OpenIndy – Eine Open-Source-Software für Industriemesssysteme

B. Eng. Martin LUX, B. Sc. Jens WAMBACH, B. Sc. Benedikt RAULS,

Prof. Dr.-Ing. Fredie KERN, Dipl.-Ing. (FH) Heiko PALUSZEK

Zusammenfassung

OpenIndy ist ein studentisches Softwareentwicklungsprojekt im Rahmen des Masterstudienganges Geoinformatik und Vermessung der FH-Mainz (Prof. Dr.-Ing. Fredie Kern). Angeregt wurde und unterstützt wird OpenIndy durch die sigma3D GmbH, Mainz (Dipl.-Ing. (FH) Heiko Paluszek).

Mit dem langfristig angelegten Projekt sollen Kompetenzen aus dem Bereich des Softwareengineerings und der Programmierung erworben werden, zugleich ein Verständnis für Messprozesse im Bereich der Industrievermessung entwickelt und praktisch umgesetzt werden. OpenIndy besteht in seiner ersten Phase aus einem grundlegenden Konzept und der Implementierung einer Basisversion, die über Plugins vielfältig erweitert und angepasst werden kann. OpenIndy ist Open-Source und soll insbesondere für studentische, aber auch wissenschaftliche Entwicklungsarbeiten an der FH-Mainz genutzt werden.

1 Überblick

Die Arbeitspakete, welche im 15 Semesterwochenstunden umfassenden Modul „Projektarbeit“ im Wintersemester 2013/2014 bearbeitet wurden, lassen sich in drei Bereiche unterteilen. Zunächst sollte ein Konzept für eine Open-Source-Software entwickelt werden, mit der typische Messaufgaben der Industrievermessung vollständig von der Datenerfassung, über die Qualitätskontrolle und Visualisierung bis zur Protokollierung bearbeitet werden können. Dieses Konzept beinhaltet folgende wesentliche Anforderungen an OpenIndy:

- Es können 3D – Messsysteme verschiedener Art (Lasertracker, Tachymeter, etc.) und von verschiedenen Herstellern generisch ausgelesen und angesteuert werden.
- Es gibt ein offenes XML – Format für den Datenaustausch.
- Verwendete Auswertealgorithmen können leicht angepasst, ergänzt oder ersetzt werden; es werden möglichst keine harten Anbindungen an Bibliotheken, wie z.B. LAPACK/BLAS, vorgenommen,
- OpenIndy eignet sich als einfach erweiterbare Entwicklungsplattform für studentische Abschlussarbeiten.
- Sämtliche Auswerteprozesse sind transparent gestaltet und deren Ergebnisse vollständig einsehbar (White – Box).
- Die Verarbeitung ausgehend von originären Messwerten bis hin zu den abgeleiteten Zielparametern erfolgt streng und konsequent nach dem Varianzfortpflanzungsgesetz.
- Die GUI- Gestaltung ist modern und aufgabenorientiert.

Anhand des entwickelten Konzeptes wurde eine erste Basisversion implementiert. Um einer Weiterentwicklung und Nutzung der entwickelten Software möglich zu machen, erfolgte eine Veröffentlichung inklusive Dokumentation in einem Open-Source Repositorium (www.github.com/OpenIndy). Dort sind auch Tutorials und Beispieldaten für den leichten Einstieg in die Handhabung und zur Weiterentwicklung für Jederfrau/-mann hinterlegt.

2 Lizenzierung und Framework

OpenIndy ist lizenziert unter der Lesser General Public License (LGPL). Da OpenIndy in besonderem Maße für studentische und wissenschaftliche Entwicklungsarbeiten verwendet werden soll, bietet sich die Wahl einer Open – Source Lizenz an. Durch eine solche Lizenz ist es nicht nur möglich Zugriff auf eine kostenlose Spezialsoftware zu erlangen sondern die Weiterentwicklung kann auch auf mehrere Schultern verteilt werden, z.B. im Rahmen von studentischen Übungen und Praktika. Auf diesem Wege gewinnt zum einen OpenIndy an Funktionalität, zum anderen haben interessierte Studenten die Möglichkeit, ihre Fähigkeiten im Bereich der Softwareentwicklung und Messtechnik auszubauen. Die LGPL wurde ausgewählt, um auch kommerzielle Bibliotheken, wie z.B. Hersteller-SDKs zur Anbindung von Messsystemen in Form von Plugins integrieren zu können.

OpenIndy wird mit der integrierten Entwicklungsumgebung (IDE) Qt Creator[1] in der Programmiersprache C++ implementiert, welche das Cross-Plattform-Anwendungs- und GUI Framework Qt verwendet. Wichtige Entscheidungsgründe für die Wahl des Frameworks Qt und den Qt-Creator waren die Verfügbarkeit für verschiedene Betriebssysteme (MS-Windows, Mac OS, Linux, u.a.), die relativ leichte Erlernbarkeit aufgrund der hohen Abstraktion nativer APIs, die Unterstützung vielfältiger, auch hardwarenaher Funktionalitäten (Plugins, SQL, OpenGL, serielle Schnittstelle, Internet, …) und dem integriertem GUI-Designer, sowie der Möglichkeit verschiedene Compiler (MS-Visual C++, MinGW, …) nutzen zu können. Diese Flexibilität ist gerade im Hinblick auf die Einbindung in die Lehre sehr wertvoll.

