Hochschule Neubrandenburg
University of Applied Sciences

Entwicklung einer Software zur Visualisierung von Trajektorien und deren Zerlegung in geometrische Primitive mittels Gauß–Helmert-Modells

Bachelorarbeit

im Studiengang Vermessungswesen

zur Erlangung des akademischen Grades
Bachelor of Engineering

eingereicht von:
Philipp Engel

Erstprüfer: Prof. Dr.-Ing. Karl Foppe
Zweitprüfer: Prof. Dr.-Ing. Hans-Jürgen Larisch

urn:nbn:de:gbv:519-thesis2010-0494-9

September 2010

Kurzzusammenfassung

In vielen Anwendungsfällen von Natur- und Ingenieurwissenschaft wird mit Trajektorien gearbeitet. Um die Qualität solcher Bahnkurven, z. B. in Form von Fahrspuren oder Gleisstrecken, beurteilen zu können, müssen sie zerlegt werden. Im Rahmen dieser Arbeit wurde ein Ausgleichungsprogramm entwickelt, dass eine Grundlage dafür darstellem soll. Es wurden praktische Versuche mit verschiedenen Messsystemen durchgeführt, um passende Daten für eine Ausgleichung zu sammeln. Die Ergebnisse lassen sich mit Hilfe eines Punktplotters visuell interpretieren.

Inhaltsverzeichnis

1. Einleitung

Dieser kurze Einleitung soll Motivation und Problemstellung der Bachelorarbeit erläutern. Des Weiteren wird auf die typografischen Besonderheiten eingegangen und der Inhalt der beiliegenden CD-Rom aufgeführt.

1.1. Motivation

Gegenstand der angewandten Geodäsie ist u. a. die Erfassung verschiedener Objekte und Gegebenheiten in der realen Welt. Bei den dabei gewonnenen Messwerten kann es sich um Bahnkurven, sogenannte *Trajektorien*, handeln. Sie entstehen beispielsweise bei der kinematischen Vermessung von Fahrbahnwegen schienengebundener oder schienenloser Fahrzeuge. Insbesondere bei der Baumaschinensteuerung oder dem *Precision Farming* sind die verschiedenen geometrischen Primitive, aus denen die Trajektorien zusammengesetzt sind, von Interesse.

Um Aussagen über die Art der Grundfiguren und die Qualität der Messungsergebnisse treffen zu können, müssen die Trajektorien zerlegt werden. Die Ausgleichungsrechnung soll dabei ein Werkzeug zur Beurteilung der Genauigkeit sein. Mit Hilfe von Computern lassen sich die anfallenden Rechenschritte automatisch durchführen und die Ausgleichungsergebnisse visuell darstellen.

1.2. Problemstellung

Die Ausgleichung von Bahnkurven per Hand durchzuführen, ist sehr aufwendig. Dabei können schnell Rechenfehler auftreten, die in das Ausgleichungsergebnis einfließen. Computer können die anfallenden Rechenschritte sehr viel schneller und genauer durchführen. Gegenstand dieser Arbeit soll daher die Entwicklung einer Software sein, mit der Bahnkurven in geometrische Primitive wie Strecken, Kreise oder Polynome zerlegt und anschließend ausgeglichen werden können. Um diesen Vorgang zu automatisieren, muss die Software die Primitive, aus denen die Bahnkurven bestehen, selbstständig erkennen und separieren.

Aus den gegebenen Koordinaten einer Bahnkurve lässt sich ihre Form ohne Visualisierung nur schwer ableiten. Daher ist eine Komponente zum Plotten von Messwerten

sinnvoll. Auch die Ergebnisse der Ausgleichung, d. h. die erzielten Verbesserungen, sind durch eine grafische Aufbereitung besser nachzuvollziehen.

Damit überhaupt Bahnkurven für eine Ausgleichung zur Verfügung stehen, sind geeignete Messwerte zu sammeln. In der Geodäsie werden satellitengestützten und terrestrische Messverfahren mit unterschiedlichen Genauigkeiten angewandt. Die Auswirkungen der Genauigkeiten auf das Ausgleichungsergebnis sind daher ebenfalls von Interesse.

1.3. Typografische Konventionen

In dieser Bachelorarbeit werden einige typografische Konventionen angewendet, die das Verständnis des Inhalts unterstützen sollen und nachfolgend aufgelistet sind.

- *Kursiv* gesetzt werden
  - Herstellernamen und Produktbezeichnungen,
  - Eigennamen,
  - fremdsprachige Wörter,
  - Variablen,
  - Funktions- und Klassennamen.
- In `nichtproportionaler Schrift` gesetzt werden
  - Internetadressen und
  - Dateinamen.

Aus Gründen der Übersichtlichkeit werden Herstellernamen nach der ersten Nennung nicht weiter ausgezeichnet.

In der Ausgleichungsrechnung sind unterschiedliche Schreibweisen für Matrizen und Vektoren verbreitet. Es wird die Notation von Niemeier[1] verwendet.

[1] vgl. Niemeier, 2008

1.4. Software

Die Ergebnisse dieser Bachelorarbeit liegen in elektronischer Form auf der beiliegenden CD-Rom vor. Zum Inhalt gehören

- das entwickelte Ausgleichungsprogramm *Java Trajectory Analyser*,
- eine Installationsanleitung,
- die gesammelten Messdaten sowie
- die Bachelorarbeit im PDF-Format.

2. Messverfahren

Die Aufnahme einzelner Punkte ist ein wichtiger Gegenstand der Geodäsie und erfolgt durch spezielle Messsysteme. Am weitesten verbreitet sind dafür elektronische Tachymeter und GPS-Empfänger zur Positionsbestimmung durch den Empfang von Satellitensignalen. Im Rahmen dieser Arbeit wurden Messungen mit beiden Systemen durchgeführt, um Daten für das zu entwickelnde Ausgleichungsprogramm zu sammeln. Nachfolgend sollen sie deshalb kurz vorgestellt und erläutert werden.

2.1. Tachymetrie

Unter dem Begriff *Tachymeter* oder *Totalstation* werden elektronische Theodolite verstanden, die mit einem elektrooptischen Distanzmesser (EDM) ausgerüstet sind. Mit ihnen können nicht nur Richtungen sondern auch Distanzen bestimmt werden. Das Prinzip der Richtungsmessung ist bei Tachymetern – wenn auch elektronisch und automatisch – weitestgehend identisch mit dem von Theodoliten. Ein Sensor erfasst die Richtung entweder mit einem feststehenden Teilkreis (stationäre Methode) oder einem rotierenden Teilkreis bzw. einer rotierenden Marke (dynamische Methode).

Bei der stationären Methode wird zwischen dem Codeverfahren mit einem binär codierten Teilkreis und dem Inkrementalverfahren mit einem Strichraster auf dem Teilkreis unterschieden. Der Teilkreis kann beim Codeverfahren durch elektrooptische oder magnetische Sensoren abgetastet werden. Die Codezeichen sind dabei auf konzentrischen Spuren angebracht. Nach einer Interpolation können sie in die jeweilige Richtung übersetzt werden. Bei dem Inkrementalverfahren tastet eine Lichtschranke das Strichraster des Teilkreises ab. Ein Zähler registriert die Hell-Dunkel-Wechsel und der Mikroprozessor des Tachymeters bestimmt daraus die Richtung. Die Absolutstellung des Teilkreises kann bei diesem Verfahren nicht erfasst werden. Es existiert somit keine physische Nullrichtung und die Orientierung des Gerätes geht nach dem Ausschalten verloren.[1]

Die dynamische Methode beruht auf dem Prinzip der Zeitmessung zwischen den Durchgängen einer rotierenden Blende durch zwei Lichtschranken. Die zweite Lichtschranke ist mit dem Mittelteil des Tachymeters (*Alhidade*) verbunden und dadurch drehbar. Die Richtung lässt sich direkt aus der gemessenen Zeit ableiten und ist abhängig von der Stellung der zweiten Lichtschranke.

[1] vgl. Joeckel u. a., 2008

Abbildung 2.1.: Automatische Totalstation Trimble *S6*

Bei der elektrooptischen Distanzmessung sind vor allem zwei Verfahren verbreitet. Beim Impulsmessverfahren dient ein Wellenpaket (Puls) als Signal und wird für einen kurzen Zeitraum ausgesendet. Mittels einer Zähleinheit kann die Laufzeit ermittelt und daraus die Streckenlänge abgeleitet werden. Dagegen wird beim Phasenvergleichsverfahren einer vom Tachymeter kontinuierlich abgestrahlten Trägerwelle ein sinusförmiges Messsignal aufmoduliert. Aus dem Vielfachen der Modulationswellenlänge und der Phasenverschiebung lässt sich die Streckenlänge bestimmen. Eine Messung besteht immer aus Grob- und Feinmessung. Bei beiden Verfahren werden intern zur Genauigkeitssteigerung mehrere Einzelmessungen durchgeführt und die bestimmten Distanzen anschließend gemittelt.

Die *Tachymetrie* ist die Vermessung mit Hilfe von Tachymetern. Die Kerngebiete dieses Verfahrens sind die Punktaufnahme und -absteckung. Die gleichzeitige Bestimmung von Richtung und Strecke zu einem Reflektor ermöglicht die Aufnahme von Punkten sowohl in der Lage, als auch in der Höhe. Der umgekehrte Fall ist die Absteckung, bei der die Punkte in die Örtlichkeit übertragen werden. Um eine hohe Wirtschaftlichkeit bei Absteckaufgaben zu erreichen, haben sich elektronische Einweishilfen etabliert. Die verbale Informationsübermittlung zwischen Geräteoperator und Reflektorträger ist aber immer noch weit verbreitet. In den letzten Jahren setzten sich auch zunehmend zielsuchende und zielverfolgende Totalstationen auf dem Markt durch (Abb. 2.1). Diese Instrumente sind mit Servomotoren oder elektromagnetischen Direktantrieben ausgerüstet. Zusammen mit speziellen Zielsensoren eignen sie sich für

automatische *Monitoring*-Aufgaben, den Einsatz als Ein-Mann-Messsystem oder für die kinematische Vermessung.

Aufgrund der verschiedenen Messprinzipien können mit einem Tachymeter Richtungen schneller bestimmt werden als Strecken. Während bei allen Richtungsmessverfahren die zu ermittelnde Richtung praktisch unmittelbar zur Verfügung steht, benötigt die elektrooptische Distanzmessung aufgrund der Laufzeitverzögerung und der vielen Einzelmessungen mehr Zeit. Das hat bei der kinematischen Tachymetrie zur Folge, dass Strecken gegenüber der statischen Vermessung ungenauer sind und sich das Messergebnis verschlechtert.

2.2. Global Positioning System

In der heutigen Zeit sind *Global Navigation Satellite Systems* (GNSS) nicht mehr aus unserem Alltag wegzudenken. Einst entwickelt, um dem U.S.-Militär eine schnelle Echtzeitpositionsbestimmung zu ermöglichen, sind GNSS-Empfänger mittlerweile in Autos, Flugzeugen, Schiffen und vielen anderen Geräten, die eine absolute Positionsbestimmung benötigen, zu finden. Selbst in Bereiche, für die die Genauigkeit von GNSS nicht ausreicht, hält die Positionsbestimmung mittels GPS, GLONASS oder demnächst Galileo Einzug. In der Geodäsie ist die Vermessung mittels Satelliten nicht mehr wegzudenken.

Die Entwicklung des *Navigational Satellite Timing and Ranging – Global Positioning System*, kurz NAVSTAR-GPS bzw. GPS, begann am 17. April 1973. Die Luftwaffe der Vereinigten Staaten wurde mit der Entwicklung eines Ortungssystems beauftragt, das dem Nutzer unabhängig von dessen eigener Bewegung seine präzise dreidimensionale Position, seine Geschwindigkeit sowie die aktuelle Zeit überall auf der Welt liefern kann und von Wettereinflüssen unabhängig ist. Am 27. Juni 1977 wurde der erste NAVSTAR-Satellit gestartet und am 17. Juli 1995 das System offiziell in Betrieb genommen.[2]

Das grundlegende Navigationsprinzip von GPS beruht auf der Messung sogenannter Pseudoentfernungen zwischen dem Empfänger und den Satelliten. Die Bestimmung der Distanz zwischen Empfangsantenne und Satellit erfolgt durch Korrelation des Satellitensignals mit einer im Empfänger erzeugten Kopie des Codes. Dabei werden Aussendezeitpunkt am Satelliten und Empfangszeitpunkt am Empfänger miteinander verglichen.

Bei GPS-Messungen wird zwischen relativer und die absoluter Punktbestimmung unterschieden. Für die absolute Punktbestimmung ist, im Gegensatz zur relativen, lediglich ein Empfänger erforderlich, wobei nur die Codemessung angewandt wird. Ein Referenzempfänger berechnet die Entfernungen zu den einzelnen Satelliten. Nach der Bestimmung des Empfängeruhrfehlers werden die Sollentfernungen mit den um den

[2] vgl. Bauer, 2003

Empfängeruhrfehler verbesserten Pseudoentfernungen verglichen und die erhaltenen Differenzen als Korrekturwerte an die Messwerte des Nutzers angebracht. Die Übertragung der Korrekturdaten erfolgt je nach Anwendungsbereich über Mittel- bzw. Ultrakurzwelle, GPRS/UMTS oder Satellit.

Die Codemessung ist die klassische Lösungsmethode bei GPS-Messungen. Im Empfänger wird ein Referenzsignal generiert, welches mit dem empfangenen Signal verglichen und zur Deckung gebracht wird. Aus der daraus entstehenden Verschiebung wird eine Laufzeit berechnet. Das Produkt aus Laufzeit und Ausbreitungsgeschwindigkeit des Signals entspricht der Distanz zwischen Empfänger und Satellit. Im Gegensatz zur Codemessung wird bei der Phasenmessung nicht das Signal, sondern die Trägerwelle ausgewertet. Die Codemessung wird lediglich genutzt, um eine Näherungslösung zu errechnen. Aus dem Vergleich mit dem vom Empfängergerät generierten Signal kann ein Teil der Distanz innerhalb einer Wellenlänge mit einer Auflösung im Submillimeterbereich bestimmt werden.

