Hochschule der Medien Stuttgart
Studiengang Medieninformatik Bachelor

Projekt: 3D Scanner

Wintersemester 09/10

David Bertram
Carsten Brandt
Jan Distel
Markus Brouwer

Inhaltsverzeichnis

1 Einleitung

1.1 Hintergrund und Idee

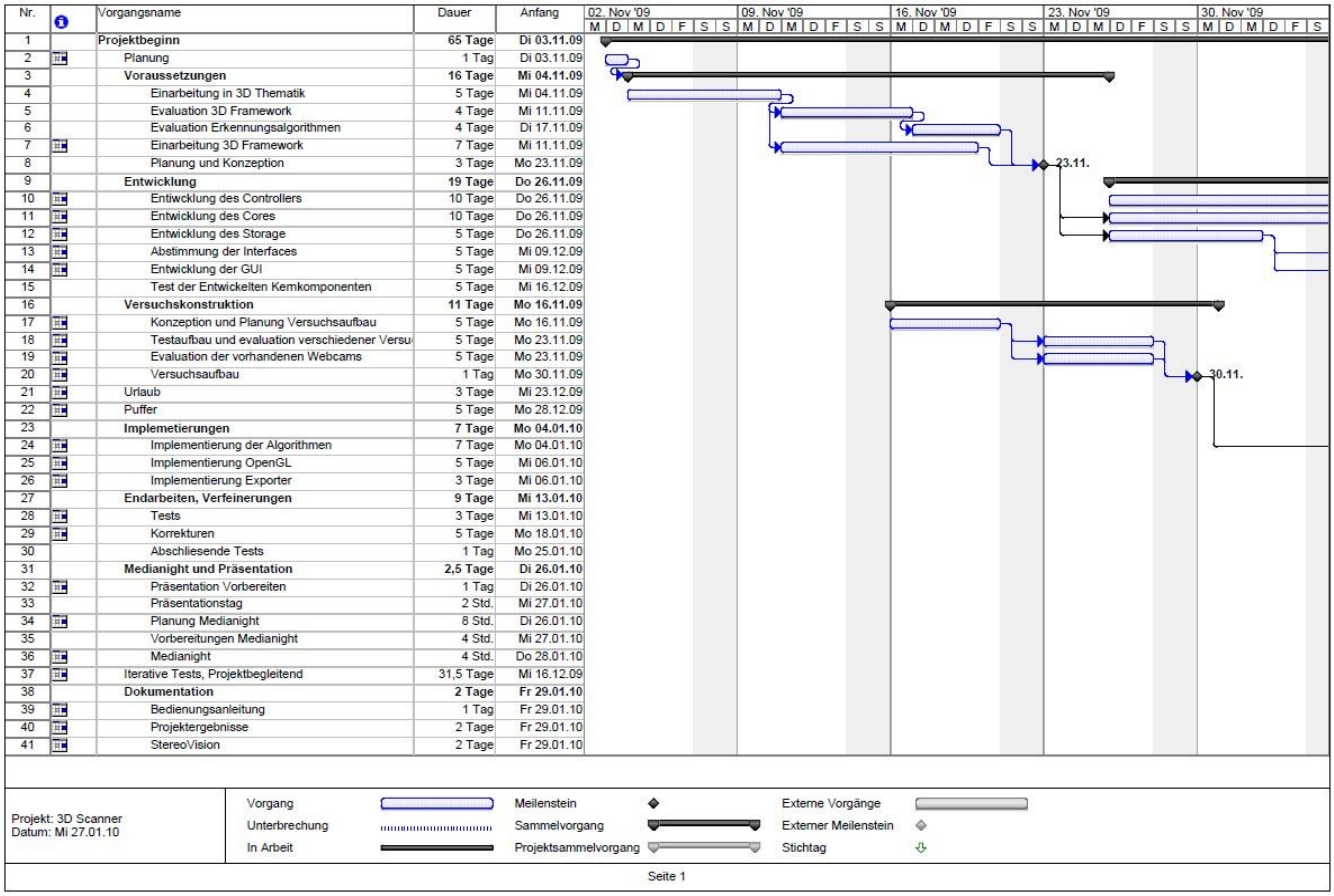
Im professionellen Bereich existieren 3D-Scanner-Systeme deren Preis die 10.000 Euro schnell übersteigen kann. Diese Systeme beinhalten meist vor kalibrierte Kamerasysteme, auch mit weit mehr als 2 Kameras, regelrechte Kamera-Arrays, sowie entsprechende API's oder direkt verwendbare Software um aus dem optischen Input die Tiefeninformation zu berechnen. Dank steigender Rechenleistung und der Verfügbarkeit von einfachen Webcams für unter 10 Euro wird der kostengünstige Einsatz von ComputerVisions-Systemen allerdings auch im Low-Budget-Bereich immer interessanter. So vertreibt beispielsweise Sony seit einigen Jahren eine Kamera für ihre Spielkonsole und Microsoft hat für Ende 2010 ein ähnliches Produkt angekündigt (Natal). Um diese Systeme zu betreiben reicht die vergleichsweise niedrige Rechenleistung der jeweiligen Spielkonsolen aus und teilweise wird von erstaunlichen Anwendungsmöglichkeiten berichtet. Daher entstand die Idee ein entsprechendes System mittels handelsüblicher Komponenten aufzubauen und zu versuchen eine 3D-Erfassung einfacher, kleiner Objekte umzusetzen. Da Intel bereits um die Jahrtausendwende ihre anfangs intern entwickelte Library OpenCV der OpenSource-Gemeinde kostenlos zur Verfügung gestellt hat und mindestens eine Robotik-Firma (WillowGarage) seither aktiv daran arbeitet diese Library weiter zu entwickeln, fiel die Wahl der Library auf eben jenes OpenCV.