3 Konzept

3.1 Begriffe und Datenmodell

Mit OpenIndy sollen Aufgaben im Bereich der Industrievermessung bearbeitet werden können. Um dieses anvisierte Anwendungsfeld näher zu spezifizieren, haben sich die Autoren an folgende Ausführungen von Möser (Möser, S. 7) für den Begriff für Industrievermessung orientiert:

> „Als Anfang der 50er Jahre die Ingenieurbauwerke immer größer und höher wurden, wurde der Begriff „Ingenieurvermessung“ geprägt. Er ist heute nach DIN 18710 definiert als: „Vermessung im Zusammenhang mit der Aufnahme, Projektierung, Absteckung und Überwachung von Bauwerken oder anderen Objekten“. Ende der 80er Jahre kamen Präzisionsmessungen zur Fertigungs- und Qualitätskontrolle beim Automobil-, Flugzeug- und Schienen-

[1] Qt Dokumentation

> fahrzeugbau hinzu. Damit wurde die „Industrievermessung“ mit Messaufgaben höchster Präzision im Nahbereich als Spezialgebiet der Ingenieurgeodäsie beschrieben. Parallel dazu entwickelte sich die rechnergestützte Koordinatenmessung im Maschinenbau mit „Methoden zur indirekten Bestimmung der gesuchten Maße auf dem Umweg über die aus einzelnen Punkten im Raum mittels analytischer Geometrie zu ermittelnden Formelemente, wie Punkt, Gerade, Ebene, Kreis, Zylinder.“ “.

Damit also soll OpenIndy den Vermessungsingenieur unterstützen bei 3D-Messaufgaben höchster Relativgenauigkeit (1ppm oder 0,01mm auf 10m) im Maschinen-, Fahrzeug- und Anlagen.

Zentrales Element in der „OpenIndy – Welt“ ist das „Feature“. Das „Feature“ ist ein abstraktes Objekt (Abbildung 1). Es kann die Ausprägung „Geometry“, „Station“, „Coordinatesystem“ oder „Transformation- Parameter“ haben. Unter „Geometry“ werden geometrische Raumprimitive, wie z.B. Punkt, Linie, Ebene, Kreis, Zylinder, Kugel, etc. zusammengefasst. Der Standpunkt, genauer die konkrete Aufstellung eines Messsystems, ist als Klasse „Station“ modelliert. OpenIndy kann mit beliebig vielen kartesischen Koordinatensystemen umgehen, welche über „Coordinatesystem“ im Detail modelliert werden. Zur Umrechnung zwischen den Koordinatensystemen werden die „Feature“ vom Typ „Transformation-Parameter“ verwendet. Diese enthalten alle nötigen Parameter um zwischen zwei Koordinatensystemen transformieren zu können. Jeder „Station“ ist ein Stationskoordinatensystem fest zugeordnet.

Unter Nominalwerten wird in OpenIndy eine Sollgeometrie mit ihren Sollwerten als Attribute (z.B. x, y, z, Radius) verstanden. Jedes „Feature“ kann Nominalwerte in verschiedenen Objektkoordinatensystemen besitzen.

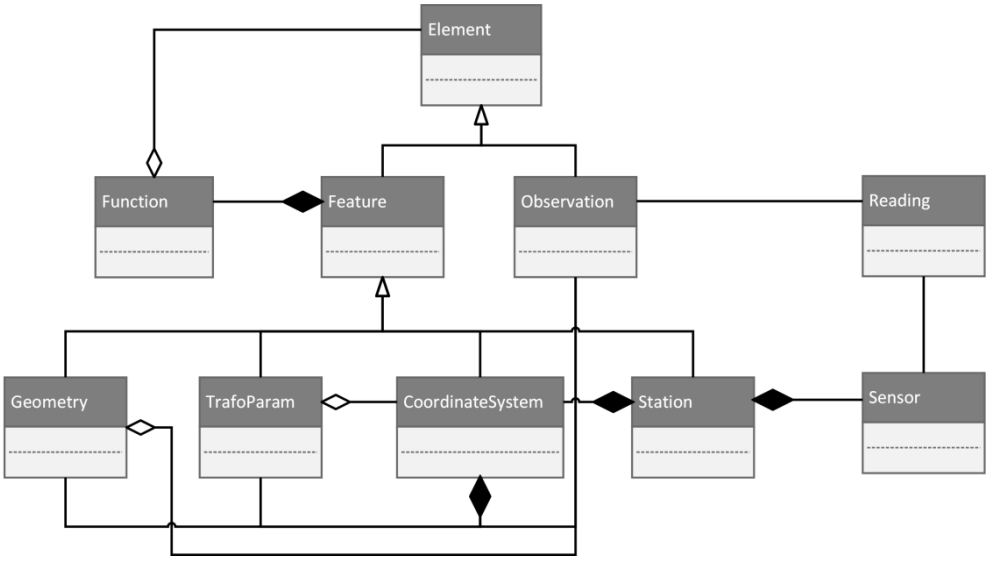

Abb. 1: Klassendiagrammausschnitt der Features in OpenIndy.

