, aber auch das Betriebssystem beschreiben, im Folgenden in Bezug auf Rechnerarchitektur bzw. Betriebssystem benutzt.

Framework Rahmenapplikation die grundsätzliche Strukturen und Funktionen zur Entwicklung von Programmen bereitstellt. Stellt nicht nur Klassen und Funktionen bereit wie eine Library, sondern in der Regel auch die grobe Anwendungsarchitektur.

Modularität Das Aufteilen eines Ganzen in logische Teilmodule, welche wiederverwendbar sind.

Logging Aufzeichnen von Daten während eines Vorgangs

MLT-Datei ShotCut-Dateiformat

Code Convention Programmierrichtlinien

Offset Zeitversatz (hier: in Videos)

Standalone Software (Eine Software die ohne andere Software funktioniert)