1.2 Umfang

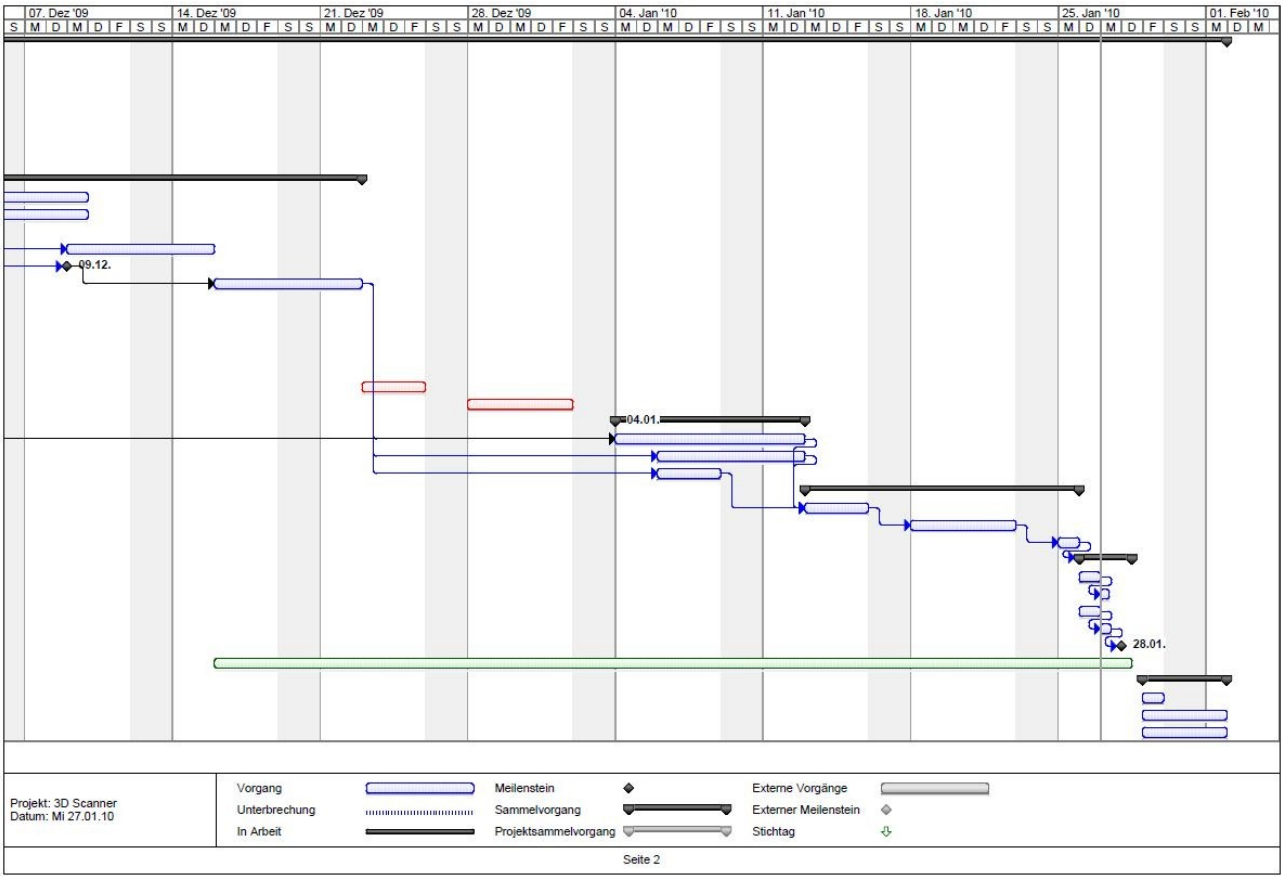
Im Laufe des Projekts wurde mehrfach diskutiert was man genau erreichen möchte und welche Methoden angewandt werden sollen. So stand zwischenzeitlich auch zur Debatte ob man statt reinen „Stereo-Systemen“ auch 3 oder mehr Kameras zusammenschalten soll. Dies konnte leider nicht direkt umgesetzt werden, da die vorhandenen Algorithmen jeweils nur Stereodaten als Input verarbeiten und ein Erweitern im verfügbaren Zeitrahmen nicht umsetzbar gewesen wäre. Also einigte man sich darauf, zunächst zu versuchen aus Stereodaten brauchbare Tiefenmaps zu generieren, sowie eine Oberfläche zu entwickeln, die dem Benutzer erlaubt die verwendeten Algorithmen auszuwählen und die Konfiguration der entsprechenden Parameter anzupassen. Im nächsten Schritt sollten Tiefenmaps aus mehreren Perspektiven miteinander zu 3D-Daten kombiniert werden. Diese Ergebnisse würden durch ein PostProcessing verfeinert, geglättet, mit Farbinformation versehen und anschließend in entsprechenden Formaten für eine externe Weiterverarbeitung abgespeichert werden. Hierfür sollte eine Anwendung programmiert werden, die eigenständig die verfügbaren Kameras erkennt, konfiguriert und vom Benutzer die übrigen zur Konfiguration des Stereo-Systems notwendigen Einstellungen abfragt. Vor dem eigentlichen Erfassen der Tiefeninformationen müssen die Verzerrungen der billigen Kameras durch Kalibrierung erfasst und berechnet werden um danach die jeweils gewünschte 3D-Erfassung durchführen zu können.