Die von den Messsystemen produzierten Messwerte werden in OpenIndy als „Readings“ bezeichnet. „Readings“ enthalten den Messwert, wie er ursprünglich erfasst wurde. Das können sowohl kartesische Koordinaten, als auch polare Elemente sein, oder einfach nur eine Strecke bzw. eine Richtung. „Observations“ sind immer kartesische Koordinaten, die je nach gewähltem Koordinatensystem transformiert werden.

Jeder „Station“ ist ein „Sensor“ zugeordnet. Ein „Sensor“ ist ein Messinstrument oder Messsystem, das über ein Plugin in seiner Funktionalität (z.B. Art der erfassbaren Messwerte, Messmodus, Korrelationen, Reduktionen, …) und seiner a-priori-Genauigkeiten genauer definiert wird. Das Plugin stellt quasi eine generische Schnittstelle zu den sehr speziellen Hersteller-SDKs dar.

Eine zentrale Rolle im Datenmodell stellt die abstrakte Klasse „Function“ dar. Unter „Function“ wird ein Berechnungsalgorithmus verstanden, mit dem aus einer Menge von Eingabewerten („Geometry“, „Transformation-Parameter“, …) die konkreten Zahlenwerte für die Parameter eines geometrischen Primitives unter Beachtung des Varianzfortpflanzungsgesetz bestimmt werden.

Ebenso wie „Sensor“ sind auch „Functions“ als Plugin konzipiert. Jeder „Geometry“ können beliebig viele „Functions“ zugeordnet werden. Die Reihenfolge in der „Functions“ einer „Geometry“ zugeordnet sind, entspricht der Reihenfolge in der die „Functions“ ausgeführt werden. In Abbildiung 2 sind die verschiedenen „Function“ - Typen zu sehen. „Fit – Functions“ haben den Zweck bei Überbestimmung aus „Observations“ ausgeglichene Geometrieparameter zu berechnen. Solche Algorithmen sind z.B. in (Drixler 1993) zu finden. Mit Hilfe von „Construct – Functions“ können Geometrien anhand anderer Geometrien konstruiert werden (z.B. Schnitt einer Ebene mit einer Linie um einen Punkt zu konstruieren). „Object transformations“ können verwendet werden, um bereits berechnete Geometrien zu verändern (z.B. einen Punkt entlang einer Linie zu verschieben). Um Transformationsparameter für die Transformation zweier Koordinatensysteme zu berechnen, sind „System transformations“ vorgesehen. Unter „Geodetic functions“ werden Berechnungen zur geodätischen Neupunktbestimmung, wie z.B. Vorwärtsschnitt zusammengefasst..

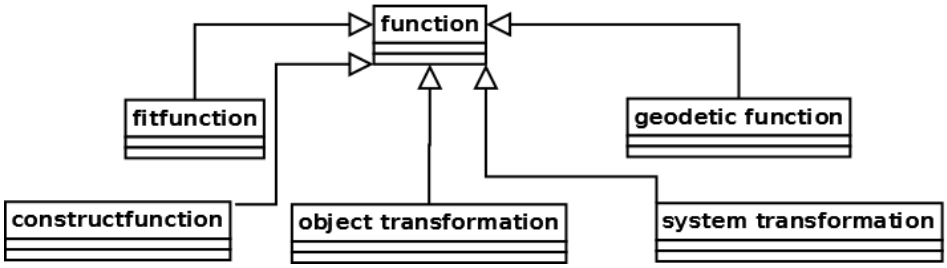

Abb. 2: Functions in OpenIndy.

3.2 Plugin – Konzept

OpenIndy liegt ein Plugin – Konzept zugrunde. Nahezu die gesamte Funktionalität ist in Plugins ausgelagert. Das Hauptprogramm dient in erster Linie der Steuerung des Zusammenspiels der einzelnen Plugin – Komponenten und der grafisch unterstützten Benutzerführung. Es gibt insgesamt drei grundlegende Bereiche, die in Plugins implementiert werden

können. Funktionen („Function“), Sensoren („Sensor“) und Netzausgleichungen („Networkadjustment“) (Abbildung 3). Das Pluginkonzept sieht vor, eigene Erweiterungen hinsichtlich Sensoren, Funktionen und Netzausgleichungen in einen gesonderten Plugin (DLL Datei) zu implementieren und dieses anschließend auszuliefern. Für Anwender ist es so möglich, die Funktionalität ihrer eigenen OpenIndy – Installation durch die Wahl geeigneter Plugins an ihre konkreten Anforderungen anzupassen – evtl. auch über etwaige lizenzrechtliche Grenzen hinweg. Auf diese Weise kann OpenIndy für viele unterschiedliche Arbeitsgebiete und Aufgaben optimiert, angewendet und ausgetestet werden. Die eigenen Entwicklungen können sich dabei auf die Implementierung der Berechnungsalgorithmik konzentrieren. Veränderungen auf der Plugin-Seite haben in der Regel keine Auswirkungen auf das GUI-Design.

