Hochschule Mittweida (FH)
University of Applied Sciences

MITTWEIDA
ROSSWEIN

Ing. Thomas Panner

Visualisierung von 1-, 2- und 3 dimensionalen Zeitreihen

eingereicht als

DIPLOMARBEIT

an der

HOCHSCHULE MITTWEIDA

UNIVERSITY OF APPLIED SCIENCES

Fachbereich Informationstechnik & Elektrotechnik

Bruck/Leitha, 2009

Erstprüfer: **Prof. Dr.-Ing. Jürgen Ruck**
Zweitprüfer: **Dipl.-Ing. Dr. Thomas Grünberger**

Vorgelegte Arbeit wurde verteidigt am: 8. Mai 2009

Bibliographische Beschreibung:

Panner, Thomas:
Visualisierung von 1-, 2- und 3 dimensionalen Zeitreihen. - 2009 - 74 S. Bruck/Leitha, Hochschule Mittweida, Fachbereich Informationstechnik & Elektrotechnik, Diplomarbeit, 2009

Referat:
Ziel der Diplomarbeit ist es, ein System zur Darstellung von mehrdimensionalen Zeitreihen zu entwickeln, welches zur Visualisierung der Ergebnisse von Sensoren zur Qualitätskontrolle an Laserschweißnähten eingesetzt werden soll.
In dieser Arbeit wird zuerst ein Überblick über Qualitätskontrollsysteme in der Laserprozesstechnik vermittelt. Neben aktuell verwendeten Sensoren zur Gewinnung der Daten werden die Grundlagen der Visualisierung präsentiert und die Grenzen des aktuellen Visualisierungstools aufgezeigt. Nach der Auswahl einer Toolbox werden systematische Untersuchungen in Bezug auf Performance und Funktion durchgeführt. In weiteren Kapiteln werden Design, Implementierung und Verifikation des neu entwickelten und auf der gewählten Toolbox basierenden Visualisierungssystems präsentiert. Die Zusammenfassung der Ergebnisse sowie ein Ausblick auf mögliche Optimierungen schließen diese Arbeit ab.

Inhaltsverzeichnis

1 Einleitung

1.1 Themenvorstellung

Moderne Messsysteme werden immer leistungsfähiger, wodurch sowohl die Messfrequenz als auch die Auflösung der ermittelten Daten steigt. Für die Analyse und Optimierung der zugrundeliegenden Prozesse stellt die aussagekräftige Aufbereitung der Daten ein elementares Hilfsmittel dar. Die grafische Darstellung ermöglicht es dem Betrachter meist rasch einen Überblick über die Daten zu erhalten.

Die Lasermaterialbearbeitung zählt aktuell zu den innovativsten Technologien der industriellen Fertigung. Die Anwendungen für dieses Fertigungsverfahren nehmen laufend zu, wobei dadurch auch der Bedarf an Qualitätssicherungs- und Optimierungssystemen steigt. Die Verbindung (Schweißnaht oder Lötnaht) ist dabei zumeist am Produkt sichtbar aber nicht mehr zerstörungsfrei prüfbar. Da diese Technik auch zunehmend für die Verbindung sicherheitsrelevanter Bauteile verwendet wird, kommt der Qualitätskontrolle wachsende Bedeutung zu (vgl. /13/ ff).

Abhängig vom Verwendungszweck werden unterschiedliche Anforderungen an die Qualität der Verbindungen gestellt. So sind beispielsweise Poren und Risse problematisch im Zusammenhang mit stark belasteten Verbindungen. Speziell in der Automobilindustrie ist die 100% - Kontrolle Standard. Dies bedeutet, dass jede Verbindung in Echtzeit geprüft und bewertet werden muss. Kann durch das Kontrollsystem keine eindeutige Bewertung durchgeführt werden, so müssen die Messdaten des betroffenen Prozessschrittes dem Anlagenbediener entsprechend aufbereitet präsentiert werden. Dieser hat dann die letztgültige Entscheidung zu treffen, ob der Vorgang als fehlerfrei oder fehlerhaft bewertet wird. Als Entscheidungsgrundlage dienen sowohl die Messdaten der Sensoren als auch die visuelle Beurteilung des Prüflings. Aufgrund der laufend verkürzten Produktionszeiten bei gleichzeitig steigenden Datenmengen der Sensoren ergibt sich eine wachsende Diskrepanz zwischen dem Zeitbedarf für die Datenvisualisierung und der dafür zur Verfügung stehenden Zeit.
Diese Arbeit befasst sich mit der Darstellung von 1-, 2- und 3dimensionalen Zeitreihen aus dem Bereich der Laserprozesskontrolle, deren rasche Darstellung und interaktive Manipulation einen wesentlichen Faktor für die effiziente Überwachung und Optimierung der Prozesse darstellen.

Vorstellung der Firma Plasmo Industrietechnik

Im Jahr 1994 wurde die Entwicklung eines Sensorsystems zur Überwachung des Laserschneidens begonnen. Die Ursprünge der Idee des Plasmamonitorings entstammen einer Kooperation von Wild-Austria und den Austrian Research Centers Seibersdorf. Die Idee bestand in der Regelung des Laser-Schneidprozesses. 1998 wurde das erste Produkt unter der Marke plasmo vorgestellt sowie die erste industrielle Implementierung vorgenommen. Der ProcessObserver (PsO) genannte Sensor erkennt die wesentlichen Fehlerarten durch Unregelmäßigkeiten im Prozess. Zusätzlich können mit dem Sensorsystem effizient die Bearbeitungsparameter optimiert werden.

ProfileObserver (PeO), der Sensor zur Geometrievermessung wurde im Jahr 2001 vorgestellt. Der Sensor dient zur Erfassung und Bewertung von dreidimensionalen Geometrien. Das Haupteinsatzgebiet für den ProfileObserver ist die Kontrolle von Schweiß- und Lötnähten, wobei die Nahtgeometrie schnell und mit hoher Auflösung erfasst wird, um damit Fehler wie beispielsweise Poren im Größenbereich von einem zehntel Milimeter zu detektieren.

Im Jahr 2003 wurde die Plasmo Industrietechnik GmbH gegründet. In die Gründungszeit fielen die ersten großen Installationen im Bereich Automobilbereich. Als weiteres Produkt für die Laserprozesstechnik wurde der Plasmo PowerObserver entwickelt. Das nach dem Kalorimeterprinzip arbeitende Gerät dient zur raschen und genauen Laserleistungs-messung nach der Fokussieroptik ohne die Produktivität der Anlage zu beeinträchtigen.

Der PositionController stellt ein Messsystem dar, das durch Einsatz eines Bildverarbeitungssystemes den Fügespalt zwischen den Blechen detektiert und damit die Position des Schweißwerkzeuges nachregeln kann. Optional kann auch nach dem Schweißprozess die Position der Schweißnaht bestimmt werden.

Plasmo zählt mittlerweile zu den weltweit führenden Unternehmen für Qualitätssicherungs- und Kontrollsysteme in der Lasermaterialbearbeitung. Heute bietet Plasmo eine breite Palette an Lösungen, die von Laserleistungsmessung, Schweißparameterüberwachung, Schweißnahtgeometrievermessung, industrieller Bildverarbeitung und Analysesoftware für die Qualitätssicherung bis hin zu ausgedehnten Serviceangeboten reicht.

1.2 Motivation der Arbeit

Die Datenvisualisierung stellt ein stark wachsendes Gebiet der Computertechnik dar. Speziell die Darstellung von großen Datenmengen erweist sich nach wie vor als eine Herausforderung für die ausführende Hard- und Software. Basierend auf den Erfahrungen der Abteilungen Entwicklung, Vertrieb und Service in der Plasmo Industrietechnik wurden Weiterentwicklungen der bestehenden Visualisierungssoftware erarbeitet.

Der Wunsch des Vertriebes war die Implementierung weiterer Alleinstellungsmerkmale um die Produktlinie noch besser am Markt positionieren zu können. Ein Vorschlag war die Realisierung eines virtuellen Kamerafluges entlang der Naht. Dieser sogenannte flyover-Modus scheiterte bisher jedoch sowohl an der Performance der bestehenden Visualisierung als auch am Konzept der derzeit eingesetzten Visualisierungstoolbox Measurement Studio von National Instruments.

Aus Anwendersicht bietet die aktuelle Software ausreichende Funktionalität für die Analyse der 2D und 3D Messdaten. Eine wesentliche Funktion für die Bewertung von Schweiß- und Lötnähten ist die visuelle Beurteilung. Dafür ist es erforderlich, die aktuelle Betrachtungsposition innerhalb der Datenprofil-Visualisierung interaktiv zu verändern. Die aktuelle Visualisierungssoftware benötigt relativ leistungsstarke Hardware, um die notwendige Interaktivität zu gewährleisten. Durch die zusätzlich Darstellung von 1D-Messdaten (beispielsweise PsO-Daten) könnte der Zeitbedarf für die Analyse von Nahtstellen weiter reduziert werden.

Die Eignung einer Quellcodebasis für unterschiedliche Betriebssysteme stellt ebenso eine Anforderung der Entwicklungsabteilung dar wie auch Flexibilität und Erweiterbarkeit der Visualisierungstoolbox. Diese Anforderungen werden von der aktuell verwendeten Tool-box nur teilweise erfüllt. Das derzeit eingesetzte Visualisierungssystem ist nur für Microsoft Windows verfügbar, wobei dessen Funktionalität zwar konfigurierbar jedoch nicht erweiterbar ist.

1.3 Ziel der Arbeit

Ziel der vorliegenden Arbeit ist nach einem Überblick über den aktuellen Stand der Computergrafik eine Einführung in die Thematik der Visualisierung auf dem Gebiet der Qualitätskontrolle in der Laserprozesstechnik. Aufgrund der Rahmenbedingungen soll eine derzeit am Markt verfügbare Toolbox für die Visualisierung der Sensordaten ausgewählt werden. Nach der Selektion eines geeigneten Basissystems erfolgt die Erstellung einer Testapplikation, die als Grundlage für die weiteren Untersuchungen dient.
Die Analyse des Verlaufes von Verarbeitungszeit und Speicherbedarf in Abhängigkeit der Darstellungsart und Datengröße führt zur technischen Bewertung der unterschiedlichen Varianten. Durch den Vergleich der Ergebnisse unterschiedlicher Systeme wird der Einfluss der Systemzusammenstellung auf die Performance untersucht. Aufgrund des Erscheinungsbildes wird auch eine subjektive Bewertung durchgeführt. Nach der Untersuchung von Maßnahmen zur Optimierung der Performance erfolgt die Zusammenfassung und bildet damit den Abschluß des ersten Hauptteiles der Arbeit.
Basierend auf den Ergebnissen des ersten Hauptteiles soll eine Klassensammlung entwickelt werden, die die Erstellung weiterer Visualisierungsanwendungen auf Grundlage der gewählten Toolbox ermöglicht.
Die Zusammenführung der unterschiedlichen Sensordaten zu einer einheitlichen Datenrepräsentation ist ebenso wie die Definition einer Schnittstelle für die Implementierung einer Offlineauswertung angestrebt. Zusätzlich zu den Visualisierungsmöglichkeiten der bestehenden Auswertungssoftware soll die Umsetzbarkeit neuer Visualisierungansätze wie beispielsweise des flyover-Modus geprüft werden.

1.4 Kapitelübersicht

Kapitel 2

Dieses Kapitel enthält eine Übersicht über die verwendeten Sensorsysteme sowie den aktuellen Stand der Technik in Bezug auf Grafikhard- und Software. Es erfolgt eine Zusammenfassung mit Bewertung der Vor- und Nachteile der verfügbaren Systeme.

Kapitel 3

Die Definition der Rahmenbedingungen für das zu entwickelnde System sowie der Auswahlprozess für die Werkzeuge als auch der Entwicklungsprozess sind in Kapitel 3 beschrieben. Weiters wird ein Überblick über die Verifikation des Systems gegeben. Den Abschluss bildet die Zusammenfassung des Systemdesignes und der Vergleich mit dem aktuellen System.

Kapitel 4

Details der Implementierung sowie dabei besonders zu berücksichtigende Faktoren werden in diesem Kapitel beschrieben.

Kapitel 5

Kapitel 5 stellt sowohl eine Zusammenfassung der erreichten Ergebnisse als auch einen Überblick über deren zukünftige Verwendung dar. Weiters werden mögliche Erweiterungen aufgezeigt und Thesen aufgestellt.

Kapitel 6 (Anhang)

Den Abschluss bildet Kapitel 6 mit Verzeichnissen der Tabellen, Abbildungen und Abkürzungen sowie ein Literaturverzeichnis.

2 Stand der Technik

2.1 Grundlagen der computergestützten Visualisierung

2.1.1 Ziele und Aufbau

Die Grundaufgabe der computergestützten Visualisierung ist die Umwandlung abstrakter Daten in eine für den Menschen leichter erfassbare Form. Die visuelle Repräsentation ermöglicht die Nutzung des menschlichen Sehsinnes zur Aufnahme der Information (vgl. /5/ S. 3 ff).

Die Kernkomponente der Visualisierung ist die Rendering-Pipeline wobei eine Verarbeitungskette als Pipeline bezeichnet wird. Der Rendering-Vorgang beschreibt dabei die Transformation einer abstrakten polygonalen Szenenbeschreibung in ein zweidimensionales Bild (vgl. /16/ S.124). Die Rendering-Pipeline besteht aus der Anwendungsstufe, der Geometriestufe und der Rasterungsstufe. Die einzelnen Hauptstufen der Pipeline können wieder in Pipelines unterteilt sein, wobei Teile davon auch parallel abgearbeitet werden können (vgl. /3/ S. 12ff).

Anwendungsstufe

In dieser Stufe hat der Entwickler die volle Kontrolle über die Abläufe und Implementierung. Die zu visualisierende Geometrie wird in der Anwendungsstufe erzeugt. Externe Eingriffe in die Visualisierung wie beispielsweise durch Eingabegeräte werden in der Anwendungsstufe bearbeitet. Die zu erstellenden Geometrien werden an die Geometriestufe übergeben (vgl. /3/ S. 14Ff, /1/, /5/, /7/ und /8/).

Geometriestufe

Die Aufgabe dieser Stufe besteht darin, die darzustellende Geometrie in grafische Primitiven umzusetzen. Typische Grafikprimitiven sind Linien, Dreiecke und Polygone. Weitere Aufgaben sind die Projektion von 3D nach 2D, die Begrenzung der auszugebenden Koordinaten (Clipping) sowie auch die Umrechnung der Koordinaten der Geometrien (Weltkoordinaten) auf Bildschirmkoordinaten (vgl. /3/ S. 16ff).

Rasterstufe

Die Ausgabegeräte für Computergrafik sind fast ausschließlich Rastergeräte. Dies bedeutet, dass der Ausgabebereich rasterförmig unterteilt ist. Die Ansteuerung erfolgt jeweils nur an diskreten Rasterpunkten. Die Daten für die Grafikausgabe werden von der

Geometriestufe als auf Bildschirmkoordinaten transformierte Primitiven übergeben. Der notwendige Prozess zur Umwandlung der Primitiven in diskrete Rasterpunkte wird als Rasterung bezeichnet (vgl. /10/ S. 54ff).

2.1.2 Grafikhardware

Basis-Hardware

Erste Grafikkarten setzten nur den Inhalt des Videospeichers (Grafik) in elektrische Signale um. Die Aufbereitung des Grafikspeichers erfolgte nur durch die Anwendungssoftware.

Erste Beschleunigungsfunktionen

Aufgrund der gestiegenen Anforderungen wurden erste Beschleunigungsfunktionen in die Hardware integriert. Ein Beispiel dafür sind die BitBlockTransfer – Operationen die die Bildpunkte eines angegebenen Quellrechteckes in ein Zielrechteck kopieren und dabei die definierte Operation ausführen. Mögliche Operationen sind unter anderem logische Verknüpfungen, kopieren, setzen, löschen (vgl. /19/).

Erweiterte Beschleunigungsfunktionen

Mit den steigenden Anforderungen an die Grafikleistung der Systeme wurden komplexere Funktionen in die Hardware integriert. Die Grafikkarten wurden für die Beschleunigung der Verarbeitung von Grafikprimitiven wie Linien, Polygonen und gefüllten Polygonen optimiert. Der Ansatzpunkt für Hardwarebeschleunigung bewegte sich von der Rasterstufe zunehmend in Richtung der Anwendungsstufe.

Stand der Technik

Aktuelle Grafikkarten bieten auch die Möglichkeit Bilder und Muster hardwarebeschleunigt auf die Oberfläche von dreidimensionalen Objekten zu projizieren sowie Oberflächeneffekte wie beispielsweise dreidimensionale Strukturen oder Spiegelungen zu erzeugen.

Ein weiteres Merkmal aktueller Grafikhardware besteht darin, dass der Anwendungssoftware eine einheitliche Programmierschnittstelle geboten wird. Funktionen die nicht bereits in der Hardware realisiert sind werden durch die Treibersoftware nachgebildet. Damit ist es möglich, Anwendungen zu erstellen die weitgehend unabhängig von der eingesetzten Grafikhardware sind. Relevant wird diese erst, wenn die erzielbare Systemperformance ausschlaggebend ist.

2.1.3 Standard-Programmierschnittstellen

Die Standardisierung einer Programmierschnittstelle, dem sogenannten Application Programming Interface (API), wird zumeist von dem Wunsch getragen, die Softwareerstellung zu vereinfachen und damit die Akzeptanz einer Plattform auszuweiten beziehungsweise die Portabilität von Software zu erhalten oder zu steigern. Im Bereich der Programmierschnittstellen für die Grafikausgabe haben sich zwei Standards etabliert:

- OpenGL
  (Verfügbar unter anderem für Microsoft Windows, Mac OS X, Linux, Unix)
- DirectX (Verfügbar für Microsoft Windows)

Aktuelle Grafikkarten aller Hersteller stellen in Kombination mit der Treibersoftware diese beiden Programmierschnittstellen zur Verfügung.

OpenGL

OpenGL entwickelte sich aus IRIS GL der Firma Silicon Graphics Inc. (SGI). Bei IRIS GL handelte es sich ursprünglich um eine Sammlung von Funktionen für zweidimensionale Grafik. Diese wurde für die Anwendung im Bereich der dreidimensionalen Grafik zur Verwendung mit den Hochleistungssystemen von Silicon Graphics erweitert. Bei diesen Systemen handelte es sich um dezidierte Grafikcomputer, die auch über spezielle Hardware zur Beschleunigung von Grafikoperationen wie beispielsweise Matrixtransformationen verfügten.

Zur Erhöhung der Portabilität von IRIS GL – Software wurde der offene Standard OpenGL definiert. Um die Verbreitung von OpenGL zu beschleunigen und die Weiterentwicklung zu sichern, wurde das OpenGL Architecture Review Board (ARB) von den Firmen Intel, SGI, Digital Equipment Corporation, IBM und Microsoft gegründet. Im Jahr 1992 wurde die OpenGL – Spezifikation 1.0 vorgestellt.

Die Khronos Group übernahm 2006 infolge der drohenden Insolvenz von SGI die Verantwortung für den OpenGL – Standard vom OpenGL ARB. Aktueller Standard ist OpenGL 3.1 (vgl. /2/ S. 33ff).

DirectX

DirectX wurde im Jahr 1995 von Microsoft vorgestellt. Ziel der Einführung von DirectX war die Verbreitung von Microsoft Windows als Betriebssystemplattform für Spiele. Dies sollte durch die Vereinfachung der Grafikprogrammierung erreicht werden, da davor überwiegend das textbasierte Betriebssystem MS-DOS mit direktem Grafikzugriff als Spieleplattform verwendet wurde.

Die DirectX – API umfasst unter anderem die Teilbereiche

- DirectDraw: Funktionen für grafische Ausgaben
- DirectInput: Schnittstelle zu Eingabegeräten
- DirectSound: Ansteuerung der Soundhardware

Die aktuelle Version ist DirectX 10.1, welche mit Windows Vista Service Pack 1 ausgeliefert wird (vgl. /10/).

2.1.4 Aktuelle Hardware-Entwicklung

Die aktuellen Grafikkarten erreichen bereits Rechenleistungen, die die Leistung der Computersysteme übertreffen, in die sie eingebettet sind. Für die Erzielung dieser Leistung werden auf den Grafikkarten zunehmend spezielle Recheneinheiten verbaut. Diese als Graphics Processing Unit (GPU) bezeichneten Einheiten sind für den massiv parallelen Einsatz optimiert. Daraus resultiert die Verwendung für Aufgabenstellungen, die von ihrer Struktur zu der Architektur der Grafiksysteme passen. Sind die GPU-Einheiten auch für andere Aufgaben verwendbar, so werden diese als General Purpose GPU (GPGPU) bezeichnet. Als Beispiele seien hier die Berechnung von dreidimensionalen finiten Differenzen und die Implementierung der Vector Signal Image Processing Library (VSIPL), einer Funktionsbibliothek zur Signalverarbeitung, genannt (vgl. /11/).

Derzeit sind vorwiegend zwei Anbieter mit GPGPU – Systemen vertreten:

- NVIDIA: Compute Unified Device Architecture (CUDA) – System (vgl. /11/)
- ATI: Stream – System (vgl. /12/)

NVIDIA mit CUDA

Hardwarebeispiel GeForce GTX 295: (vgl. /21/)

GPU-Einheiten: 2
Prozessor-Kerne je GPU: 240
GPU-Takt: 1,242 Ghz
Spitzenrechenleisung: 1 TFLOPS (single precision)
Speichertakt 1 GHz
Speicheranbindung je GPU: 448 Bit
Speichergröße je GPU: 896 MB

Entwicklungsumgebung: CUDA – Libraries, Programmierung in C, wobei die Algorithmen dann parallel in den GPU-Einheiten abgearbeitet werden können. Die Verwendung von Fortran und C++ für die Programmierung ist geplant.

Besonderheit: mit der Scalable Link Interface (SLI)-Technologie von NVIDIA können bis zu 4 GPU-Einheiten gemeinsam in einem PC-System verwendet werden, wodurch sich eine theoretische Rechenleistung von bis zu 2 TFlops ergibt.

ATI mit Stream

Hardwarebeispiel FireStream 9270: (vgl. /20/)

GPU-Einheiten: 1
Prozessor-Kerne je GPU: 800
GPU-Takt: keine Angabe
Spitzenrechenleisung: 1,2 TFLOPS (single precision), 240 GFLOPS (double precision)
Speichertakt 850 MHz
Speicheranbindung je GPU: 256 Bit
Speichergröße je GPU: 2 GB

Entwicklungsumgebung: Stream SDK, die Programmierung erfolgt mittels einer C-Erweiterung für explizit parallele Abläufe, die Brook+ genannt wird.

Besonderheit: mit der CrossFireX-Technologie von ATI können 2 oder mehr Grafikkarten parallel in einem PC-System verwendet werden.

2.2 Sensorsysteme

Die Eingangsdaten für die Visualisierung sind Messdaten von Sensoren aus der Überwachung von Schweiß-, Schneid- und Lötvorgängen in der Laserprozesstechnik. Für die Prozessüberwachung werden unterschiedliche Sensoren eingesetzt wobei die Plasmo Industrietechnik selbst zwei Sensorsysteme entwickelt hat. Zusätzlich finden in Abhängigkeit der Applikation weitere Sensorarten Verwendung.

Messsysteme der Plasmo Industrietechnik

ProcessObserver (PsO)

Der plasmo ProcessObserver dient zur Detektion von Unregelmäßigkeiten und damit von Fehlern in Schweiß- Schneid- und Bohrvorgängen. Das System besteht aus dem Sensorkopf mit den optischen Komponenten zur Einkoppelung des Signales und der Auswerteeinheit, welche die Sensorelemente und die Verarbeitungselektronik enthält (vgl. Abb.1). Der Sensor basiert auf Photodioden, welche integral sowohl über den Ort als auch über definierte Wellenlängenbereiche die Emmision des Prozesses erfassen. Die typischen Wellenlängenbereiche hierfür sind der sichtbare Bereich und der kurzwellige Infrarotbereich.

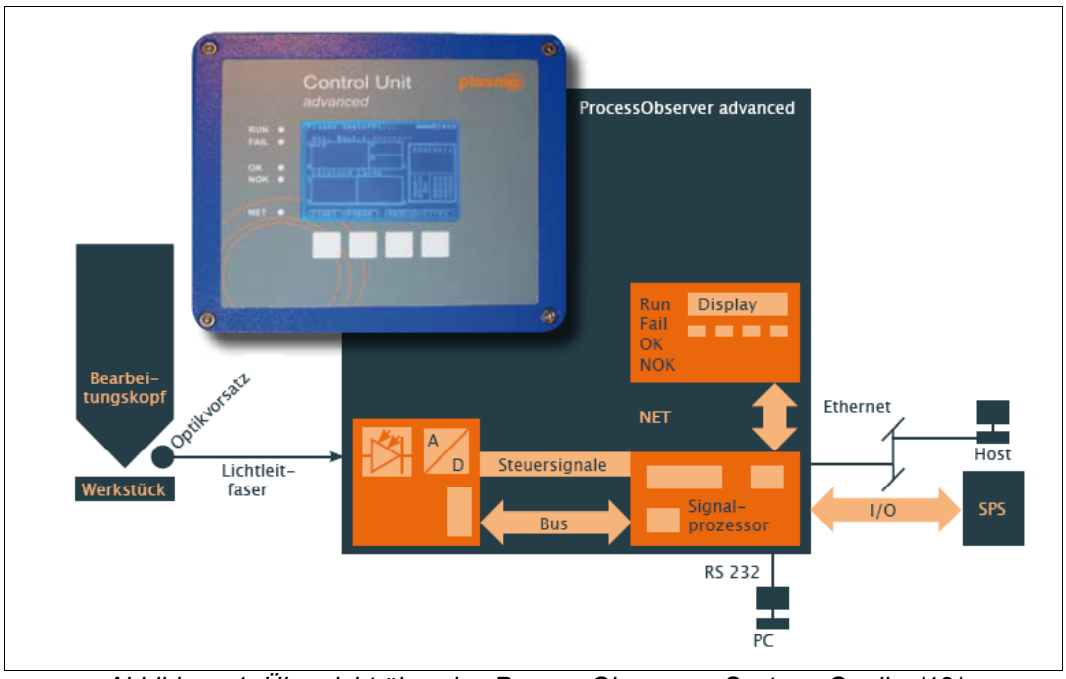


Abbildung 1: Übersicht über das ProcessObserver - System, Quelle: /13/

Abbildung 2 zeigt die Anwendung des ProcessObservers für die Kontrolle eines Laserschneidprozesses. Das Schliffbild einer Fehlerstelle sowie die korrespondierenden PsO-Daten zeigt Abbildung 3.