2 Projektmanagement (Sebastian Stadtrecher)

2.1 Projektablauf



Nr.	Vorgangsname	Dauer	Anfang
1	**Projektbeginn**	**65 Tage**	**Di 03.11.09**
2	Planung	1 Tag	Di 03.11.09
3	**Voraussetzungen**	**16 Tage**	**Mi 04.11.09**
4	Einarbeitung in 3D Thematik	5 Tage	Mi 04.11.09
5	Evaluation 3D Framework	4 Tage	Mi 11.11.09
6	Evaluation Erkennungsalgorithmen	4 Tage	Di 17.11.09
7	Einarbeitung 3D Framework	7 Tage	Mi 11.11.09
8	Planung und Konzeption	3 Tage	Mo 23.11.09
9	**Entwicklung**	**19 Tage**	**Do 26.11.09**
10	Entiwcklung des Controllers	10 Tage	Do 26.11.09
11	Entwicklung des Cores	10 Tage	Do 26.11.09
12	Entwicklung des Storage	5 Tage	Do 26.11.09
13	Abstimmung der Interfaces	5 Tage	Mi 09.12.09
14	Entwicklung der GUI	5 Tage	Mi 09.12.09
15	Test der Entwickelten Kernkomponenten	5 Tage	Mi 16.12.09
16	**Versuchskonstruktion**	**11 Tage**	**Mo 16.11.09**
17	Konzeption und Planung Versuchsaufbau	5 Tage	Mo 16.11.09
18	Testaufbau und evaluation verschiedener Versu	5 Tage	Mo 23.11.09
19	Evaluation der vorhandenen Webcams	5 Tage	Mo 23.11.09
20	Versuchsaufbau	1 Tag	Mo 30.11.09
21	Urlaub	3 Tage	Mi 23.12.09
22	Puffer	5 Tage	Mo 28.12.09
23	**Implemetierungen**	**7 Tage**	**Mo 04.01.10**
24	Implementierung der Algorithmen	7 Tage	Mo 04.01.10
25	Implementierung OpenGL	5 Tage	Mi 06.01.10
26	Implementierung Exporter	3 Tage	Mi 06.01.10
27	**Endarbeiten, Verfeinerungen**	**9 Tage**	**Mi 13.01.10**
28	Tests	3 Tage	Mi 13.01.10
29	Korrekturen	5 Tage	Mo 18.01.10
30	Abschliesende Tests	1 Tag	Mo 25.01.10
31	**Medianight und Präsentation**	**2,5 Tage**	**Di 26.01.10**
32	Präsentation Vorbereiten	1 Tag	Di 26.01.10
33	Präsentationstag	2 Std.	Mi 27.01.10
34	Planung Medianight	8 Std.	Di 26.01.10
35	Vorbereitungen Medianight	4 Std.	Mi 27.01.10
36	Medianight	4 Std.	Do 28.01.10
37	Iterative Tests, Projektbegleitend	31,5 Tage	Mi 16.12.09
38	**Dokumentation**	**2 Tage**	**Fr 29.01.10**
39	Bedienungsanleitung	1 Tag	Fr 29.01.10
40	Projektergebnisse	2 Tage	Fr 29.01.10
41	StereoVision	2 Tage	Fr 29.01.10


07. Dez '09
14. Dez '09
21. Dez '09
28. Dez '09
04. Jan '10
11. Jan '10
18. Jan '10
25. Jan '10
01. Feb '10
09.12.
04.01.
28.01.
Projekt: 3D Scanner
Datum: Mi 27.01.10
Vorgang
Unterbrechung
In Arbeit
Meilenstein
Sammelvorgang
Projektsammelvorgang
Externe Vorgänge
Externer Meilenstein
Stichtag
Seite 2

3 Technik (Carsten Brandt, Markus Brouwer, David Bertram)

3.1 Cross Plattform

MacOS: Zuerst einmal gibt es unter MacOS generelle Probleme mit den Treibern zu Webcams. Hier werden nur die Wenigsten unterstützt. So werden außer der nicht mehrr produzierten Apple I-Sight und den in den Displays verbauten Kameras nur einige wenige der Firma Logitech unterstützt. Da zum aktuellen Betriebssystem Snow Leopard bisher kein Update seitens OpenCV heraus kam und hier noch Probleme mit Quicktime nicht behoben wurden, war es uns nicht möglich OpenCV unter MacOS zum Laufen zu bringen.

3.2 Kalibrierung

Hier wurde ein Direktes Lineares Transformationsverfahren(DLT) implementiert. Dazu werden mehrere Bilder eines Schachbrettmusters aufgenommen, welche dann verarbeitet werden um die inneren und äußeren Parameter der Kamera zu ermitteln. Der Algorithmus wertet hierzu die Ecken auf dem Muster aus. Außer dem DLT-Verfahren stehen in OpenCV keine Algorithmen (wie z.B. Hoppe, Tsai) zur Kamera-Kalibrierung zur Verfügung. Mit Hilfe von View Morphing Algorithmen können verschiedene Ansichten eines Objekts, z.B. von einer virtuellen Kamera, berechnet werden.