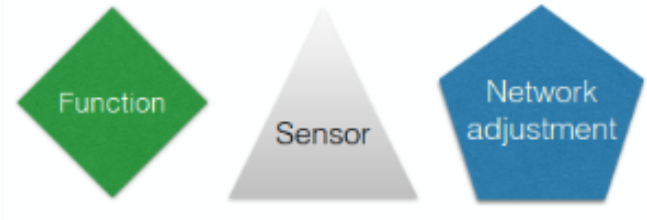

Abb. 3: Die Plugin - Schnittstellen von OpenIndy.

Wer OpenIndy um bestimmte Funktionalitäten erweitern möchte, muss sich nicht in das gesamte komplexe Datenmodell einarbeiten, sondern kann eine wohldefinierte Schnittstelle („Function“, „Sensor“, „Networkadjustment“) implementieren. Somit muss er lediglich die Schnittstelle verstehen. So lassen sich z.B. im Rahmen einer studentischen Abschlussarbeit verschiedene Approximationsalgorithmen für Kugeln miteinander vergleichen.

3.3 Default-Plugin

OpenIndy liegt nach Installation nicht als leere Hülle vor, sondern es existiert ein die Grundfunktionalität abdeckendes Plugin, das standardmäßig eingebunden wird. Das „Default – Plugin“ enthält die Implementierung zur Ansteuerung von Leica – Tachymetern über die GeoCOM Schnittstelle, sowie die Implementierung eines „Pseudotrackers“ mit dem die Handeingabe von Messwerten sowohl in kartesischer als auch in polarer Form möglich ist. Das „Default – Plugin“ beinhaltet derzeit vermittelnde Ausgleichungen für die Geometrien Punkt, Linie, Ebene und Kugel, die teilweise nach den Algorithmen von (Drixler 1993) implementiert sind. Einige sind zusätzlich auch durch die fitting-Routinen (Kern 2013) realisiert. Ebenso können Punkte, Linien, Ebenen und Kugeln entlang des Richtungsvektors einer Linie oder entlang des Normalenvektors einer Ebene um einen beliebigen Betrag verschoben werden. Des Weiteren kann der Radius einer Kugel nachträglich durch eine Funktion verändert werden. Außerdem ermöglicht das Plugin das Projizieren von Punkten in eine Ebene sowie die Berechnung des Schnittpunktes einer Linie mit einer Ebene und die Berechnung einer Schnittgeraden zweier Ebenen. Auch ist eine 7 Parameter Transformation implementiert, um die Transformationsparameter für die Transformation zweier Koordinatensysteme zu berechnen.

3.4 Berechnungsphilosophie

OpenIndy unterstützt konsequent das Konzept der Varianzfortpflanzung. Wie bereits erwähnt können Geometrien mehrere Funktionen zugeordnet werden. Jede Geometrie hat zusätzlich ein Statistik – Objekt welches durch die Funktionen benutzt und gefüllt werden kann. Dadurch können Funktionen mit Hilfe der eingehenden Statistik und ihrem funktionalen Zusammenhang Varianzfortpflanzung betreiben.

In Abbildung 4 ist dieses Konzept an einem Beispiel verdeutlicht. Ein Punkt A, der mehrfach gemessen wurde, erhält zunächst die Funktion „Best Fit“. Diese berechnet den aus *n* Beobachtungen gemittelten Punkt. Ergebnis sind die neuen Koordinaten *X, Y, Z* für den gemittelten Punkt, ein Verbesserungsvektor *v* sowie die Kovarianzmatrix $\Sigma_{XX,1}$. Anschließend wird diese Punktschätzung einer Verschiebung entlang des Normalenvektors einer Ebene unterzogen. Hierzu wird dem „Feature“ eine weitere Funktion „Translate by Plane“ zugewiesen. Die Ebene könnte zuvor selbst durch eine oder mehrere Funktionen berechnet worden sein und besitzt neben ihren Ebenenparametern (Elemente i, j und k des Normalenvektors) auch eine Kovarianzmatrix $\Sigma_{XX,plane}$ für diese.