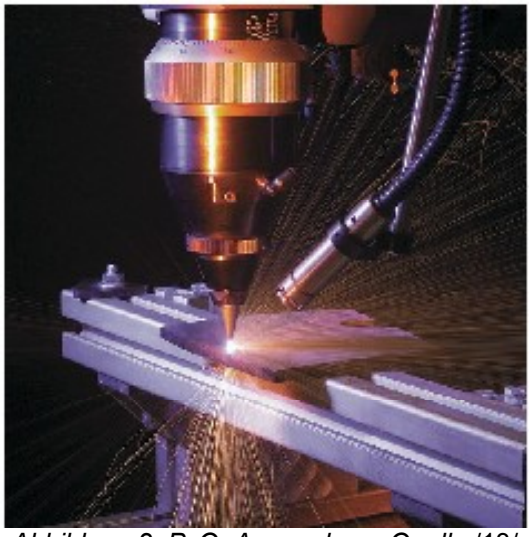


Abbildung 2: PsO–Anwendung, Quelle /13/

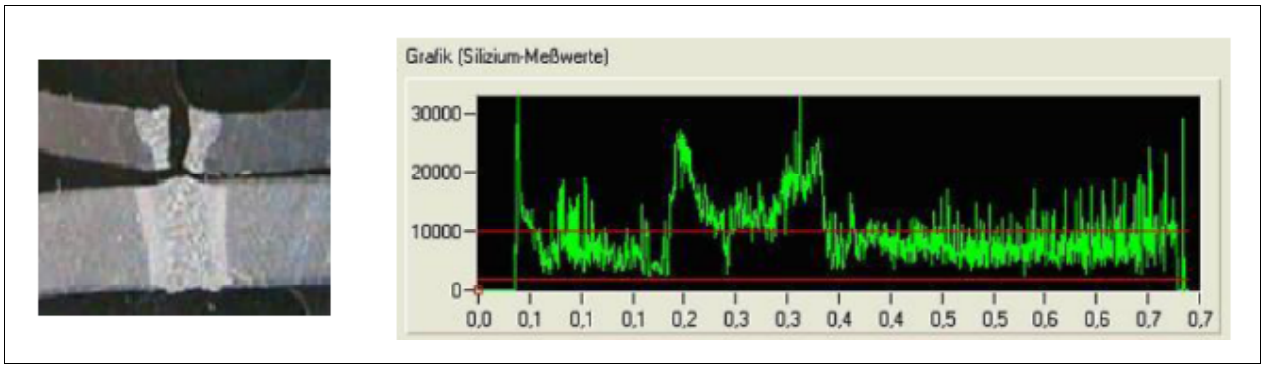


Abbildung 3: Schliffbild einer Fehlerstelle mit PsO-Daten, Quelle: /13/

Eckdaten des PsO (Auszug, die vollständige Spezifikation befindet sich im Anhang):

- Samplingrate: maximal 10 kHz
- Kanalanzahl: 2
- Datengröße je Kanal und Sample: 16 Bit
- Datenschnittstelle: Ethernet

Das vollständige Datenblatt des PsO befindet sich im Anhang in Abbildung 52.

ProfileObserver (PeO)

Das Sensorsystem plasmo ProfileObserver basiert auf dem Lichtschnittverfahren, wobei der Sensor Erweiterungen beinhaltet, die zusätzlich zur Ermittlung des 3D-Profiles auch die Aufnahme eines hochauflösenden 2D-Graustufenbildes ermöglichen. Angewendet wird dieses System zur Geometrievermessung von Schweiß- und Lötnähten. Der Laserlichtschnitt basiert auf der Detektion der Laserposition auf dem Kamerasensor wobei die Detektionsposition relativ zur Sensorachse ermittelt wird. Die Abweichung von der Sensorachse hängt sowohl vom Triangulationswinkel als auch von der Höhendifferenz zwischen dem Schnittpunkt Kameraachse - Laserlinie und dem Messpunkt ab. Das Grundprinzip des Laserlichtschittes wird in Abbildung 4 dargestellt, das simulierte Kamerabild zum Aufbau aus Abbildung 4 ist in Abbildung 5 ersichtlich.

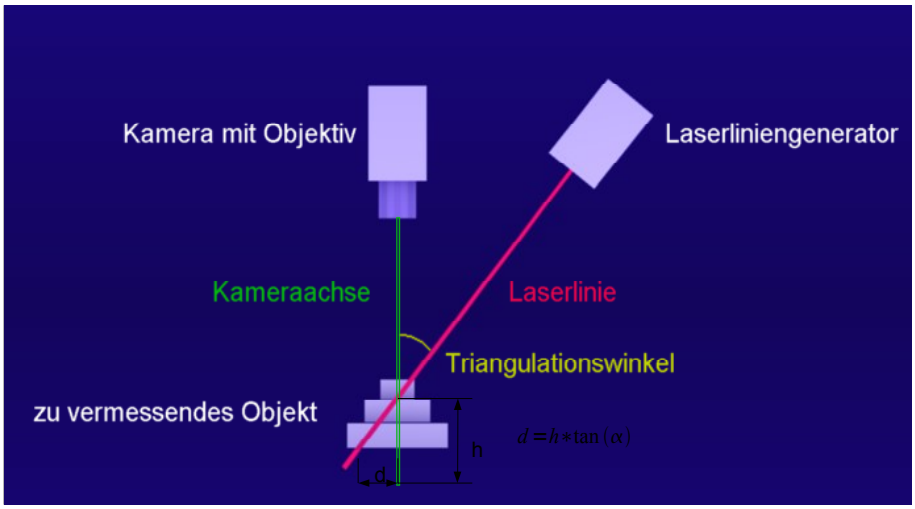


Abbildung 4: Laserlichtschnitt - Grundprinzip

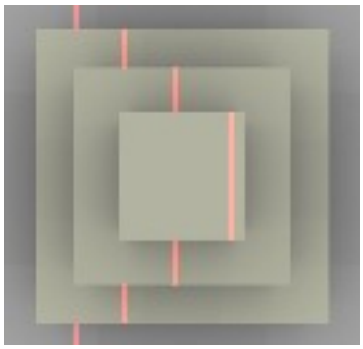


Abbildung 5: Laserlichtschnitt - simuliertes Kamerabild

Die berechneten Höhendaten einer Lichtschnitt-Auswertung werden als Profil bezeichnet.

Ein typisches Anwendungsszenario bei dem der PeO mittels Roboter über die zu kontrollierende Naht geführt wird ist in Abbildung 6 ersichtlich. Abbildung 7 zeigt eine Großaufnahme der zu prüfenden Naht mit aktiver Laserlinie.

Abbildung 6: PeO-System, Quelle:/13/

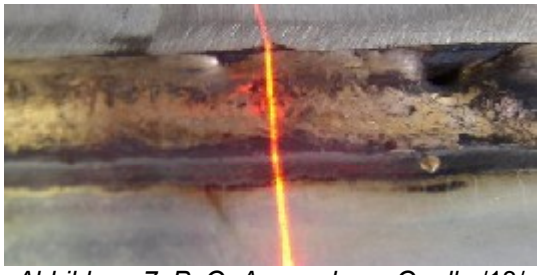


Abbildung 7: PeO–Anwendung, Quelle:/13/

Eckdaten des PeO (Auszug, die vollständige Spezifikation befindet sich im Anhang)

- Samplingrate: maximal 10kHz
- Kanalanzahl: 2 (3D- und Graustufendaten)
- Datengröße je Kanal und Profil: 1536 Punkte, 16 Bit (3D-Daten)
  3092 Punkte, 8 Bit (Graustufenbild)
- Datenschnittstelle: Ethernet

Das vollständige Datenblatt des PeO befindet sich im Anhang in Abbildung 53.

Weitere Messsysteme

Pyrometer dienen zur Erfassung von Wärmeemission, wobei bei bekanntem Emissionsgrad die Ermittlung der Temperatur ermöglicht wird. Die Anbindung an die Messdatenerfassung erfolgt wahlweise über eine serielle Schnittstelle (RS232 / RS485) oder ein Analogsignal (eingeprägter Strom, 0 bis 20mA bzw. 4 bis 20 mA). Die Messdaten des Pyrometers stellen eine Zusatzinformation für die Beurteilung der Prozessqualität dar. Dieser, in Abbildung 8 gezeigte Sensor liefert einen einkanaligen Datenstrom mit einer Auflösung von 0,1°. Das Datenblatt zu diesem Sensor befindet sich im Anhang in Abbildung 54.

Abbildung 8: Pyrometer, Quelle:/15/

2.3 Visualisierungsoftware für die Sensordaten

Software für den PsO:

Die ProcessObserver - Applikation ist in C# erstellt. Für die Darstellung der 1D-Daten kommen Steuerelemente aus dem Measurement Studio von National Instruments zum Einsatz. Zusätzlich zur Visualisierung der aufgezeichneten Daten können die PsO-Parameter für die Auswertung modifiziert und deren Auswirkung anhand einer Offline-Auswertung verifiziert werden. Abbildung 9 zeigt einen Screenshot der PeO Advanced Software.

Das Measurement Studio stellt eine integrierte Sammlung von ActiveX und .NET-Steuerelementen dar, die für die Erstellung von Mess- und Automatisierungsapplikationen verwendet werden. Als Entwicklungsumgebung für die Verwendung ist Visual Studio in den Versionen 6.0, .NET 2003, 2005 und 2008 zu verwenden. Mögliche Zielplattformen für Measurement Studio – Applikationen sind Microsoft Windows – Systeme ab Windows 98 (vgl. /22/).

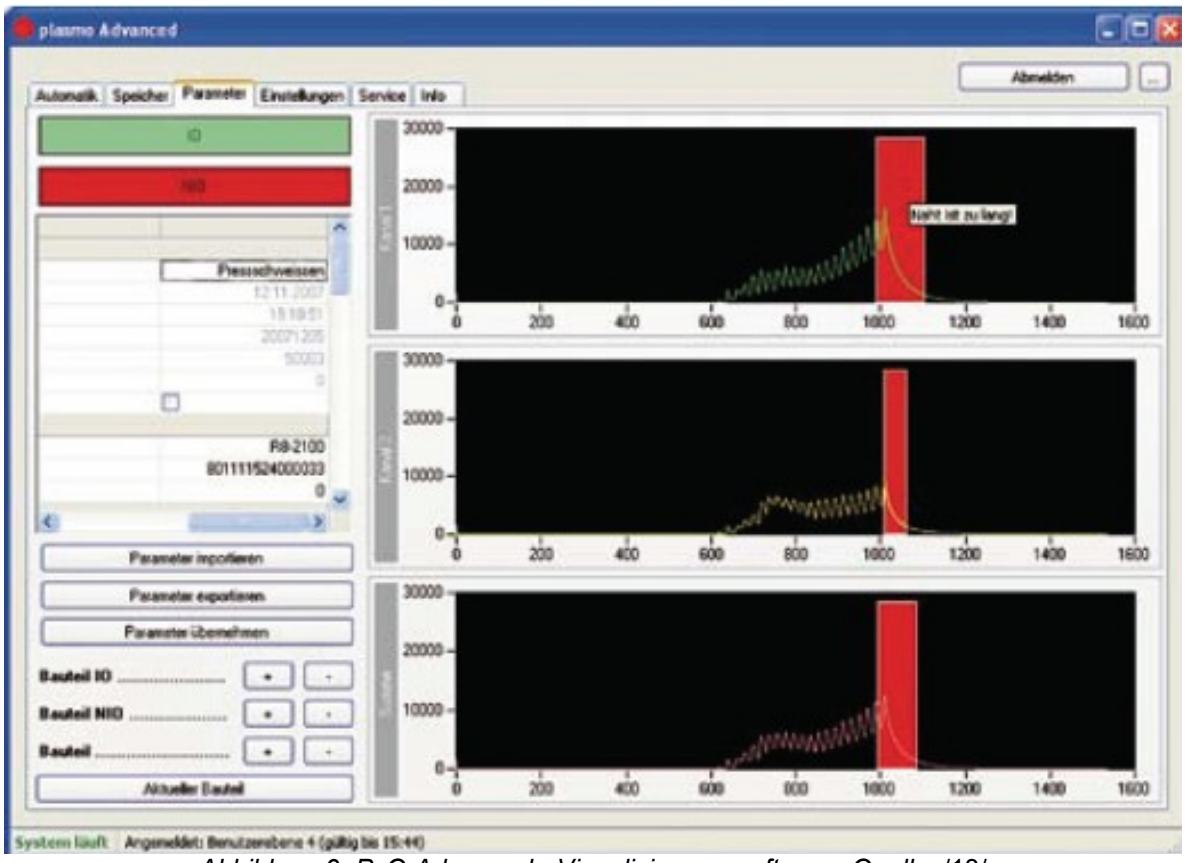


Abbildung 9: PsO Advanced - Visualisierungssoftware, Quelle: /13/

Software für den PeO:

Die Daten- und Auswertungsvisualisierung für Messdaten des PeO erfolgt aktuell mittels einer C++ - Applikation, welche für die grafische Ausgabe auf Komponenten des Measurement Studio von National Instruments zurückgreift. Zusätzlich werden für Visualisierungen im 2D – Bereich Eigenentwicklungen eingesetzt.

Die PeO-Software ermöglicht analog zur PsO-Software die Modifikation der Auswertungsparameter sowie die Überprüfung der Ergebnisse auf Basis der veränderten Parametrierung. Die grafische Oberfläche der PeO-Software ist in Abbildung 10 ersichtlich. Mit der Visualisierungssoftware sind sowohl einzelne Messdatenprofile als auch eine Fehleranzeige über den gesamten Nahtverlauf und eine zweidimensionale Ansicht der Daten sichtbar. Abbildung 11 zeigt eine zusätzliche Darstellung mit einer dreidimensionalen Ansicht der Daten und einer Einzelprofilvisualisierung.

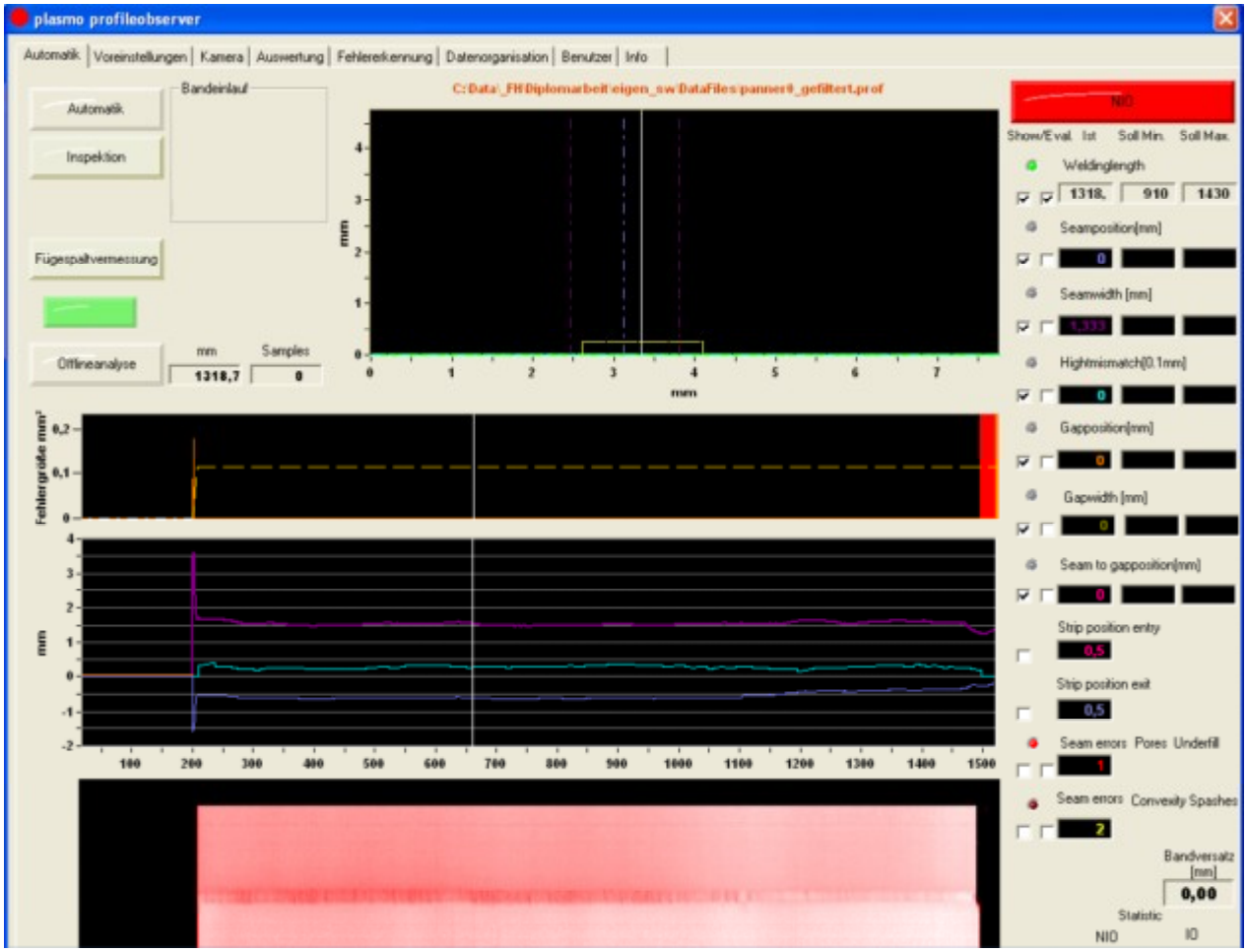


Abbildung 10: PeO–Visualisierungssoftware, Quelle: /13/

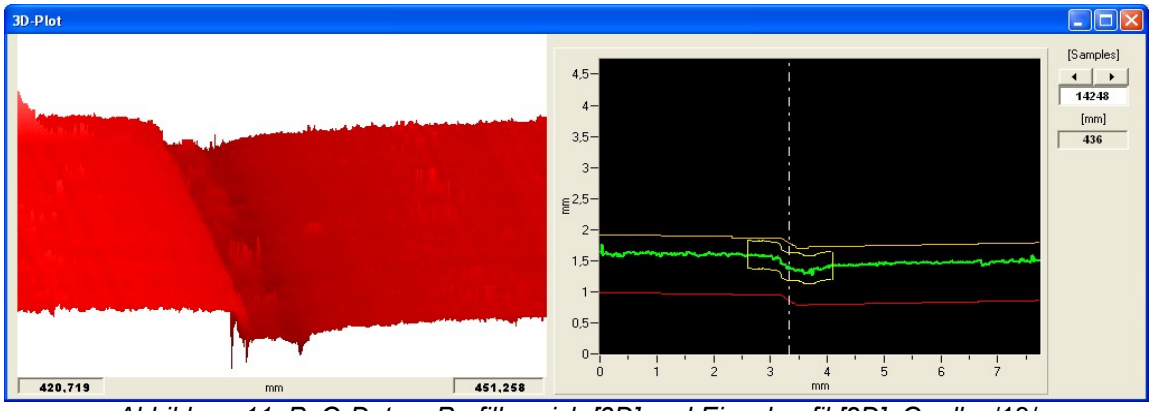


Abbildung 11: PeO-Daten: Profilbereich [3D] und Einzelprofil [2D], Quelle: /13/

Software für das Pyrometer:

Mit der dem Pyrometer beiliegenden Software erfolgt die Inbetriebnahme und der Test des Sensors. Die im Verlauf der Prozesskontrolle ermittelten Daten werden applikationsspezifisch mit dem Messrechner für den PsO bzw. PeO erfasst und ausgewertet (siehe Abbildung 12).

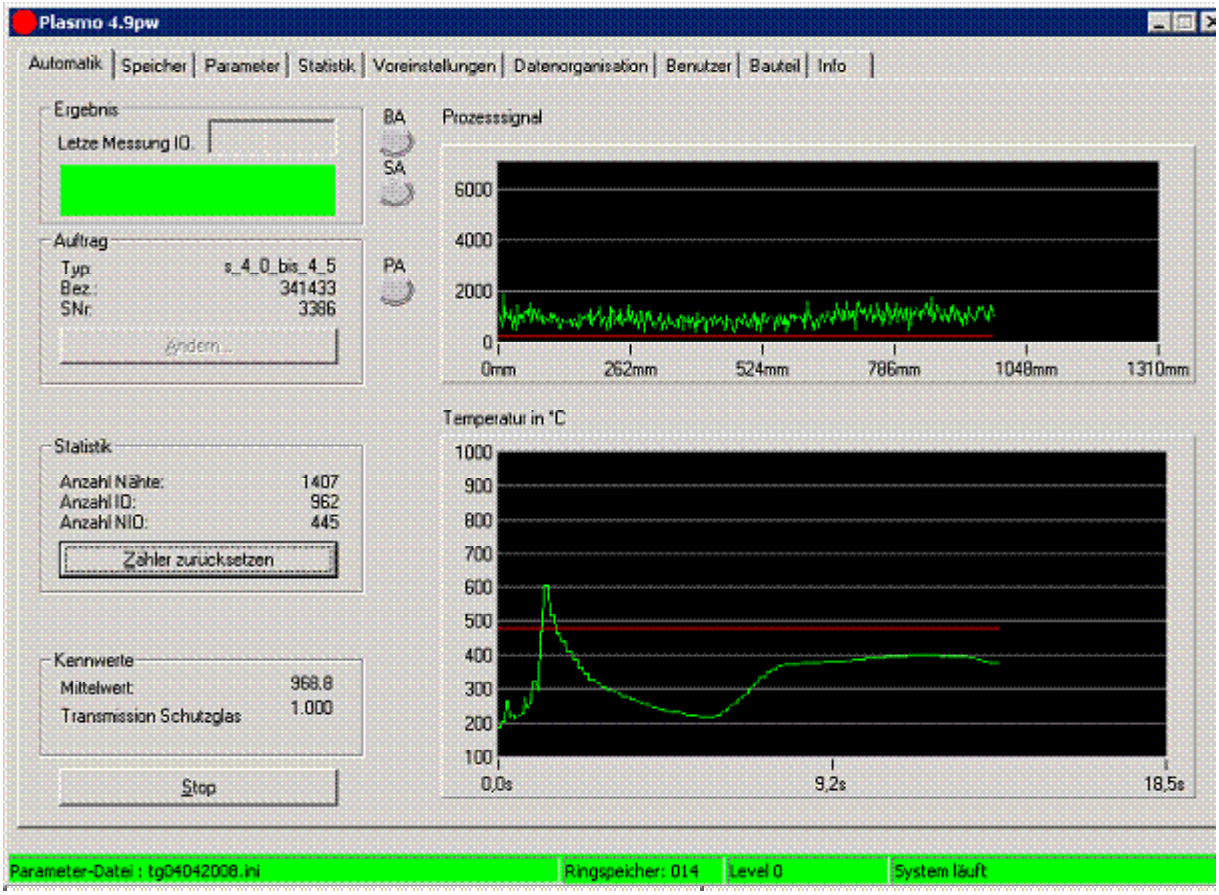


Abbildung 12: Kundenspezifische Applikation (PsO und Pyrometerdaten), Quelle: /13/

2.4 Übersicht über die zu verarbeitenden Datenmengen

Anhand der technischen Eckdaten der Sensoren lässt sich die, bei der Messdatenerfassung anfallende Datenmenge für typische Anwendungsfälle abschätzen. Die resultierende Datenmenge pro Sekunde ergibt sich aus dem Produkt von Datenpunktanzahl je Kanal, Kanalanzahl, Speicherbedarf je Datenpunkt und Messfrequenz. Die Daraus resultierenden Datenmengen sind in Tabelle 1 angeführt. Die Dauer eines Schweißvorganges beträgt in typischen Anwendungen von PsO und PeO zirka 5 bis 60s. Durch die Aufzeichnung des 3D-Profiles bei typischen Schweißanwendungen im Automobilbereich entstehen theoretische Datenmengen im Bereich von $153*10^6$ bis $1{,}8*10^9$ Byte. Reale Messungen erfolgen derzeit aufgrund der ausreichenden Auflösung mit einer Messfrequenz von maximal 4000 Hz sowie einer Breite von 512 Datenpunkten je eingelesenem Profil. Daraus entsteht ein reales Datenvolumen von $20*10^6$ bis $245*10^6$ Byte je Schweißnaht.

Sensor	Datenpunkte je Kanal [1]	Kanalanzahl [1]	Datenpunkt-Speichergröße [Byte]	Messfrequenz [s^{-1}]	Datenmenge [Byte/s]
PsO	1	2	2	10000	$0{,}04*10^6$
PeO Graustufenbild	3092	1	1	10000	$30{,}92*10^6$
PeO 3D-Daten	1536	1	2	10000	$30{,}72*10^6$

Tabelle 1: Übersicht über die zu verarbeitenden Datenmengen

Aus Tabelle 1 ist ersichtlich, dass der PsO im Vergleich zum PeO kein signifikantes Datenvolumen generiert. Deshalb wurde in dieser Arbeit vorrangig die Visualisierung der PeO-Daten behandelt.