3.3 Kameras

Die Ergebnisse der verschiedenen Algorithmen hängt stark von den verwendeten Kameras ab. So gibt es hier Probleme mit dem Weißabgleich, unterschiedliche Belichtungen, Farbverfälschungen und Kissenverzerrungen, sogenannten "Distortions". Besonders problematisch wird dies vor allem in den Unterschieden bei den beiden Kameras einer Stereoeinheit. Hinzu kommen Probleme mit den Treibern der USB-Kameras. Hier brechen schnell die Frames per Second ein oder einzelne Kameras versagen ihren Dienst.
Bei Recherchen im Internet stießen wir auf folgende Anbieter die vor kalibrierte Stereo-Kamerasysteme vertreiben:

http://www.videredesign.com/index.php?id=2
http://www.ptgrey.com/products/stereo.asp

3.4 Warping

Beim Warping, handelt es sich um ein Verfahren, welches die Bilder der Kameras einer Stereoeinheit so abstimmt, das sie den gleichen Bildausschnitt wieder geben. D.h. wenn in beiden Bildern der Kameras eine bestimmte Fläche zu sehen ist, bei uns z.B. das Schachbrett, werden Bildausschnitte so verändert, dass beide nichts außer dem Schachbrett wieder geben.
Dies haben wir erreicht, in dem wir ein Schachbrett als bekanntes Bild genommen haben, welches die Kameras aufnehmen. Über die Ecken der Felder des Schachbrettes, die in beiden Bildern gefunden werden, wurden Berechnungen vorgenommen, um zu bestimmen, wie die Kameras ausgerichtet sind und im Raum zueinander stehen. Diese Ergebnisse haben wir in Transformationsmatrizen gespeichert und wieder verwendet um die Bilder so zu transformierten, dass die Quadrate nicht mehr als Trapeze bzw. Parallelogramm, sonder als Quadrat wieder gegeben werden. Danach müssen die Bildausschnitte nur noch an das Schachbrett angepasst werden.

3.5 Algorithmen

3.5.1 Graph Cut

Der Graph Cut Algorithmus unterteilt den Graphen in Bereichen in denen Matches gefunden werden können ein und begrenzt so den Bereich in dem einen korrespondierender Punkt von Bild 1 in Bild 2 gesucht wird. Dies geschieht mittels einer Berechnung einer Wahrscheinlichkeit, wo sich der korrespondieren Punkt befinden kann.

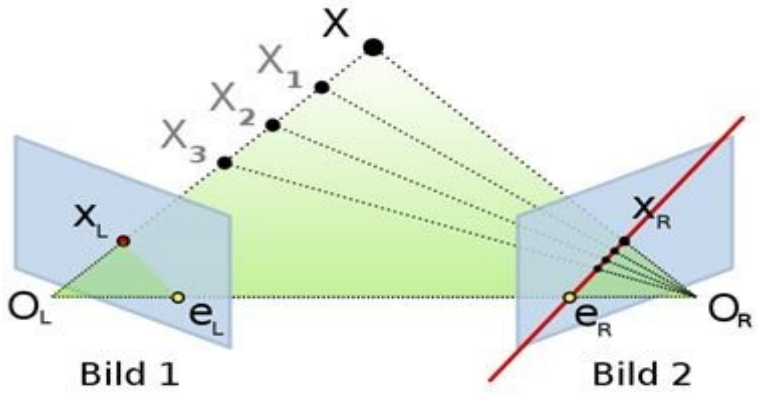
3.5.2 Block Matching

Beim Block Match Algorithmus wird angenommen, dass jeder Block X in Bild A nahezu identisch ist mit einem Block Y in Bild B

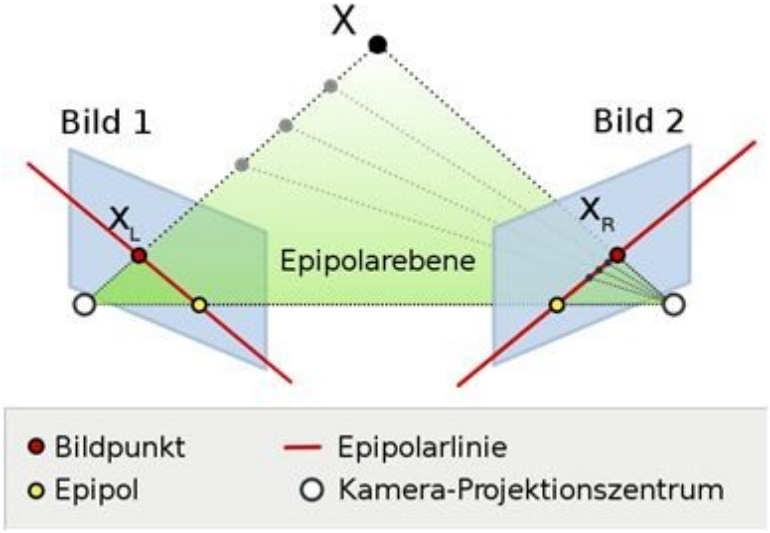
- unterteile Bild A in Blöcke der Größe m×n (z.B. 8×8)
- finde den zu jedem Block in A den ähnlichsten Block in B
  (als Maß für die Übereinstimmung kann beispielsweise der *mittlere quadratische Fehler verwendet* werden)
- berechne Disparitäten für einzelne Pixel anhand der Blockdisparitäten
- Disparitätswerte der einzelnen Pixel ergeben eine *dichte Disparitätenkarte* aus der man eine Tiefenkarte berechnen kann