Das als Differenzial-GPS (DGPS) bezeichnete Verfahren nutzt sowohl Code- als auch Phasenmessung und liefert Genauigkeiten im Bereich von einigen Millimetern bis wenigen Zentimetern. Hauptproblem ist die Bestimmung der Anzahl ganzer Wellenzyklen zwischen Sender und Empfänger und die daraus entstehende Phasenmehrdeutigkeit (engl. *ambiguity*). Für die Bestimmung der Mehrdeutigkeit stehen verschiedene – geometrische und analytische – Verfahren zur Verfügung.

Unter *Real Time Kinematic* (RTK) wird eine kinematische Echtzeitvermessung auf Basis von GNSS verstanden. Es handelt sich um ein differentielles Verfahren, das die Phase des Trägersignals auswertet. Eine Referenzstation ist mit einem Funksender verbunden, der die von Satelliten empfangenen Daten korrigiert und aussendet. Ein sogenannter *Rover*, ein tragbarer GPS-Empfänger, ist ebenfalls mit einer Funkverbindung ausgestattet und empfängt das von der Referenzstation ausgestrahlte Signal. Über seine eigene GPS-Antenne erhält er zusätzlich Daten direkt von den Satelliten. Die beiden Datensätze werden im Rover weiterverarbeitet, um die Phasenmehrdeutigkeiten aufzulösen und eine hochgenaue Position in Bezug auf den Referenzempfänger zu erhalten. Die Genauigkeit der Basislinienbestimmungen beträgt 0,5 bis 5 cm.

Der »Satellitenpositionierungsdienst der deutschen Landesvermessung« (SAPOS) ist der Korrekturdienst der Bundesländer und wurde 1995 als Infrastrukturmaßnahme und als Substitut der herkömmlichen Lagefestpunktfelder eingerichtet. Der Satellitenpositionierungsdienst deckt ganz Deutschland mit ca. 270 vernetzten Referenzstationen ab. Da nicht nur alle Bundesländer sondern auch Nachbarländer wie Österreich, Schweiz oder Frankreich in die Vernetzungen einbezogen sind, bestehen an den Ländergrenzen keine Spannungen. Der Dienst bietet die drei Bereiche »Echtzeit-Positionierungs-Service« (EPS), »Hochpräziser Echtzeit-Positionierungs-Service« (HEPS) und »Geodätischer Postprocessing-Positionierungs-Service« (GPPS) mit verschiedenen Genauigkeitsstufen an.

3. Allgemeinfall der Ausgleichungsrechnung

Um Aussagen über die Qualität von Beobachtungen und daraus abgeleiteter Größen treffen zu können, sind Überbestimmungen erforderlich. Dazu müssen mehr Messungen durchgeführt werden, als es zur eindeutigen Festlegung der Unbekannten erforderlich wäre. Aufgrund von unvermeidlichen zufälligen Abweichungen geraten die Bestimmungsgleichungen in Widerspruch zu einander. Die redundanten Beobachtungen lassen eine mehrfache Bestimmung der gesuchten Größen zu, jedoch mit unterschiedlichen Ergebnissen. Die Aufgabe der Ausgleichungsrechnung ist es, auf Basis der durchgeführten Messungen die mutmaßlich besten Werte für die Unbekannten zu liefern, indem minimale Verbesserungen nach der *Methode der kleinsten Quadrate* an die Beobachtungen angebracht werden. Demnach lautet die Forderung der Ausgleichung

$$\mathbf{v}^{\mathrm{T}}\mathbf{P}\mathbf{v} \Rightarrow min. \tag{3.1}$$

Der *Allgemeinfall der Ausgleichungsrechnung*, auch Gauss–Helmert-*Modell* genannt, beschreibt eine Situation, bei der zwischen den Beobachtungen $(L_1, L_2, \ldots, L_n)$ und den Unbekannten $(X_1, X_2, \ldots, X_u)$ nur ein einzelner funktionaler Zusammenhang formuliert wird. Im Gegensatz dazu hängt bei der *Ausgleichung nach vermittelnden Beobachtungen* (Gauss–Markov-*Modell*) eine Messgröße funktional von allen Unbekannten ab.[1] Der Ausgleichungsansatz ist daher

$$\mathbf{F}(\tilde{\mathbf{L}}, \tilde{\mathbf{X}}) = \mathbf{0}. \tag{3.2}$$

In der Geodäsie werden überwiegend trigonometrische und andere nichtlineare Funktionen verwendet, so dass der funktionale Zusammenhang im Normalfall ebenfalls nichtlinear ist. Für die Ausgleichung ist aber ein lineares Modell erforderlich, weshalb in Abschnitt 3.2 näher auf die Linearisierung eingegangen wird. Zuvor soll allerdings ein kurzer Überblick über die verschiedenen Fehlerarten gegeben werden.

3.1. Fehlerarten

Jede beobachtete Messgröße weist einen Fehler (Abweichung) zu ihrem Sollwert, der wahren Größe, auf. Die Ursache liegt in der Beeinflussung des Messergebnisses durch

[1] vgl. Gränicher, 1996; Niemeier, 2008

die Unvollkommenheit der Messgeräte und Messverfahren, durch Umwelteinflüsse und durch den Beobachter selbst. Mit keinem Messgerät lässt sich der genaue Sollwert bestimmen, auf jeden Messvorgang wirken äußere Einflüsse. Als Umwelteinflüsse müssen z. B. Temperatur und Luftdruck, magnetische oder elektrische Felder berücksichtigt werden. Der Anteil des Beobachters an der Abweichung hängt von seinen persönlichen Eigenschaften und Fähigkeiten ab.

In der Geodäsie wird grundsätzlich zwischen drei Arten von Fehlern unterschieden. Ein *grober Fehler* (Ausreißer) liegt vor, wenn ein Messwert aufgrund von Zielverwechslungen, defekten Messgeräten oder Mängeln im Messverfahren erheblich vom wahren Wert abweicht. Grobe Fehler werden nicht korrigiert, sondern entfernt. Unter *systematischen Fehlern* versteht man Messabweichungen, die sich auf eine funktional beschreibbare Ursache zurückführen lassen. Sie werden bestimmt und in der Auswertung berücksichtigt. Die als *zufällige Fehler* bezeichneten Restabweichungen können weder nach Betrag noch nach Richtung beschrieben werden. Ihr Einfluss wird durch Überbestimmungen gesenkt.[2]

3.2. Funktionales Modell

Der erste Schritt bei der Ausgleichungsrechnung ist das Festlegen der funktionalen Beziehungen zwischen den Beobachtungen und den Unbekannten. Die Beobachtungen $(L_1, L_2, \ldots, L_n)$ bilden den *Beobachtungsvektor* $\mathbf{L}$. Er stellt eine Näherung für die *wahren Werte* $\tilde{\mathbf{L}}$ dar, die im Allgemeinen unbekannt sind. Mit Hilfe der Methode der kleinsten Quadrate kann aber eine Schätzung, die *ausgeglichenen Beobachtungen* $\hat{\mathbf{L}}$, angegeben werden. Diese werden durch Addition des Beobachtungsvektors zum *Vektor der Verbesserungen* gebildet:

$$\hat{\mathbf{L}} = \mathbf{L} + \mathbf{v}. \tag{3.3}$$

Der *Parametervektor* $\mathbf{X}$ umfasst u Unbekannte $(X_1, X_2, \ldots, X_u)$. Er besteht, wie der Beobachtungsvektor, ebenfalls aus Zufallsgrößen und besitzt daher wahre Werte, für die der Vektor $\tilde{\mathbf{X}}$ verwendet wird. Die Schätzungen der Unbekannten lassen sich im *ausgeglichenen Parametervektor* $\hat{\mathbf{X}}$ zusammenfassen. Des Weiteren sind für die Ausgleichung die Näherungswerte $\mathbf{X}_0$ erforderlich. Wenn für $\mathbf{X}_0$ nur grobe Schätzwerte vorliegen, besteht die Möglichkeit der iterativen Verbesserung, indem das Ergebnis der Ausgleichung als Näherungslösung für weitere Ausgleichungen verwendet wird.

Der Widerspruch, den die Beobachtungen $\mathbf{L}$ und die genäherten Unbekannten $\mathbf{X}_0$ zwangsweise bei Anwendung von (3.2) hervorrufen, wird im *Widerspruchsvektor*

$$\mathbf{w} = \mathbf{F}(\mathbf{L}, \mathbf{X}_0) \tag{3.4}$$

[2] vgl. Jäger u. a., 2005

zusammengefasst. Um den Widerspruch in Gleichung (3.4) zu beseitigen, wird ein Vektor mit *genäherten Beobachtungen* $\mathbf{L}_0$ eingeführt, so dass

$$\mathbf{F}(\mathbf{L}_0, \mathbf{X}_0) = \mathbf{0} \tag{3.5}$$

gilt. Das funktionale Modell der Ausgleichung

$$\mathbf{F}(\hat{\mathbf{L}}, \hat{\mathbf{X}}) = \mathbf{0} \tag{3.6}$$

kann durch eine Taylorentwicklung in ein lineares funktionales Modell transformiert werden, für das die partiellen Ableitungen nach den Beobachtungen und den Unbekannten zu bilden sind:

$$\mathbf{F}(\hat{\mathbf{L}}, \hat{\mathbf{X}}) = \mathbf{F}(\mathbf{L}, \mathbf{X}_0) + \frac{\partial \mathbf{F}(\mathbf{L}, \mathbf{X}_0)}{\partial \mathbf{L}} (\hat{\mathbf{L}} - \mathbf{L}) + \frac{\partial \mathbf{F}(\mathbf{L}, \mathbf{X}_0)}{\partial \mathbf{X}_0} (\hat{\mathbf{X}} - \mathbf{X}_0) = \mathbf{0}. \tag{3.7}$$

Die partiellen Ableitungen lassen sich jeweils in einer Matrix zusammenfassen; für die Unbekannten $(X_1, X_2, \ldots, X_u)$ in der A-Matrix (*Designmatrix*)

$$\mathbf{A} = \begin{bmatrix} \frac{\partial F_1(\mathbf{L},\mathbf{X})}{\partial X_1} & \frac{\partial F_1(\mathbf{L},\mathbf{X})}{\partial X_2} & \cdots & \frac{\partial F_1(\mathbf{L},\mathbf{X})}{\partial X_u} \\ \frac{\partial F_2(\mathbf{L},\mathbf{X})}{\partial X_1} & \frac{\partial F_2(\mathbf{L},\mathbf{X})}{\partial X_2} & \cdots & \frac{\partial F_2(\mathbf{L},\mathbf{X})}{\partial X_u} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_r(\mathbf{L},\mathbf{X})}{\partial X_1} & \frac{\partial F_r(\mathbf{L},\mathbf{X})}{\partial X_2} & \cdots & \frac{\partial F_r(\mathbf{L},\mathbf{X})}{\partial X_u} \end{bmatrix} \tag{3.8}$$

und für die Beobachtungen $(L_1, L_2, \ldots, L_n)$ in der B-Matrix

$$\mathbf{B} = \begin{bmatrix} \frac{\partial F_1(\mathbf{L},\mathbf{X})}{\partial L_1} & \frac{\partial F_1(\mathbf{L},\mathbf{X})}{\partial L_2} & \cdots & \frac{\partial F_1(\mathbf{L},\mathbf{X})}{\partial L_n} \\ \frac{\partial F_2(\mathbf{L},\mathbf{X})}{\partial L_1} & \frac{\partial F_2(\mathbf{L},\mathbf{X})}{\partial L_2} & \cdots & \frac{\partial F_2(\mathbf{L},\mathbf{X})}{\partial L_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_r(\mathbf{L},\mathbf{X})}{\partial L_1} & \frac{\partial F_r(\mathbf{L},\mathbf{X})}{\partial L_2} & \cdots & \frac{\partial F_r(\mathbf{L},\mathbf{X})}{\partial L_n} \end{bmatrix}. \tag{3.9}$$

In einer übersichtlicheren Schreibweise lässt sich (3.7) mit den beiden Matrizen wie folgt darstellen:

$$\mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{v} + \mathbf{w} = \mathbf{0}. \tag{3.10}$$

3.3. Stochastisches Modell

Für den Beobachtungsvektor $\mathbf{L}$ kann eine *Kovarianzmatrix* aufgestellt werden, die die stochastischen Eigenschaften der einzelnen Beobachtungen und ihre Abhängigkeiten (*Korrelationen*) beschreibt:

$$\mathbf{\Sigma}_{LL} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22}^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2}^2 & \cdots & \sigma_{nn} \end{bmatrix}. \tag{3.11}$$

Auf der Hauptdiagonalen stehen die Varianzen; sie stellen die *a-priori*-Genauigkeiten der Beobachtungen dar. Für sie sind häufig nur relative Werte bekannt, wie z. B. die vom Hersteller angegebene Genauigkeit eines elektronischen Distanzmessers. Es wird daher eine *Varianz der Gewichtseinheit* σ_0^2 gewählt, die im Verhältnis zu alle Varianzen und Kovarianzen steht. Der Wert von σ_0^2 beeinflusst die weiteren Rechenschritte nicht, durch eine sinnvolle Wahl können Ungenauigkeiten bei der Auflösung des Gleichungssystems vermieden werden. Auf den Nebendiagonalen liegen noch die Kovarianzen, sofern die Beobachtungen statistisch von einander abhängig sind. Andernfalls hat $\mathbf{\Sigma}_{LL}$ die Form einer Diagonalmatrix.