3 Systementwicklung

3.1 Rahmenbedingungen

Aufgrund der großen Datenmengen muss das auszuwählende System die Hardwarebeschleunigung der verwendeten Grafikhardware nutzen, um eine entsprechende Interaktivität zu ermöglichen. Ziel ist es, dem Bedienpersonal zu ermöglichen 5 Fehlerstellen innerhalb von 30 Sekunden visuell zu beurteilen. Die Verwendung der Toolbox in eigenen Anwendungen, die in den Programmiersprachen C++ sowie C# erstellt wurden, muss möglich sein. Als Betriebssystem muss zumindestens Microsoft Windows unterstützt werden. Besondere Beachtung ist auch den lizenzrechtlichen Rahmenbedingungen für die Nutzung der Toolbox zu schenken, da die damit erzeugte Software kommerziell genutzt wird.
Die Verfügbarkeit für unterschiedliche Hardware- und Betriebssystemplattformen sowie auch die Verfügbarkeit der Quellcodes waren Nebenziele bei der Auswahl.

3.2 Auswahl der Entwicklungsumgebung

Im ersten Schritt wurde eine Recherche bezüglich verfügbarer Toolboxen, die für eine Visualierungslösung geeignet erscheinen, durchgeführt. Es wurden sowohl die unterstützten Betriebssysteme als auch die Lizenzierung und die Form der Verfügbarkeit (als Quellcode oder ausführbare Datei) erhoben. Da die vorliegende Diplomarbeit als Kernpunkte den Entwurf und die Implementierung des Visualisierungssystems aufweist, wird an dieser Stelle nur die ausgewählte Toolbox sowie die ausschlaggebenden Faktoren aufgezeigt.
Für den Aufbau der neuen Visualisierung wurde das Visualization Toolkit (vtk) der Firma Kitware ausgewählt. Neben der Unterstützung für verschiedene Betriebssysteme (Microsoft Windows, Linux, Unix, Mac OSX) handelt es sich um eine freie, im Quellcode verfügbare Toolbox, die neben der Funktionalität für die Visualisierung auch leistungsfähige Funktionen für die Bildvorverarbeitung und den Zugriff auf Standarddateiformate besitzt. Der kommerzielle Einsatz ist gestattet wobei nur wenige Funktionen geschützt sind. Diese sind nur nach Klärung der Verwendungsmodalitäten (Verwendungsart und optional Vergütung) mit den Autoren kommerziell nutzbar.
Dokumentationen in elektronischer und gedruckter Form sind ebenso erhältlich wie

Schulungen. Durch diese Optionen wird die Einarbeitungsphase deutlich reduziert. Die Recherche zeigte auch Anwendungen des vtk sowohl im medizinischen Bereich als auch für die Simulationsvisualisierung (vgl. /23/).

3.3 Grundeigenschaften des gewählten Systems

Das vtk ist ein plattformunabhängiges Klassensystem das für verschiedene Aufgaben im Bereich der Datenverarbeitung und Visualisierung eingesetzt werden kann. Basis des vtk ist ein Datenflusskonzept das auf den Grundelementen *Datenobjekt* und *Algorithmus* beruht. Datenobjekte repräsentieren Informationen unterschiedlicher Datentyen. Sie stellen einen Container dar der zusätzlich zu den geometrischen und topologischen Daten auch Attributdaten speichern kann. Algorithmen beziehungsweise Filter wie sie auch allgemein genannt werden verarbeiten Datenobjekte und erzeugen daraus neue Datenobjekte. Durch die Verknüpfung von Datenobjekten und Filtern entsteht das Datenflussnetzwerk (vgl. Abbildung 13).

Datenquellen generieren Datenobjekte wobei sie im Gegensatz zu Filtern eingangsseitig keine Datenobjekte entgegennehmen. Typische Datenquellen sind Reader, die Daten aus Dateien einlesen und daraus Datenobjekte generieren. Datensenken nehmen Daten-obekte entgegen ohne jedoch neue Datenobjekte zu generieren. Writer sind Datensenken die die übergebenen Datenobjekte in Dateien ablegen. Mapper sind spezielle Datensenken die Datenobjekte in grafische Daten umwandeln, die danach von der Grafikeinheit gerendert und ausgegeben werden.

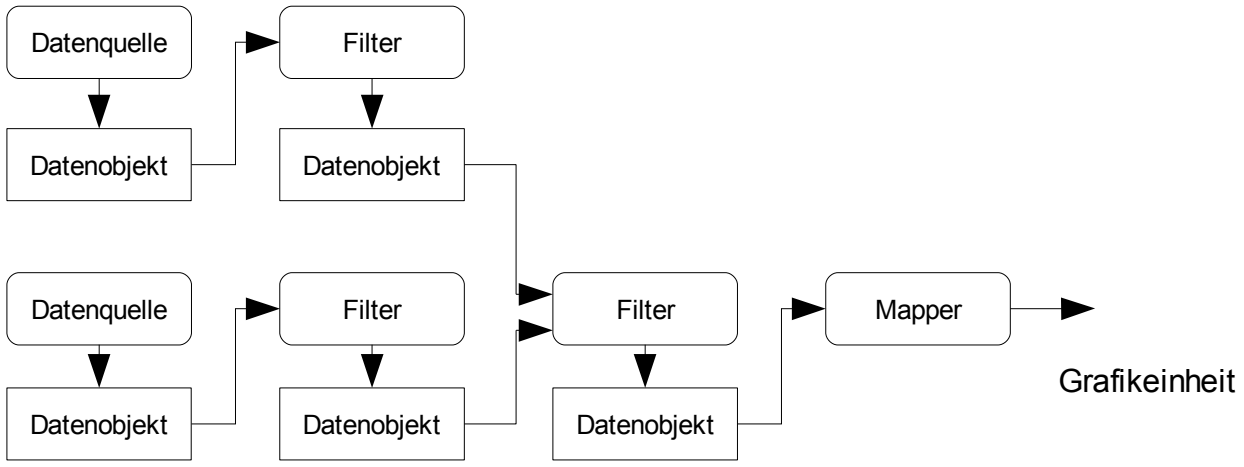


Abbildung 13: Datenflußmodell des vtk

In Abbildung 14 ist der gesamte Verarbeitungsprozess von den Daten bis zur Bildausgabe dargestellt.

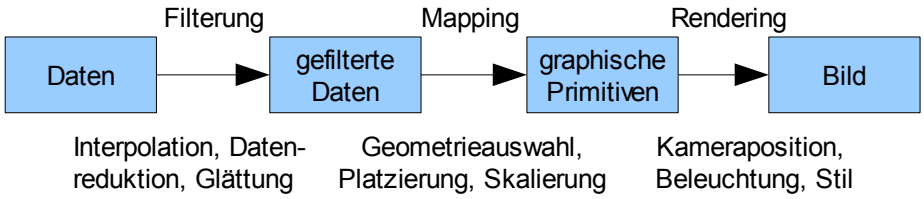


Abbildung 14: Grundprinzip der Datenverarbeitung bei vtk, Quelle: /14/

Das vtk setzt intern auf die OpenGL – Programmierschnittstelle auf und kann damit die Hardwarebeschleunigung aktueller Grafikkarten nutzen. Die Verkettung der einzelnen Verarbeitungsstufen von den Datenquellen bis zum Renderingprozess wird als Visualisierungspipeline bezeichnet. Die grafischen Primitiven die die Ausgabe des Mappers darstellen, werden dem sogenannten Actor zugeführt. Dieser ist für die geometrischen Transformationen (Verzerrung, Skalierung, Rotation, Translation) sowie Beleuchtung und Farbgebung verantwortlich. Dadurch können mehrere Instanzen eines Grafikobjektes in unterschiedlicher Erscheinungsform dargestellt werden (vgl. Abb. 15).

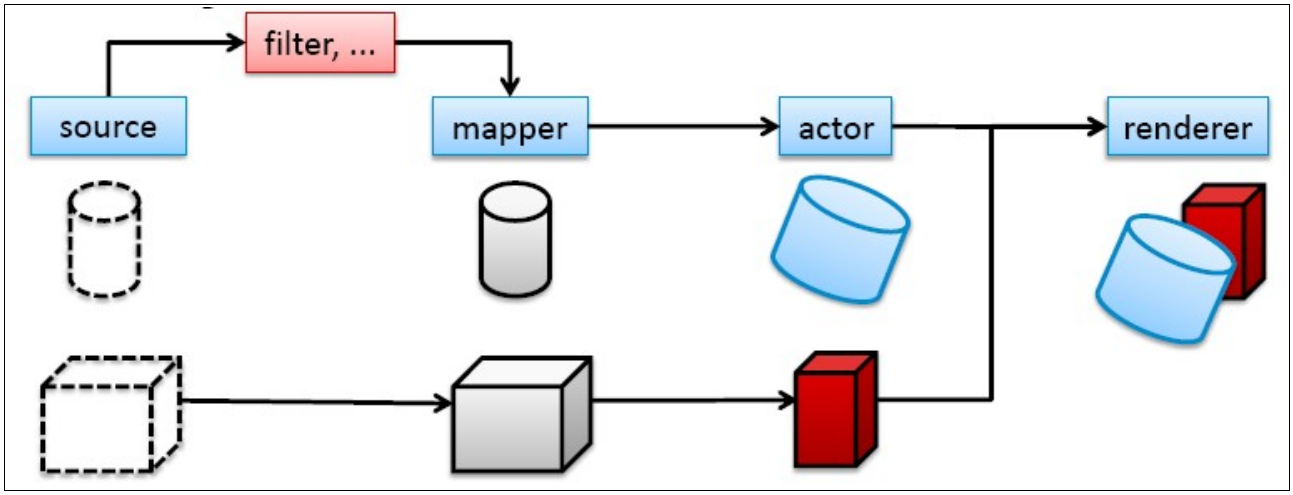


Abbildung 15: Visualisierungspipeline im vtk, Quelle: /14/

Die im vtk angewandte Form der Pipelineausführung wird als *lazy evaluation* bezeichnet (vgl. /6/ S. 24 ff). Dies bedeutet, dass die Ausführung der einzelnen Filter- und Mapperstufen nicht sofort erfolgt, sondern erst nach Anforderung von nachfolgenden Verarbeitungsstufen die das Ergebnis dieser Stufe benötigen. Diese Aktualisierung kann

jedoch auch durch die Anwendung gezielt ausgelöst werden. Dieser Aspekt ist besonders bei der Ermittlung von Verarbeitungszeiten zu beachten.

3.4 Klassenentwurf

3.4.1 Benennungskonventionen

Die eingeführten Benennungskonventionen dienen der besseren Erfassbarkeit von Zugehörigkeit und Funktion der Klasse wobei jeder Klassenname zumindestens aus einem Prefix und einer Datenartspezifikation besteht. Der Prefix dient zur Identifikaiton der Plasmo Industrietechnik – Klassen während durch die Prefixerweiterung die Zugehörigkeit zu einer Funktionsgruppe (Datenzugriff, Visualisierung) angegeben wird. Die nachfolgende Datenartspezifikation nimmt Bezug auf die zugrundeliegende Datenstruktur. Zusätzlich wird noch eine optionale Funktionsspezifikation angefügt, wobei eine fehlende Funktionsspezifikation auf eine reine Datenverwaltungsklasse hinweist.

Prefix:

PIT	Prefix für PlasmoIndustrieTechnik
SD_	Prefixerweiterung für SensorData
VIS_	Prefixerweiterung für Visualisation

Datenartspezifikaiton:

DataMem	internes, sensorunabhängiges Format
Data	externes, nicht näher spezifiziertes Format

Funktionsspezifikation (Beispiele):

Reader	Lesefunktion, nicht näher spezifiziert
ReaderExt	Lesefunktion für Daten in externem Format, Zugriff nicht spezifiziert
ReaderExtFile	Lesefunktion für Daten in externem Format, Quelle ist eine Datei

So implementiert beispielsweise die Klasse *PITSD_DataReaderExtFile* den Zugriff auf Datendateien. Die Klasse selbst gehört der Gruppe der Sensordatenzugriffsklassen an.

3.4.2 Sensordatenzugriff

Da der PsO und der PeO einen Großteil der Anwendungen abdecken und die zugekauften Sensoren (z.B. Pyrometer) herstellerspezifische Protokolle und Datenformate verwenden, wurde in dieser Arbeit vorrangig der Zugriff auf Sensoren der Plasmo Industrietechnik betrachtet. Das Konzept für den Datenzugriff wurde jedoch so aufgebaut, dass herstellerspezifische Eigenschaften in den dafür abgeleiteten Klassen implementiert werden können.

ProcessObserver

Die PsO-Applikation generiert eindimensionale Zeitreihen wobei die Zeitdifferenz zwischen den Messwerten über die Samplingrate gegeben ist. Um vom Relativbezug durch die Messfrequenz auf die absoluten Zeitwerte umrechnen zu können, wird ein Zeitsignal zu Beginn der Messung erfasst. Zusätzlich werden die Informationen mit der Qualitätsbewertung (fehlerfrei oder fehlerhaft) generiert. Die Messdaten werden zusammen mit dem Parametersatz, der Qualitätsbewertung und der Zeitreferenz in einer Textdatei abgelegt.

ProfileObserver

Die PeO-Applikation generiert 2D und 3D – Daten, die zusammen mit dem aktuellen Parametersatz in einer Datei abgelegt werden. Die Parameter werden als Text, die Messdaten im Anschluss daran in binärer Form ablegt. Zusätzlich wird bei der Auswertung ein Datensatz je Profil generiert, der folgende Daten beinhaltet:

- fortlaufende Nummer des aktuellen Messdatenprofiles
- Position der linken Schweißnahtgrenze innerhalb des Profiles
- Position der rechten Schweißnahtgrenze innerhalb des Profiles
- Kennzahl Profil-Auswertung 1 (z.B. Fehlerwert)
- Kennzahl Profil-Auswertung 2 (z.B. Temperatur)

Die zeitliche Zuordnung relativ zum Start der Messung erfolgt über die Messfrequenz, der absolute zeitliche Bezug wird analog zum PsO über ein Zeitsignal am Beginn der Messung hergestellt.

Zugriffsarten

Dies Sensordaten von PsO und PeO können sowohl im Speicher vorliegen, als Datei auf einem Datenträger gespeichert sein oder als Datenstrom über das Netzwerk zugeführt werden. Der Zugriff im Speicher ist nach der lokalen Datenerfassung von einem Sensor oder einem lokalen Ladevorgang möglich, wobei „lokal“ die Ausführung der Messdatenerfassung beziehungsweise des Datenträgerzugriffes auf jenem Computer bezeichnet, der auch auf die Daten für Visualisierung zugreift. Für die Datenzuführung über das Netzwerk wird auf socketbasierte Verbindungen zurückgegriffen.
Im Folgenden werden die wesentlichen Klassen, die für den geordneten Zugriff auf die Daten entworfen wurden, vorgestellt.

Basisklassen

Klasse PITSD_DataMem

- Implementiert die Struktur der sensorunabhängigen Darstellung der Daten und Verwaltungsinformationen im Speicher sowie auch die entsprechenden Zugriffsmethoden. Damit wird die sensorunabhängige Schnittstelle zwischen den Messdaten und der Visualisierung gebildet.

Klasse PITSD_DataMemReader

- Implementiert den Zugriff auf *PITSD_DataMem* – Objekte

Klasse PITSD_DataReaderExt

- Implementierung der Basisklasse für den Zugriff auf externe Datenquellen, wobei extern bedeutet, dass der Aufbau der Datenstrukturen von einer anderen Anwendung vorgegeben ist. Diese Klasse dient zur Transformation der Daten von einem sensorspezifischen in ein sensorunabhängiges Format (PITSD_DataMem).

Abgeleitete Klassen

Klasse PITSD_DataReaderExtFile

- Von *PITSD_DataReaderExt* abgeleitete Klasse die den Zugriff auf Datendateien implementiert.

Klasse PITSD_DataReaderExtMem

- Von *PITSD_DataReaderExt* abgeleitete Klasse die den Zugriff auf Daten, die bereits im sensorspezifischen Format im Speicher vorliegen, ermöglicht.

Klasse PITSD_DataReaderExtSocket

- Von *PITSD_DataReaderExt* abgeleitete Klasse die den Zugriff auf Daten über eine Socketverbindung implementiert.

Diese abgeleiteten Klassen dienen als Basisklassen für die Implementierung der Sensorspezifika. So implementiert die Klasse *PITSD_DataReaderExtFilePeO* den Zugriff auf Datendateien des ProfileObserver.

Abbildung 16 gibt einen Überblick über die Klassen für den Datenzugriff wobei aus Gründen der Übersichtlichkeit der Klassenprefix *PITSD_* weggelassen wurde.

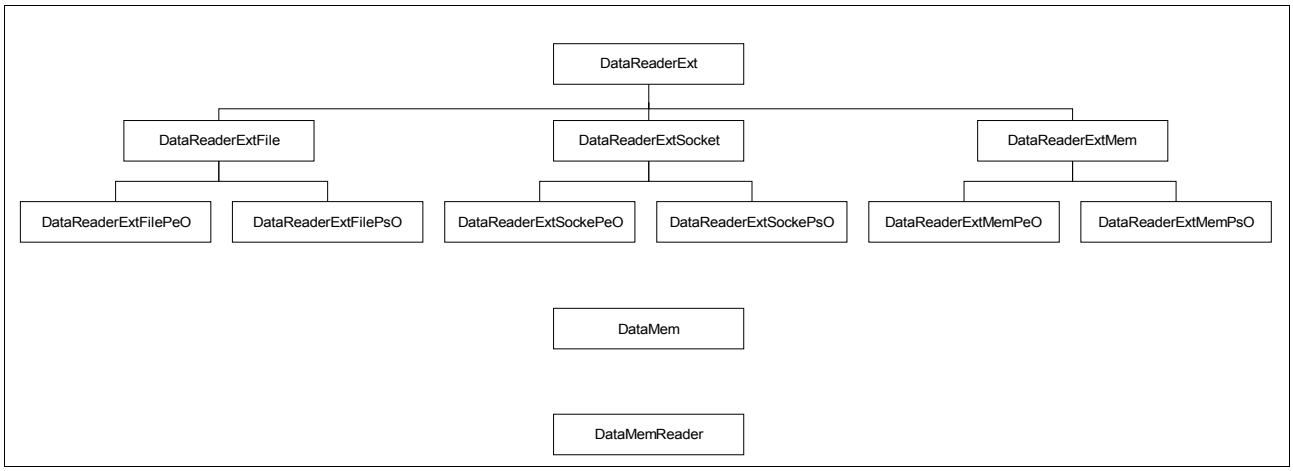


Abbildung 16: Klassenübersicht: Datenzugriffsklassen (Prefix PITSD_)

3.4.3 Visualisierung

Eine weitere Gruppe von Klassen generiert aus den Daten von DataMem-Objekten jene Objektinstanzen und Strukturen, die der Renderingstufe zugeführt werden. Die implementierten Klassen erzeugen aus den 1-, 2- und 3dimensionalen Daten in Abhängigkeit der geforderten Darstellungsart entsprechende grafische Repräsentationen. Die entstandenen grafischen Primitiven werden an ein klasseninternes Mapperobjekt übergeben, welches mit einem Actor-Objekt verbunden ist (vgl. Abbildung 15). Diese Actor-Objekte bilden die Ausgangsdaten der Visualisierungsklassen und werden in der Anwendung an die Anzeigeklasse übergeben, die die Anbindung an die grafische Benutzeroberfläche herstellt. Abbildung 17 zeigt die Übersicht über die Visualisierungsklassen.

Basisklassen:

Klasse PITVIS_DataMemVis

- Implementiert die Basisklasse für die Visualisierung. In dieser Klasse sind die Verarbeitungsabläufe von der Umwandlung in Primitiven bis zur Erzeugung und Initialisierung der Mapper- und Actor-Objekte abstrakt implementiert. Die konkrete, auf die Visualisierungsart abgestimmte Implementierung erfolgt in den abgeleiteten Klassen.

Klasse PITVIS_DataDisplay

- Implementiert die Basisklasse für die Anbindung an die grafische Benutzeroberfläche. Die betriebssystemabhängigen Zugriffe werden in den abgeleiteten Klassen implementiert.

Abgeleitete Klasen:

Klasse PITVIS_DataMemVis1D

- Ableitung der Klasse *PITVIS_DataMemVis* die die Visualisierung eindimensionaler Datenreihen implementiert.

Klasse PITVIS_DataMemVis2D

- Ableitung der Klasse *PITVIS_DataMemVis* die die Visualisierung zweidimensionaler Datenreihen implementiert.

Klasse PITVIS_DataMemVis3D

- Ableitung der Klasse *PITVIS_DataMemVis* die die Visualisierung dreidimensionaler Datenreihen implementiert.

Klasse PITVIS_DataDisplayWin

- Ableitung der Klasse *PITVIS_DataDisplay* die die Anbindung der Visualisierung an das Betriebssystem Microsoft Windows herstellt.

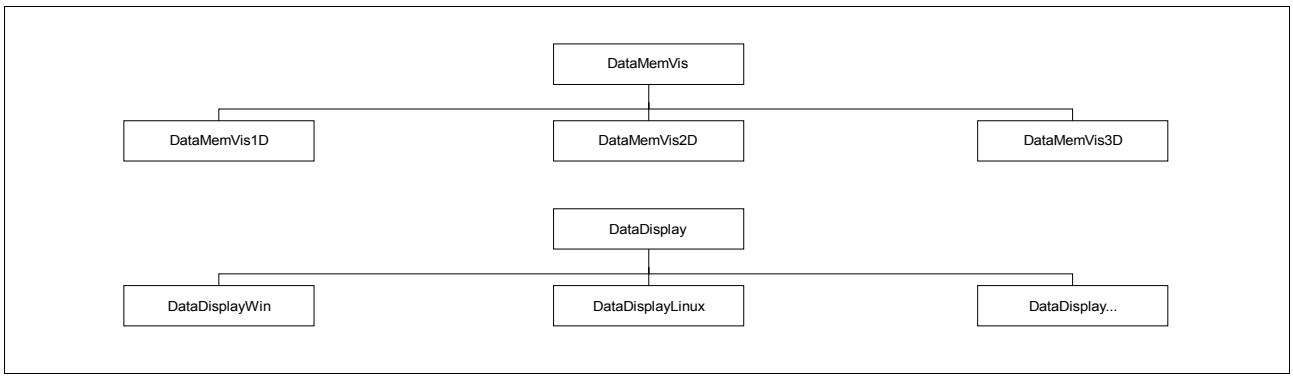


Abbildung 17: Klassenübersicht Visualisierungsklassen (Prefix PITVIS_)

Weiterführende Informationen zum objektorientierten Entwurf sind in /8/ und /9/ zu finden.

3.5 Neue Visualisierungsansätze

3.5.1 Farbzuordnung

Der bisherige Ansatz beruht darauf, jedes Profil als einen Block zusammengehörender Datenpunkte zu betrachten und zu visualisieren. Die Zuordnung zwischen Höhenwert (Z-Koordinate) und dargestellter Farbe erfolgt mittels einer Tabelle, wobei jedem Z-Wert eine Farbkombination bestehend aus je einer Komponente für Rot, Grün und Blau zugewiesen wird. Bei aktuellen Visualisierungssystemen kann zumeist nur eine Zuordnungstabelle (Lookup-Table, LUT) an einen 3D-Datensatz zugewiesen werden. Diese Zuordnung erfolgt daher sowohl unabhängig von der Position des Messpunktes innerhalb des Profiles als auch von der Position innerhalb der Zeitreihe.
Bezogen auf die Anwendung lassen sich die zwei Bereiche “Rand” und “Naht” unterscheiden. Zusätzlich kann jedem Bereich ein Fehlerstatus zugewiesen werden, wobei derzeit nur der Fehlerstatus je Profil, jedoch nicht je Profilpunkt bereitgestellt wird. Aus der Kombination von zwei Bereichen (“Rand” und “Naht”) mit zwei Fehlerstati (“fehlerbehaftet” und “fehlerfrei”) ergeben sich die vier mögliche Gruppen

- Randbereich – fehlerfrei (entspricht Gruppe 1 in Abb. 18)
- Nahtbereich – fehlerfrei (entspricht Gruppe 2 in Abb. 18)
- Nahtbereich – fehlerbehaftet (entspricht Gruppe 3 in Abb. 18)
- Randbereich – fehlerbehaftet (entspricht Gruppe 4 in Abb. 18)

Diese vier Gruppen werden in der bisher eingesetzten Lösung nicht berücksichtigt. Der neue Visualisierungsansatz basiert auf einer Auftrennung dieser direkten Zuordnung zwischen Z-Wert und Farbinformation, ohne jedoch die farbliche Differenzierung innerhalb der Gruppe aufzugeben. Ermöglicht wird dies durch die Verwendung eines zusätzlichen Skalares je Datenpunkt, der für die Zuordnung der Farbe verwendet wird. Die verwendete LUT wird aus bis zu vier Farbteilbereichen zusammengesetzt. Der Wert des Skalars resultiert aus der Summe des gruppenabhängigen Offsets und dem Z-Wert.