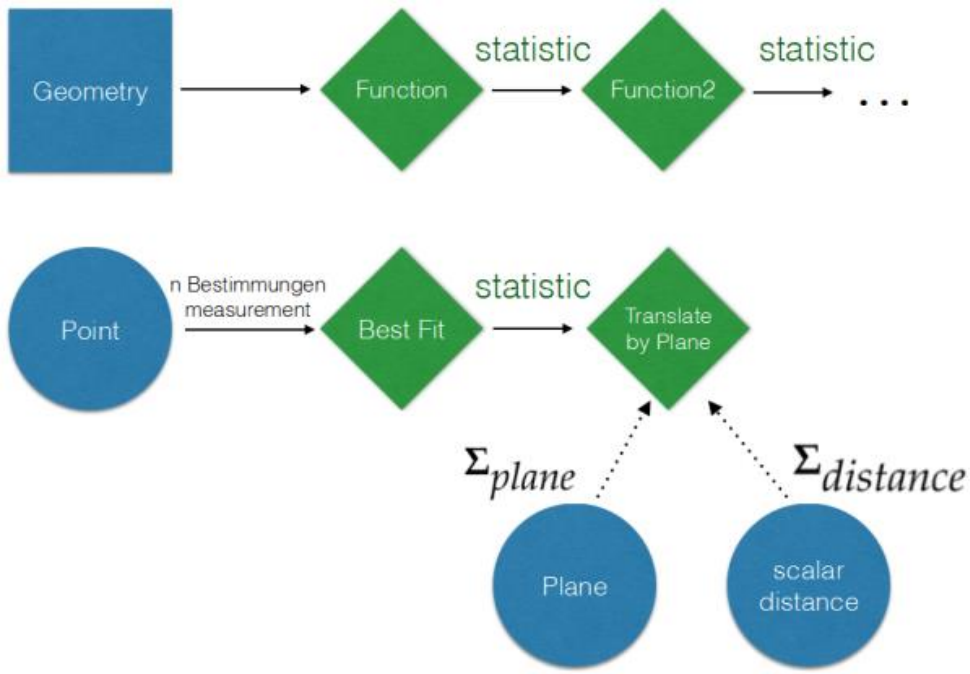

Abb. 4: Beispiel zur Berechnungsphilosophie in OpenIndy.

Auch der Verschiebebetrag kann fehlerbehaftet sein und somit eine von der Einheitsmatrix abweichende Kovarianzmatrix $\Sigma_{XX,distance}$ besitzen. Der funktionale Zusammenhang der Funktion „Translate by Plane“ soll wie folgt ausgedrückt werden:

$$\vec{X}_2 = \vec{X}_1 + d * \vec{n}_{plane} \tag{1}$$

In dieser Formel stehen im Vektor $\vec{X}_1$ die Koordinaten des Punktes A nach Anwendung der Funktion „Best Fit“. Der skalare Wert *d* repräsentiert den gewünschten Verschiebebetrag und der Vektor $\vec{n}_{plane}$ den normierten Normalenvektor der Ebene. Der Vektor $\vec{X}_2$ enthält nach Anwendung der Funktion die neuen, verschobenen Koordinaten des Punktes A. Zusammengefasst sind die Eingangsgrößen der Funktion „Translate by Plane“ demzufolge die alten Koordinaten $\vec{X}_1$ des Punktes A sowie die Kovarianzmatrix $\vec{\Sigma}_{XX,1}$ ebendieses Punktes, der Verschiebebetrag *d* mitsamt seiner Varianz $\Sigma_{XX,distance}$ und der Normalenvektor $\vec{n}_{plane}$ der Ebene sowie die Kovarianzmatrix $\Sigma_{XX,plane}$ der Ebene. Die neuen Koordinaten $\vec{X}_2$ des Punktes A können somit einfach nach Gleichung (1) berechnet werden. Um auch eine Aussage über die Genauigkeit des verschobenen Punktes A treffen zu können wird das Varianzfortpflanzungsgesetz (2) angewandt.

$$\Sigma_{FF,2} = F * \Sigma_{XX} * F^T \tag{2}$$

Die Matrix *F* repräsentiert dabei den funktionalen Zusammenhang der Funktion „Translate by Plane“ und hat die Dimension (3,7). In der Matrix Σ_{XX} sind die Kovarianzmatrizen der Ebene ($\Sigma_{XX,plane}$ mit der Dimension (3,3)), des Verschiebebetrages ($\Sigma_{XX,distance}$ mit der Dimension (1,1)) und des alten Punktes ($\Sigma_{XX,1}$ mit der Dimension (3,3)) zusammengefasst und kann wie folgt aufgebaut sein::

$$\Sigma_{XX} = \begin{pmatrix} \Sigma_{XX,p} & 0 & 0 \\ 0 & \Sigma_{XX,d} & 0 \\ 0 & 0 & \Sigma_{XX,1} \end{pmatrix} \tag{3}$$

Die Matrix $\Sigma_{FF,2}$ enthält nach Ausrechnen der Gleichung (2) die Varianzen und Korrelationen der Koordinaten des verschobenen Punktes A.

Diese Kette von Funktionen, die aneinander gereiht und nacheinander ausgeführt werden, kann beliebig fortgesetzt werden. Dabei wir jedes Mal konsequent die Kovarianzmatrix des veränderten Punktes aus den fehlerbehafteten Elementen durch die Anwendung des Varianzfortpflanzungsgesetzes berechnet und an die darauffolgende Funktion weitergegeben. Die Kette beginnt dabei in der Regel mit den a-priori-Genauigkeiten der eingesetzten Messsysteme. Diese sind vom Anwender z.B. auf Grundlage von Hersteller-Angaben in OpenIndy einzugeben.