3.5.3 Standard

Der Standard Algorithmus Nimmt den Punkt X der sich im Bild 1 als Punkt XL darstellt und berechnet einen Strahl von XL nach X. Dabei sind die Punkte X1-X3, die auf dem Strahl liegen, ebenso mögliche Bildpunkte die genauso wie X dem Punkt XL zugerechnet werden.



Danach wird über die Punkte OL, OR und X eine Ebene gebildet. Die Linie, welche sich durch den Schnitt mit den Bildern ergibt wird Eipipolarlinie genannt. Jetzt wird die Suche nach dem Pixel XR, der mit XL korrespondiert auf diese Linie Beschränkt.



3.6 Aufbau

Für den Aufbau war Ursprünglich geplant es möglich zu machen bis zu vier Stereoeinheiten (zwei Kameras dicht nebeneinander) von bis zu vier Seiten auf ein Objekt zu richten. Dabei sollte es möglich sein ein Objekt von der größe ca. 15x15x15 (H,B,T) vollständig zu erfassen. Aufgrund der recht schlechten Tiefenmaps haben wir zu Demonstrationszwecken zwei Stereoeinheiten an einem Ständer befestigt und auf ein Schachbrett, welches in einem durchsichtigen Kunststoffrahmen eingebettet ist gerichtet. Zur Verbesserung der Ergebnisse wurde zusätzlich hinter dem Schachbrett eine weiße Fläche geschaffen um störende zusätzliche Information zur Berechnung der Tiefenmaps zu beseitigen.

3.7 Beleuchtung

Eine gleichmäßige Beleuchtung der zu scannenden Objekte führt zu einem erheblich besserem Ergebnis. Hier sind Schlagschatten besonders störend aufgefallen. Aufgrund von fehlendem Budget konnten wir hier keine weiteren Tests durchführen.

4 Arbeitsumgebung

4.1 Open Computer Vision (OpenCV)

Bei OpenCV handelt es sich um eine opensource Programmbibliothek für C/C++. Sie beinhaltet eine große Sammlung an Algorithmen für maschinelles Sehen und Bildverarbeitung. Die Entwicklung von OpenCV wurde von Intel ins Leben gerufen und wird heute hauptsächlich von Wilow Garage betreut.

4.2 CA Labor

Das CA-Labor in der Hochschule der Medien beinhaltet einiger Vernetzte Rechner die an die Anforderungen des jeweiligen Projektes angepasst werden können und mit verschiedenen Entwicklungsumgebungen ausgestattet sind.

4.3 Visual Studio 2008

Visual Studio 2008 ist eine Entwicklungsumgebung für verschiedene Hochsprachen wie C/C++ und Java. Es dient dazu klassische Windows-Programme und Webseiten zu entwickeln. Das Hauptaugenmerk liegt auf der Entwicklung mit dem von Microsoft Entwickelten .NET Framework.

4.4 Qt

Dies ist ein ursprünglich von der Firma Trolltech entwickeltes Oberflächen-Framework zum entwickeln von Crossplatform GUIs. Die Entwicklung von GUIs basiert hier auf C++.

4.5 C++

Wurde 1979 von B. Stroustrup bei AT&T als Erweiterung von C entwickelt. Es handelt sich um eine höhere Programmiersprache die mehrere Programmierparadigmen wie objektorientierte, generisch und Prozedurale Programmierung unterstützt.

4.6 .NET

Eine Software-Plattform welche verschiedene APIs und Services für Programmierer enthält, die es ermöglichen einfach auf Betriebssystemfunktionen zuzugreifen und diese zu nutzen. .NET wurde von Microsoft entwickelt und ist bisher nur für Windows in vollem Umfang erhältlich.

4.7 Microsoft Windows

War ursprünglich eine grafische Erweiterung für MS-DOS und ist mittlerweile der Markenname für das Betriebssystem von Microsoft. Die aktuelle Version ist Windows7 und insgesamt haben die Windows-Betriebssysteme einen Markt.Anteil von ca. 93%.

4.8 Apache Subversion (SVN)

Dies ist eine frei Software die Programmierern die Versionsverwaltung ihrer Projekte erleichtert. Es besteht aus einem zentralen Archiv und die Versionierung wird mit Hilfe einer einfachen Revisionszählung umgesetzt. Zwischen Arbeitsplatz der Programmierers und dem zentralen Archiv werden dann jeweils nur die Unterschiede zur letzten Revision übertragen.
So ist es möglich von unterschiedlichen Standorten immer an der richtigen Version weiter zu entwickeln.

5 GUI (Jan Distel)

5.1 Aufgaben der GUI

Die GUI in diesem Projekt soll selbsterklärend und den Benutzer von alleine durch das Programm leiten. Durch feste Abfolgen und vorgegebene Auswahlmenüs sollen Fehler durch falsche Eingaben verhindert werden.