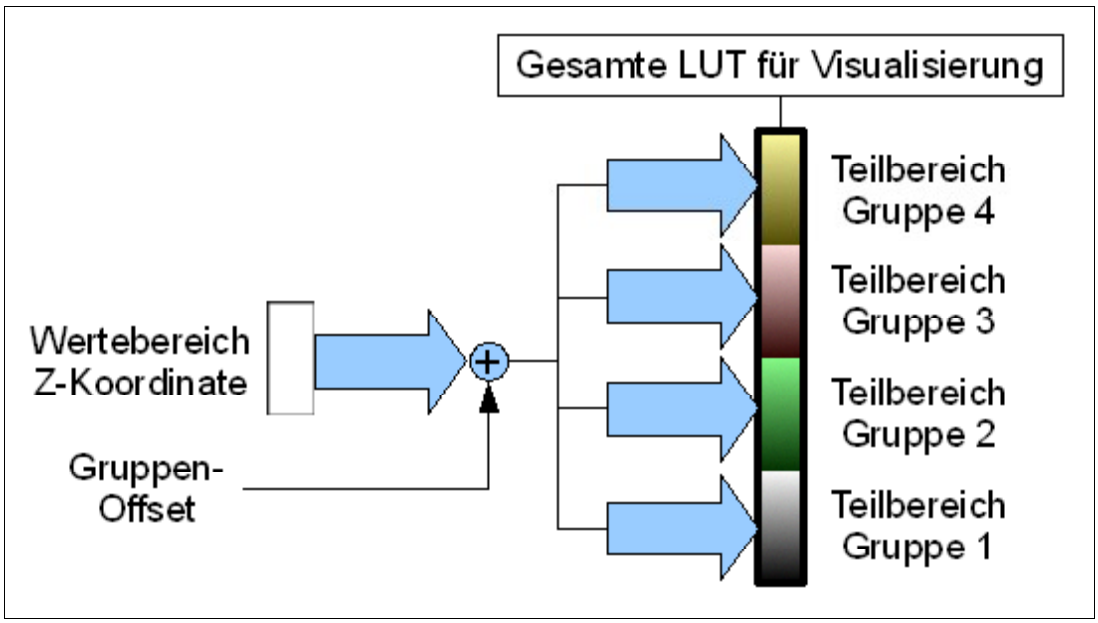


Abbildung 18: Prinzip der Mehrbereichs-LUT

Besitzt die Z-Koordinate beispielsweise einen Wertebereich von 0 bis 255 so wird eine LUT generiert, bei der die Werte 0 bis 255 den Farbinformationen für schwarz bis weiß zugeordnet werden („fehlerfreier Randbereich“). Die Werte von 256 bis 511 werden mit dem Farbverlauf von dunkelgrün bis hellgrün belegt („fehlerfreier Nahtbereich“). Die dritte Gruppe (Werte 512 bis 767) werden mit dem Farbverlauf von dunkelrot bis hellrot besetzt („fehlerbehafteter Nahtbereich“). Der vierte Teilbereich, der den Werten von 768 bis 1023 entspricht, wird mit den Farbwerten von dunkelbraun bis hellbraun belegt („fehlerbehafteter Randbereich“). Abhängig von der Bewertung wird der Gruppenoffset mit den Werten 0, 256, 512 oder 768 besetzt. In Summe mit dem Wert der Z-Koordinate ergibt sich der Tabellenindex innerhalb der LUT.

Abbildung 19 zeigt eine Fehlerstelle in der derzeit implementierten Darstellung wobei die Höheninformation direkt in eine Graustufe umgewandelt wird. Im Gegensatz dazu zeigt Abbildung 20 diese Fehlerstelle mit aktiver Mehrbereichs-LUT. In dieser Darstellung werden nur die Gruppen 1 bis 3 verwendet. Der Nahtbereich wird grün eingefärbt, als fehlerhaft bewertete Bereiche werden rot markiert. Der Randbereich wird in Graustufen dargestellt, wobei hier nicht zwischen fehlerbehaftet und fehlerfrei differenziert wird (Wunsch der Plasmo Industrietechnik).

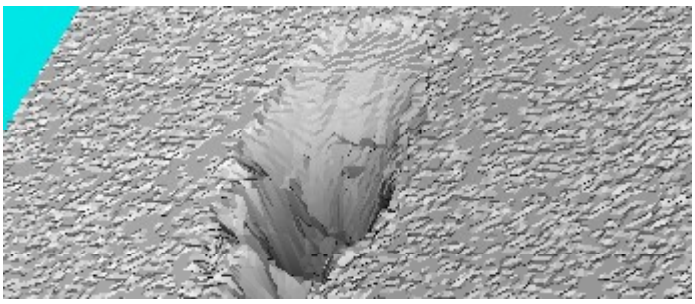


Abbildung 19: Bildausschnitt: 3D-Daten mit Einfach-LUT

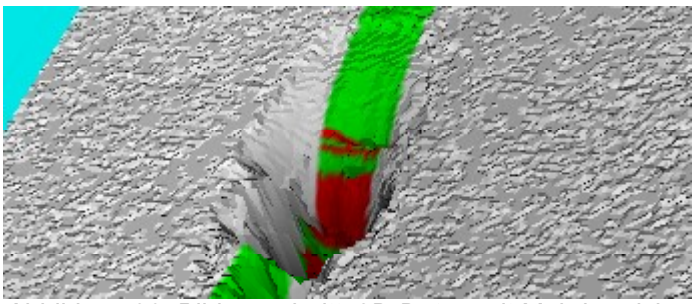


Abbildung 20: Bildausschnitt: 3D-Daten mit Mehrbereichs-LUT (3 Bereiche)

3.5.2 Einblendung von Zusatzdaten

Um noch mehr Informationen für die visuelle Bewertung der Messdaten zur Verfügung zu haben, wurde das Visualisierungskonzept so erweitert, dass orthogonal zur Darstellungsebene der 3D-Daten eine zweite Darstellungsebene eingeführt wurde. Diese Zusatzebene kann beispielsweise zur Einblendung von Messdaten des PsO oder der Größe des von der PeO-Auswertung ermittelten Fehlerwertes verwendet werden. Abbildung 21 zeigt den fehlerhaften Beginn einer Schweißnaht mit eingeblendetem PeO-Fehlerwert.

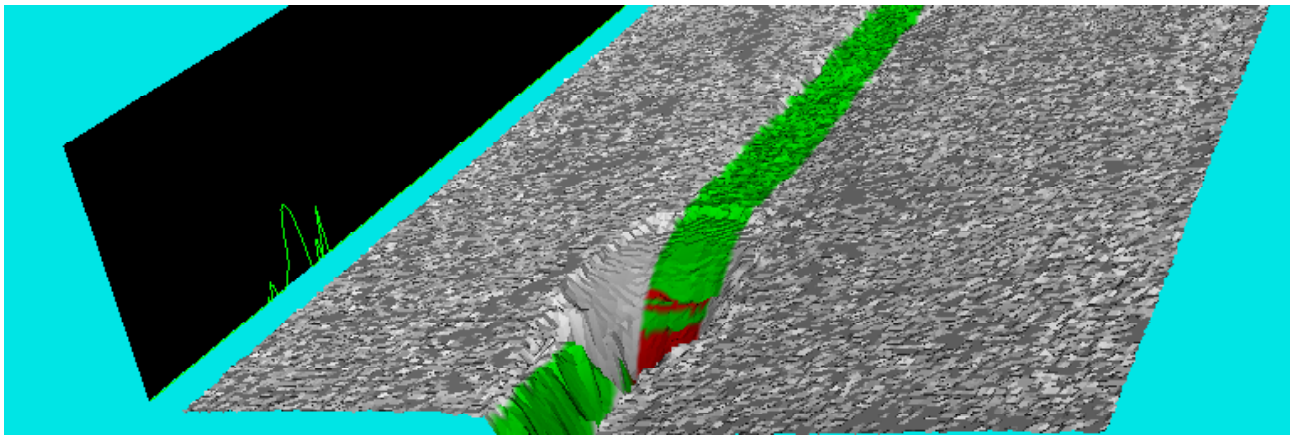


Abbildung 21: Bildausschnitt: Darstellung einer Schweißnaht mit Einblendung des PeO-Fehlerwertes

Durch die Darstellung der zusätzlichen Auswertungsdaten kann die Größe des Fehlers auch abgeschätzt werden ohne die Visualisierungsperspektive zu verändern. Da die Visualisierung nur eine Linie je Datenprofil generiert wirkt sich der zusätzliche Zeit- und Speicherbedarf für die Erstellung und Anzeige dieser Daten nicht signifikant auf die Gesamtzeit für die Visualisierung aus.

3.6 Bewertung der Leistung der 3D-Visualisierung

Um die Bewertung der unterschiedlichen Darstellungsarten und Rendermodi sowie Untersuchungen bezüglich der Skalierbarkeit durchführen zu können, musste zuerst eine entsprechende Testapplikation erstellt werden, die den raschen Zugriff auf diese Parameter sowie Werkzeuge zur Ermittlung der Ergebnisse bereitstellt. Dies betrifft sowohl die objektiven Ergebnisse wie Verarbeitungszeiten als auch subjektiv beurteilte Kriterien wie das Erscheinungsbild.

3.6.1 Kriterien und Testplanung

Die Leistung des Visualisierungssystems kann nach unterschiedlichen Gesichtspunkten bewertet werden. Die Gewichtung der einzelen Kriterien hängt von der angestrebten Anwendung ab und kann nicht unabhängig von den Beschränkungen des ausführenden Systems betrachtet werden. Neben den Kriterien selbst sind die Auswahl der Testhardware und die Zusammenstellung der Testszenarien wesentliche Grundelemente der Analyse.

Testkriterien:

Als Kriterien wurden der Speicher- und Zeitbedarf ausgewählt wobei der Speicherbedarf gerennt nach Maximal- (MemMax) und Endwert (MemFinal) aufgezeichnet wurde. Der Zeitbedarf wurde für einzelne Verarbeitungsschritte aufgezeichnet (vgl. Abb. 22). In der Auswertung wurden die Teilzeiten zu den Bereichen Vorverarbeitung (Preprocessing) und Grafikausgabe (Draw) zusammengefasst. Der Grund für die geteilte Zeitmessung liegt in den unterschiedlichen Komponenten, die die einzelnen Verarbeitungsschritte abarbeiten. So wird das Preprocessing Rechenleisung der Zentraleinheit (CPU), während die Grafikausgabe überwiegend die Grafikkarte belastet. Zusätzlich wurde die Gesamtzeit (SumProc) für die Bildausgabe ermittelt, da dies jene Zeit ist, die vom Anwender der Visualisierung wahrgenommen wird. Die Zeit für den Ladevorgang der Messdaten wurde

hier nicht berücksichtigt, da diese nicht direkt im Zusammenhang mit der Visualisierungsleistung steht.

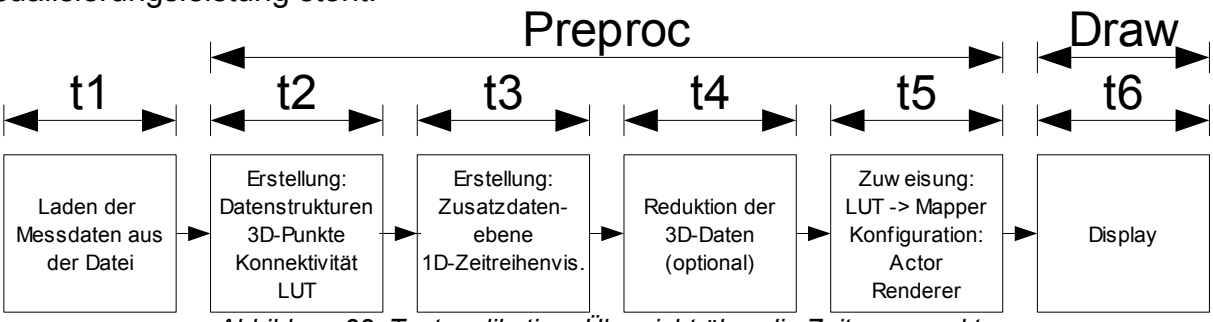


Abbildung 22: Testapplikation: Übersicht über die Zeitmesspunkte

Testszenarien:

Ein Ziel der Untersuchung war die objektive Bewertung der unterschiedlichen Darstellungsarten. Dazu wurden die Visualisierungsparameter Testdatendatei, Profilanzahl und Ausgabemodus fixiert und die Werte für Zeit- und Speicherbedarf bei variabler Darstellungsart ermittelt. Der Einfluss des Rendermodus auf die Ausführungszeit und den Speicherbedarf wurde analog dazu untersucht. Als zusätzliches Ergebnis wurde die Skalierbarkeit des Zeit- und Speicherbedarfes in Abhängigkeit der darzustellenden Profilanzahl überprüft. Bei der Art der Grafikausgabe wird bei OpenGL zwischen einem gepufferten Modus, dem sogenannten retained Modus und dem Direktmodus, der als immediate Modus bezeichnet wird, unterschieden. Im retained Modus werden in sogenannte Displaylisten Zusammenstellungen von OpenGL-Primitiven zwischengespeichert, was sich durch einen schnelleren Grafikneuaufbau wie er beispielsweise nach einer Objektrotation notwendig ist, bemerkbar macht. Dies wird jedoch durch einen höheren Zeitaufwand für die Generierung der Listen und einen größeren Speicherbedarf für die Listenspeicherung erzielt (vgl. /2/, S.423 ff).

Als Datenquelle für die Testsfälle wurde die Datei panner0_gefiltert.prof verwendet. Sie beinhaltet die Daten einer Schweißnaht wobei 43173 Profile mit einer Breite von 512 Datenpunkten aufgezeichnet wurden.

Vergleich der Darstellungsarten:

Testdaten:	panner0_gefiltert.prof
Darstellungsarten:	Punkt / Linie_A / Linie_B / Linie_AB / Dreieck / Dreieck_reduziert (Einstellung 2, 3 und 5)
Profilanzahl:	2000
Rendermodus:	retained

Analyse der Skalierbarkeit:

Testdaten:	panner0_gefiltert.prof
Darstellungsart:	Dreiecksflächen in voller Auflösung
Profilanzahl:	250 / 500 / 1000 / 1500 / 2000 / 4000 / 6000 / 8000 / 10000 / 15000
Rendermodus:	retained

Vergleich retained / immediate Rendering:

Testdaten:	panner0_gefiltert.prof
Darstellungsart:	Dreiecksflächen in voller Auflösung
Profilanzahl:	250 / 500 / 1000 / 1500 / 2000 / 4000 / 6000 / 8000 / 10000 / 15000
Rendermodus:	retained / immediate

Testhardware:

Für die Untersuchung wurden zwei Rechnersysteme gewählt, die ähnliche Leistungsdaten aufweisen, jedoch Grafikhardware unterschiedlicher Hersteller verwenden. Als Vergleichssystem wurde ein wesentlich leistungsfähigeres PC-System herangezogen, wobei auf diesem aufgrund der zeitlich begrenzten Verfügbarkeit nur ein Teil der Tests durchgeführt wurde.

Testumgebungen:

Jedes der drei verfügbaren PC-Systeme wurde als Testsetup definiert (vgl. Tabelle 2). Aufgrund von Unregelmäßigkeiten in den Messdaten zu Setup 1 (vgl. Abbildungen 24 und 25) wurde eine entsprechende Recherche durchgeführt. Da die Recherche ergab, dass der OpenGL-Treiber für diese Grafikkarte abhängig von der Anwendung sehr leistungsschach ist, wurde ein zweites Setup mit der Testhardware 1 erstellt (vgl. /17/ ff). Hierbei wurde der ATI - OpenGL – Treiber in der Datei atioglxx.dll durch einen entsprechenden NVIDIA-Treiber nvoglnt.dll ersetzt. Die Konfiguration mit der originalen Treiberdatei wurde als Testsetup 1A, jene mit dem NVIDIA-Treiber als Testsetup 1B geführt. Die Messergebnisse , die mit Setup 1A ermittelt wurden verblieben in der vorliegenden Arbeit, um damit auch den Einfluß der Treiberqualität auf die Visualisierung zu dokumentieren.

Bezeichnung	CPU	Arbeitsspeicher	Grafikhardware	Betriebssystem	OpenGL-Treiber
Setup 1A	Intel Celeron T2080, 1.73 GHz	2 GB	ATI Radeon XPRESS 200M 256 MB HyperMemory	Windows XP Home	atioglxx.dll
Setup 1B	Intel Celeron T2080, 1.73 GHz	2 GB	ATI Radeon XPRESS 200M 256 MB HyperMemory	Windows XP Home	nvoglnt.dll
Setup 2	Intel Celeron E1200, 1,6GHz	1 GB	Nvidia GeForce 7050 / nForce 610i 256 MB	Windows XP Professional	nvoglnt.dll
Setup 3	Intel Xeon X5450 3 GHz, QuadCore	4 GB	Nvidia Quadro FX 1700, 512 MB	Windows XP Professional	nvoglnt.dll

Tabelle 2: Performancetest - Übersicht über Testhardware und Setups

Messungen:

Alle Messungen wurden zweifach ausgeführt und die Ergebnisse gemittelt, sofern keine auffallenden Differenzen zwischen den Messungen auftraten. Anderenfalls wurde die Messung auf Messfehler untersucht und gegebenenfalls wiederholt. Zu Beginn jedes Testlaufes wurde der aktuelle Speicherstatus als Bezugspunkt für den weiteren Speicherverlauf festgelegt. Nach Beendigung jedes Testlaufes wurde sowohl der maximale Speicherbedarf als auch der Speicherbedarf nach erfolgter Visualisierung aufgezeichnet.

3.6.2 Testergebnisse

Setup 1A – Vergleich der Darstellungsarten:

Die Daten aus Tabelle 3 enthalten die Messergebnisse bezüglich Zeit- und Speicherbedarf in Abhängigkeit der verwendeten Darstellungsart. Die Messwerte sind zur übersichtlicheren Bewertung in den Abbildungen 23 und 24 in Diagrammform dargestellt.

	Zeitbedarf [s]			Speicherbedarf [kB]	
type	**Preproc**	**Draw**	**SumProc**	**MemMax**	**MemFinal**
pkt	3,927	0,758	4,685	104968	40750
line_a	3,946	0,647	4,592	67094	41635
line_b	4,027	2,816	6,843	475426	101585
line_ab	4,065	5,336	9,401	445480	124770
tri_full	4,194	31,670	35,863	627306	624712
tri_red_2_2_1	7,614	2,092	9,706	209172	94860
tri_red_3_3_1	6,439	0,945	7,384	108940	58188
tri_red_5_5_1	5,819	0,341	6,159	57724	40684

Tabelle 3: Zeit- und Speicherbedarf in Abhängigkeit der Darstellungsart, Setup 1A

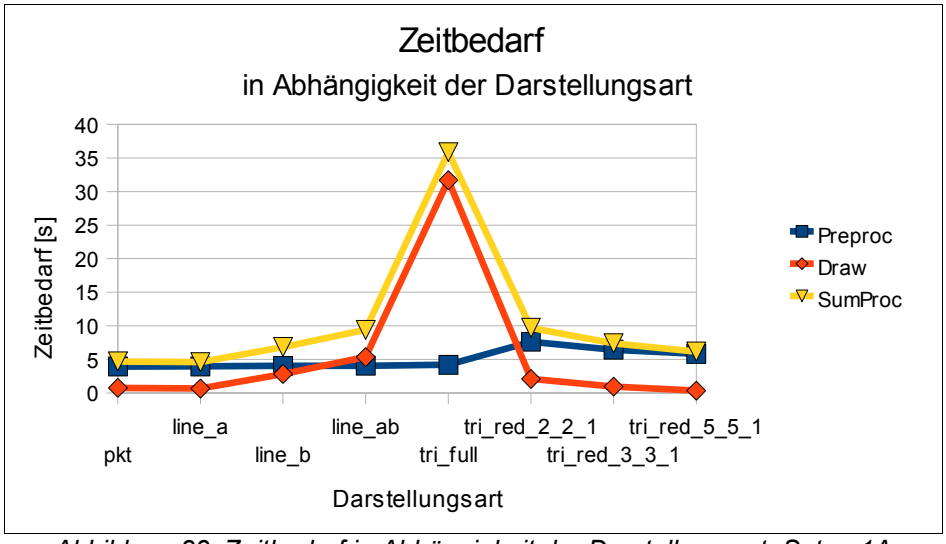


Abbildung 23: Zeitbedarf in Abhängigkeit der Darstellungsart, Setup 1A

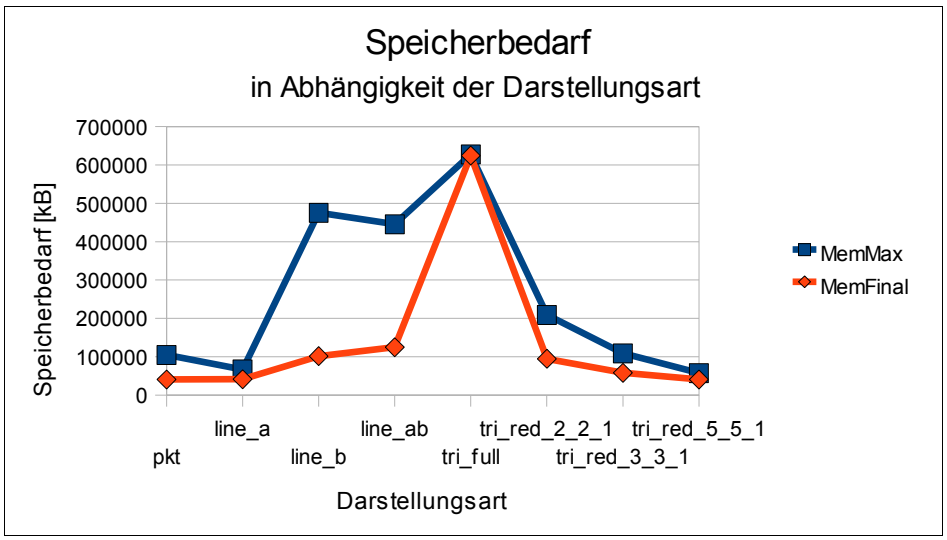


Abbildung 24: Speicherbedarf in Abhängigkeit der Darstellungsart, Setup 1A

Setup 1A – Skalierbarkeit:

Die Testserie konnte nur bis 10000 Profile erfolgreich durchgeführt werden. Bei 15000 Profilen wurde der Test nach 3600 Sekunden abgebrochen, da noch kein Ergebnis vorlag (vgl. Tabelle 4).

	Zeitbedarf [s]			Speicherbedarf [kB]	
Profilanzahl	**Preproc**	**Draw**	**SumProc**	**MemMax**	**MemFinal**
250	0,534	1,492	2,026	62162	23660
500	1,045	2,935	3,980	127322	59775
1000	2,109	5,857	7,966	255660	121058
1500	3,121	8,869	11,989	383914	184050
2000	4,224	31,859	36,082	642206	638412
4000	8,376	9,031	17,407	1022648	416216
6000	12,536	25,169	37,705	1405558	528568
8000	16,820	18,250	35,070	1398804	830050
10000	20,925	24,211	45,136	1370040	1041482
15000	*Abbruch*	*Abbruch*	*Abbruch*	*Abbruch*	*Abbruch*

Tabelle 4: Zeit- und Speicherbedarf in Abhängigkeit der Profilanzahl, Setup 1A

Auffallend an den Messergebnissen bei 2000 und 6000 Profilen ist ein deutlicher Anstieg des Zeit- und Speicherbedarfes (siehe Abbildungen 25 und 26). Aufgrund der wiederholten Messung konnte ein Messfehler ausgeschlossen werden. Da die Vorverarbeitungszeit nahezu linear skaliert, wurden die Abweichungen auf den Treiber zurückgeführt.

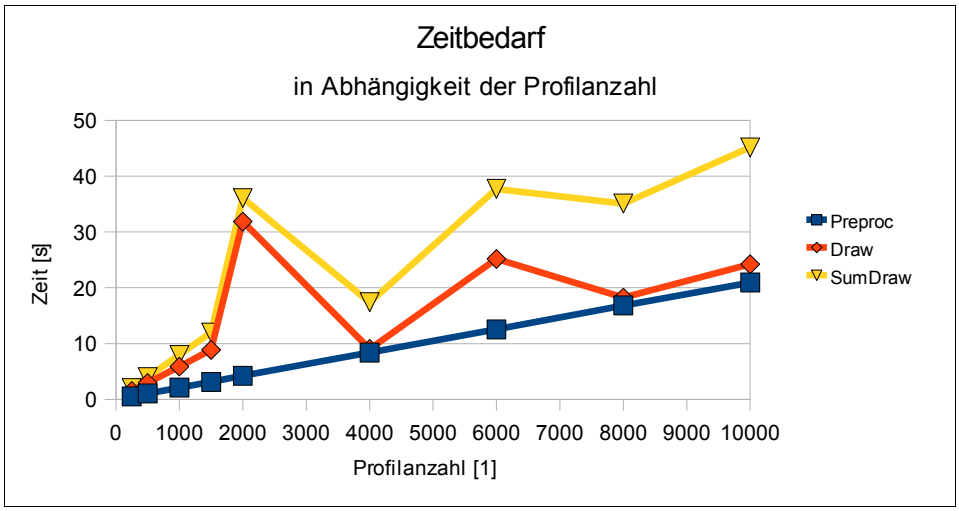


Abbildung 25: Verarbeitungszeit in Abhängigkeit der Profilanzahl, Setup 1A

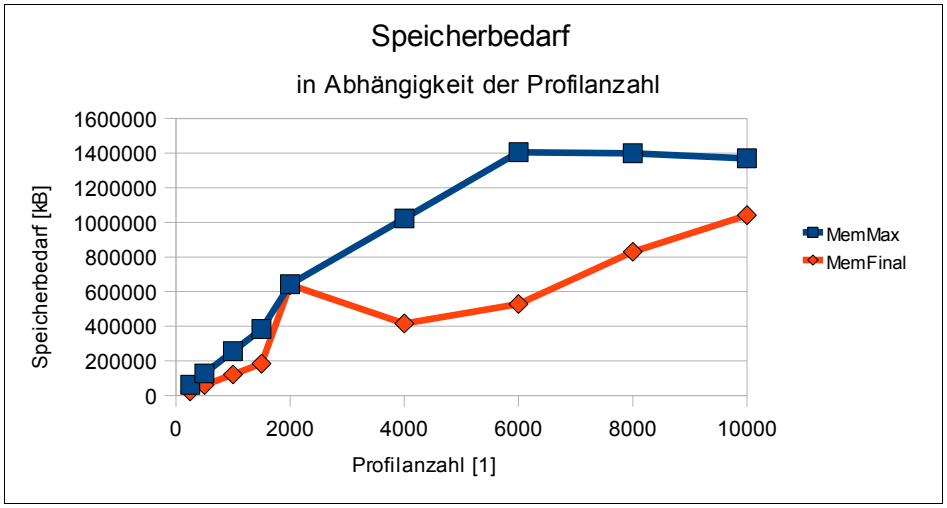


Abbildung 26: Speicherbedarf in Abhängigkeit der Profilanzahl, Setup 1A

Setup 1B – Vergleich der Darstellungsarten:

Wie in Tabelle 5 ersichtlich ist, wurde durch den geänderten OpenGL-Treiber im Vergleich zu Setup 1A eine deutliche Steigerung der Verarbeitungsgeschwindigkeit bei reduziertem Speicherbedarf erreicht. Dieses Verhalten wird durch die Diagramme in den Abbildungen 27 und 28 veranschaulicht.