Um die *Kofaktormatrix* zu erhalten, muss die Kovarianzmatrix mit der Varianz der Gewichtseinheit in das gleiche Verhältnis gesetzt werden:

$$\mathbf{Q}_{LL} = \frac{1}{\sigma_0^2}\mathbf{\Sigma}_{LL}. \tag{3.12}$$

Sie beschreibt die Genauigkeitsrelationen zwischen den Beobachtungen.[3] Die *Gewichtsmatrix*

$$\mathbf{P} = \mathbf{Q}_{LL}^{-1} \tag{3.13}$$

ist die Inverse der Kofaktormatrix und enthält die umgekehrt proportionale Gewichtung zu den Varianzen. Da das stochastische Modell nur theoretische Größen enthält, ist dessen Zuverlässigkeit noch zu prüfen. Dazu dient der statistische Test, der in Abschnitt 3.5 erläutert wird.

3.4. Ausgleichungsalgorithmus

Mit Aufstellen des funktionalen Modells lassen sich Matrix $\mathbf{A}$, Matrix $\mathbf{B}$ und Widerspruchsvektor $\mathbf{w}$ berechnen. Zur Lösung der Ausgleichungsaufgabe ist der Multiplikator $\mathbf{k}$, der sogennante *Korrelatenvektor*, nach der Methode von Lagrange zu bestimmen. Dazu wird die Forderung (3.1) mit der Bedingung (3.10) verknüpft:

$$\mathbf{F}(\mathbf{v}, \hat{\mathbf{x}}) = \mathbf{v}^{\mathrm{T}}\mathbf{P}\mathbf{v} - 2\mathbf{k}^{\mathrm{T}}(\mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{v} + \mathbf{w}). \tag{3.14}$$

[3] vgl. Niemeier, 2008

Von dieser Funktion sind anschließend die partiellen Ableitungen nach $\mathbf{v}$ und $\hat{\mathbf{x}}$ zu bilden. Mit ihnen kann ein Gleichungssystem, die *Normalgleichungen*, aufgestellt werden:

$$\begin{bmatrix} \mathbf{B}\,\mathbf{P}^{-1}\,\mathbf{B}^{\mathrm{T}} & \mathbf{A} \\ \mathbf{A}^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ \hat{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} -\mathbf{w} \\ \mathbf{0} \end{bmatrix}. \tag{3.15}$$

Das Gleichungssystem lässt sich durch die Inversion

$$\begin{bmatrix} \mathbf{k} \\ \hat{\mathbf{x}} \end{bmatrix} = - \begin{bmatrix} \mathbf{B}\,\mathbf{P}^{-1}\,\mathbf{B}^{\mathrm{T}} & \mathbf{A} \\ \mathbf{A}^{\mathrm{T}} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{w} \\ \mathbf{0} \end{bmatrix} \tag{3.16}$$

auflösen und damit $\mathbf{k}$, $\hat{\mathbf{x}}$ sowie $\mathbf{v}$ berechnen. Sind sowohl das funktionale als auch das stochastische Modell fehlerfrei und die Näherungswerte hinreichend genau, wird auch die Schlussprobe mit

$$\mathbf{F}(\hat{\mathbf{L}}, \hat{\mathbf{X}}) = \mathbf{0} \tag{3.17}$$

erfüllt.

3.5. Statistischer Test

Der statistische Test dient der Überprüfung des funktionalen und stochastischen Modells und der Abschätzung der Qualität des Ausgleichungsergebnisses. Dazu ist die theoretische *a-priori*-Varianz der Gewichtseinheit σ_0^2 auf eine signifikante Abweichung von der empirischen *a-posteriori*-Varianz der Gewichtseinheit s_0^2 zu testen. Im Folgenden sei die empirische Varianz o. B. d. A. größer als die theoretische Varianz:

$$s_0^2 > \sigma_0^2. \tag{3.18}$$

Für den statistischen Test werden zwei Hypothesen aufgestellt. Die Nullhypothese H_0 besagt, dass beide Varianzen bei einer Irrtumswahrscheinlichkeit von α den gleichen Erwartungwert besitzen. Bei der Alternativhypothese H_A wird davon ausgegangen, dass unter Berücksichtigung des Signifikanzniveaus der Erwartungswert von s_0^2 größer als der Erwartungswert von σ_0^2 ist:

$$H_0 \colon\ E\{s_0^2\} = \sigma_0^2, \tag{3.19}$$

$$H_A \colon\ E\{s_0^2\} > \sigma_0^2. \tag{3.20}$$

Als Testverteilung lassen sich sowohl die *Chi-Quadrat-Verteilung* (abgekürzt χ^2-Verteilung) als auch die Fisher-*Verteilung* (abgekürzt F-Verteilung) anwenden. Zwischen beiden besteht zusammen mit der Anzahl der Freiheitsgrade f die Beziehung

$$\chi_f^2 = f \cdot F_{f,\infty}. \tag{3.21}$$

Im Ausgleichungsprogramm wird für den Varianzentest die F-Verteilung verwendet.

Um eine Entscheidung bzgl. der Wahl der Hypothese treffen zu können, wird die Testgröße (Stichprobenfunktion) mit dem Quantil (Prüfgröße) der Verteilung verglichen.[4] Als Testgröße wird für die F-Verteilung

$$\hat{F}_{f,\infty} = \frac{s_0^2}{\sigma_0^2} \tag{3.22}$$

verwendet. Das Quantil

$$F_{f,\infty,1-\alpha} \tag{3.23}$$

ist entweder aus Tabellen zu entnehmen oder zu berechnen. Die Entscheidung über die Annahme der Null- oder der Alternativhypothese lässt sich durch einen Vergleich von Testgröße und Quantil treffen:

$$\hat{F} \leq F_{f,\infty,1-\alpha} \Rightarrow H_0 \text{ annehmen,} \tag{3.24}$$

$$\hat{F} > F_{f,\infty,1-\alpha} \Rightarrow H_A \text{ annehmen.} \tag{3.25}$$

Mit Hilfe der API-Erweiterung *JSci* für *Java* lassen sich die Quantile unter Angabe der Freiheitsgrade und der Irrtumswahrscheinlichkeit automatisch berechnen. Als Vorgriff auf Kapitel 4 sei an dieser Stelle erwähnt, dass die im Ausgleichungsprogramm für die Berechnung der Quantile genutzte Klasse *FDistribution*[5] keine unendlichen Freiheitsgrade verarbeiten kann. Zwar kennt *Java* (symbolische) Unendlichkeit, die Quantile werden aber numerisch berechnet. Als Folge handelt es sich bei den ermittelten Quantilen lediglich um Schätzwerte, für die ein Freiheitsgrad von $10^9 - 1$ angegeben wird. Dieser Unterschied besitzt jedoch für das Ergebnis des statistischen Tests praktisch keine Relevanz und ist eher von theoretischer Bedeutung.

[4] vgl. Foppe, 2009

[5] *Javadoc*: `http://jsci.sourceforge.net/api-0.94/JSci/maths/statistics/FDistribution.html`

4. Umsetzung

Das Kernstück dieser Bachelorarbeit ist die Entwicklung einer Ausgleichungssoftware zur Visualisierung von Bahnkruven und deren Zerlegung in geometrische Grundfiguren. In diesem Kapitel werden die Umsetzung und die dabei auftretende Probleme ausgeführt. Zunächst soll auf die funktionalen Zusammenhänge der verschiedenen Primitive und die Berechnung der für die Ausgleichung notwendigen Näherungswerte eingegangen werden. Darauf folgend wird die Wahl der Programmiersprache und der passenden Entwicklungsumgebung erläutert und abschließend der Aufbau und die Bedienung der Ausgleichungssoftware beschrieben.

4.1. Ausgleichung

In der Geodäsie werden Messreihen gewonnen, aus denen sich häufig funktionale Zusammenhänge herleiten lassen. Um Bahnkurven in geometrische Primitive zu zerlegen, müssen deren funktionalen Zusammenhänge bekannt sein. Sie bilden eine notwendige Bedingung zum Aufstellen des funktionales Modells.[1] Für den unter Abschnitt 3.2 beschriebenen Vektor $\mathbf{X}_0$ sind des Weiteren Näherungswerte erforderlich, weshalb nun folgend zu jedem Primitiv die jeweils in der Ausgleichungssoftware implementierte Lösung aufgezeigt werden soll.

4.1.1. Strecken

Die Strecke als geometrische Figur ist bei Trajektorien immer dann relevant, wenn Abschnitte ohne Krümmung vorhanden sind. Prinzipiell lässt sich die lagemäßige Differenz zweier Punkte immer als Strecke auffassen, in diesem Zusammenhang sind aber nur Strecken relevant, die mindestens aus drei beobachteten Punkten bestehen. Wegen ihres einfachen funktionalen Zusammenhangs und den nur aus zwei Werten bestehenden Näherungen lassen sich die Berechnungen im Ausgleichungsprogramm schnell integrieren.

[1] vgl. Niemeier, 2008

Funktionaler Zusammenhang von Strecken

Als Strecke wird in der Geometrie jede gerade Linie bezeichnet, die durch zwei Endpunkte begrenzt ist. Für den funktionalen Zusammenhang zwischen Beobachtungen und Strecken wird die Geradengleichung (4.1) angewandt.

$$f(x) = mx + n \tag{4.1}$$

Die Steigung m und der Schnitt der Y-Achse n lassen sich durch Anfangspunkt $A\,(x_A,\, y_A)$ und Endpunkt $E\,(x_E,\, y_E)$ der Strecke wie folgt bestimmen:

$$m = \frac{y_E - y_A}{x_E - x_A}\,, \tag{4.2}$$

$$n = y_A - (m \cdot x_A). \tag{4.3}$$

Der funktionale Zusammenhang

$$mx + n - y = 0 \tag{4.4}$$

ergibt sich damit aus dem Ausgleichungsansatz (3.2) und der Geradengleichung (4.1). Die partiellen Ableitungen nach den Unbekannten

$$\frac{\partial f}{\partial m} = x\,, \tag{4.5}$$

$$\frac{\partial f}{\partial n} = -1 \tag{4.6}$$

von Gleichung (4.4) führen schließlich zu der A-Matrix

$$\mathbf{A} = \begin{bmatrix} x_1 & -1 \\ x_2 & -1 \\ \vdots & \vdots \\ x_n & -1 \end{bmatrix}. \tag{4.7}$$

Näherungswerte von Strecken

Zur Berechnung der Näherungskoordinaten wird wieder die Geradengleichung (4.1) herangezogen. Mit Hilfe des Anfangspunktes A und des Endpunktes E lassen sich die Steigung m und der Schnitt der Y-Achse n berechnen. Sie stellen die genäherten Unbekannten dar und bilden zusammen den Vektor

$$\mathbf{X}_0 = \begin{bmatrix} m \\ n \end{bmatrix}. \tag{4.8}$$

Als Anfangs- und Endpunkt sind jene beiden Streckenpunkte zu wählen, die den größten relativen Abstand

$$s = \sqrt{(x_E - x_A)^2 + (y_E - y_A)2} \tag{4.9}$$

zu einander aufweisen.

4.1.2. Kreise

Als Teil von Bahnkurven sind Kreise vorwiegend nur in Form von Kreisbögen von Bedeutung, obgleich sich auch geschlossene Kreise ausgleichen lassen. Für die Berechnung sind immer mindestens drei beobachtete Punkte nötig, damit die erforderlichen Näherungswerte berechnen werden können.

Funktionaler Zusammenhang von Kreisen

Ein Kreis mit dem Mittelpunkt $(x_M,\, y_M)$ und dem Radius r kann in der Ebene durch die Koordinatengleichung

$$(x - x_M)^2 + (y - y_M)^2 = r^2 \tag{4.10}$$

beschrieben werden, wobei $(x,\, y)$ ein beliebiger Punkt auf dem Bogen des Kreises sei. Der funktionale Zusammenhang lässt sich durch Umstellung von Gleichung (4.10) als

$$\sqrt{(x - x_M)^2 + (y - y_M)^2} - r = 0 \tag{4.11}$$

herleiten. Die für das funktionale Modell erforderlichen partiellen Ableitungen von Gleichung (4.11) nach den Unbekannten liefern die A-Matrix

$$\mathbf{A} = \begin{bmatrix} -\frac{y_1 - y_M}{\sqrt{(x_1 - x_M)^2 + (y_1 - y_M)^2}} & -\frac{x_1 - x_M}{\sqrt{(x_1 - x_M)^2 + (y_1 - y_M)^2}} & -1 \\ -\frac{y_2 - y_M}{\sqrt{(x_2 - x_M)^2 + (y_2 - y_M)^2}} & -\frac{x_2 - x_M}{\sqrt{(x_2 - x_M)^2 + (y_2 - y_M)^2}} & -1 \\ \vdots & \vdots & \vdots \\ -\frac{y_n - y_M}{\sqrt{(x_n - x_M)^2 + (y_n - y_M)^2}} & -\frac{x_n - x_M}{\sqrt{(x_n - x_M)^2 + (y_n - y_M)^2}} & -1 \end{bmatrix}. \tag{4.12}$$

Näherungswerte von Kreisen

Die Gleichung (4.10) erlaubt eine eindeutige Beschreibung von Kreisen in der Ebene. Darüber hinaus kann ein Kreis durch drei Punkte $P_1\,(x_1,\, x_1)$, $P_2\,(x_2,\, x_2)$ und $P_3\,(x_3,\, x_3)$ des Kreisbogens festgelegt werden. Zu beachten ist dabei, dass die Genauigkeit der Lösung von der relativen Lage der Punkte abhängt. Vorteilhaft ist eine Verteilung der Punkte über den gesamten Kreisbogen (z. B. am Anfang, am Ende und in der Mitte). Die Beziehung zwischen dem Kreismittelpunkt und den drei Bogenpunkten drückt ein lineares Gleichungssystem aus, dass in Determinatenschreibweise wie folgt definiert ist:

$$\det \begin{bmatrix} x_M^2 + y_M^2 & x_M & y_M & 1 \\ x_1^2 + y_1^2 & x_1 & y_1 & 1 \\ x_2^2 + y_2^2 & x_2 & y_2 & 1 \\ x_3^2 + y_3^2 & x_3 & y_3 & 1 \end{bmatrix} = 0. \tag{4.13}$$