4 Benutzeroberfläche

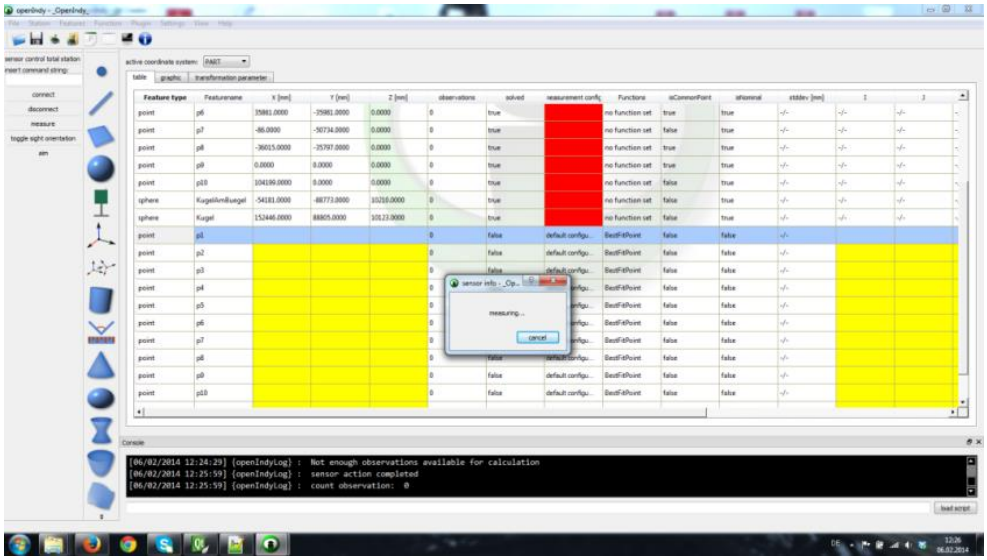

Abb. 5: Tabellarische Ansicht von OpenIndy.

Die Entwicklung der grafischen Oberfläche richtete sich nach dem momentanen „State of the Art“ von kommerziellen Softwarelösungen (Abbildung 5). Die Benutzeroberfläche von OpenIndy soll der Philosophie von „one model, many views“ entsprechen. OpenIndy hat eine interne Datenlogik, die die Daten verwaltet, alle Beziehungen der Features kennt und unabhängig von einer konkreten Darstellung (Tabelle, Grafik, …) ist. Die Benutzeroberfläche ermöglicht mehrere Sichten auf diese Datenebene, um die Messaufgabe optimal zu unterstützen, Unstimmigkeiten leicht zu entdecken und Statusinformationen einfach zu visualisieren.

Zum einen hat OpenIndy eine tabellarische Ansicht, die eine zeilenweise Repräsentation der Geometrien und Features ermöglicht. Hier werden neben Metadaten, wie Name und Featuretyp, auch Informationen zu dessen Attributen (*X*, *Y*, *Z* Koordinaten), der Anzahl von Messungen, der Messkonfiguration und der zugewiesenen Funktionen dargestellt. Die Zeilen-Reihenfolge spiegelt dabei in der Regel den zeitlichen Messablauf wieder.

In einer zweiten Tabellenansicht werden die Transformationsparameter angezeigt. Zu jedem Transformationsparameter werden das Start- sowie Zielkoordinatensystem und die einzelnen Transformationsparameter angezeigt.

Neben den tabellarischen Ansichten, verfügt OpenIndy auch über eine 3D Grafikansicht. Diese nutzt OpenGL und ermöglicht die grafische Repräsentation der Feature und ihrer Beziehungen untereinander.

Als dritte Repräsentationsmöglichkeit existiert in OpenIndy eine Konsole. In dieser werden Informationen, insb. Warn-, Fehler- und Statusmeldungen zu aktuell durchgeführten Aktionen und Funktionen ausgegeben.

5 OpenIndy im Einsatz

Das Konzept von OpenIndy ist mit dem Ziel entwickelt worden, um Messaufgaben aus dem Bereich der Industrievermessung, aber auch für konventionelle geodätische Aufgaben, zu lösen. An einem konkreten Anwendungsbeispiel soll der Umgang mit OpenIndy im industriellen Bereich erläutert werden. Damit stellen wir zugleich auch den derzeitigen Implementierungsstand hinsichtlich der Komplexität von Messaufgaben dar.

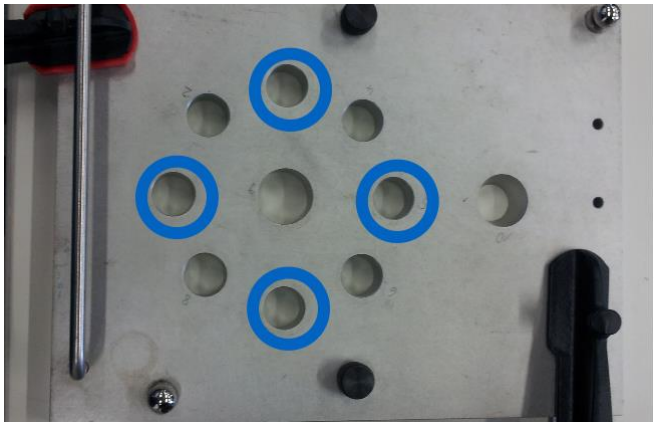

Abb. 6: Messobjekt der beispielhaften Messaufgabe.

Wie in Abbildung 6 dargestellt liegt ein Messobjekt mit 10 kreisförmigen Bohrungen sowie zwei Kugeln, eine rechts oben und eine links unten, vor. Die Sollwerte für diese 12 Geometrien wurden vorab mit einem Lasertracker bestimmt. Somit liegen Koordinaten für die Kreismittelpunkte sowie die Kugelmittelpunkte und –radien als gegeben vor. Ziel der Messung ist es nun die 12 Geometrien erneut mit einem Tachymeter zu messen, die Soll-Geometrien (10 Punkte und zwei Kugeln) aus der Lasertrackermessung als Nominalwerte einzulesen und anschließend eine Transformation zwischen dem Tachymeter – System und dem Nominal – System zu berechnen. Als Passpunkte sollen dabei die 4 Bohrungs-Mittel-Punkte dienen, die in Abbildung 6 blau umkreist sind.