	Zeitbedarf [s]			Speicherbedarf [kB]	
type	**PreProc**	**Draw**	**SumProc**	**MemMax**	**MemFinal**
pkt	1,030	0,443	1,473	61951	57700
line_a	0,981	0,456	1,437	46174	44555
line_b	1,085	1,185	2,270	118326	114442
line_ab	1,143	1,561	2,704	150156	148546
tri_full	1,245	3,233	4,478	206946	205313
tri_red_2_2_1	3,927	0,949	4,876	93450	90570
tri_red_3_3_1	3,113	0,452	3,565	57114	56486
tri_red_5_5_1	2,752	0,192	2,943	44138	37822

Tabelle 5: Zeit- und Speicherbedarf in Abhängigkeit der Darstellungsart, Setup 1B

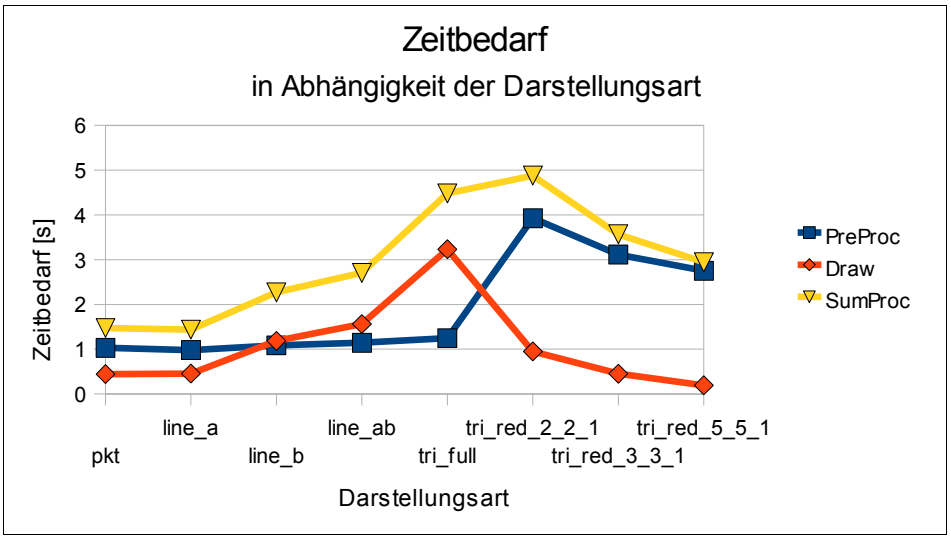


Abbildung 27: Zeitbedarf in Abhängigkeit der Darstellungsart, Setup 1B

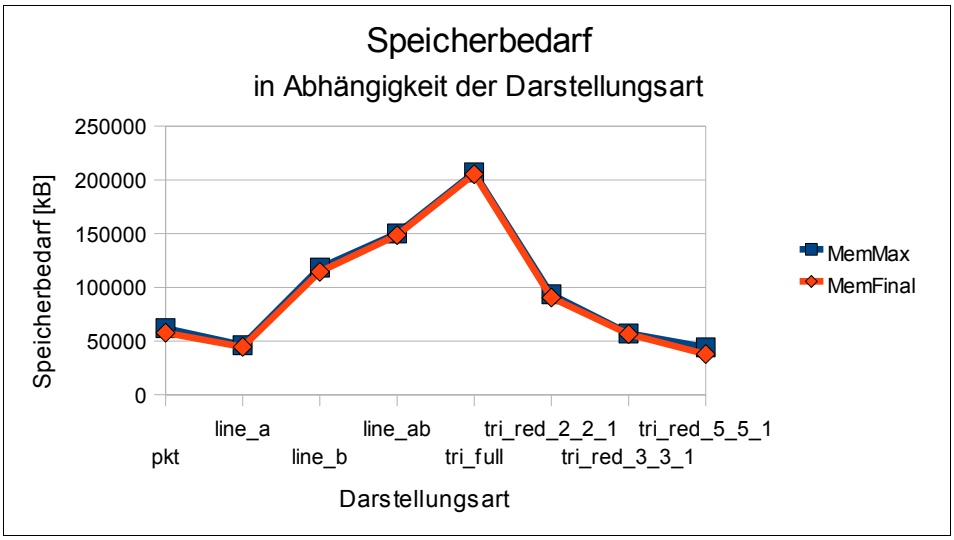


Abbildung 28: Speicherbedarf in Abhängigkeit der Darstellungsart, Setup 1B

Setup 1B – Skalierbarkeit:

Die in Tabelle 6 aufgeführten Messergebnisse der Skalierungsversuche belegen die deutliche Verbesserung der Systemleistung.

	Zeitbedarf			Speicherbedarf	
Profilanzahl	**Preproc**	**Draw**	**SumDraw**	**MemMax**	**MemFinal**
250	0,151	0,482	0,633	26590	25782
500	0,307	0,917	1,223	52558	51392
1000	0,614	1,719	2,332	102850	102222
1500	0,904	2,483	3,387	153966	153610
2000	1,237	3,246	4,482	206598	204559
4000	2,445	6,267	8,711	410195	409138
6000	3,601	9,282	12,882	615834	613628
8000	4,929	12,330	17,258	822464	817450
10000	6,072	15,426	21,498	1025192	1020672
15000	9,307	34,133	43,440	1403038	1386960

Tabelle 6: Zeit- und Speicherbedarf in Abhängigkeit der Profilanzahl, Setup 1B

Mit dem modifizierten OpenGL – Treiber trat ein deutlich reduzierter und nahezu linearer Verlauf des Zeit- und Speicherbedarfes auf (vgl. Abbildungen 29 und 30). Der Anstieg der Darstellungszeit bei 15000 Profilen beruht auf der vollständigen Auslastung des physikalischen Arbeitsspeichers, wodurch teilweise auf die Auslagerungsdatei zu Lasten der Systemgeschwindigkeit zugegriffen wurde.

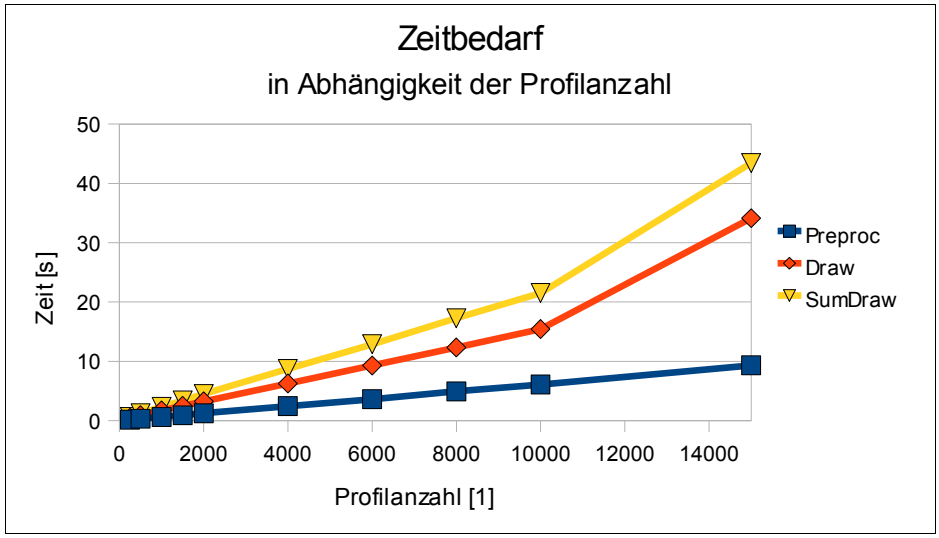


Abbildung 29: Zeitbedarf in Abhängigkeit der Profilanzahl, Setup 1B

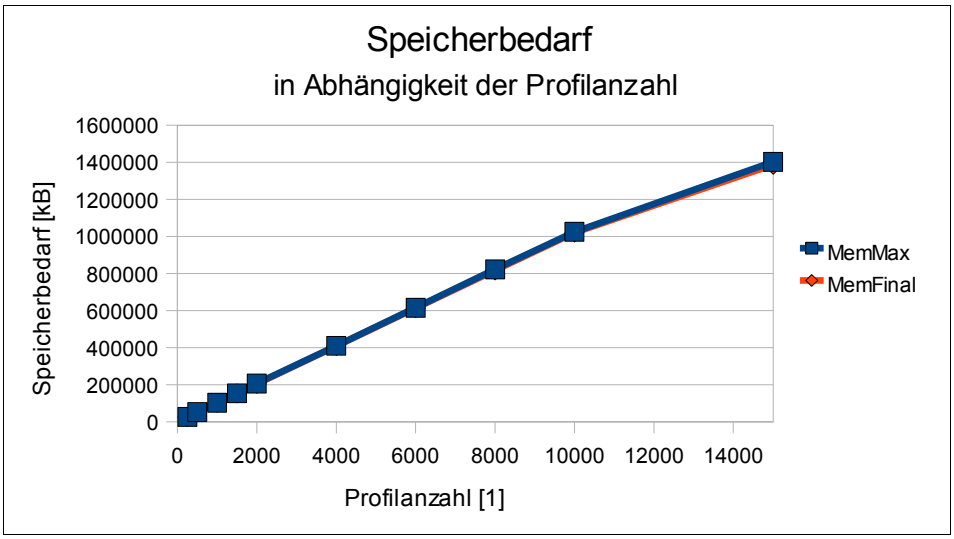


Abbildung 30: Speicherbedarf in Abhängigkeit der Profilanzahl, Setup 1B

Vergleich retained / immediate Rendering:

Der Vergleich zwischen den beiden Rendermodi wurde nur mit Setup 1B ausgeführt, da dies nur einen Anhaltspunkt in Bezug auf Zeit- und Speicherbedarf darstellt. Wie jedoch aus den Abbildungen 31 und 32 ersichtlich ist, ergab sich ein linearer Verlauf sowohl bei Zeit- als auch Speicherbedarf. Diese Linearität war über den gesamten Bereich gegeben.

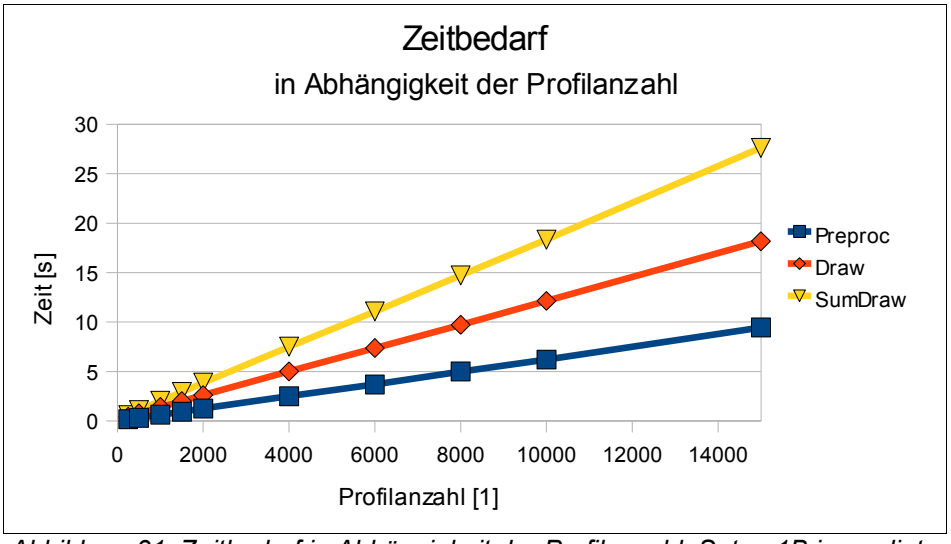


Abbildung 31: Zeitbedarf in Abhängigkeit der Profilanzahl, Setup 1B immediate

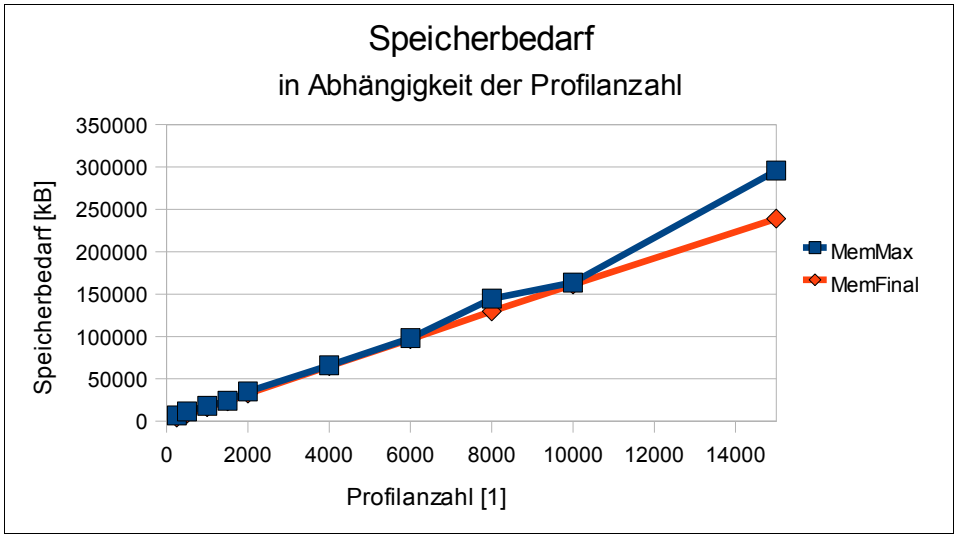


Abbildung 32: Speicherbedarf in Abhängigkeit der Profilanzahl, Setup 1b immediate

Aus dem Vergleich mit der Zeitbedarfsuntersuchung dieses Setups im retained Modus (vgl. Abbildungen 29 und 30) zeigt sich, dass der Zeitbedarf für die Datenaufbereitung in beiden Fällen gleich bleibt. Die notwendige Zeit für den Ausgabevorgang steigt im Immediatemodus linear an. Der Speicherbedarf (MemFinal) steigt ebenfalls linear, liegt aber bei 17% des retained Rendermodus.

Ein wesentlicher Unterschied zwischen den beiden Rendermodi zeigt sich beim Vergleich der Grafikausgabezeit. Diese Listen beschleunigen die erneute Grafikgenerierung, wie sie beispielsweise nach einer Rotation der Kamera notwendig ist. Die Listenerzeugung benötigt jedoch zusätzliche Rechenzeit und Speicherkapazität. Übertrifft der Gesamtspeicherbedarf den physikalisch verfügbaren Speicher, so wird auf virtuellen Speicher ausgewichen (z.B. Swapping oder Paging, vgl. /18/ S.282 ff) was aufgrund der deutlich erhöhten Zugriffs- und Transferzeiten zu einem überproportionalen Anstieg des Zeitbedarfs führt. Dieses Verhalten ist im Verlauf der Zeitbedarfskurve für den retained Rendermodus (SumProc_r) ab 15000 Profilen ersichtlich (vgl. Abbildung 33). SumProc_i zeigt die Gesamtzeit, die im immediate Modus bis zur Ausgabe der Grafik benötigt wird. SumProc_r zeigt analog den Zeitbedarf für Generierung und Ausgabe der Grafik im retained Modus.

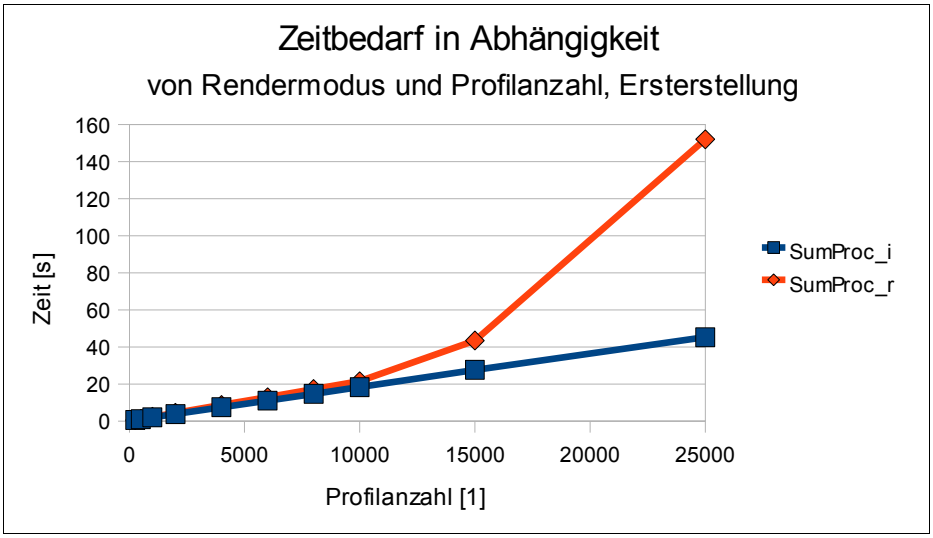


Abbildung 33: Zeitbedarf in Abhängigkeit von Rendermodus und Profilanzahl bei Ersterstellung

Für den erneuten Aufbau der Grafik (Redraw) benötigt die Ausführung im immediate Modus mehr Zeit als in der retained Version (vgl. Abbildung 34). Bei höheren Datenmengen traten allerdings im retained Modus Speicherprobleme auf, da Speicheranforderungen nicht erfüllt werden konnte. Der Test konnte nicht mit mehr als 15000 Profilen durchgeführt werden, da dann ein Programmabbruch aufgrund des Speichermangels auftrat.

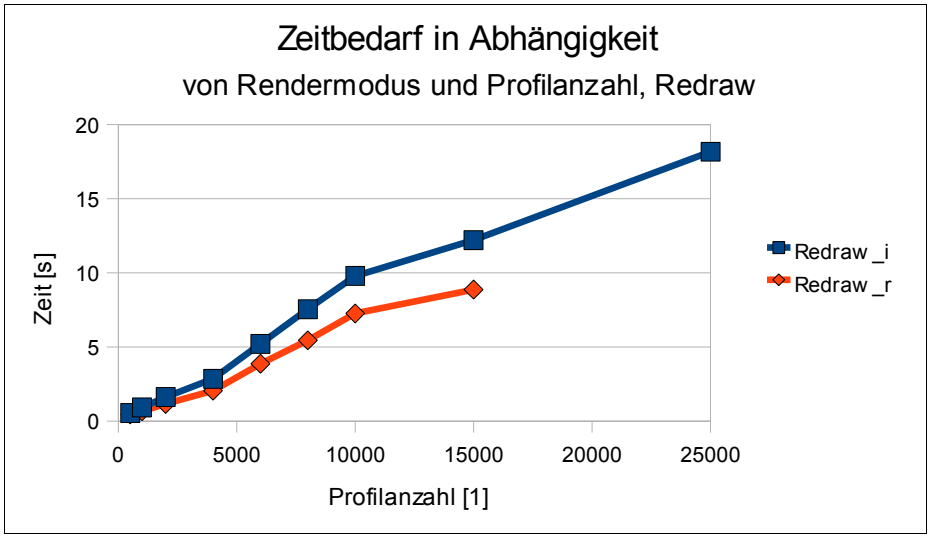


Abbildung 34: Zeitbedarf in Abhängigkeit von Rendermodus und Profilanzahl, Redraw

Setup 2 – Vergleich der Darstellungsarten:

Wie aus den Messergebnissen in Tabelle 7 sowie den daraus entstandenen Diagrammen in den Abbildungen 35 und 36 ersichtlich ist, ergibt sich eine ähnliche Abhängigkeit von der Darstellungsart wie bei Setup 1B, die geringere Rechenleistung des Systems wird aber in der deutlich gesteigerten Vorverarbeitungszeit ersichtlich.

	Zeitbedarf [s]			Speicherbedarf [kB]	
type	**Preproc**	**Draw**	**SumProc**	**memmax**	**memfinal**
pkt	3,806	0,347	4,153	64514	53132
line_a	3,781	0,288	4,068	47060	43730
line_b	3,886	1,135	5,021	161028	100510
line_ab	3,930	1,358	5,288	224486	146488
tri_full	3,986	2,417	6,403	481554	254614
tri_red_2_2_1	7,090	1,917	9,006	177806	96936
tri_red_3_3_1	6,054	0,919	6,973	92342	67558
tri_red_5_5_1	5,502	0,334	5,836	52840	44678

Tabelle 7: Zeit- und Speicherbedarf in Abhängigkeit der Darstellungsart, Setup 2

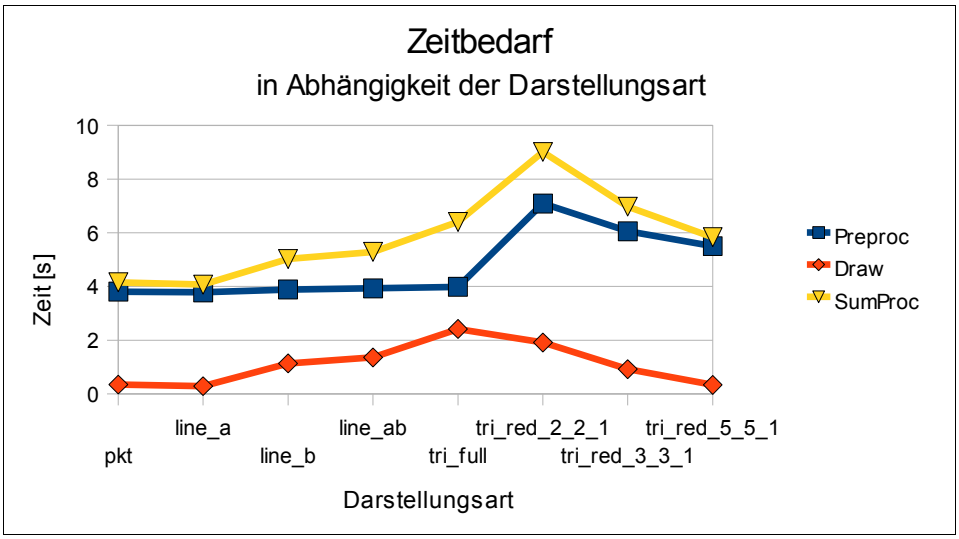


Abbildung 35: Zeitbedarf in Abhängigkeit der Darstellungsart, Setup 2

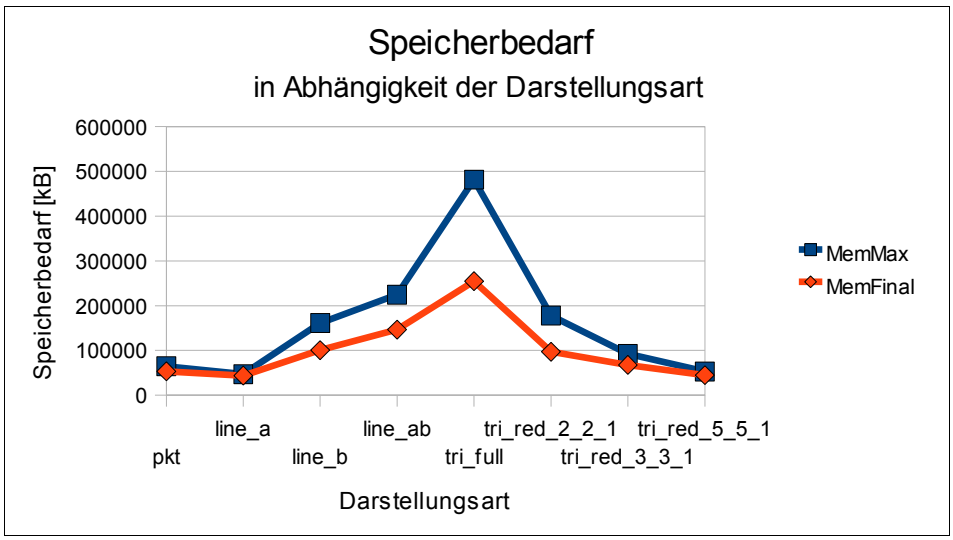


Abbildung 36: Speicherbedarf in Abhängigkeit der Darstellungsart, Setup 2

Setup 2 – Skalierbarkeit:

Bei 6000 und 8000 Profilen wurde keine korrekte Visualisierung erreicht (keine Darstellung der 3D-Daten). Aufgrund des Zeitbedarfes, der bereits bei 6000 und 8000 Profilen auftrat wurde der Testlauf mit 10000 und 15000 Profilen nicht mehr durchgeführt. Diese Datensätze sind in der Tabelle mit „n.d.“ gekennzeichnet (vgl. Tabelle 8 und Abbildungen 37 und 38).