Die Koordinaten des Kreismittelpunktes lassen sich durch Lösen von (4.13) berechnen, wie die Gleichungen (4.14) und (4.15) zeigen.

$$x_M = \frac{\det \begin{bmatrix} x_1^2 + y_1^2 & y_1 & 1 \\ x_2^2 + y_2^2 & y_2 & 1 \\ x_3^2 + y_3^2 & y_3 & 1 \end{bmatrix}}{2 \cdot \det \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}} \tag{4.14}$$

$$y_M = \frac{\det \begin{bmatrix} x_1 & x_1^2 + y_1^2 & 1 \\ x_2 & x_2^2 + y_2^2 & 1 \\ x_3 & x_3^2 + y_3^2 & 1 \end{bmatrix}}{2 \cdot \det \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}} \tag{4.15}$$

$$r^2 = \frac{\det \begin{bmatrix} x_1 & y_1 & x_1^2 + y_1^2 \\ x_2 & y_2 & x_2^2 + y_2^2 \\ x_3 & y_3 & x_3^2 + y_3^2 \end{bmatrix}}{\det \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}} + x_M^2 + y_M^2. \tag{4.16}$$

Der Radius kann durch (4.16) oder durch Einsetzen von x_M und y_M in (4.10) ermittelt werden. Alternativ dazu ist eine geometrische Lösung möglich, da der Mittelpunkt des Umkreises eines Dreiecks der Schnittpunkt der Mittelsenkrechten ist. Zusammen ergeben x_M, y_M und r den Vektor der genäherten Unbekannten

$$\mathbf{X}_0 = \begin{bmatrix} x_M \\ y_M \\ r \end{bmatrix}. \tag{4.17}$$

Die Genauigkeit der Beobachtungen hat direkten Einfluss auf die Genauigkeit der genäherten Unbekannten. Für eine Genauigkeitssteigerung können Kreismittelpunkt und Radius auch iterativ bestimmt werden, sofern mehr als drei Bogenpunkte zur Verfügung stehen. Dazu werden Permutationen mit je drei Bogenpunkten gebildet. Für n Bogenpunkte entstehen

$$\frac{\binom{n}{3}}{3!} = \frac{\frac{n!}{(n-3)!}}{6} \tag{4.18}$$

Permutationen. Die Berechnung erfolgt analog zu (4.13), wobei die Summe der Teilergebnisse noch gemittelt wird. Anstatt des arithmetischen Mittels ist auch der Median anwendbar, um z. B. extrem abweichende Messwerte (Ausreißer) auszuschließen.

Aufgrund der großen Anzahl von Permutationen ist diese Form der Berechnung der Näherungswerte vergleichsweise aufwendig, zumal sie bereits eine Art Ausgleichung darstellt und bei hinreichend genauen Beobachtungen überflüssig ist.

4.1.3. Polynome

Viele Trajektorien lassen sich vollständig oder teilweise mittels Polynomen approximieren. Diese Tatsache macht sie für eine Ausgleichung besonders interessant. Darüber hinaus ist die Implementierung hinsichtlich ihres einfachen funktionalen Zusammenhangs und der iterativen Berechnung der partiellen Ableitungen vergleichsweise unproblematisch.

Funktionaler Zusammenhang von Polynomen

Eine Funktion $f : \mathbb{R} \to \mathbb{R}$ mit

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \tag{4.19}$$

für $a_i \in \mathbb{R}$, $n \in \mathbb{N}_0$, $i = 0, 1, \ldots, n$ und $a_n \neq 0$ wird als Polynom vom Grad n bezeichnet. Ein Polynom vom Grad 0 hat die Form $f(x) = a_0$. Die Umstellung von (4.19) führt zu dem funktionalen Zusammenhang

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \ - y = 0. \tag{4.20}$$

Für die A-Matrix eines Polynoms mit $f(x) = ax^2 + bx + c$ ergibt sich nach partieller Ableitung nach den Unbekannten die Form

$$\mathbf{A} = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{bmatrix}. \tag{4.21}$$

Aus (3.8) ist ersichtlich, dass die Anzahl der Elemente in den Zeilen der Matrix mit dem Grad des Polynoms zunimmt. Da sich die partiellen Ableitungen der A- und B-Matrix jeweils iterativ berechnen lassen, kann die Software sehr einfach erweitert werden, um beispielsweise auch Polynome siebten oder achten Grades auszugleichen. In Abschnitt 4.3 soll darauf näher eingegangen werden.

Näherungswerte von Polynomen

Gegenstand der Berechnung von Näherungswerten für Polynome sind lineare Transformationen der Form

$$\mathbf{A} \cdot \mathbf{X}_0 = \mathbf{b}, \tag{4.22}$$

wobei $\mathbf{A}$ eine quadratische Matrix und $\mathbf{b}$ ein bekannter Vektor ist. Aus den gegebenen Koordinaten der Beobachtungen und dem Grad des Polynoms kann der Vektor der Unbekannten $\mathbf{X}_0$ berechnet werden. Bei einem Polynom vom Grad n stellt das lineare Gleichungssystem eine Zusammenfassung von n Linearkombinationen dar. Die *Koeffizientenmatrix* $\mathbf{A}$ enthält die Variablen des Polynoms mit ihren Exponenten und der Vektor $\mathbf{b}$ die Funktionswerte. Zur Lösung des linearen Gleichungssystems wird Gleichung (4.22) nach $\mathbf{X}_0$ umgeformt:

$$\mathbf{X}_0 = \mathbf{A}^{-1} \cdot \mathbf{b}. \tag{4.23}$$

Der Vektor der genäherten Unbekannten enthält schließlich die Koeffizienten des approximierten Polynoms der Beobachtungen. Für eine möglichst gute Approximation sollten die zur Lösung des Gleichungssystems herangezogenen Beobachtungen gleichmäßig über das Polynom verteilt sein.

4.2. Programmiersprache und Entwicklungsumgebung

Für die softwaretechnische Umsetzung des in Kapitel 3 beschriebenen Ausgleichungsalgorithmus muss zunächst eine geeignete Programmiersprache gewählt werden. Prinzipiell eignen sich dafür alle TURING-vollständigen[2] Sprachen. Bei der Visualisierung von Trajektorien und deren Zerlegung in geometrische Primitive handelt es sich um eine rein mathematische Problemstellung. Daher ist eine Beschränkung auf eine für mathematische Berechnungen optimierte Programmiersprache, die auch Schnittstellen für die grafischen Ausgabe von Streudiagrammen bereitstellt, sinnvoll.

Zur Definition eigener Datenstrukturen und einer späteren Erweiterung der Ausgleichungssoftware eignet sich die Anwendung der *objektorientierten Programmierung* (OOP). Dabei handelt es sich um ein Paradigma zum Umgang mit der zunehmenden Komplexität von Softwaresystemen. Der Grundgedanke kann durch die drei Elemente *Vererbung*, *Kapselung* und *Polymorphie* beschrieben werden. Ein Programm lässt sich in Objekte zerlegen, deren Definition durch Klassenstrukturen erfolgt. Klassen können ihre Attribute und Methoden an andere Klassen vererben und ihre Implementierung nach außen hin verbergen, d. h. den direkten Zugriff auf ihre Struktur verwehren und diesen stattdessen über definierte Schnittstellen kapseln. Die Objekte reagieren des Weiteren kontextabhängig auf Nachrichten und verhalten sich damit polymorph.

[2] eine Programmiersprache heißt TURING-vollständig, wenn sich jedes berechenbare Problem mit ihr lösen lässt (siehe auch Socher, 2003, S. 116 f.)

Die objektorientierte Programmierung ist in der modernen Softwareentwicklung weit verbreitet und wird gemeinhin als Ablösung der *prozeduralen Programmierung*[3] angesehen.[4] Zwar kann die objektorientierte Programmierung eine sinnvolle Strukturierung nicht garantieren – die Implementation wird nicht allein deshalb einfacher und übersichtlicher, weil der Programmierstil objektorientiert ist –, sie lässt aber eine bessere Abschätzung der Komplexität zu, erleichtert die spätere Erweiterung des Programms und fördert die Wiederverwendbarkeit einzelner Programmteile. Am Beispiel des Ausgleichungsprogramms bedeutet das konkret, dass elementare Datenstrukturen, wie die des zweidimensionalen Punktes, von so essenzieller Bedeutung sind, dass sie einer besonderen Betrachtung bedürfen, um die Komplexität des Programms nachhaltig zu verringern.

4.2.1. Matlab

Vor allem im akademischen Bereich ist das proprietäre *Matlab* des amerikanischen Herstellers *The MathWorks* verbreitet. Die Softwarelösung ist, im Gegensatz zu Computeralgebrasystemen, primär für numerische Berechnungen ausgelegt und erlaubt neben Matrixoperationen auch das Plotten von Funktionen, die Entwicklung von Benutzerschnittstellen sowie die Kopplung mit Programmen, die in anderen Programmiersprachen wie *C/C++* oder *Fortran* geschrieben sind. Damit erfüllt *Matlab* die für die Entwicklung eines Ausgleichungsprogramms erforderlichen, formalen Anforderungen.

Unter der Prämisse, die Umsetzung des Ausgleichungsprogramms rein objektorientiert auszurichten, eignet sich *Matlab* jedoch nur noch bedingt. Obwohl OOP unterstützt wird, unterscheiden sich Syntax und Aufrufkonventionen signifikant von denen anderen Programmiersprachen. Die Entwicklung umfangreicher Programme wird dadurch vor allem am Anfang erschwert. Des Weiteren ist die Funktionalität in Bezug auf OOP beschränkt und wirkt im Vergleich umständlich und nicht mehr zeitgemäß. So wird beispielsweise zwischen *Value*-Klassen, die bei Aufruf von Methoden immer Objektkopien liefern, und *Handle*-Klassen, die Referenzen zurückgeben, unterschieden; ein Umstand, der anderen Hochsprachen, wie *Java* oder *C++*, fremd ist. Auch die Einbindung grafischer Benutzeroberflächen ist aufwendig, da diese Fähigkeit erst mit späteren Versionen hinzukam. Schließlich sind in *Matlab* geschriebene Programme nicht ohne eine installierte *Matlab*-Version lauffähig. Eine einfache Weitergabe entwickelter Software ist somit nicht möglich. Auch wenn sich der Ausgleichungsalgorithmus in *Matlab* schnell implementieren lässt, sollte eine Alternative gefunden werden, um das Programm im vorgegebenen Zeitrahmen fertigstellen zu können.

[3] Paradigma, bei dem Programme in Teilprobleme, sogenannte *Prozeduren*, zerlegt werden
[4] vgl. Claussen, 1998, S. 2

4.2.2. Java

Als Alternative zu *Matlab* wurde die von *Sun Microsystems* entwickelte objektorientierte Programmiersprache *Java* gewählt. In *Java* geschriebene Programme werden in Zwischencode, sogenannten *Bytecode*, übersetzt und von einer Laufzeitumgebung, der *Java Virtual Machine*, ausgeführt. Dadurch sind *Java*-Programme plattformunabhängig, d. h. auf allen Prozessorarchitekturen und Betriebssystemen lauffähig, für die Bytecode-Interpreter existieren. Eine Neuübersetzung des Quellcodes ist somit nicht nötig. Dieser Vorteil zeichnet *Java* auch gegenüber *Matlab* aus: zum Ausführen entwickelter Software ist keine proprietäre Umgebung erforderlich. Des Weiteren vereint *Java* viele Vorteile moderner Programmiersprachen, wie automatische Speicherbereinigung oder *Multithreading*[5]. Zugleich ist *Java* eine der am aktivsten unterstützten Plattformen und mittlerweile Industriestandard.

Für die Entwicklung eines Ausgleichungsprogramms wirkt sich aber nachteilig aus, dass der Umfang der Standard-Bibliothek von *Java* in Bezug auf mathematische Funktionen beschränkt ist. Sie bietet nicht die Möglichkeiten einer auf numerische Berechnungen hin optimierten Programmiersprache wie z. B. *Fortran*. Die für die Ausgleichung nötigen Matrizenoperationen können daher ohne Einbindung externer Bibliotheken nicht durchgeführt werden. Nachfolgend sollen kurz einige API[6]-Erweiterungen vorgestellt werden, die diesen Nachteil kompensieren. Alle Pakete vereint, dass sie grundlegende Matrizenoperationen wie Addition oder Multiplikation bereitstellen und gleichzeitig unter freien Lizenzen stehen, die eine kostenfreie Nutzung erlauben.

Das vom *National Institute of Standards and Technology* (NIST) des U.S.-Handelsministeriums und dem *Matlab*-Hersteller *The MathWorks* entwickelte *Jama*[7] soll laut offizieller Webseite in Zukunft das *Java*-Standardpaket für lineare Algebra werden. Neben der eigentlichen Matrizenrechnung können auch verschiedene Matrizenzerlegungen durchgeführt werden, die für diese Arbeit aber nicht weiter von Bedeutung sind.

Die weitaus umfangreichere Bibliothek *Colt*[8] wird von der Europäischen Organisation für Kernforschung (CERN) entwickelt und stellt neben Matrixoperationen und -zerlegungen u. a. auf Effizienz ausgelegte Datenstrukturen (Listen, mehrdimensionale und assoziative Felder) sowie zusätzliche mathematische Funktionen bereit.

Für das im Rahmen dieser Arbeit zu entwickelnde Ausgleichungsprogramm wird *Jama* für die Matrizenrechnung und *JSci*[9] für die Berechnung von F-Verteilungen, die für die spätere statistische Auswertung der Ausgleichungsergebnisse von Bedeutung sind, genutzt. Das *Jama*-Paket bietet sich aufgrund der einfachen Integration und dem klar umrissenen Umfang, der für das Programm ausreichend ist, an.