5.2 Anforderungen an die GUI

Um das Programm flexibel zu halten sollte es eine Cross-Platform Unterstützung geben. Des Weiteren die Möglichkeit 3D-Objekte in OpenGL darzustellen. Ansonsten gab es keine speziellen Anforderungen außer der Programmiersprache die auf C++ basieren muss um es einfach mit OpenCV kombinieren zu können.

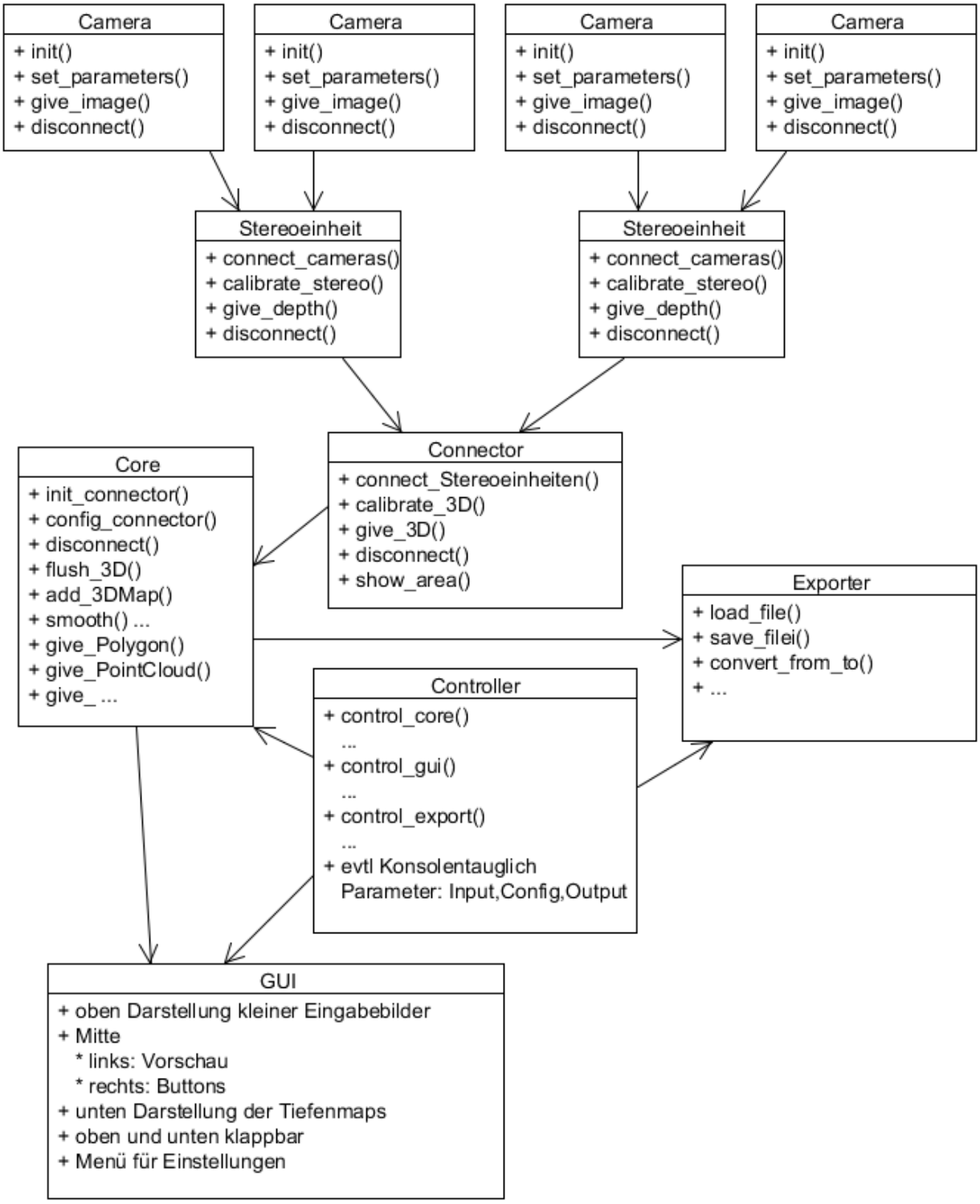
5.3 Ablauf bei der GUI Erstellung

Unsere Anforderungen an die GUI, wie Cross-Platform und trotzdem auf C++ basierend, brauchten eine spezielle Entwicklungsumgebung welche die nötigen zusätzlichen Klassen und Erweiterungen enthält das alles umzusetzen und gleichzeitig mit unseren Hauptbibliotheken von OpenCV kompatibel ist.