	Zeitbedarf			Speicherbedarf	
Profilanzahl	**Preproc**	**Draw**	**SumDraw**	**MemMax**	**MemFinal**
250	0,500	0,328	0,828	46138	24078
500	0,993	0,649	1,642	114622	46038
1000	1,994	1,270	3,264	231604	101788
1500	2,976	1,845	4,821	375360	179828
2000	3,967	2,415	6,382	490110	254560
4000	7,960	50,076	58,036	704624	391348
6000	11,937	74,671	86,608	765314	286162
8000	16,020	225,841	241,861	765068	270548
10000	*n.d.*	*n.d.*	*n.d.*	*n.d.*	*n.d.*
15000	*n.d.*	*n.d.*	*n.d.*	*n.d.*	*n.d.*

Tabelle 8: Zeit- und Speicherbedarf in Abhängigkeit der Profilanzahl, Setup 2

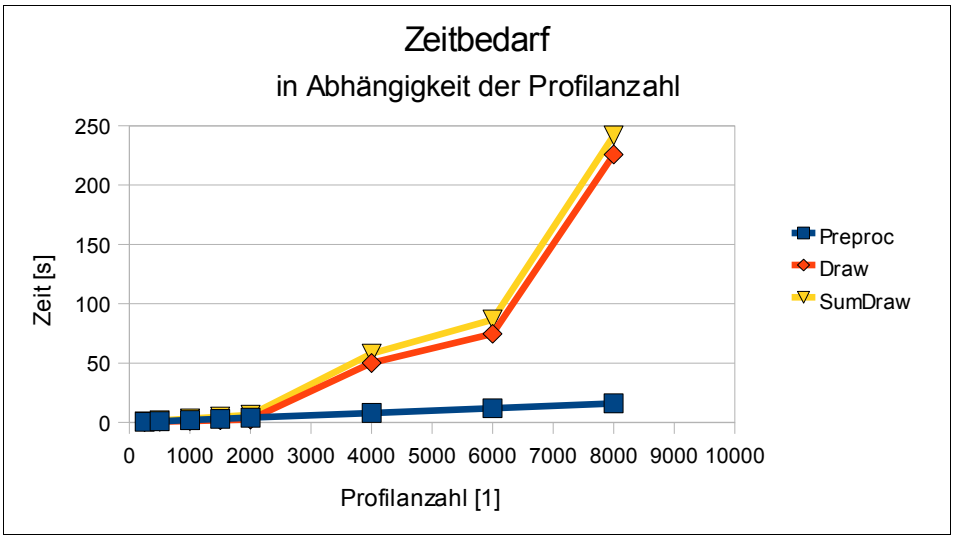


Abbildung 37: Zeitbedarf in Abhängigkeit der Profilanzahl, Setup 2

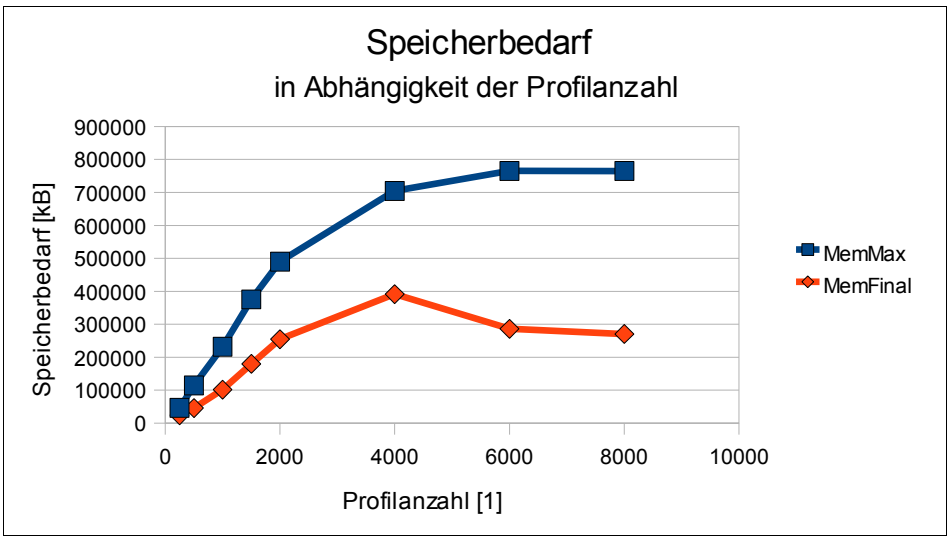


Abbildung 38: Speicherbedarf in Abhängigkeit der Profilanzahl, Setup 2

Setup 3 – Vergleich der Darstellungsarten:

Diese Messung diente nur dem Vergleich, da die eingesetzte Grafikkarte eine relativ hohe Leistung bieten sollte. Wie das Diagramm in Abbildung 39 zeigt, war die Darstellungs-geschwindigkeit höher als bei den restlichen Systemen, die Gesamtleistung war aber nur unwesentlich höher als bei Setup 1B.

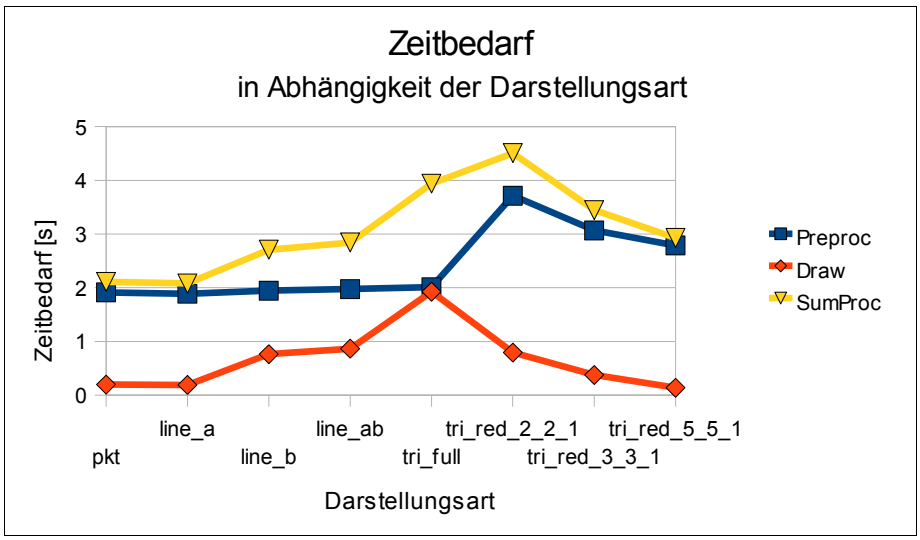


Abbildung 39: Zeitbedarf in Abhängigkeit der Darstellungsart, Setup 3

3.6.3 Subjektive Bewertung:

Für die Bewertung der subjektiven Kriterien wurde je ein Vertreter aus jeder der Abteilungen Vertrieb, Service, Entwicklung und Geschäftsführung der Plasmo Industrietechnik eingeladen. Nach der Präsentation der möglichen Darstellungsarten musste jeder Bewerter 10 Punkte auf die möglichen Darstellungsarten gemäß des subjektiven Eindruckes verteilen. Zur Auswahl standen die in die Testapplikation implementierten Darstellungsarten Punkt, Linie_A (profilinterne Verbindung), Linie_B (Verbindung korrespondierender Punkte), Linie_AB (Kombination von Linie_A und Linie_B) und Tri (Dreiecksflächen). Zusätzlich wurde die bisher verwendete Visualisierungssoftware als Option angeboten.

Darstellungs-Art	Punkt	Linie_A Profil	Linie_B Korrespond.	Linie_AB A+B	Tri	System Aktuell
Bewerter 1	0	0	0	5	0	5
Bewerter 2	0	0	0	4	3	3
Bewerter 3	0	0	0	5	5	0
Bewerter 4	0	0	0	0	10	0
SUMME	**0**	**0**	**0**	**14**	**18**	**8**

Tabelle 9: Subjektive Bewertung der Darstellungsarten

Die beurteilenden Personen entschieden sich für die Darstellung der Daten als Dreiecks-flächen beziehungsweise als Linienraster (Variante Linie_AB). Der Vorteil der Dreiecksflächen-Darstellung liegt darin, dass auch bei starker Vergrößerung noch die Nahtstruktur erkennbar bleibt. Diesen Vorteil bietet die Liniendarstellung nur bedingt, weist

jedoch in Bezug auf Verdeckungen bei Nahteinbrüchen den Vorteil auf, dass der Verlauf der Naht innerhalb der Einsenkung erfassbar bleibt.

3.6.4 Grenzen des Systems

Die vorangegangenen Versuche haben gezeigt, dass drei wesentliche Faktoren für die Begrenzung des Systems verantwortlich sind:

- der zur Verfügung stehende Arbeitsspeicher
- die vom Anwender festzulegende Maximaldauer für die Darstellung der Daten
- die Qualität des Grafiktreibers

Arbeitsspeicher

Dieser Einflußfaktor ist theoretisch linear abhängig von der Anzahl der darzustellenden Datenmenge (Profilanzahl und Profilbreite). Dadurch kann in Abhängigkeit des verfügbaren Arbeitsspeichers die darstellbare Datenmenge abgeschätzt werden. Beeinflussen lässt sich der Speicherbedarf durch die Wahl der Darstellungsart und des Rendermodus.

Maximaldauer der Darstellung

Der Zeitbedarf für die Darstellung hängt analog zum benötigten Speicher linear von der dargestellten Datenmenge ab. Damit läßt sich theoretisch eine Abschätzung bezüglich der innerhalb der Zeitvorgabe visualisierbaren Datenmenge treffen. Die Beeinflussung der Darstellungsdauer ist sowohl über die Darstellungsart als auch über den Rendermodus möglich.

Qualität des Grafiktreibers

Die Qualität des Grafiktreibers stellt den kritischsten Einflußfaktor dar, da dadurch sowohl die beiden anderen Faktoren (Zeit- und Speicherbedarf) massiv beeinflußt werden als auch die Eingriffsmöglichkeiten und die Abschätzbarkeit der Qualität sehr gering sind. Die Auswirkungen reichen sogar so weit, dass die Funktion durch unvollständige beziehungsweise fehlende Darstellung der Grafik teilweise nicht mehr gegeben war. Dieses fehlerhafte Verhalten konnte bei den bestehenden Systemen durch den Wechsel in den Immediatemodus behoben werden, ging jedoch zu Lasten der Darstellungsgeschwindigkeit.

3.6.5 Vergleich mit der bisher eingesetzten Lösung

Geschwindigkeit:

Nachfolgend wurden Vergleichsmessungen zwischen der bisher eingesetzten Visualisierungssoftware (basierend auf Measurement Studio) und dem neu entwickelten Visualisierungssystem (basierend auf dem vtk) durchgeführt (vgl. Tabelle 11). Gemäß der Anforderung einer gesteigerten Interaktivität wurde die Zeit bis zur ersten Erstellung der Datenvisualisierung (Bezeichnung „Ersterst." in der Tabelle) sowie die Zeit für den Wiederaufbau der Grafik nach einer Änderung der Perspektive ermittelt (Bezeichnung „Redraw" in der Tabelle). Das neue Visualisieriungssystem wurde sowohl ohne als auch mit Datenreduzierung (mittels Quadric Clustering, Qualitätseinstellung für X, Y und Z = 3, 3, 1) getestet, wobei die Dreiecksflächen als Darstellungsart gewählt wurden. Die Messungen wurden mit Testsetup 1B durchgeführt.

	Originalsoftware		**neue Visualisierung unreduziert**		**neue Visualisierung reduziert: QC 3/3/1**	
Profil-anzahl	**Ersterst.**	**Redraw**	**Ersterst. r / i**	**Redraw r / i**	**Ersterst. r / i**	**Redraw r / i**
1000	1,42	1,20	2,33 / 2,03	1,17 / 1,63	2,3 / 2,0	0,21 / 0,32
2000	2,13	2,45	4,48 / 3,87	2,05 / 2,86	3,77 / 3,7	0,40 / 0,54
4000	4,44	4,70	8,71 / 7,50	3,88 / 5,22	7,41 / 7,2	0,60 / 0,78
10000	10,78	11,00	21,50 / 18,35	8,85 / 12,21	18,37 / 17,85	1,28 / 1,72
20000	20,69	18,80	151,5 / 36,45	- / 21,94	- / 35,72	-/ 3,03

Tabelle 10: Vergleich der Visualisierungszeiten zwischen bestehendem und neuem System, Zeiten in s

In der Darstellung ohne Datenreduktion ist die Originalsoftware schneller als die neue Visualisierung. Dies liegt unter Umständen an der speziell für diese Anwendung optimierten Komponenten des Measurement Studio. Bei Einsatz der Datenreduktion liegt der Zeitbedarf des neuen Systems für den Neuaufbau der Grafik bei 11,6% im Vergleich zu der bestehenden Software.

Funktionalität:

Die Funktionalität der neuen Visualisierung übertrifft durch die erfolgreiche Implementation der zwei neuen Darstellungsansätze die aktuell eingesetzte Software wobei diese Neuerungen bei Bedarf abgeschaltet werden können. Durch die Erweiterte Informationspräsentation ist eine raschere und klarere Beurteilung der Naht möglich. Die Navigation der neuen Visualisierung entspricht noch nicht voll den Erfordernissen, da die Veränderung von Kameraposition und Perspektive noch schwer zu kontrollieren ist.

3.6.6 Ergebnisse

Die Analyse der Leistung des Visualisierungssystems hat ergeben, dass die Entscheidung für die optimale Darstellungsart nicht systemunabhängig getroffen werden kann. Es besteht die Möglichkeit, basierend auf den vorhandenen Systemressourcen Rechenzeit, Grafikleistung und Arbeitsspeicher jene Darstellungsart auszuwählen, die diese Ressoucen bestmöglich ausnutzt. Alternativ kann von einer vorgegebenen Darstellungsart ausgehend entweder die Hardwareanforderung oder das, unter Einhaltung der Ressourcenrahmenbedingungen darstellbare Datenvolumen definiert werden.

Um diesen Sachverhalt zu verdeutlichen wurde in Abbildung 40 der Zeitbedarf in Abhängigkeit der Darstellungsart für das Setup 1B getrennt nach Vorverarbeitung und Grafikausgabe dargestellt. Für eine Grobeinteilung der Eignung einer Darstellungsart in Abhängigkeit der Leistungsfähigkeit des benutzten PC-Systems wurde der Datenbereich des Diagrammes in vier gleich große Bereiche geteilt. Auffallend daran ist, dass keine der Darstellungsarten viel Vorverarbeitungszeit bei hoher Grafikleistung benötigt. Weiters zeigte sich, dass die acht analysierten Darstellungsarten in drei Gruppen eingeteilt werden können:

- Punkt und Liniendarstellung (geringer Vorverarbeitungs- und Ausgabezeitbedarf)
- Dreiecksdarstellung ohne Datenreduktion (Hauptzeitbedarf bei der Ausgabe)
- Dreiecksdarstellung mit Datenredukton (Hauptzeitbedarf bei der Vorverarbeitung)

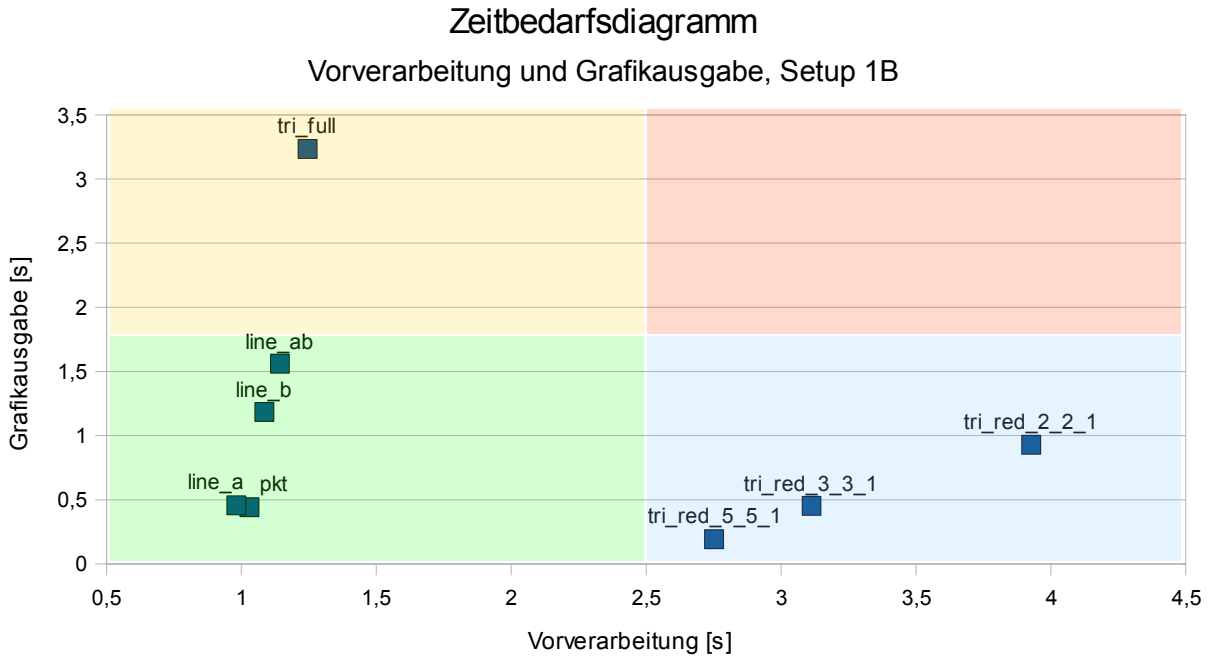


Abbildung 40: Zeitbedarf in Abhängigkeit der Darstellungsart, getrennt nach Vorverarbeitung und Ausgabe, Setup 1B

Zusätzlich zu der ressourcenbedingten Gruppierung ist bei der Auswahl auch auf die subjektiv zu beurteilenden Kriterien Rücksicht zu nehmen. So wurden von den Mitarbeitern der Plasmo Industrietechnik die Darstellung in unreduzierter Dreiecksvolldarstellung favorisiert.

Ausgehend von der zeitlichen Anforderung, 5 Fehlerstellen in 30 Sekunden visualisieren zu können (vgl. Kap. 3.1) wobei dazu derzeit zwischen 500 und 2000 Profile angezeigt werden, währen alle Darstellungsarten geeignet, zumindestens 500 Profile innerhalb von 2 Sekunden zu visualisieren. Die verbleibende Zeit von 4 Sekunden je Fehlerstelle bleibt dem Bedienpersonal, um die Nahtstelle zu bewerten.

Sowohl die Punkt- und Liniendarstellung (profilinterne Verbindung, line_a) als auch die flächige Dreiecksdarstellung (unter Ausnutzung der Datenreduktion) eignen sich aufgrund der geringen Ausgabedauer von unter 0,5s dafür, für den flyover-Modus eingesetzt zu werden. Hierbei würde die Bewegung noch mit zumindestens 2 Bildern pro Sekunde animiert werden.

Ein wesentliches Ergebnis der durchgeführten Untersuchungen besteht darin, welchen Einfluß die Treiberqualität auf die Gesamtleistung des Visualisierungssystems hat. Es konnte gezeigt werden, dass durch den Wechsel des OpenGL-Treibers bei Testsystem 1 die Zeit für die Ausgabe von 2000 Profilen als unreduzierte Dreiecke von zirka 32 Sekunden (Setup 1A) auf 3,2 Sekunden (Setup 1B) reduziert wurde. Diese Beschleunigung der Ausgabe um den Faktor 10 ist durch leistungsfähigere Hardware kaum auszugleichen. Für den Aufbau eines leistungsfähigen Visualisierungssystems ist daher eine Überprüfung der Treiberqualität unabdingbar.

4 Implementierung

4.1 Installation und Inbetriebnahme des vtk

Installation und Inbetriebnahme des vtk stellen einen mehrstufigen Prozess dar. Das Toolkit wird als compiler- und betriebssystemunabhängiges Quellcode-Paket zum Download bereitgestellt. Vor der Verwendung muß das Projekt für das gewünschte Betriebssystem sowie den gewählten Compiler konfiguriert werden. Nach Abschluß des Konfigurationsvorganges stehen Projektdateien für den Zielcompiler zur Verfügung, aus denen durch den Übersetzungsvorgang ausführ- oder linkbare Dateien generiert werden.

Der Ablauf der vtk-Installation stellt sich wie folgt dar:

- Download des Tools Cmake, Quelle: /14/
- Download der Sourcefiles, Datenfiles und Beispiele, Quelle: /14/
- Konfiguration der Projektfiles mittels CMake (Pfade, Einstellungen, Compiler-System, Projektumfang), siehe Abbildungen 41 und 42
- Erstellung der compilerabhängigen Projektfiles durch Cmake
- Übersetzung der vorkonfigurierten Projekte (z.B. Auswahl Debug/Release)
- Erstellung der Binärdateien (dynamisch linkbare und Bibliotheksdateien), siehe Abbildung 43
- Projekt “Install” ausführen
- Erstellung der Verzeichnisstruktur (bin/include/lib)

Weiterführende Informationen zu Cmake sind in /4/ zu finden.