[5] gleichzeitige Ausführung mehrerer leichtgewichtiger Prozesse (*Threads*)
[6] Programmierschnittstelle (engl. *application programming interface*)
[7] `http://math.nist.gov/javanumerics/jama/`
[8] `http://acs.lbl.gov/software/colt/`
[9] `http://jsci.sourceforge.net`

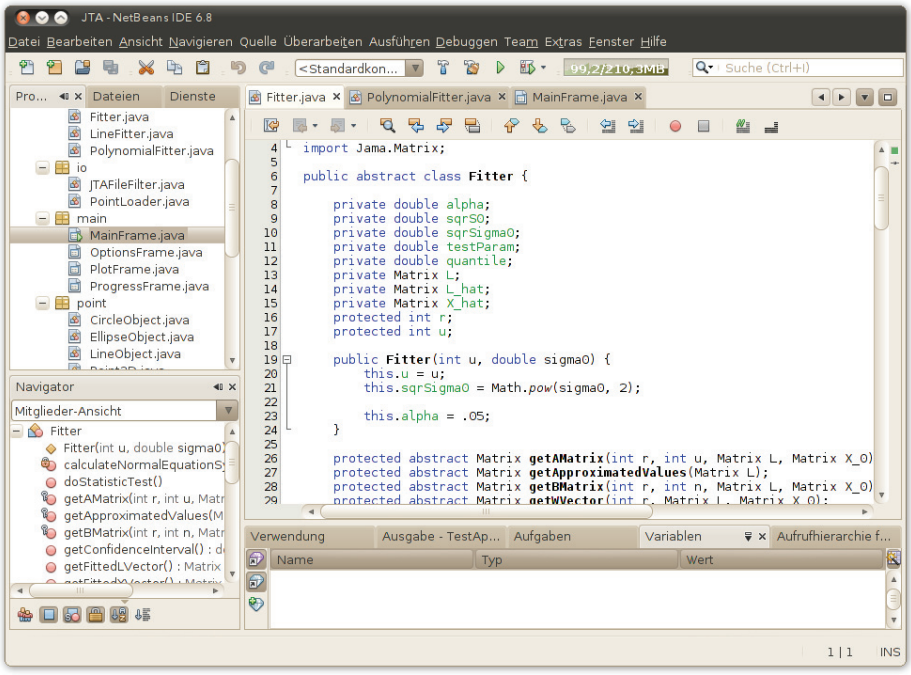

Abbildung 4.1.: Integrierte Entwicklungsumgebung *NetBeans IDE*

4.2.3. Entwicklungsumgebung

Ein weiterer Vorteil von *Java* ist die freie Verfügbarkeit von integrierten Entwicklungsumgebungen (IDEs). Mit ihnen kann die Entwicklung von Software erheblich vereinfacht werden. Je nach Umfang vereinen sie Komponenten wie Texteditor, Compiler, Linker und Debugger. Insbesondere die Texteditoren sind um hilfreiche Werkzeuge erweitert. Der Programmierer kann beispielsweise direkt auf Software-Dokumentationen (*Javadoc*) zugreifen oder den Quelltext vervollständigen und ganze Funktionen automatisch erzeugen lassen. Für umfangreiche Projekte werden Versionsverwaltung, Projektmanagement und UML[10]-Modellierung unterstützt.

Für *Java* sind vor allem *Eclipse* und die *NetBeans IDE* (Abb. 4.1) verbreitet. Der Funktionsumfang der beiden Open-Source-Projekte lässt sich über *Plug-ins* zusätzlich erweitern. Die Wahl der Entwicklungsumgebung hängt stark von persönlichen Präferenzen ab. Beide Systeme haben ihre Vor- und Nachteile, unterscheiden sich bei den Grundfunktionen aber kaum. Die *NetBeans IDE* zeichnet sich durch eine wohl durchdachte und übersichtliche Oberfläche und den integrierten *GUI*[11]*-Builder* für *Swing*[12] aus. Der *GUI-Builder* minimiert den Arbeitsaufwand beim Erstellen von

[10] *Unified Modeling Language*: Sprache zur Spezifikation, Visualisierung und Konstruktion von Softwaresystemmodellen
[11] *Graphical User Interface*: grafische Benutzerschnittstelle
[12] Komponentenbibliothek zum Erstellen grafischer Oberflächen unter *Java*

grafischen Benutzerschnittstellen. Ohne ein solches Werkzeug sind Aussehen und Eigenschaften von visuellen Komponenten per Hand im Quelltext festzulegen. Die Wahl der Entwicklungsumgebung viel vor allem aus diesem Grund auf die *NetBeans IDE*.

4.3. Aufbau und Bedienung

Die *Java Trajectory Analyser* genannte Ausgleichungssoftware wurde vollständig objektorientiert in *Java* unter Nutzung der *NetBeans IDE* geschrieben. Durch diese Entscheidung ließen sich viele anfallende Aufgaben im Voraus durch den Programmierstil, die Bibliotheken der Programmiersprache und den Möglichkeiten der Entwicklungsumgebung erledigen. Klassen und Pakete kapseln Teilprobleme, *Swing* stellt grafische Komponenten wie Menüs, Listen oder Tabellen zur Verfügung und mit dem *GUI-Builder* der *NetBeans IDE* können diese einfach zu Oberflächen verbunden werden.

Vor der eigentlichen Umsetzung müssen jedoch Überlegungen angestellt werden, welche Merkmale die Software aufweisen soll. Schließlich besteht sie nicht nur aus der reinen Umsetzung des Ausgleichungsalgorithmus. Konkret sind folgende Teilprobleme zu lösen:

1. Eine Punktdatei laden.
2. Die geladenen Punkte in einer Tabelle auflisten.
3. Einzelne Punkte zu geometrischen Figuren (Punktobjekte) zusammenfassen.
4. Punktobjekte mittels Gauss–Helmert-Modells ausgleichen.
5. Die Ausgleichungsergebnisse durch einen Varianzentest (F-Verteilung) prüfen.
6. Punkte durch einen Punktplotter anzeigen lassen.
7. Die Verbesserungen ausgeglichener Punkte im Plotter darstellen.

Die Software soll mindestens die in Abschnitt 4.1 beschriebenen Primitive Strecke, Kreis und Polynom ausgleichen können. Eine automatische Zerlegung von Bahnkurven in diese Primitive ist ebenfalls wünschenswert, auch wenn der Fokus zuerst auf der manuellen Auswahl von Punkten liegt.

4.3.1. Laden und Verwalten von Punkten

Für die erste Funktion – das Laden von Punkten – wurde eine eigene Klasse namens *PointLoader* geschrieben. Die *Swing*-Komponente *JFileChooser* stellt einen Dialog zum Öffnen von Dateien bereit. Der Dialog reicht die ausgewählte Datei an ein *PointLoader*-Objekt weiter, das sie verarbeitet. Die Punkte sind in einer einfachen Punktdatei gespeichert. Dabei handelt es sich um Textdateien, in denen zeilenweise Punktnummer, Punktcode, X- und Y-Koordinate sowie die Höhe aufgelistet sind

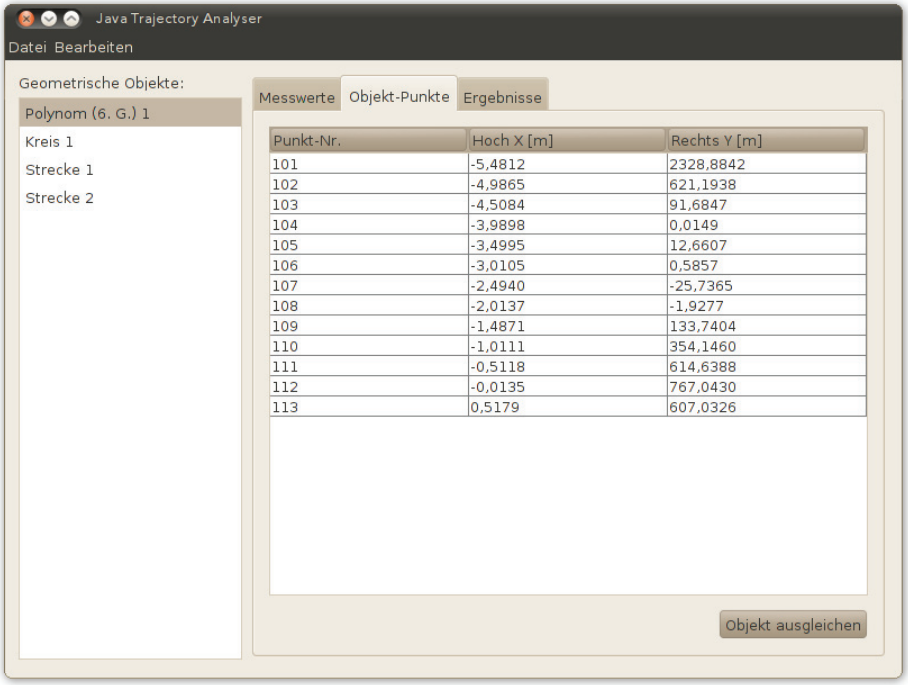

Abbildung 4.2.: Ausgleichungsprogramm *Java Trajectory Analyser* mit geladenen Messwerten

(Auflistung 5.2). Mittels eines *regulären Ausdrucks*[13] werden die Zeilen zerlegt und in *Point2D*-Objekte umgewandelt, die sich wiederum zu einem *PointObject* zusammenfassen lassen (Abb. 4.3).

Die *Point2D*-Klasse dient der Kapselung von Punktnummer und Koordinaten, während die *PointObject*-Klasse eine Menge von *Point2D*-Objekten verwaltet. Wie das UML-Klassendiagramm in Abbildung 4.3 weiter zeigt, sind die Klassen *CircleObject*, *LineObject* und *PolynomialObject* von *PointObject* abgeleitet. In ihnen sind die Namen der geometrischen Primitive und die für die Ausgleichung nötige Mindestanzahl an Punkten festgelegt. Für Polynome zweiten bis sechsten Grades sind von *PolynomialObject* weitere Klassen abgeleitet, die zusätzlich den jeweiligen Grad speichern.

Die *PointObjectManager*-Klasse hat die Aufgabe, Objekte von Typ *PointObject* zu verwalten. Durch diese Aufteilung lässt sich eine saubere Strukturierung vom einfachsten Datentyp, dem Punkt, zum umfangreichsten, der geometrischen Figur, erreichen. Der anfängliche Mehraufwand beim Entwurf der Klassen lohnt sich, da somit Datenstrukturen definiert werden, die programmweit nutzbar sind. Bei einer späteren

[13] Reguläre Ausdrücke (*RegExp*) sind Muster, mit denen sich Zeichenketten beschreiben lassen. Für die Zeile einer Punktdatei lautet der Ausdruck beispielsweise »`^\s* (\w+) \s+ (\d+) \s+ ([\-\d \.]+) \s+ ([\- \d \.]+) \s* ([\- \d \.]*)`«.

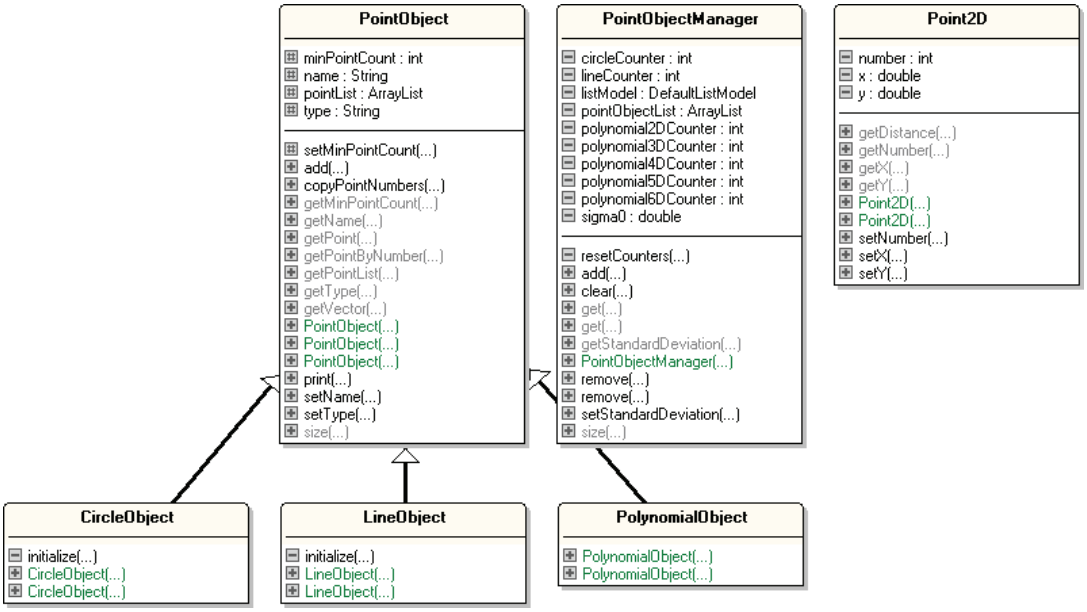

Abbildung 4.3.: UML-Klassendiagramm für Klassen aus dem Paket *point* (ohne von *PolynomialObject* abgeleitete Klassen für Polynome 2. bis 6. Grades)

Erweiterung der Ausgleichungssoftware sind nur noch geringe Änderungen durchzuführen.

4.3.2. Punkte plotten

In der Menüleiste kann der Benutzer durch Wahl von »Bearbeiten« und »Messwerte plotten ...« der Punktplotter aufrufen. Er gibt eine Übersicht über die Bahnkurve und zeigt zu jedem Punkt die entsprechende Punktnummer. Für den Fall, dass die Verbesserungen sehr klein ausfallen und somit im Plot nicht sichtbar sind, lässt sich der Maßstabsfaktor über  und  vergrößern bzw. verkleinern.