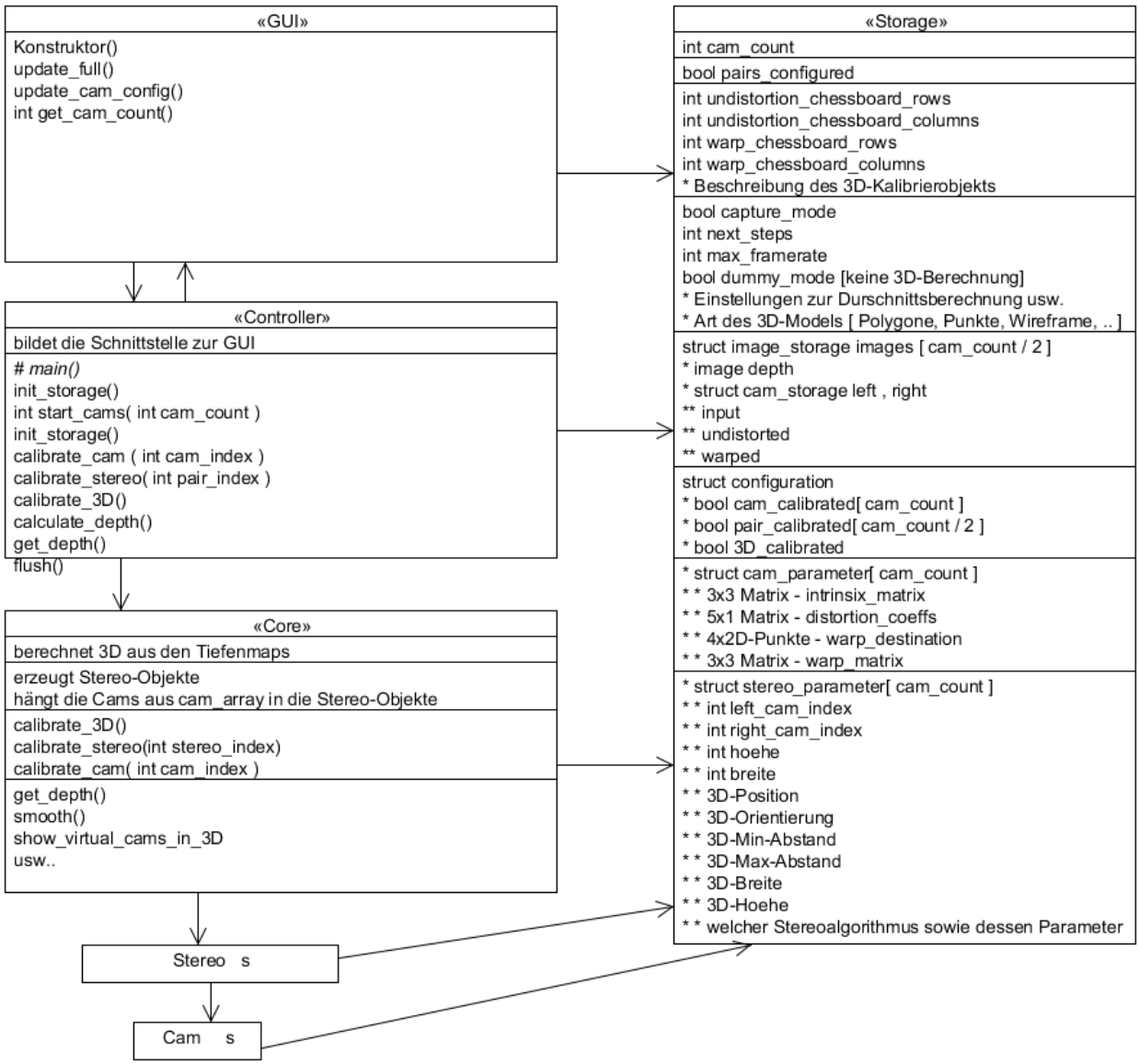
Die Lösung in diesem Fall war Qt, eine Entwicklungsumgebung die auf C++ basiert, OpenGL Unterstützung enthält und Cross-Platforming ermöglicht. Leider war auf Grund der anderen Klassen eine lange Einlernphase nötig in der die Tutorials der Herstellerfirma Trolltech durchgearbeitet wurden.
Anschließend ging es an die Umsetzung der GUI. Bis Mitte Dezember war der Prototyp der GUI fertig und es wurde Zeit die GUI mit dem OpenCV Code in Visual Studio 2008 zu integrieren. Hier kamen die ersten Probleme auf. Qt ließ sich nicht ohne weiteres in Visual Studio integrieren. Die einzige Lösung des Problems wäre es gewesen sich die kommerzielle Version von Qt zu kaufen, da diese standardmäßig eine Integration in Visual Studio beinhaltet.
Da es mittlerweile nur noch 20 Tage bis zum Präsentationstermin waren beschlossen wir eine Notlösung. Die GUI sollte komplett in Visual Studio 2008 mit Hilfe von WinForms und .NET Unterstützung erstellt werden. Also kamen wieder einige Tage Einlernzeit und anschließend sofort die Neuerstellung der GUI so gut es in der Zeit ging.