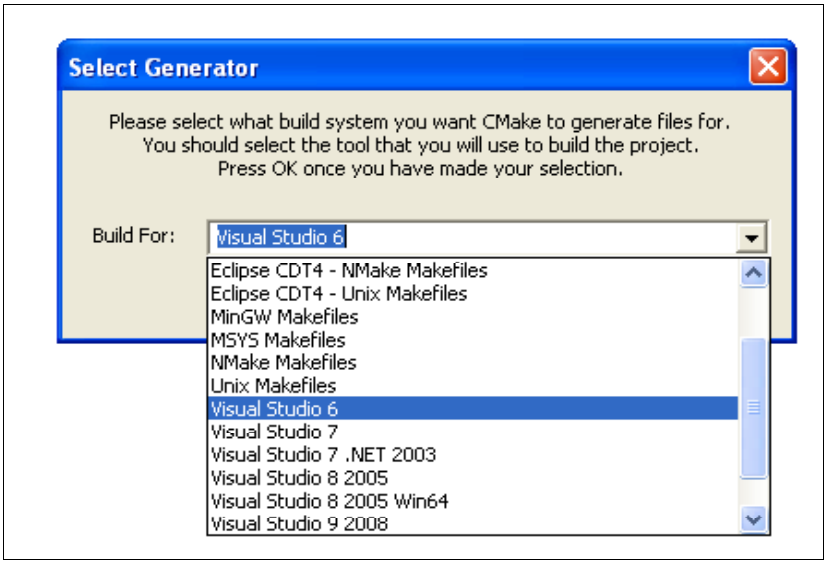


Abbildung 41: CMake: Auswahl des Compilerzielsystems

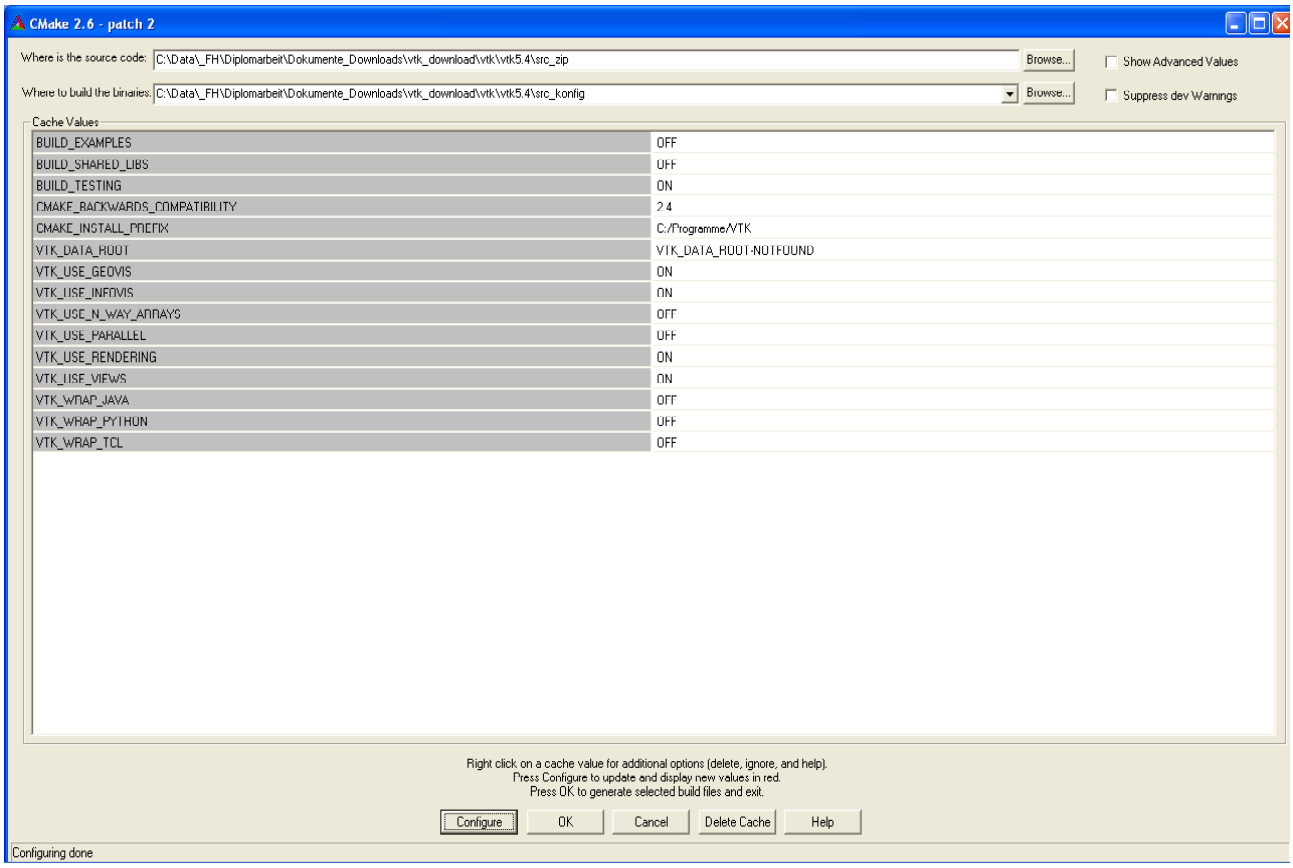


Abbildung 42: CMake: fertig konfiguriertes Projekt

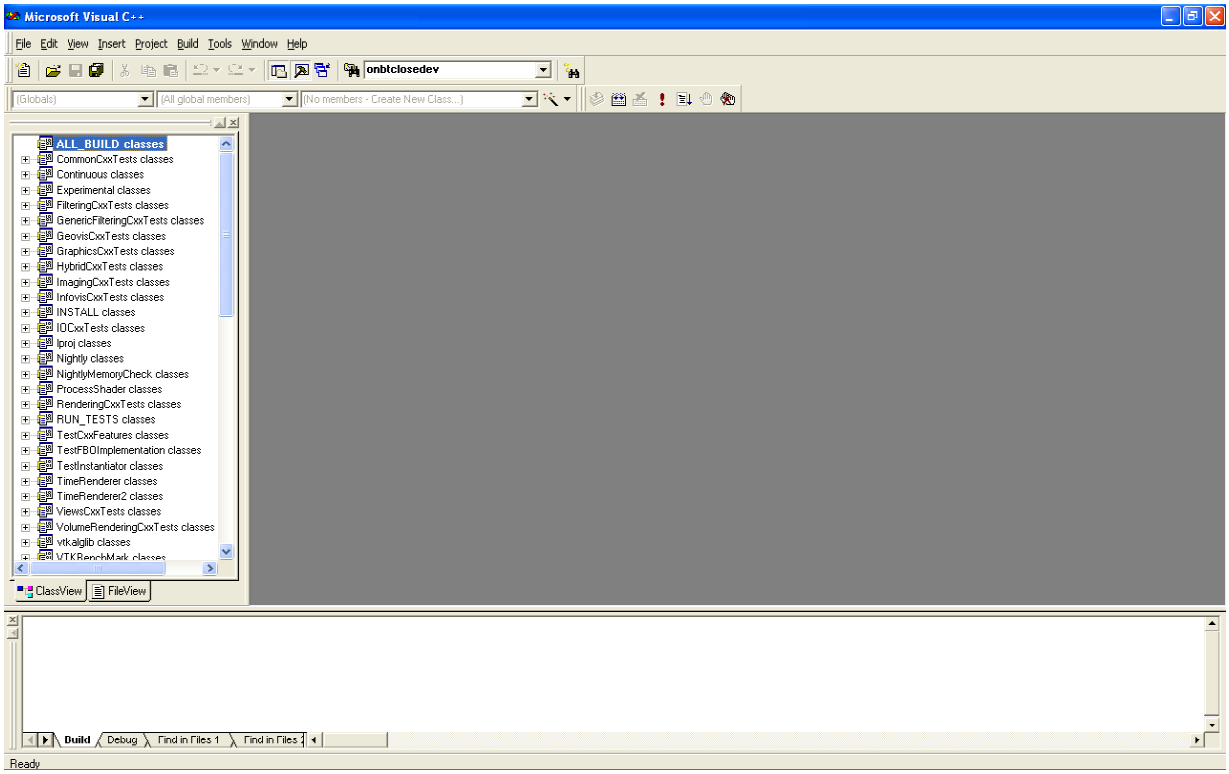


Abbildung 43: Microsoft Visual C++ 6.0 mit geladenem vtk-Projekt

4.2 Erstellung der Testapplikation

Zur Sicherstellung der korrekten Installation des vkt wurde eine einfache Applikation erstellt, welche Grundfunktionalitäten des vtk verwendet.

- Erstellung einer dialogbasierten MFC-Applikation
- Bereitstellung der Ausgabefläche für einfache Grafik-Primitiven
- Erstellung der Verarbeitungspipeline
- Erstellung von Testdaten
- Test der Applikation

In der ersten Erweiterung wurde die Funktionalität für den Import von Daten aus PeO-Datenfiles implementiert. Zusätzlich erfolgte die Implementierung der unterschiedlichen Anzeigemodi.

- Punkt: jeder Punkt der Messdaten wird durch einen Punkt in der Grafik repräsentiert
- Linie A: die Messpunkte eines Datenprofiles werden durch Linien verbunden
- Linie B: Verbindung korrespondierender Punkte benachbarter Profile durch Linien
- Linie AB: Kombination aus den Varianten Linie A und Linie B
- Dreiecksflächen: Aus je drei Punkten wird ein gefülltes Dreieck generiert

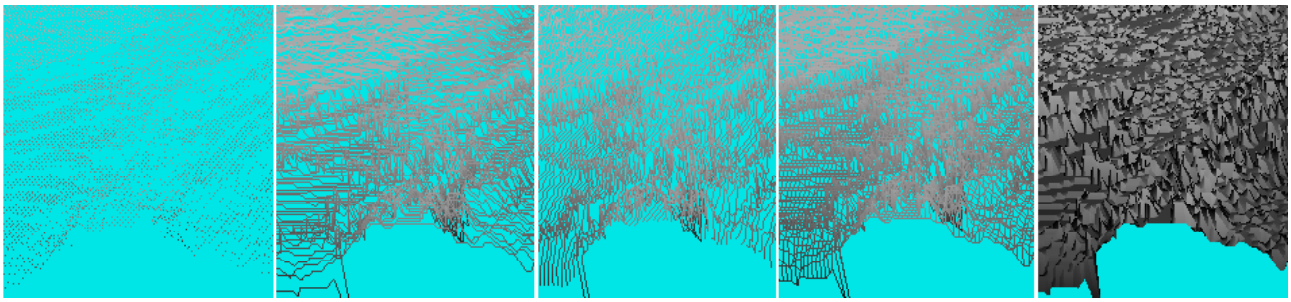


Abbildung 44: Darstellungsarten: Punkt / Linie_A / Linie_B / Linie_AB / Dreiecksflächen

Die Funktionen zur Reduktion der Daten wurden im zweiten Erweiterungsschritt implementiert. Alle vom vtk bereitgestellten Reduktionsverfahren wurden implementiert:

- None – Verarbeitung ohne Datenreduktion
- DecimatePro – Verwendet eine lineares Abstandskriterium zur Auswahl von zu entfernenden Punkten und Kanten (vgl. /6/ S. 92ff)
- QuadricDecimation – Verwendet eine quadratisches Kriterium zur Auswahl von zu entfernenden Punkten und Kanten (vgl. /24/, /6/ S. 92ff).
- QuadricClustering – (vgl. /25/, /6/ S. 92ff)
  das schnellste der im vtk verfügbaren Datenreduktionssysteme, verfügt auch über Funktionen, die die stückweise Reduktion sehr großer Datenbestände ermöglicht

Die Einführung des flyover-Modus war Inhalt der nächsten Erweiterung. Dabei werden die Kamera und das Mesh in eine vordefinierte Ausrichtung gebracht. Danach wird das Mesh automatisch schrittweise an der Kamera vorbeigeführt und die Visualisierung aktualisiert. Für die Anpassung an unterschiedliche Verarbeitungsgeschwindigkeiten kann die Anzahl der Profile pro Visualisierungsschritt parametriert werden.

Die aktuelle Version der Testapplikation wurde um Funktionen zur Messung der Verarbeitungszeit der einzelnen Pipelinestufen erweitert (vgl. Abbildung 19). Für die

Zeitmessung wurden Funktionen verwendet, die auf einen hochauflösenden Timer des Betriebssystem Microsoft Windows zugreifen. Zur Bestimmung des Speicherbedarfes wurde eine eigenständige Applikation erstellt, die in einem definierbaren zeitlichen Raster den Speicherstatus des Betriebssystems ermittelt.

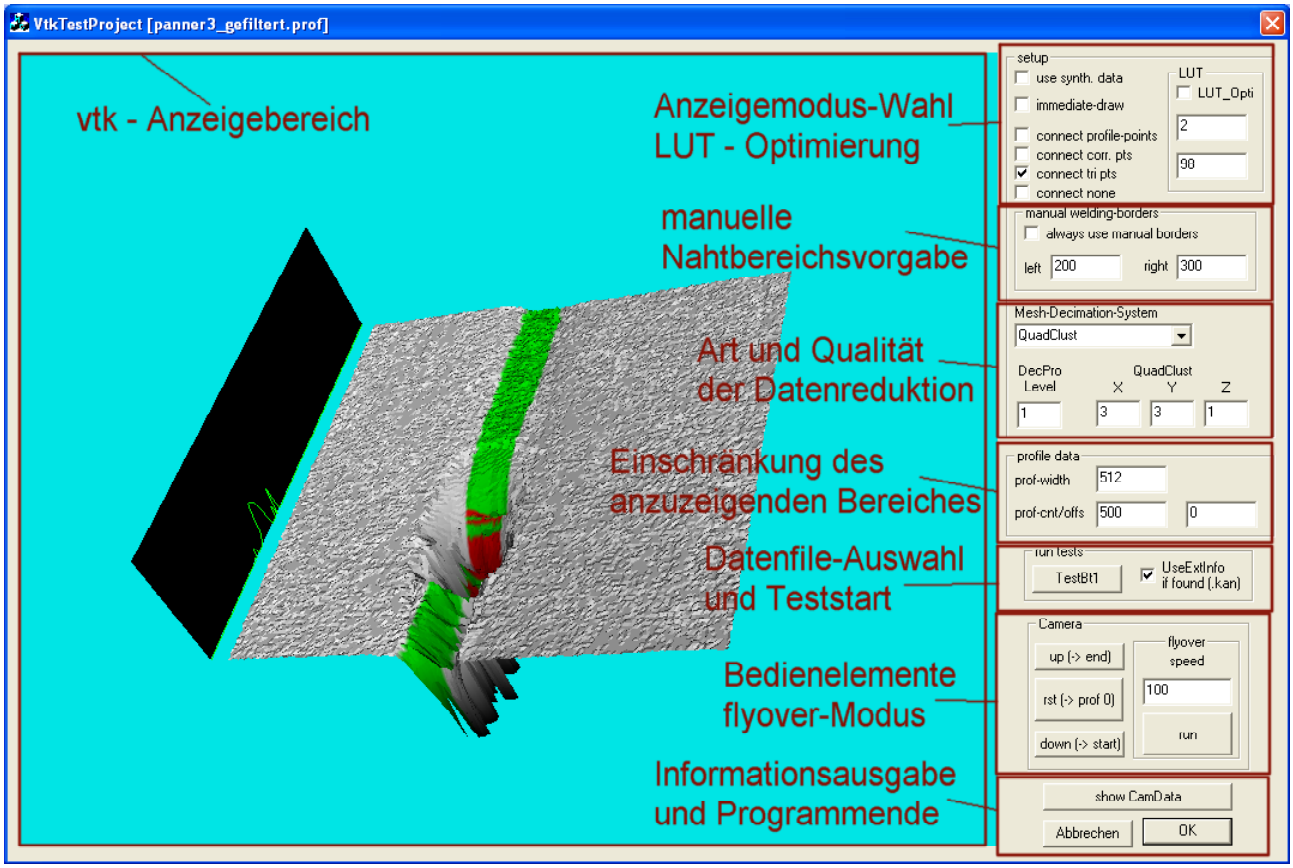


Abbildung 45: Screenshot der Testapplikation für 1D- und 3D-Visualisierung

Für die Implementierung der 2D-Testvisualisierung wurde eine Version der Testapplikation so modifiziert, dass zusätzliche Ausgabebereiche existieren. Hier wurde die Möglichkeit geschaffen, Messdaten des PeO zu laden und in den folgenden drei Darstellungsarten auszugeben:

- flächige 3D-Darstellung die frei im 3D-Raum bewegt werden kann.
- Darstellung der Messdaten als zweidimensionales Graustufenbild. Die Bildebene liegt parallel zur Bildschirmebene
- Darstellung der Messdaten als zweidimensionales Graustufenbild auf einer frei im 3D-Raum beweglichen Ebene

Zusätzlich wurden in dieser Testanwendung die Möglichkeiten der Ausgabe von Zeichenprimitiven auf einer Ebene getestet. Ein Screenshot der Applikation nach dem Laden von Messdaten ist in Abbildung 46 ersichtlich.

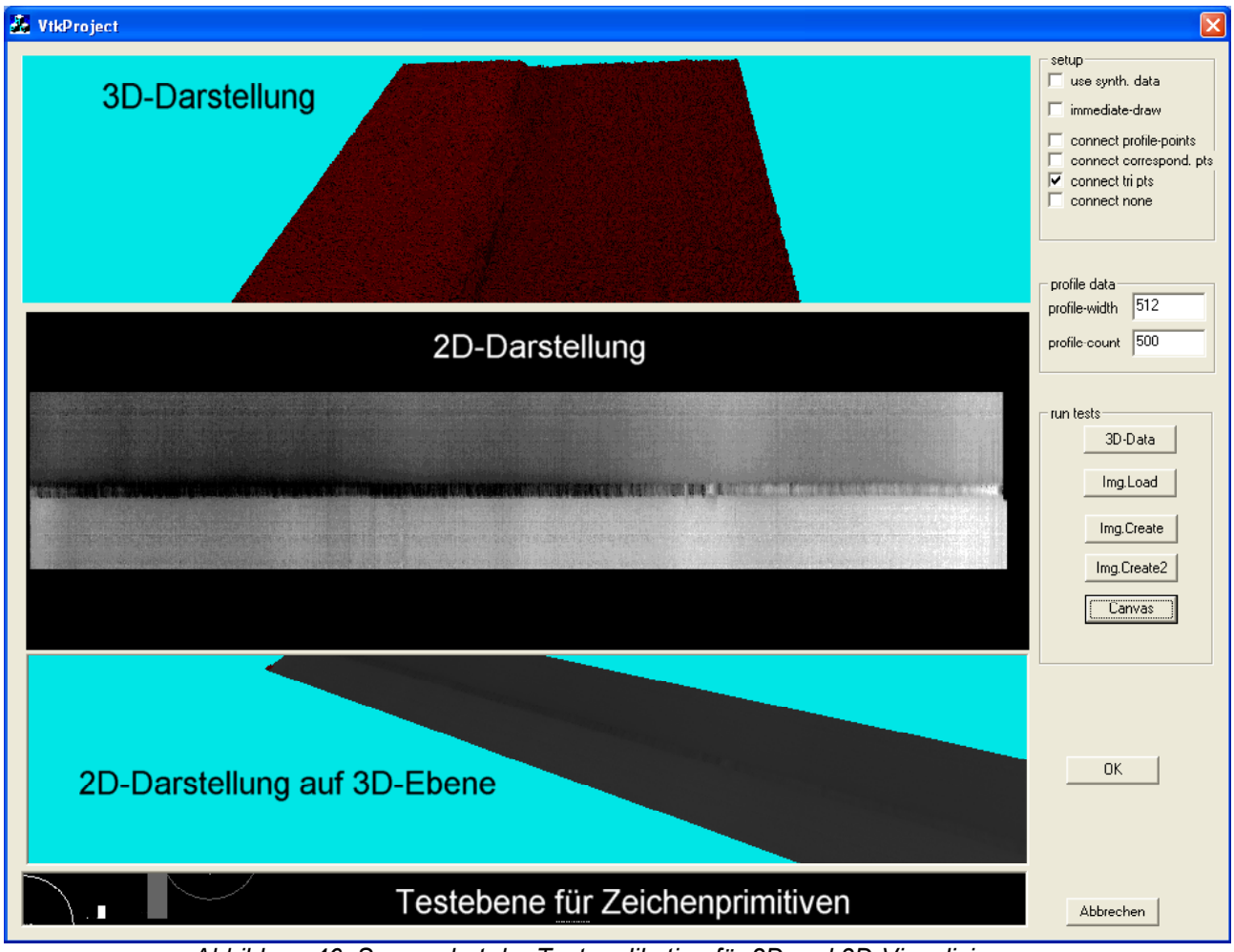


Abbildung 46: Screenshot der Testapplikation für 2D und 3D-Visualisierung

4.3 Datenreduktion

Die Gesamtverarbeitungszeit für die Messdaten teilt sich in die beiden Bereiche Vorverarbeitung und Grafikausgabe auf. Die Grafikausgabe stellt bei steigender Datenmenge zunehmend die Engstelle bei der Erreichung der geforderten Leistung dar. Die Zeit, die für die Ausgabe benötigt wird kann entweder durch leistungsfähigere Hardware oder durch eine Reduktion der auszugebenden Datenmenge erreicht werden. Das vtk bietet die drei Datenreduktionsverfahren DecimatePro, QuadricDecimation und QuadricClustering wobei das letztgenannte Verfahren die Implementatierung des schnellsten Algorithmus beinhaltet. Die implementierte Version des QuadricClustering-Verfahrens unterstützt zusätzlich noch eine stückweise Reduktion der Daten. Dies bietet den Vorteil, dass nicht die gesamte Anzahl an Messdatenprofilen im Speicher als 3D-Daten vorliegen muss. Es ist hierbei möglich, Teile der Daten zu laden, die Reduktion durchzuführen und den Speicher der Originaldaten wieder freizugeben. Diese Funktion wies in der verwendeten Version des vtk einen Fehler auf weshalb dieser Zusatznutzen nicht erzielt werden konnte. Weiterführende Literatur zu diesem Thema ist unter /26/, /6/ S.92ff, /24/, /25/ verfügbar. Nachfolgend werden Bildausschnitte der mit unterschiedlichen Qualitätseinstellungen reduzierten Daten aus der Datei panner3_gefiltert.prof (Profil 1 bis 1000) gezeigt (siehe Abbildungen 47 bis 51). Die in Abhängigkeit der Qualitätseinstellung entstehenden Polygonane sind in Tabelle 11 aufgeführt.

Qualitätseinstellung	Polygonanzahl	Abbildung
1/1/1	1020978	47
2/2/1	254490	48
3/3/1	112216	49
5/5/1	40198	50
7/7/1	20304	51

Tabelle 11: Polygonreduktion mittels QuadricClustering

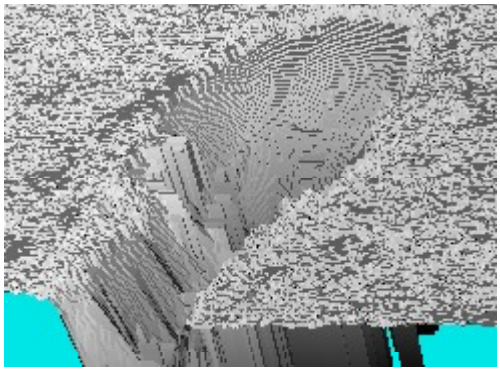


Abbildung 47: Datenreduktion, Einstellung 1/1/1

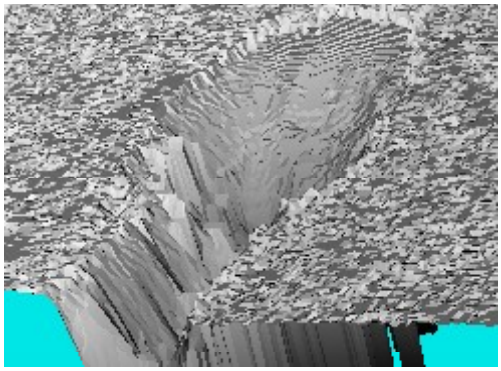


Abbildung 48: Datenreduktion, Einstellung 2/2/1

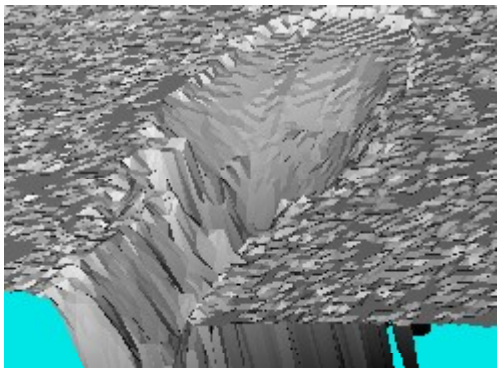


Abbildung 49: Datenreduktion, Einstellung 3/3/1

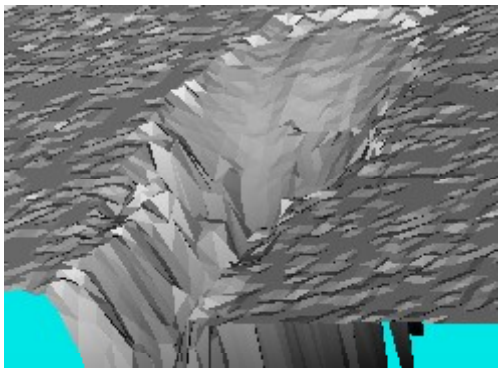


Abbildung 50: Datenreduktion, Einstellung 5/5/1

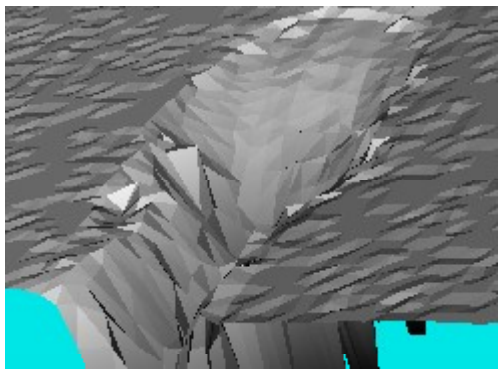


Abbildung 51: Datenreduktion, Einstellung 7/7/1

5 Zusammenfassung und Ausblick

5.1 Erreichte Ziele

Die vorliegende Arbeit hat gezeigt, dass durch den Einsatz von Datenreduktionsverfahren die Messdaten so aufbereitet werden können, dass damit auch auf leistungsschwächeren Systemen eine interaktve Visualisierung und Analyse ermöglicht wird.
Mit dem vtk wurde eine Toolbox ausgewählt, die sowohl in Bezug auf die verfügbaren Plattformen als auch durch die bereitgestellte Funktionalität über die gestellten Anforderungen der Visualisierung hinausreicht und für weitere Anwendungen im Bereich der Bildverarbeitung eingesetzt werden kann.
Im Rahmen dieser Arbeit entstand weiters eine Testapplikation, die zur Bewertung von Funktionalität und Leistung des vtk aber auch des verwendeten Computersystems dient und zum Vergleich von Systemen beziehungsweise unterschiedlichen Systemkonfigurationen herangezogen werden kann. Im Zuge der Untersuchungen wurde auch gezeigt, welchen Einfluß die Qualität des Grafiktreibers auf die Leistung des Visualisierungssystems ausübt. So kann durch einen mangelhaften Treiber die Funktionstüchtigkeit des Systems bis zur Unbrauchbarkeit reduziert werden.

Die Klassensammlung die im Verlauf dieser Diplomarbeit entstand, weist die folgenden Funktionsgruppen auf:
Datenzugriffsklassen:

Diese dienen dem Zugriff auf externe Messdaten unterschiedlicher Sensoren, wobei eine einheitliche interne Datenrepräsentation erstellt wird. Dadurch wird die Entwicklung von Funktionen zur Datenweiterverarbeitung vereinfacht, da nur mehr eine Schnittstelle zu den internen Daten implementiert werden muss. Die Implementierung zusätzlicher Sensoranbindungen wird durch den objektorientierten Ansatz und das gewählte Klassendesign vereinfacht. Es sind nur jene Funktionalitäten neu zu implementieren, die sich von der Basisklasse unterscheiden beziehungsweise in dieser noch nicht enthalten sind. Die Schnittstelle zur Offlineauswertung kann nun von einem Objekt gebildet werden, welches von der Klasse PITSD_DataMemReader abgeleitet ist und die benötigten Messdaten an die Auswertung weiterreicht.

Visualisierungsklassen:

Die implementierten Klassen kapseln die Funktionalität des vtk und stellen eine vereinfachte Schnittstelle für die Visualisierungsanwendung zur Verfügung. Der vollständige Zugriff auf die Art der Darstellung als auch auf die optionale Datenreduktion führen zu einer optimierten Anpassbarkeit an die Anforderungen der Anwendung und die Leistung des Zielsystems. Dadurch wird der Aufwand für die Erstellung neuer leistungsstarker Anwendungen reduziert.

5.2 Einsatz des Erreichten

Die Klassensammlung wird nach internen Tests voraussichtlich noch 2009 in der Visualisierung für ein Kundenprojekt eingesetzt. Durch den plattformunabhängigen Aufbau können nun auch Visualisierungslösungen auf Basis von Linux, Unix und Mac OS X – Systemen mit geringen Softwareanpassungen erstellt werden.

Aufgrund der rascheren Datenvisualisierung und der damit gesteigerten Interaktivität wird eine effizientere Analyse der Prozessdaten ermöglicht und führt damit zu einer gesteigerten Produktivität und Qualität des Prozesses. Dies wird zusätzlich durch die erweiterte Informationsvisualisierung unterstützt, da sich das Personal an der Anlage damit einen schnelleren und exakteren Überblick über die Qualität der Produktion verschaffen kann.

Der neu implementierte flyover-Modus, der einen virtuellen Flug über die Naht ermöglicht, stellt einen weiteren Wettbewerbsvorteil dar. Diese Art der Datendarstellung stellt ein Alleinstellungsmerkmal in Bezug auf den Mitbewerb dar, und bietet eine besonders eindrucksvolle Vertriebsunterstützung.