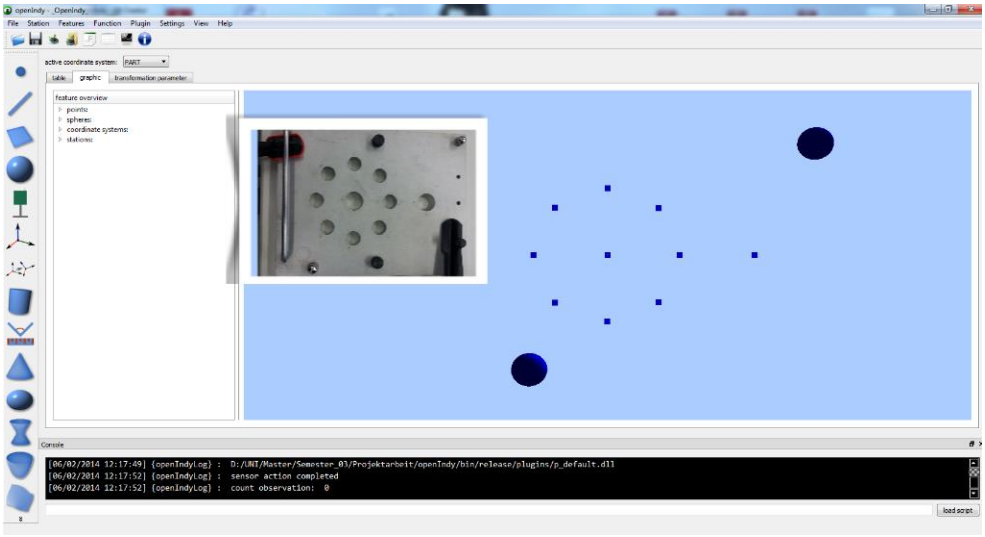

Abb. 7: 3D Grafikansicht des Messobjektes in OpenIndy.

Die eingelesenen Nominal – Geometrien sind in Abbildung 7 im 3D-Fenster von OpenIndy in blau dargestellt.

Für die Tachymetermessung wird ein Kugelreflektor verwendet. Die 10 Bohrungen werden zunächst gemessen, indem der Kugelreflektor in die Bohrung gelegt wird. Gesucht sind allerdings die Mittelpunkte der kreisrunden Bohrungen, die in der oberen Objektebene liegen. In Abbildung 8 ist diese Problematik skizziert. Gesucht ist der Punkt an der Pfeilspitze. Dazu wird die Ebene anhand mehrerer gemessener Oberflächenpunkte durch die „Best-Fit-Function“ bestimmt und alle 10 Punkte in diese Ebene entlang der Ebenennormalen projiziert. In der „OpenIndy – Welt“ bedeutet das, dass alle 10 Punkt-Geometrien jeweils zwei Funktionen besitzen: Die erste Funktion ist die Funktion „Best Fit Point“, die den Mittelpunkt aus *n* Beobachtungen berechnet. Die zweite Funktion, „Project in Plane“, projiziert den Punkt in die Ebene. Neben den 10 Punkten sind auch die Parameter der zwei Kugeln zu bestimmen.

Nachdem alle Geometrien sowohl im Nominalkoordinatensystem (Lasertracker-Aufstellung), als auch im Tachymeterkoordinatensystem vorliegen kann über eine 7 – Parameter –Transformation die Position und Orientierung des Tachymeters zum Nominal-System bestimmt werden. Als Passpunkte dienen dabei, wie bereits erwähnt, die 4 in Abbildung 6 blau markierten Punkte.

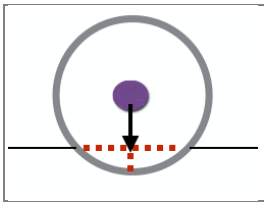

Abb. 8:
Projektion des Kugelmittelpnktes in die obere Objektebene bei Messung einer Bohrung mit einem Kugelreflektor.

Um das Ergebnis der Transformation beurteilen zu können sind in Abbildung 9 die Koordinaten des Punktes 9 im Nominalsystem gegenübergestellt. Wie zu erkennen ist stellt der Punkt im Nominalsystem den Ursprung des Koordinatensystems dar. Die „actual“ – Koordinaten des Punktes 9 sind die in das Nominalsystem transformierten Stationskoordinaten, repräsentieren also die Tachymetermessung. Die Abweichungen liegen zwischen ein und zwei zehntel Millimeter. Die Abweichungen sind deshalb so gering, weil die Transformation mit Maßstab gerechnet wurde, sodass die Messabweichungen sich zum Teil in einem von eins abweichenden Maßstab wiederspiegeln.

Feature name	X [mm]	Y [mm]	Z [mm]
p9 **(actual)**	-0.1188	0.1031	0.1953
p9 **(nominal)**	0.0000	0.0000	0.0000

Abb. 9: Vergleich der Koordinaten eines Punktes nach der Transformation.