Das Paket *JSci* bietet mit der Klasse *JScatterGraph*[14] eine Möglichkeit zum Anzeigen von Streudiagrammen. Diese grafische Komponente ist aber nur sehr einfach aufgebaut. Punkte werden aufgrund von starken Rundungen nur sehr ungenau in das Koordinatensystem eingezeichnet, so dass geometrische Formen schwer zu erkennen sind. Punktnummern werden ebenfalls nicht angezeigt, eine Zuordnung einzelner Punkte ist somit nicht möglich. Schließlich lassen sich einzelne Punkte oder Punktgruppe nicht farblich markieren und eine Methode zur Darstellung der Verbesserungen ist ebenfalls nicht vorhanden. Eine andere, frei verfügbare Komponente, die diese Anforderungen erfüllt, existiert nicht. Daher war es nötig, einen neuen Punktplotter zu

[14] *Javadoc*: `http://jsci.sourceforge.net/api-0.94/JSci/swing/JScatterGraph.html`

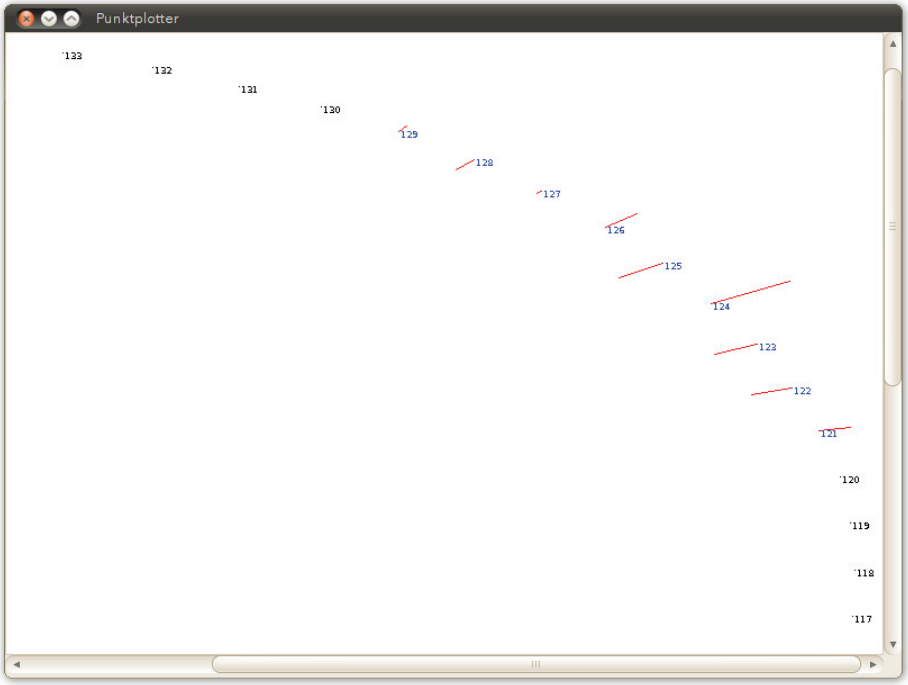

Abbildung 4.4.: Teil eines Kreisbogens (blau) mit 50-fach vergrößerten Verbesserungen (rot) im Punktplotter

programmieren, der diese Fähigkeiten besitzt. Als Grundlage wurde eine einfache *JPanel*-Komponente der *Swing*-Schnittstelle um die nötigen Zeichenfunktionen erweitert und in die Ausgleichungssoftware integriert.

4.3.3. Primitive ausgleichen

Die Tabelle »Messwerte« listet alle geladenen Punkte auf. Für eine Ausgleichung sind diese Punkte noch zu geometrischen Primitiven zusammenzufassen. Mit der Maus kann der Anwender einzelne Punkte auswählen. Durch gleichzeitiges Drücken von Strg lassen sich mehrere Zeilen, mit ⇧ ganze Zeilenbereiche markieren. Das Kontextmenü der Tabelle (rechte Maustaste) führt alle wählbaren Primitive auf. Die Liste »geometrische Objekte« enthält die erzeugten Primitive mit einer nachgestellten Kennnummer. Zum Löschen eines markierten Primitivs ist lediglich Entf zu drücken.

Wählt der Anwender ein Element aus der Liste »geometrische Objekte« aus, werden die zu diesem Primitiv zählenden Punkte unter dem Reiter »Objekt-Punkte« angezeigt. Die Ausgleichung wird über die Schaltfläche »Objekt ausgleichen« gestartet. Das Programm übergibt dann das Primitiv an die jeweilige Ausgleichungsklasse und

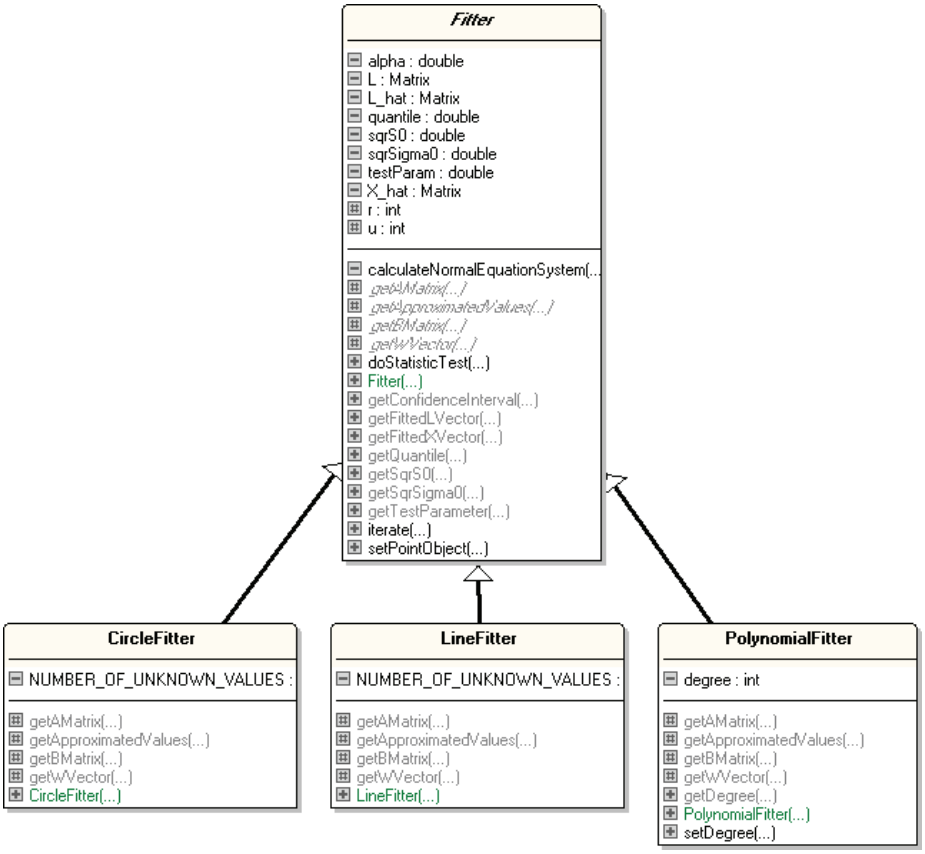

Abbildung 4.5.: UML-Klassendiagramm der Ausgleichungsklassen

zeigt die ausgeglichenen Beobachtungen und das Ergebnis des statistischen Test unter »Ergebnisse« an.

Bei der Ausgleichungsklasse *Fitter* (Abb. 4.5) handelt es sich um eine *abstrakte Klasse*, d. h. von ihr lässt sich kein Objekt instanziieren. In ihr ist lediglich der reine Ausgleichungsalgorithmus, der für alle Primitive gleich ist, implementiert. Die Berechnung der Näherungswerte, des Widerspruchsvektors, der A- und der B-Matrix erfolgt getrennt in den abgeleiteten Klassen *CircleFitter*, *LineFitter* und *PolynomialFitter*. Eine Besonderheit besteht in Bezug auf *PolynomialFitter*. Die Klasse kann Polynome beliebigen Grades ausgleichen, da alle Berechnungen iterativ durchgeführt werden. Der Grad wird im Vorfeld über die Funktion *setDegree()* festgelegt.

Für den statistischen Test wird, wie unter Abschnitt 3.5 bereits erläutert, die Klasse *FDistribution* aus dem Paket *JSci* verwendet. Nach Angabe der beiden Freiheitsgrade und der Sicherheitswahrscheinlichkeit gibt sie das zugehörige Quantil der Verteilungsfunktion zurück.

Die Sicherheitswahrscheinlichkeit ist intern auf 0,95 % festgelegt. Eine Funktion, die dem Benutzer ermöglicht den Wert zu ändern, ist noch nicht implementiert. Auch ein Speichern der Ergebnisse ist gegenwärtig nicht möglich. Aus praktischen Erwägungen heraus ist diese Funktion aber zu ergänzen. In Abschnitt 6.2 soll noch auf weitere, fehlende Funktionen eingegangen werden.

5. Exemplarischer Einsatz

Ein weiterer Teil dieser Arbeit war neben der Entwicklung der Ausgleichungssoftware auch das Sammeln von Messdaten. Damit sollen die für Ausgleichung nötigen Daten gesammelt und die Software einem praxisnahem Test unterzogen werden. Auch mögliche Fehler im Quelltext lassen sich dadurch leichter aufdecken. Für die Messungen mittels GPS und Tachymeter wurden ein Traktor und eine Modelleisenbahn als Versuchsobjekte genutzt. Künstlich erzeugte Messwerte sollen ein direkten Vergleich zwischen unterschiedlichen Messverfahren ermöglichen In diesem Kapitel sollen die drei Messverfahren vorgestellt und erläutert werden. Die Ergebnisse werden in Kapitel 6 diskutiert.

5.1. Modelleisenbahn

Das erste Versuchsobjekt ist eine digitale Modelleisenbahn der Firma *Märklin* in Nenngröße I (Maßstab 1:32) mit einer Modellspurweite von 45 mm (Abb. 5.1). Durch unterschiedlich lange und gekrümmte Gleisstücke sind verschiedener Teststrecken realisierbar. Die Infrarotfernbedienung erlaubt die Bewegung des Modells in verschiedenen Geschwindigkeitsstufen.

Um Bahnkurven für eine Ausgleichung zu erhalten, wird die Position der Eisenbahn kontinuierlich durch ein Tachymeter erfasst. Das Instrument, eine *S6*-Totalstation von *Trimble* (Abb. 2.1), besitzt eine Richtungsmessgenauigkeit von 0,3 mgon und eine Streckenmessgenauigkeit von 2 mm + 2 ppm bei Standardmessungen bzw. 0,6 mgon und 4 mm + 2 ppm im *Tracking*-Modus. Für diesen Versuch wird ausschließlich der *Tracking*-Modus verwendet, in dem das Tachymeter ein auf dem Anhänger der Bahn montierten Reflektor automatisch verfolgt und dessen Koordinaten fortlaufend abspeichert. Bei dem Reflektor handelt es sich um ein sogenanntes 360°-Prisma, welches den Messstrahl aus jeder beliebigen Richtung zurück zum Tachymeter reflektiert, ohne dass es zum Instrument gedreht werden muss.

Es standen ein Prisma des Tachymeterherstellers und ein *GRZ4*-Prisma der Firma *Leica* zur Verfügung (Abb. 5.2). Das Trimble-Prisma besteht aus sechs in das Gehäuse eingelassene Rundprismen, während bei der Leica-Version sechs Prismenblöcke zu einem zusammengefügt sind. Diese bilden ein Antiprisma, bei dem Ober- und Unterseite parallel und verdreht zueinander liegen. Der Vorteil des *GRZ4* ist, dass der Versatz zwischen den einzelnen Prismen geringer ist, als beim Trimble-Modell und

Abbildung 5.1.: Märklin-Modelleisenbahn mit *GRZ4*-Prisma auf dem Anhänger

Abbildung 5.2.: 360°-Prisma von Trimble (links) und *GRZ4*-Prisma von Leica (rechts)

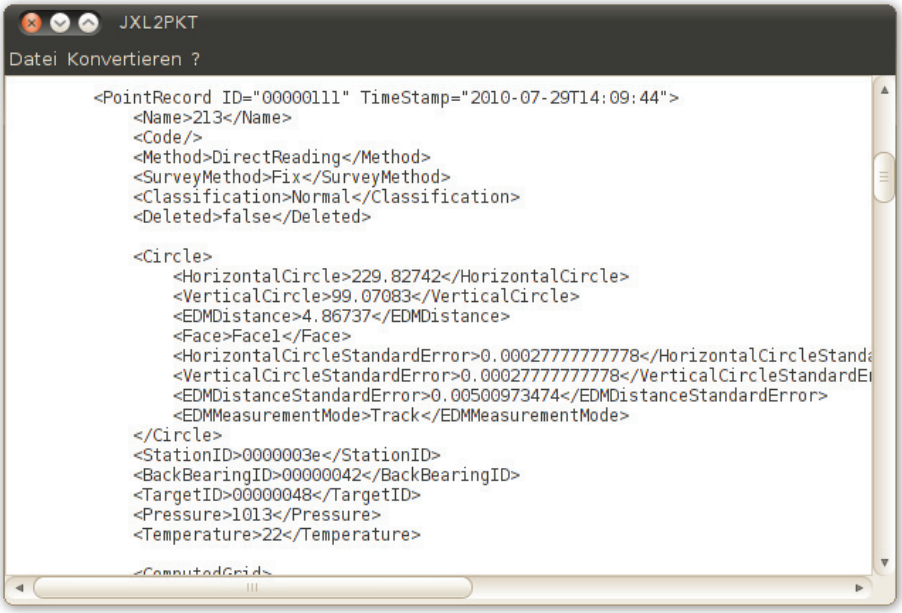

Abbildung 5.3.: Konvertierungsprogramm *JXL2PKT* mit geladenen Messwerten

damit die Messgenauigkeit steigt. In Abschnitt 6.1 soll darauf eingegangen werden, ob sich dieser Unterschied aus den Ausgleichungsergebnissen ableiten lässt.