Die Verfügbarkeit des Quellcodes erleichtert nicht nur die Anpassung an veränderte Anforderungen sowie Implementierung neuer Funktionen sondern ermöglicht auch die rasche Analyse und Behebung von Fehlern, wodurch die Entwicklungszeiten reduziert und die Kundenzufriedenheit gesteigert werden kann.

5.3 Erweiterungsmöglichkeiten

Verwendung der stückweisen Datenreduktion zur Minimierung des Speicherbedarfes

Das Basiskonzept des vtk sieht vor, die zu visualisierenden Daten vollständig im Speicher aufzubereiten und dann der Ausgabe zuzuführen. Bei Verwendung der Datenreduktion wird daher zuerst das gesamte Mesh im Speicher generiert. Nach Anwendung des Reduktionsfilters (QuadricClustering), welches ein neues, reduziertes Mesh erzeugt, kann der Speicher für das unreduzierte Mesh wieder freigegeben werden. Die dadurch erzeugten Speicherauslastungsspitzen können dann bei Verwendung der Auslagerungsdatei die Gesamtperformance des Systems negativ beeinflussen. Der Aufbau des QuadricClustering-Filters erlaubt auch die Zuführung von Teilstücken des unreduzierten Meshes, was die temporär Speicherausnutzung deutlich reduzieren kann. Hier lag jedoch in der aktuellen vtk-Version 5.4 noch ein Fehler vor, der die Berücksichtigung in dieser Arbeit verhinderte.

Navigation innerhalb der 3D – Darstellung

Die derzeit als Standardimplementierung bestehende Navigation innerhalb des vtk weicht von den üblichen Bedienkonventionen ab, und ist daher besonders bei einem Wechsel von einer anderen Visualisierungssoftware nicht intuitiv bedienbar. Hier bietet sich die Ableitung einer Klasse von der Navigationsbasisklasse *vtkInteractorStyle* an. Die neu entstandene Klasse kann dann entsprechend dem Bedienkonzept adaptiert werden (vgl. /6/ S. 283 ff).

Anpassung der Rendereinstellung an die darzustellende Datenmenge

Der Immediate-Rendermodus erzielt ab einer systemabhängigen Größe des zu visualisierenden Datensatzes deutliche Performancevorteile gegenüber jenem Modus, der Displaylisten zur Beschleunigung des Bildaufbaues generiert. Der zusätzliche Bedarf an Ressourcen für die Speicherung der Listen zwingt das Betriebssystem bei Überschreitung des physikalisch verfügbaren Arbeitsspeicher zur Verwendung des virtuellen Speichers. Dies geht zu Lasten der Performance. Zur Auswahl des datenvolumenbedingten optimalen Rendermodus ist eine vorerst manuelle später auch eine automatische Konfiguration der Umschaltgrenzen denkbar.

5.4 Ausblick

Aktuell wird beim Hersteller des vtk, der Firma Kitware bereits eine Toolboxerweiterung entwickelt, die auf die GPGPU-Funktionalität moderner NVIDIA-Grafikkarten aufbaut. Der zu erwartende Leistungsschub durch den Einsatz der massiv parallel arbeitenden GPGPU-Systeme wird in den nächsten Jahren deutliche Verbesserungen sowohl in der Qulität als auch der Geschwindigkeit der Darstellung erbringen. Bestärkt wird diese Entwicklung durch die Bestrebungen, die GPGPU-Schnittstelle zu Standardisieren. Mit OpenCL, der Open Computing Language hat die Khronos Group, die bereits den OpenGL-Standard verwaltet, Anfang 2009 die Spezifikation für eine API veröffentlicht, die die standardisierte Programmierung in heterogenen Parallelverarbeitungsumgebungen aus CPU und GPU-Systemen ermöglicht (vgl. /27/).

5.5 Thesen

Die Performance der Visualisierung ist ein Produkt aus Hardware und Software.

Die Ergebnisse aus Kapitel 3.6 haben gezeigt, dass nur ein System bei dem Hard- und Software optimal aufeinander abgestimmt ist auch eine gute Visualisierungsleistung erbringt. Maßgeblich dafür ist es jedoch, auch die Softwaretreiber der Grafikhardware als Teil der Systems anzusehen.

Die Systeme zur Visualisierung großer Datenmengen werden zunehmend vom Einsatz von GPU-Hardware in Grafikkarten dominiert.

Die derzeit verfügbaren Anwendungen, die bereits die GPU-Hardware für Aufgaben außerhalb reiner Grafikoperationen verwendet, zeigt welches Potenzial in den GPU-Systemen vorliegt. Speziell durch die voranschreitenden Standardisierung der GPU-API wird dieses Potenzial zunehmend genutzt.

Die Vorhersagbarkeit bezüglich Skalierbarkeit der Applikation wird von der Software stärker beeinflusst als von der Hardware.

In den, in dieser Arbeit durchgeführten Untersuchungen konnte gezeigt werden, dass bereits der Einsatz eines schlecht implementierten Treibers die Visualisierungsleistung um den Faktor 10 reduziert. Umgekehrt bedeutet dies eine Steigerung um den Faktor 10 durch austausch einer Softwarekomponente. Derartige Steigerungen sind durch Wechsel einer Hardwarekomponente schwer möglich.

Datenvisualisierung zeigt die Grenzen aktueller Computersysteme auf.

Trotz des Einsatzes von Datenreduktionsverfahren stoßen aktuelle Visualisierungs-systeme rasch an absehbare Grenzen. Die Ursache liegt daran, dass für ein optimales Ergebnis alle Einflussfaktoren (hautpsächlich Rechenleistung, Grafikleistung und Arbeitsspeicher) über genug Reserven verfügen müssen. Gelangt nur ein Einflussfaktor an seine Grenzen so beeinflusst das die Gesamtleistung. Die aktuell verfügbaren Systeme können ohne Probleme durch real erzeugte Messdaten an und über die Grenzen der interaktiven Visualisierung gebracht werden.

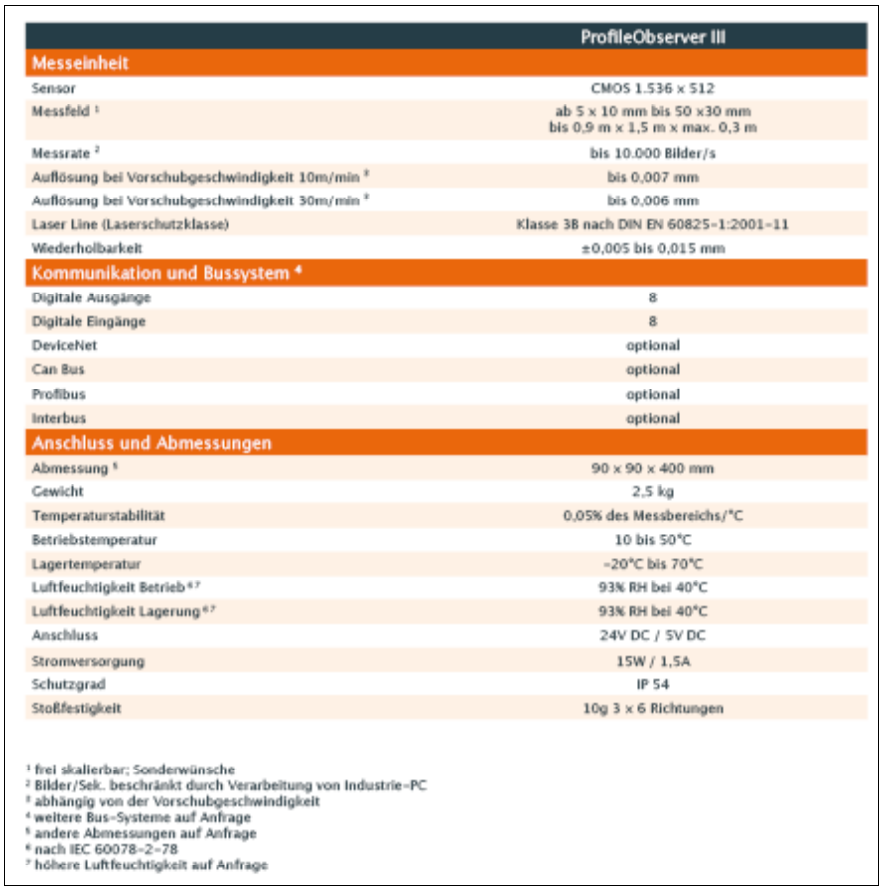
6 Anhang

6.1 Datenblätter

Messeingänge	
Optische Sensoreingänge für Lichtleitfaser [1]	2
Analog Eingang 0 – 20V [2]	1
Analog Eingang 0 – 20mA [3]	1
Messeinheit	
Optische Sensoreingänge für Lichtleitfaser	bis 10kHz/16bit integriert
Analog Eingang 0 – 20V	bis 500KHz/10bit
Analog Eingang 0 – 20mA	bis 500 KHz/10bit
Kommunikation	
Fast Ethernet 100Mbit/s	1
CAN Bus [4]	1
RS 232 [5]	2 x 230kbit/s
SPS Eingänge [6]	16
SPS Ausgänge [7]	16
DeviceNet	optional
CANopen	optional
Profibus	optional
Interbus	optional
TwinCAT ADS	optional
RSLinx	optional
Profinet	optional
Anzeige & Bedienung	
LCD Display	240 x 128
Tastatur	4 Tasten
LED Status Anzeigen	5
Akkustische Signalgeber	1
Prozessoreinheit	
CPU	Power PC 266MHz
Flash	max. 32MB
RAM	max. 256MB
Compact Flash Karte (CF)	max. 16GB
Anschluss & Abmessungen	
Netzanschluss	85 – 260 V~, 40 – 400 Hz, 40 VA
Abmessungen	230 (B) x 200 (H) x 150 (T) mm
Gewicht	7,2 kg
Umgebungstemperatur Betrieb	5 – 50° C
Umgebungstemperatur Lagerung	10 – 50° C
Lüfterlos	ja
Luftfeuchtigkeit Betrieb [8]	93% RH bei 40°C
Luftfeuchtigkeit Lagerung [8]	93% RH bei 40°C

[1] Photodiode 400–1.800nm
[2] Eingangswiderstand >100kΩ
[3] Eingangswiderstand 1kΩ
[4] bis 1Mbit/s unterstützte Standards sind CAN 1.2, CAN 2.0A, CAN 2.0B Passiv und CAN 2.0B Aktiv, DeviceNet-Adressierung
[5] Standard
[6] Extern versorgt, max. 24V
[7] Extern versorgt, max. 24V/1A. Summe max. 3A. Bei gleichzeitiger Belastung aller Ausgänge max. 200mA je Ausgang.
[8] Nach IEC 60078 – 2 –78

Abbildung 52: Datenblatt des ProcessObserver Advanced, Quelle:/13/



	ProfileObserver III
Messeinheit	
Sensor	CMOS 1.536 x 512
Messfeld [1]	ab 5 x 10 mm bis 50 x30 mm bis 0,9 m x 1,5 m x max. 0,3 m
Messrate [2]	bis 10.000 Bilder/s
Auflösung bei Vorschubgeschwindigkeit 10m/min [3]	bis 0,007 mm
Auflösung bei Vorschubgeschwindigkeit 30m/min [3]	bis 0,006 mm
Laser Line (Laserschutzklasse)	Klasse 3B nach DIN EN 60825-1:2001-11
Wiederholbarkeit	±0,005 bis 0,015 mm
Kommunikation und Bussystem [4]	
Digitale Ausgänge	8
Digitale Eingänge	8
DeviceNet	optional
Can Bus	optional
Profibus	optional
Interbus	optional
Anschluss und Abmessungen	
Abmessung [5]	90 x 90 x 400 mm
Gewicht	2,5 kg
Temperaturstabilität	0,05% des Messbereichs/°C
Betriebstemperatur	10 bis 50°C
Lagertemperatur	-20°C bis 70°C
Luftfeuchtigkeit Betrieb [6,7]	93% RH bei 40°C
Luftfeuchtigkeit Lagerung [6,7]	93% RH bei 40°C
Anschluss	24V DC / 5V DC
Stromversorgung	15W / 1,5A
Schutzgrad	IP 54
Stoßfestigkeit	10g 3 x 6 Richtungen

[1] frei skalierbar; Sonderwünsche
[2] Bilder/Sek. beschränkt durch Verarbeitung von Industrie-PC
[3] abhängig von der Vorschubgeschwindigkeit
[4] weitere Bus-Systeme auf Anfrage
[5] andere Abmessungen auf Anfrage
[6] nach IEC 60078-2-78
[7] höhere Luftfeuchtigkeit auf Anfrage

Abbildung 53: Datenblatt des ProfileObserver III, Quelle:/13/

	IS 5	IGA 5
Messbereiche:	600 ... 2000°C (MB 20) 800 ... 2500°C (MB 25) 1000 ... 3000°C (MB 30)	350 ... 1800°C (MB 18) 250 ... 2000°C (MB 20) 400 ... 2500°C (MB 25) 500 ... 3000°C (MB 30)
Teilmessbereich:	Beliebig innerhalb des Grundmessbereichs (Mindest-Messbereichsumfang 51°C)	
IR-Detektor:	Si-Photoelement	InGaAs-Photoelement
Spektralbereich:	0,8 ... 1,1 µm	1,45 ... 1,8 µm
Spannungsversorgung:	24 V DC ± 25%, stabilisiert, Restwelligkeit < 50 mV	
Leistungsaufnahme:	≤ 3 W (inkl. aktivem Laserpilotlicht)	
Analogausgang:	0 ... 20 mA oder 4 ... 20 mA, umschaltbar, eingeprägter Gleichstrom, temperaturlinear	
Schnittstelle:	Wahlweise RS232 oder RS485 (adressierbar), halbduplex, Baudrate 1,2 bis 38,4 kBd	
Auflösung:	0,1°C an Schnittstelle Am Analogausgang < 0,1% des eingestellten Teilmessbereiches, jedoch mind. 0,1°C	
Isolation:	Spannungsversorgung, Analog- und Digitalausgang sind gegeneinander galvanisch getrennt	
Parameter:	Einstellbar an Messumformer-Rückseite: Emissionsgrad, Einstellzeit, Analogausgang 0 ... 20 mA oder 4 ... 20 mA, Online / Offline für Einstellungen über PC/ Pyrometer. Zusätzlich über Schnittstelle lesbar und einstellbar: Teilmessbereich, Einstellungen für Maximalwertspeicher, Adresse, Baudrate. Über Schnittstelle nur lesbar: Messwert, Geräteinnentemperatur Bei Pyrometern mit PID-Regler per Software einstellbar: Sollwert, Proportionalband, Nachstell-/Verzugszeit, Stellgrößenbegrenzung.	
Maximalwertspeicher:	Eingebauter Einfach- bzw. Doppelspeicher. Löschen durch eingestellte Zeit t_{clear} (off; 0,01 s; 0,05 s; 0,25 s; 1 s; 5 s; 25 s), extern, über Schnittstelle oder auch automatisch bei neuem Messgut, Hold-Funktion zum halten der Temperaturanzeige auf dem aktuellen Wert (nicht bei Pyrometern mit PID-Regler)	
Emissionsgrad ε:	0,2 ... 1 einstellbar über Schalter im Gerät oder mit der Software *InfraWin* in Stufen von 0,01	
Erfassungszeit t_{90}:	≤ 2 ms, einstellbar auf 0,01 s; 0,05 s; 0,25 s; 1 s; 3 s; 10 s	
Messunsicherheit: (T_{Umg} = 25°C, ε = 1, t_{90} = 1 s)	< 350°C: 0,5% vom Messwert in °C + 1°C 350 ... 1500°C: 0,3% vom Messwert in °C + 1°C > 1500°C: 0,5% vom Messwert in °C + 1°C	
Wiederholbarkeit:	0,2% vom Messwert in °C + 1°C (T_{Umg} = 25°C, ε = 1, t_{90} = 1 s)	
Rauschäquivalente Temperaturdifferenz (NETD):	0,4°C (t_{90} = min=2 ms); 0,1°C (t_{90} = 0,01 s) (ε = 1, T_{Umg} =10 ... 40°C)	
Ausrichthilfe:	Wahlweise Laserpilotlicht (Laserklasse 2, max. Laserleistung < 1 mW, λ = 630 - 660 nm) oder Durchblickvisier oder Videomodul	
Schutzart:	IP65 (nach DIN 40050)	
Zul. Umgebungstemperatur:	0 ... 70°C	
Lagertemperatur:	-20 ... 70°C	
Gewicht:	550 g	
Gehäuse:	Edelstahl, Abmessungen siehe Zeichnungen rechts	
Bestandene EMV-Prüfungen:	Entspr. EU-Richtlinien über elektromagnetische Verträglichkeit	

Abbildung 54: Datenblatt: Pyrometer Impac IS5 / IGA5,Quelle:/15/

6.2 Abbildungsverzeichnis

6.3 Tabellenverzeichnis

6.4 Literaturverzeichnis

/1/ Apetri, Marius: 3D-Grafik Programmierung.
2. Auflage. Heidelberg: mitp, 2008

/2/ Wright, Richard S.; Lipchak, Benjamin; Haemel, Nicholas: OpenGL SuperBible.
4. Edition. Boston: Addison Wesley, 2007

/3/ Akenine-Möller, Tomas; Haines, Eric; Hoffmann, Naty: Real-Time Rendering.
3. Edition. Wellesley: A K Peters, Ltd., 2008

/4/ Martin, Ken; Hoffmann, Bill: Mastering CMake.
4. Edition. Colombia: Kitware Inc., 2007

/5/ Schroeder, Will; Martin, Ken; Lorensen, Bill: The Visualization Toolkit.
4. Edition. Colombia: Kitware Inc., 2006

/6/ Kitware, Inc.: The VTK Users's Guide.
Updated for VTK5. Colombia: Kitware Inc., 2006

/7/ Pavlidis, Theo: Algorithmen zur Grafik und Bildverarbeitung.
5. Auflage. Hannover: Heise, 1994

/8/ Beier, Ekkehard: Objektorientierte 3D-Grafik.
1. Auflage. Bonn; Albany: International Thomson Publishing GmbH

/9/ Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John: Entwurfsmuster. 1. Auflage. Bonn: Addison-Wesley-Longmann, 1996

/10/ Microsoft: Programming Guide for Direct3D 10.
URL:<http://msdn.microsoft.com/de-de/library/bb205067(en-us,VS.85).aspx>, verfügbar am 7.2.2009

/11/ NVIDIA Corp.: CUDA zone. URL <http://www.nvidia.com/object/cuda_home.html>, verfügbar am 25.1.2009

/12/ ATI Corp.: GPU Technology for Accelerated Computing. URL<http://ati.amd.com/technology/streamcomputing/index.html>, verfügbar am 25.1.2009

/13/ Plasmo: Plasmo Industrietechnik GmbH. URL<http://www.plasmo.eu>, verfügbar am 14.11.2008

/14/ Kitware: Kitware Inc. URL<http://www.kitware.com>, verfügbar am 14.11.2008

/15/ Impac: Pyrometer IGA 5. URL <http://www.impacinfrared.com/DE/produkte/pyrometer/p28.html>, verfügbar am 7.2.2009

/16/ Rezk-Salama, Christof: Volume Rendering Techniques for General Purpose Graphics Hardware. URL <http://subs.emis.de/LNI/Dissertation/Dissertation3/GI-Dissertations.03-12.pdf>, verfügbar am 7.4.2009

/17/ Blender Wiki: Blender 3D: Troubleshooting. URL<http://en.wikibooks.org/wiki/Blender_3D:_Troubleshooting>, verfügbar am 3.2.2009

/18/ Schneider, Uwe: Taschenbuch der Informatik.
6.Auflage. München: Carl Hanser Verlag, 2007

/19/ Microsoft: BitBlt. URL<http://msdn.microsoft.com/en-us/library/aa930997.aspx>, verfügbar am 21.4.2009

/20/ ATI: AMD FireStream 9250. URL <http://ati.amd.com/technology/streamcomputing/product_firestream_9270.html>, verfügbar am 19.3.2009

/21/ NVIDIA: GeForce GTX 295. URL <http://www.nvidia.com/object/product_geforce_gtx_295_us.html>, verfügbar am 19.3.2009

/22/ National Instruments: Visual Studio Test, Measurement and Automation Software. URL <http://www.national-instruments.com/mstudio/>, verfügbar am 19.3.2009

/23/ Kitware: The Visualization Toolkit – Applications. URL<http://www.vtk.org/VTK/resources/applications.html>, verfügbar am 17.12.2009

/24/ Garland, Michael;Heckbert, Paul S.: Surface Simplification Using Quadric Error Metrics. URL<http://mgarland.org/files/papers/ quadrics.pdf>, verfügbar am 28.12.2008

/25/ Lindstrom, Peter: Out-of-Core Simplification of Large Polygonal Models. URL <http://www.cc.gatech.edu/~lindstro/papers/siggraph2000/siggraph2000.pdf>, verfügbar am 28.12.2008

/26/ Knapp, Michael: Mesh Decimation using VTK. URL <http://www.cg.tuwien.ac.at/courses/Seminar/SS2002/Knapp_paper.pdf>, verfügbar am 28.12.2008

/27/ Khronos Group: OpenCL The Open Standard for Heterogeneous Parallel Programming. URL <http://www.khronos.org/developers/library/overview/opencl_overview.pdf>, verfügbar am 20.2.2009

6.5 Abkürzungsverzeichnis

API	Application Programming Interface Definition der Schnittstelle einer Komponente zur Anwendung
ARB	Architecture Review Board Zusammenschluss von Firmen um Entwürfe für Standards zu begutachten
CPU	Central Processing Unit Zentraleinheit des Verarbeitungssystems
CUDA	Compute Unified Device Architecture, ermöglicht die Verwendung des Grafikprozessors für Anwendung außerhalb Grafischer Anwendungen
DLL	Dynamic Link Library, Funktionsbibliothek die erst zur Laufzeit angebunden Wird
Flops	floatingpoint operations per second Rechenleistungskennzahl, die die Anzahl der ausgeführten Fließkomma-Operationen pro Sekunde angibt
GPGPU	General Purpose Graphics Processing Unit Elektronikkomponente zur Verarbeitung und Ausgabe grafischer Daten keine Spezialisierung für eine bestimmte Anwendung
GPU	Graphics Processing Unit Elektronikkomponente zur Verarbeitung und Ausgabe grafischer Daten meist spezialisiert für eine Anwendung (z.B. Pixel-Shader / Vertex-Shader)
IBM	International Business Machines Hersteller von Computersystemen
Lib	Library, Funktionsbibliothek die zumeist thematisch zusammenhängende Funktionen enthält
LUT	LookUp-Table Tabelle für die Zuweisung von Ausgangsdaten zu Eingangsdaten
MFC	Microsoft Foundation Classes Klassenbibliothek von Microsoft, die viele Funktionen von MS – Windows Kapselt
MS	Microsoft Hersteller von Betriebssystemen, Standard- und Entwicklungssoftware
MS-DOS	DiscOperatingSystem Textorientiertes Betriebssystem der Firma Microsoft
OpenGL	Open Graphics Library Offenes Standard-API für den Zugriff auf Grafikhardware
PeO	Profile – Observer Sensor zur 2D und 3D – Erfassung von Oberflächen Kombination aus Flächen- und Lichtschnittsensor
PsO	Process – Observer Sensor zur multispektralen Erfassung von Plasmasignalen
SGI	Silicon Graphics Inc. Hersteller von Hochleistungsgrafikworkstations, entwickelte OpenGL
SLI	Scaleable Link Interface Technologie, die von der Firma NVIDIA eingesetzt wird, um bis zu 4 GPU-Einheiten in einem PC zu verbinden
VGA	Video Graphics Adapter Elektronikkomponente zur Verarbeitung und Ausgabe grafischer Daten

7 Eigenständigkeits – Erklärung

Ich erkläre, dass ich die hier vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Bruck/Leitha, 30.04.2009

Erstbegutachter:

Prof. Dr.-Ing. Jürgen Ruck

Fachbereich Informationstechnik & Elektrotechnik

Hochschule Mittweida (FH), University of Applied Sciences

Technikumplatz 17

D-09648 Mittweida, Deutschland

Zweitbegutachter:

Dipl.-Ing. Dr. Thomas Grünberger

Plasmo Industrietechnik GmbH

Dresdner Strasse 81 - 85

A-1200 Wien, Österreich

8 Danksagung

Ich möchte mich herzlich bei allen bedanken, die mich bei der Erstellung dieser Arbeit und auch während des vorangegangenen Studiums so tatkräftig unterstützt haben. Ohne deren Unterstützung wäre diese Arbeit nicht möglich gewesen.

Im Besonderen gilt dies für meine Frau Maria und meine Kinder Marie-Theres und Julia, die durch ihr großes Verständnis die Phasen, in denen ich wenig Zeit für sie hatte, ertragen haben. Danken möchte ich auch meinen Eltern, ohne deren Unterstützung ich diesen Weg voraussichtlich nicht gehen hätte können.

Großer Dank gilt auch meinem Firmenbetreuer und Zweitbegutachter Dipl.-Ing. Dr. Thomas Grünberger, der dieses Diplomarbeitsthema initiiert hat und dessen wertvolle Anregungen und Erfahrungen maßgeblich zum Erfolg dieser Diplomarbeit beigetragen haben.

Abschließend möchte ich mich noch bei Prof. Dr.-Ing. Jürgen Ruck für die fachliche und organisatorische Unterstützung herzlichst bedanken. Er hatte auch außerhalb der regulären Dienstzeiten ein offenes Ohr für Fragen.