6 Alleinstellungsmerkmale von OpenIndy

OpenIndy ist Open – Source, was bei anderer Software im Bereich Messtechnik i.d.R. nicht der Fall ist. Das ermöglicht eine kostenlose Nutzung und jedem die aktive Teilnahme bzw. Teilhabe durch Weiterentwicklung an dem Projekt. Bei der Entwicklung von OpenIndy wurde zudem darauf geachtet keine externen Abhängigkeiten in Form von Fremdbibliotheken einzubauen, ohne die OpenIndy nicht mehr funktionieren würde. Für die Vektor- und Matrizenalgebra wird defaultmäßig die Open – Source Bibliothek Armadillo (Sanderson) verwendet. Durch eine abstrakte Schnittstelle für Routinen zur linearen Algebra ist die Bibliothek jedoch jederzeit durch eine andere austauschbar. Eine Besonderheit von OpenIndy ist zudem sein ausgeprägtes Plugin – Konzept. Dadurch kann die Funktionalität auf spezielle Anforderungen angepasst werden ohne dabei das gesamte Datenmodell von OpenIndy kennen oder verstehen zu müssen. Die Komplexität der Entwicklung neuer Funktionen oder Sensoren beschränkt sich dadurch auf die Entwicklung der Funktionen selbst und nicht auf das Nachvollziehen bereits vorhandener Module. Ebenfalls einmalig nach Kenntnis der Autoren ist die konsequente Integration und Unterstützung des Konzepts der Varianzfortpflanzung. Zusammen mit der offenen Darstellung aller Zwischen- und Endergebnisse von Berechnungen sowie deren Statistik, bietet OpenIndy damit ein großes Maß an Transparenz. OpenIndy wurde auf drei verschiedenen Plattformen (Windows, Mac und Linux) entwickelt und getestet und kann somit auf diesen Plattformen eingesetzt werden. Außerdem wurde für OpenIndy ein offenes XML – Austauschformat entworfen, das zugleich zur persistenten Speicherung aller Projektinformationen und Messdaten dient. Dadurch können andere Programme OpenIndy Projekte auf einfache Weise über Standard-Parser importieren und nutzen. Zudem ist damit ein Weg aufgezeigt, wie OpenIndy in einer Auswertekette verschiedener Programmsysteme eingebunden werden könnte (Batchbetrieb).

7 Ausblick

OpenIndy ist unter GitHub veröffentlicht und verfügbar (https://github.com/openindy). Die Autoren hoffen durch diese Social Media Plattform weitere Mitstreiter - Entwickler und/oder Anwender - zu finden, die das Projekt unterstützen, weitere Ideen und Anregungen einbringen und neue Funktionalitäten implementieren.

OpenIndy wird gemäß der eingangs gestellten Zielvorgabe im Rahmen der Lehre von Prof. F. Kern im Masterstudiengang Geoinformatik und Vermessung an der Fachhochschule Mainz in den folgenden Semestern weiterentwickelt. Damit verbunden ist die Hoffnung mehr Studierende für das spannende, vielfältige und immer wichtiger werdende Berufsfeld der Industrievermessung zu begeistern.

Die mit viel Fleiß und großem Engagement am Projekt beteiligten Studierenden konnten eine hohe Kompetenz sowohl im Softwareengineering, als auch in der Messtechnik nach und nach aufbauen. Sie planen sich auch bei ihren Masterarbeiten mit OpenIndy zu „beschäftigen“.

Die Sigma3D GmbH wird das Projekt weiterhin mit ihrer Praxiserfahrung unterstützen.

Literatur

Drixler, Erwin (1993): *Analyse der Form und Lage von Objekten im Raum*. Dissertation. Deutsche Geodätische Kommission bei der Bayerischen Akademie der Wissenschaften. Reihe C Heft Nr. 409

Kern, Fredie (2010): Fitting - Programm für die Approximation von geometrischen Formen. Version 1.0.1 *https://github.com/zanzaraz/fitting*

National Science Foundation *http://www.netlib.org/lapack/*

National Science Foundation *http://www.netlib.org/blas/*

Möser, Michael (2007): *20 Jahre Industrievermessung*. In: Tagungsband zum 3. Dresdener Ingenieurgeodäsietag, Schriftenreihe des Geodätischen Instituts der TU Dresden, Nr. 4, S. 7-19. *http://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_forst_geo_und_hydrowissenschaften/fachrichtung_geowissenschaften/gi/ig/termine/papers/moeser.pdf* (Zugriff 18.2.2014)

Qt Dokumentation *qt-project.org/doc/* Version 5.1

Sanderson, Conrad *http://arma.sourceforge.net/*

Lux, Martin
Fachhochschule Mainz
Martin.lux@students.fh-mainz.de

Wambach, Jens
Fachhochschule Mainz
Jens.Peter.Wambach@students.fh-mainz.de

Rauls, Benedikt
Fachhochschule Mainz
Benedikt.Rudolf.Wilhelm.Rauls@students.fh-mainz.de

Prof. Dr. Kern, Fredie
Fachhochschule Mainz
Fachbereich Technik - Lehreinheit Goeinformatik und Vermessung
fredie.kern@fh-mainz.de

Dipl. – Ing Paluszek, Heiko
Sigma3D GmbH
paluszek@sigma3d.de