Im *Tracking*-Modus des Tachymeters kann aus sechs Messverfahren gewählt werden. In der Einstellung *Festzeit* erfolgt das Abspeichern der Prismenkoordinaten jeweils nach Ablauf einer angegebenen Zeitspanne, während bei *Feststrecke* analog dazu das Messobjekt einen festgelegten Abstand zu der zuletzt gespeicherten Position aufweisen muss. Diese Einstellungen können mit der Wahl von *Zeit und Strecke* sowie *Zeit oder Strecke* auch kombiniert werden. Die letzte Option ist *Stop and go*, bei der nur gespeichert wird, wenn sich das Messobjekt bewegt. Für diesen Versuch wurden Messungen mit den ersten beiden Einstellungen vorgenommen.

Die Messdaten des Tachymeters lassen sich in verschiedene Formate exportieren, u. a. in ein herstellereigenes XML[1]-Format (Trimble JobXML). Diese enthalten nicht nur die aufgemessenen Punkte, sondern auch Metadaten über die Datei, Reduktionen und Informationen zum verwendeten Koordinatensystem (Auflistung 5.1).

Um die aufgemessenen Punkte in das Ausgleichungsprogramm einlesen zu können, wurde ein Hilfsprogramm in *Java* geschrieben (Abb. 5.3). Mit diesem können JobXML-Dateien in ein einfacheres ASCII[2]-Format umgewandelt werden. Nach der Konvertie-

[1] *Extensible Markup Language*: medienübergreifende Auszeichnungssprache zur strukturierten Beschreibung von Informationen
[2] *American Standard Code for Information Interchange*: einfache Zeichenkodierung für Textdateien

rung stehen Nummer, Code, X- und Y-Koordinate sowie die Höhe jedes Punktes zeilenweise in einer Textdatei (Auflistung 5.2).

```
1 <PointRecord ID="00000439" TimeStamp="2010-03-12
      T16:35:47">
2      <Name>359</Name>
3      <Code/>
4      <Method>DirectReading</Method>
5      <SurveyMethod>Fix</SurveyMethod>
6      <Classification>Normal</Classification>
7      <Deleted>false</Deleted>
8
9      <Circle>
10         <HorizontalCircle>89.73801</HorizontalCircle>
11         <VerticalCircle>96.4179</VerticalCircle>
12         <EDMDistance>5.31861</EDMDistance>
13         <Face>Face1</Face>
14         <HorizontalCircleStandardError>0.00027777777778
               </HorizontalCircleStandardError>
15         <VerticalCircleStandardError>0.00027777777778</
               VerticalCircleStandardError>
16         <EDMDistanceStandardError>0.00501063722</
               EDMDistanceStandardError>
17         <EDMMeasurementMode>Track</EDMMeasurementMode>
18     </Circle>
19     <StationID>0000001a</StationID>
20     <BackBearingID>0000001e</BackBearingID>
21     <TargetID>00000024</TargetID>
22     <Pressure>1010</Pressure>
23     <Temperature>20</Temperature>
24
25     <ComputedGrid>
26         <North>0.02416733906901</North>
27         <East>5.2852279861871</East>
28         <Elevation>10.805488467565</Elevation>
29     </ComputedGrid>
30 </PointRecord>
```

Auflistung 5.1: Einzelner Punkt einer JobXML-Datei

```
1     359     0000     0.0241     5.2852    10.8054
```

Auflistung 5.2: Auszug aus Textdatei mit Punktnummer, Punktcode, X-Koordinate, Y-Koordinate und Höhe (v. l. n. r.)

Abbildung 5.4.: John-Deere-Traktor mit *StarFire-iTC*-Empfänger (Dach) und Topcon *HiPer Pro* (rechter Außenspiegel)

5.2. Traktor

Ein weiterer Messversuch wurde mit einem vom Fachbereich Agrarwirtschaft der Hochschule Neubrandenburg bereitgestellen *John-Deere*-Traktor durchgeführt (Abb. 5.4). Dieser ist mit einem DGPS-Empfänger für *Precision Farming* ausgestattet. Das übersetzt als *Präzisionsackerbau* bezeichnete Verfahren dient der Steuerung von landwirtschaftlichen Nutzfahrzeugen sowie der Erfassung von Pflanzen- und Bodenkennwerten zur effizienteren Bearbeitung von Agrarflächen. Das System erlaubt beispielsweise das Führen des Traktors mittels automatischem Lenksystem, die Einhaltung vorgegebener Fahrspuren (Anschlussfahren) oder die automatische Dokumentation der durchgeführten Arbeitsgänge. Vordefinierte Bahnen können durch die Absteckung von Flurstücksgrenzen und anschließender Eingabe der Koordinaten in das System oder dem manuellen Abfahren des Gebietes festgelegt werden. Der eingebaute *StarFire-iTC*-Empfänger von John Deere berücksichtigt zudem Bodenunebenheiten und Seitenneigungen des Fahrzeugs und korrigiert diese automatisch.[3]

[3] vgl. John Deere, 2010

Der Traktor sollte eine Grünfläche vor dem Laborgebäude 2 der Hochschule GPS-gesteuert abfahren und die Positionsdaten im regelmäßigen Abstand abspeichern. Die für das DGPS-Verfahren nötige Basisstation wurde auf dem Dach des Laborgebäudes aufgestellt. Da sich die Rohdaten der Messung nicht aus dem *StarFire*-System auslesen lassen, wurde ein zusätzlicher GPS-Empfänger von *Topcon* an einem der Außenspiegel angebracht. Die Messung erfolgt ebenfalls differentiell, anstatt der Basisstation des *StarFire*-Systems wurde aber der SAPOS-Dienst mittels *Real Time Kinematic* genutzt. Die Empfangseinheit des Topcon *HiPer Pro* ließ sich nur mit Schraubzwingen am Traktor befestigen. Dadurch war keine exakte Horizontierung der Antenne möglich und die durch den Motor verursachten Vibrationen des Außenspiegels hatten direkten Einfluss auf das Messergebnis. Des Weiteren befand sich die Antenne unterhalb des Fahrzeugdachs, wodurch sich Abschattungen nicht ausschließen lassen. In Abschnitt 6.1 soll daher untersucht werden, ob die durch den Messaufbau verursachten Messfehler eine Zerlegung der Bahnkurven in geometrische Primitive womöglich erschweren.

Zu Beginn steuerte der Fahrzeugführer den Traktor manuell, während das Steuerungssystem die Positionsdaten alle 2 m abspeicherte. Anschließend fuhr der Traktor die aufgezeichnete Strecke GPS-geleitet, mit einer Spurweite von 4 m. Die Grünfläche erwies sich für die drei parallel verlaufenden Bahnen als zu klein. Die nötigen Wendemanöver konnten nur durch das Eingreifen des Fahrzeugführers gefahren werden; uneinheitliche Kurven waren die Folge.

5.3. Simulierte Messreihen

Eine weitere Möglichkeit zum Testen des Ausgleichungsprogramms und zur Veranschaulichung der Ergebnisse verschiedener Messmethoden ist die Simulation von Messreihen. Anstatt Messungen mit Tachymetern oder GPS-Empfängern durchzuführen, werden Beobachtungen künstlich erzeugt. Dazu bieten sich beispielsweise Tabellenkalkulationsprogramme wie das freie *OpenOffice.org Calc*[4] an. Es sind lediglich die in Abschnitt 4.1 beschriebenen funktionalen Zusammenhänge der einzelnen geometrischen Primitive anzuwenden und daraus Koordinaten zu berechnen.

Den so gewonnenen »Beobachtungen« haften, abgesehen von den Rundungsfehlern der Maschinenzahlen, keine Abweichungen an. Diese müssen noch mittels Pseudozufallszahlengenerator erzeugt und anschließend angebracht werden. Das Programm *Calc* bietet dazu die Funktion *Zufallszahl()*, die pseudozufällige Zahlen im Bereich zwischen null und eins liefert. Dadurch ist es möglich, beliebig große Abweichungen zu produzieren, die als Vergleich für die Genauigkeit konventioneller Messmethoden dienen. Soll beispielsweise eine tachymetrische Aufnahme von Bahnkurven simuliert werden, sollte die Abweichung streckenabhängig mindestens 2 mm betragen. Bei einer DGPS-Messung unter Nutzung des SAPOS-Dienstes sind zwischen 0,5 und 3 cm anzubringen;

[4] `http://de.openoffice.org`

Messverfahren	**Abweichung** [m]
Tachymetrie	$\pm 0,005$
DGPS (SAPOS)	$\pm 0,030$
GPS	$\pm 5,000$

Tabelle 5.1.: Simulierte Messverfahren mit den jeweils angebrachten Abweichungen

bei einer reinen GPS-Codemessung ohne Korrekturdaten, wie sie bei handelsüblichen GPS-Navigationsgeräten durchgeführt wird, hingegen mindestens 5 m.

Prinzipiell können so alle Formen von Trajektorien erzeugt werden, die das Ausgleichungsprogramm verarbeiten kann. Für einen Vergleich verschiedener Messgeräte wurden aber lediglich drei Kreisbögen mit variablen Abweichungen zwischen 5 mm und 10 m zur Simulation von tachymetrischen, DGPS- und Handheld-GPS-Messungen erzeugt. Dadurch lässt sich erst eine vergleichbare Aussage zu den Ausgleichungsergebnissen treffen, weil die fehlerfreien Ursprungskoordinaten identisch sind.

Die Besonderheit dieser Methode ist, dass die wahren Beobachtungen, die durch den Vektor $\tilde{\mathbf{L}}$ repräsentiert werden, bekannt sind. Es wird ein Kreisbogen mit dem Mittelpunkt M (100, 100) und einem Radius von 25 m definiert. Über Polarpunktberechnung werden 41 Bogenpunkte zwischen 0 und 200 gon erzeugt. Die Richtungswinkel betragen jeweils 5 gon.

Es sollen drei verschiedene Messverfahren mit ihren typischen Genauigkeiten simuliert werden (Tabelle 5.1). Die Abweichungen werden in X- und Y-Richtung sowohl positiv als auch negativ angebracht. Die Ergebnisse dieses Versuchs sind in Abschnitt 6.1 dargestellt.

6. Diskussion und Ausblick

In diesem sollen die Ergebnisse der durchgeführten Messungen betrachtet und mögliche Erweiterungen der Ausgleichungssoftware diskutiert werden. Darüber hinaus enthält es einige kritische Betrachtungen und eine abschließende Zusammenfassung.

6.1. Ergebnisse

Nachfolgend die Auswertung der Versuche aus Kapitel 4. Die erzeugten Plots der Messreihen sind im Anhang C zu finden.

6.1.1. Modelleisenbahn

Der Versuch wurde mit zwei Prismenarten und den Messeinstellungen »Festzeit« und »Feststrecke« durchgeführt. Für die Einstellung »Feststrecke« mit 10 cm ergeben sich in den Kurven größere Sprünge sowohl für das Trimble- als auch für das Leica-Prisma (Abb. C.2 und C.3). In Abschnitt 2.1 wurde darauf eingegangen, dass die Richtungsmessung mit Tachymetern schneller ist als die Streckenmessung. Das Tachymeter hat die Richtung jeweils erst dann abgespeichert, wenn die Strecke ermittelt war. Zu diesem Zeitpunkt befand sich die Eisenbahn aber bereits an einer anderen Stelle und die Richtung hatte sich dementsprechend geändert. Die statistischen Tests fielen für die ausgeglichenen Kreisbögen trotz des Versatzes positiv aus.

Bei der Messung mit der Einstellung »Festzeit« bei 1 s zeigen sich diese Sprünge nicht, was die Annahme bestätigt (Abb. C.1). Auch hier fällt der Test für den Kreisbogen positiv aus, auch wenn er nur grob einem Halbkreis entspricht.

Ob und wie sehr die Wahl des Prismas die Messergebnisse beeinflusst, lässt sich aus den gesammelten Daten nicht ableiten. Dazu sind die Messergebnisse nicht vergleichbar. Das liegt zum einen daran, dass nicht die gleichen Punkte gemessen wurden und zum anderen, dass die Eisenbahn keine Fahrlastreglung besitzt und die Kurvenfahrt nicht mit konstanter Geschwindigkeit erfolgt. Darüber hinaus waren die Schienen nicht fest mit dem Untergrund verbunden. Es kann davon ausgegangen werden, dass die Bewegung der Bahn zu sehr kleinen Gleisverschiebungen führt.

Messverfahren	$\mathbf{y_M}$ [m]	$\mathbf{x_M}$ [m]	$\mathbf{r}$ [m]
Wahre Unbekannte	100,0000	100,0000	25,0000
Genäherte U. bei Tachymetrie	100,0015	100,0035	24,9995
Genäherte U. bei SAPOS	99,9995	99,9815	24,9915
Genäherte U. bei GPS	101,3961	96,0848	25,0486

Tabelle 6.1.: Wahren Unbekannte und genäherte Unbekannten bei simulierten Messreihen

6.1.2. Traktor

Bei den durch einen Traktor abgefahrenden Bahnkurven wurde zwei Teilstücke ausgewählt (Strecke und Bogen) und einzeln ausgeglichen. Aufgrund der verschiedenen Maßstabsfaktoren für X- und Y-Richtung wirken die Plots (Abb. C.4 und C.5) verzerrt. Das gefahrene Wendemanöver entspricht trotz Eingreifen des Fahrzeugführers einem Kreisbogen und der statistische Test fällt positiv aus. Auffallend ist, dass kurz nach Übergang der Strecke in den Bogen und vor Übergang des Bogens in die Strecke die Verbesserungen stärker zunehmen. Diese beiden Punkte scheinen nur grob auf dem Kreisbogen zu liegen.