6 Konzepte (David Bertram)

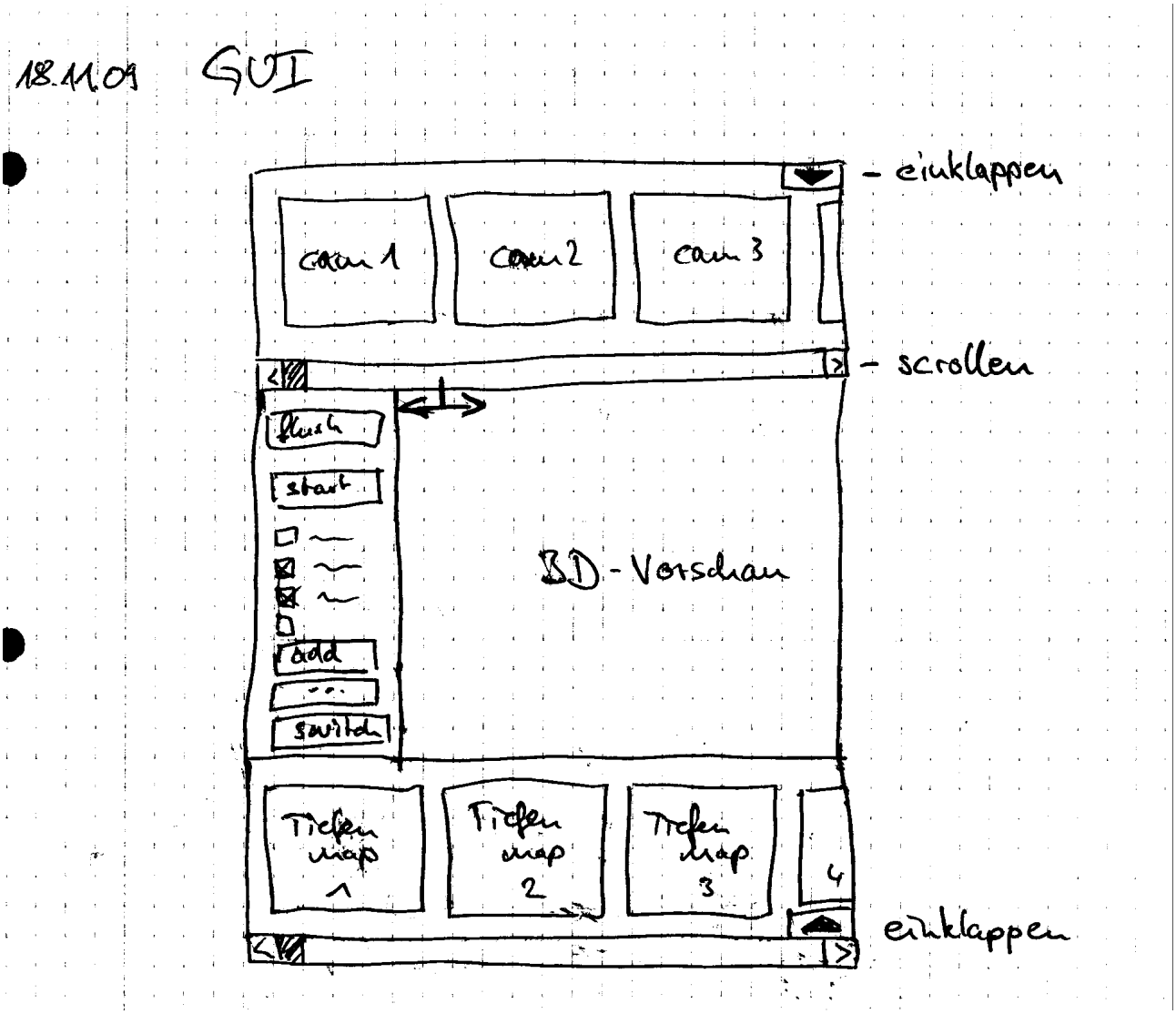
6.1 Erster Entwurf der zu verwendenden Klassen



6.2 Übersicht der implementierten Programmteile



6.3 Erster Entwurf einer GUI



7 Fazit

7.1 Soll/Ist

	Soll				
	Kalibrierung	Stereokalibrierung	Warping	Tiefenmap	3D-Objekt
Ist	x	x	x	x	-

7.2 Ausblick

Auf der Basis unserer Ergebnisse und Erfahrungen die wir gesammelt haben, wäre es möglich und bestimmten Voraussetzungen (bzgl. der Arbeitsumgebung) die Ergebnisse zu verbessern und 3D-Objekte zu erfassen. Wenn diese erfasst wären, könnte man auf diese noch Teilbereiche der Künstlichen Intelligenz anwenden um Objekte zu erkennen, bzw. ein Programm zu trainieren, das Objekte kategorisieren kann oder z.B. eine Gesichtserkennung durchführt.

7.3 ToDo

- Die GUI dahin gehend überarbeiten, dass eine Einstellung der Parameter für die Tiefenmaps möglich ist.
- Versuchen über diese Parameter die Ergebnisse der Tiefenmaps zu verbessern
- Aufbau erstellen der das Scannen von vier Seiten ermöglicht
- Für diesen Aufbau eine geeignete Belichtung (weißes Licht) herstellen
- Aus Tiefenmaps ein OpenGL-Objekt berechnen
- Das OpenGL Objekt in GUI Darstellen
- Möglichkeit das OpenGL Objekt in gängigem Format abzuspeichern

7.4 Schlussfolgerungen

Zum Schluss ist fest zu halten, dass unter den gegebenen Bedingungen - Webcams mit niedriger Auflösung, schlechtem Weißabgleich und schlechten Lichtverhältnissen – keine zufrieden stellenden Ergebnisse hinsichtlich eines 3D-Scanners zu bekommen sind. Unserer Ansicht nach bedarf es hierfür eigens Entwickelte Stereoeinheiten, die sich in einem Gehäuse befinden und Bilder zurück liefern die richtig aufeinander abgestimmt sind, also entzerrt und gutem Weißabgleich. Zusätzlich ist eine richtige Beleuchtung und ein Raum nötig, in dem man einen Aufbau gestalten kann, der eine unveränderte Arbeitsumgebung gewährleistet.