Da der Traktor die gefahrene Spur des Fahrers lediglich GPS-geleitet nachfuhr, entsprechen die gesammelten Beobachtungen nur grob einer Gerade. Der statistische Test fiel aber trotzdem positiv aus. Die Qualität der Messergebnisse hängt von der Fahrleistung des Fahrzeugführers ab. Verbesserungen wurden bei der Ausgleichung daher besonders in Y-Richtung angebracht.

6.1.3. Simulierte Messreihen

Die drei Messreihen zur Simulation von Tachymeter-, SAPOS- und GPS-Messung wurden nacheinander in das Ausgleichungsprogramm geladen und jeweils als Kreis ausgeglichen. Der statistische Test fiel nur für den Tachymeter- und SAPOS-Messreihen positiv aus.

Das negative Ergebnis für die GPS-Messreihe ohne Korrekturdaten ist auf die großen Abweichungen von bis zu $\pm 5,000\,\mathrm{m}$ zurückzuführen. Wie aus Tabelle 6.1 ersichtlich, sind für die GPS-Messung bereits die Abweichungen bei den genäherten Unbekannten mit 1,4 und 4 m sehr groß. Aus Tabelle C.1 sind die Unterschiede zwischen den wahren Beobachtungen und den ausgeglichenen Beobachtungen ersichtlich. Die geplotteten Verbesserungen sind in Abbildung C.8 dargestellt.

Die Ausgleichungsergebnisse der Tachymeter- und SAPOS-Messreihen fallen dagegen besser aus. Die Verbesserungen in den Plots mussten vergrößert werden, damit sie erkennbar sind (Abb. C.6 und C.7). Auch die Näherungswerte liegen nahe an den wahren Unbekannten (Tabelle 6.1). Die ausgeglichenen Tachymeter-Beobachtungen weichen

in Y- und X-Richtung im Durchschnitt um 1,4 bis 2,0 mm von den wahren Werten ab, gegenüber 2,3 bis 2,6 mm bei den unausgeglichenen Beobachtungen. Die ausgeglichenen SAPOS-Beobachtungen weichen dagegen um 0,8 bis 1,0 cm ab, gegenüber 1,4 bis 1,5 cm bei den unausgeglichen.

6.2. Erweiterungen

In diesem Abschnitt sollen einige Erweiterungsmöglichkeiten erläutert werden, die den Weg in das Programm aus zeitlichen Gründen nicht mehr geschafft haben. Der Ausgleichungsalgorithmus wurde beispielsweise modular aufgebaut, so dass weitere geometrische Primitive einfach implementiert werden können. Die Ausgleichungssoftware bietet genügend Potential zur Steigerung des Funktionsumfangs.

6.2.1. Import von JobXML-Dateien

Die Möglichkeit, JobXML-Dateien in Punktdateien umzuwandeln, wurde mittels des *JXL2PKT*-Hilfsprogramms ausgelagert. Ein direkter Import wäre aber eine sinnvolle Erweiterung. Es ist auch zu überlegen, ob weitere Dateiformate anderer Hersteller ebenfalls unterstützt werden sollen.

6.2.2. Speichern und Editieren von Punkten

Um geladene Messdaten zu editieren, d. h. Punktnummern oder Koordinaten zu verändern, oder einzelne Punkte zu löschen, sind die entsprechenden Methoden noch umzusetzen. Gegenwärtig müssen diese Änderungen direkt in der Punktdatei vorgenommen werden. Auch können ausgeglichene Beobachtungen nicht gespeichert werden. Ebenso ist eine Funktion zum Export der Plotterdarstellungen denkbar.

6.2.3. Ausgleichung von Ellipsen

Neben den unter Abschnitt 4.1 beschriebenen geometrischen Figuren stellt die Ellipse eine weitere Möglichkeit zur Zerlegung von Trajektorien dar. Analytisch lässt sich eine Ellipse mit dem Mittelpunkt (x_0, y_0), der großen Halbachse a, der kleinen Halbachse b und einem beliebigen Punkt (x, y) auf dem Ellipsenbogen mittels der Hauptform

$$\frac{x - x_0}{a^2} + \frac{y - y_0}{b^2} = 1 \tag{6.1}$$

beschreiben. Der funktionale Zusammenhang wird durch Umstellung von (6.1) zu

$$\frac{x - x_0}{a^2} + \frac{y - y_0}{b^2} - 1 = 0 \tag{6.2}$$

gebildet.

Da die Ellipsen zu den Kegelschnitten gehören, beschreibt die Kegelschnittgleichung

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0 \tag{6.3}$$

für $B = 0$ eine Ellipse, deren Hauptachsen in X- und Y-Richtung liegen. Sie kann durch Aufstellen eines linearen Gleichungssystems zur Berechnung der Näherungswerte von x_0, y_0, a und b herangezogen werden. Mit mindestens fünf Punkten auf dem Ellipsenbogen lässt sich das Gleichungssystem zur Bestimmung der Koeffizienten von (6.3) lösen. Für den Fall, dass $B \neq 0$, muss das Koordinatensystem noch gedreht und anschließend der Winkel zwischen Ellipsen- und Koordinatenachsen bestimmt werden.

6.3. Automatische Zerlegung von Trajektorien

Die umgesetzte Softwarelösung beruht auf der manuellen Vorauswahl der geometrischen Primitive durch den Benutzer. Dieses Verfahren ist zeitaufwendig und fehleranfällig. Eine automatische Zerlegung der Trajektorien ist von Vorteil. Um eine Bahnkurve in Grundformen aufspalten zu können, müssen diese erkannt werden.

Ein Indiz kann das Ergebnis des statistischen Tests sein. Fällt dieser für eine gewählte geometrische Figur positiv aus, wird der entsprechende Teil der Bahnkurve als solche interpretiert. Durch den Vergleich von Testergebnissen kann – statistisch gesehen – die am besten passende Figur gewählt werden. Für dieses Verfahren sind aber Limitationen zu setzen. Beispielsweise fällt der Test für ein gerades Teilstück, das als Kreisbogen ausgeglichen wird, besser aus, als bei einer Ausgleichung als Strecke. Grund dafür ist, dass bei der Suche nache Näherungswerten ein Radius von 10^6 oder größer angesetzt wird, und die Strecke dann tatsächlich auf einem Kreisbogen liegt.

Der Vergleich von Richtungswinkeln kann ebenfalls eine Möglichkeit zur Erkennung von geometrischen Formen sein. Zumindest lassen sich Strecken und Kreisbögen auf diese Art erkennen. Iterativ sind die Punkte eines Teilstücks einer Bahnkurve zu untersuchen. Es sind die Richtungswinkel zwischen dem ersten oder letzten und allen anderen Punkten zu bestimmen. Als Orientierung wird der zweite bzw. vorletzte Punkt gewählt. Handelt es sich um eine Strecke, dürfen sich die Winkel gar nicht oder nur wenig von einander unterscheiden. Bei einem Kreisbogen muss die Differenz dagegen konstant sein. Für andere geometrische Primitive, wie Polynome, ist dieses Verfahren aber zu aufwendig. Mögliche Alternativen sind daher noch zu suchen.

6.3.1. Internationalisierung

Um die Anwendung einem Nutzerkreis zugänglich zu machen, der über den deutschsprachigen hinausgeht, ist eine Lokalisierung der Oberfläche notwendig. Gegenwär-

tig sind alle Beschriftungen der visuellen Komponenten im Quelltext festgeschrieben (*hard-coded*), was eine Übersetzung aufwendig macht. Damit nicht mehrere verschiedensprachige Programmversionen gepflegt werden müssen, sind bei einer Internationalisierung die Texte in Hilfszeichenketten auszulagern, die während der Ausführung dynamisch geladen werden. Die *NetBeans IDE* kann dieses Vorhaben durch einen integrierten Internationalisierungsassistenten unterstützen.

6.4. Kritik

In diesem Abschnitt soll die Arbeit einer kurzen, kritischen Betrachtung unterzogen werden. Durch den engen Zeitrahmen ließen sich nicht alle geplanten Funktionen der Ausgleichungssoftware umsetzen. Auch werfen die durchgeführten Experimente Fragen auf, die sich nur mit weiteren Versuchen beantworten lassen. Welche Genauigkeiten lassen sich bei der Ausgleichung von Polynomen erzielen? Können entsprechende Gleisverbindungen aufgebaut werden? Wie lassen sich etwaige Messabweichungen bei der Wahl verschiedener Prismen bestimmen?

Auch eine automatische Zerlegung von Trajektorien in geometrische Primitive konnte nicht implementiert werden. Zum Testen der dafür umzusetzenden Algorithmen werden auch weitere Experimente benötigt, um passende Messdaten zu sammeln. Sollen die dabei zu erzielenden Lösungen auch produktiven und nicht nur im akademischen Bereich eingesetzt werden, ist eine weitestgehend fehlerfreie Interpretation durch die Software notwendig. Diese Arbeit kann für eine solche Untersuchung als Grundlage dienen.

Unter diesen Gesichtspunkten scheint auch die Konzentration auf eine benutzerfreundliche Oberfläche für die Ausgleichungssoftware von Nachteil. Die Entwicklung einer grafischer Benutzeroberfläche (GUI) ist ungeachtet der Fähigkeiten moderner integrierter Entwicklungsumgebungen zeitaufwendig. Alternativ zu der gewählten Lösung können die implementierten Funktionen der GUI – das Auflisten der geladenen Punkte mittels Tabelle, das Zusammenfassen von Koordinaten zu geometrischen Objekten und das Anzeigen der Ausgleichungsergebnisse – auch innerhalb einer Konsole, d. h. einer textbasierten Ein- und Ausgabeschnittstelle (CLI[1]), umgesetzt werden. Lediglich für den Punktplotter ist ein grafisches Fenster nötig. Die eingesparte Entwicklungszeit hätte sich zur Integration weiterer Algorithmen einsetzen lassen. Zurückblickend scheinen die dadurch entstehenden Einschränkungen bei der Benutzerfreundlichkeit im Vergleich zu einem erhöhten Funktionsumfang hinnehmbar. Bei Wahl eines passenden Entwurfsmusters (beispielsweise *Model View Controller*[2]) ließe sich die Programmlogik von der Benutzerschnittstelle trennen und die GUI noch im nachhinein einbinden.

[1] engl. *Command Line Interface*

[2] Entwurfsmuster zur Strukturierung von Programmen in die Teile Datenmodell, Präsentation und Steuerung

6.5. Zusammenfassung

Das entwickelte Ausgleichungsprogramm besitzt Potenzial für weitere Anwendungsfälle, auch wenn es weiterentwickelt werden muss. Es hat sich gezeigt, dass sich der objektorientierte Ansatz bei der Programmierung auszahlt. Im späteren Verlauf der Entwicklungsarbeit ließen sich weitere Primitive wie Polynome einfacher hinzufügen. Das gilt auch für eine mögliche Einbindung von Ellipsen. Die dafür nötigen Grundlagen sind bereits umgesetzt.

Um die Software produktiv einsetzen zu können, ist eine automatische Zerlegung der Trajektorien nötig. Der gegenwärtige Stand reicht zwar durchaus für den akademischen Bereich, vor allem in der Lehre, aus, es lassen sich aber noch einige Anwendungsfälle finden. Diese Anwendungsfälle müssen aber gesondert betrachtet werden. Dies kann durch weitere Experimente, z. B. im Bereich der Baumaschinensteuerung, geschehen.

Auch wenn der Fokus zu sehr auf der Entwicklung einer optisch ansprechenden und benutzerfreundlichen Oberfläche lag und weitere, sinnvolle Funktionen außen vor gelassen werden mussten, sind die Grundfähigkeiten, das Ausgleichen von geometrischen Formen und die Visualisierung von Bahnkurven, implementiert.

A. Verzeichnisse

Literaturverzeichnis

[Bauer 2003] Bauer, M.: *Vermessung und Ortung mit Satelliten: GPS und andere satellitengestützte Navigationssysteme*. Heidelberg : Wichmann, 2003

[Claussen 1998] Claussen, U.: *Objektorientiertes Programmieren: Mit Beispielen und Übungen in C++*. Berlin u. a. : Springer, 1998

[Foppe 2009] Foppe, K.: Ausgleichungsrechnung mit Interpretation der Ausgleichungsergebnisse. Neubrandenburg : Gesellschaft zur Förderung der Geodäsie an der Hochschule Neubrandenburg e. V., 2009

[Gränicher 1996] Gränicher, H.: *Messung beendet – was nun?: Einführung und Nachschlagewerk für die Planung und Auswertung von Messungen*. Stuttgart, Zürich : Teubner, vdf Hochschulverlag, 1996

[Joeckel u. a. 2008] Joeckel, R. ; Stober, M. ; Huep, W.: *Elektronische Entfernungs- und Richtungsmessung und ihre Integration in aktuelle Positionierungsverfahren*. Heidelberg : Wichmann, 2008

[John Deere 2010] John Deere: *Agrar-Management-Systemlösungen (AMS)*. `http://www.deere.de/de_DE/products_ag/ams1/index_ams.html`. Version: 2010. – [Online; Stand 28. August 2010]

[Jäger u. a. 2005] Jäger, R. u. a.: *Klassische und robuste Ausgleichungsverfahren: Ein Leitfaden für Ausbildung und Praxis von Geodäten und Geoinformatikern*. Heidelberg : Wichmann, 2005

[Niemeier 2008] Niemeier, W.: *Ausgleichungsrechnung: Statistische Auswertemethoden*. Berlin, New York : de Gruyter, 2008

[Socher 2003] Socher, R.: *Theoretische Grundlagen der Informatik*. München, Wien : Hanser, 2003

Abbildungsverzeichnis

B. Eidesstattliche Erklärung

Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Neubrandenburg, den 23. September 2010

. . . . . .
(Unterschrift des Kandidaten)

C. Anhang

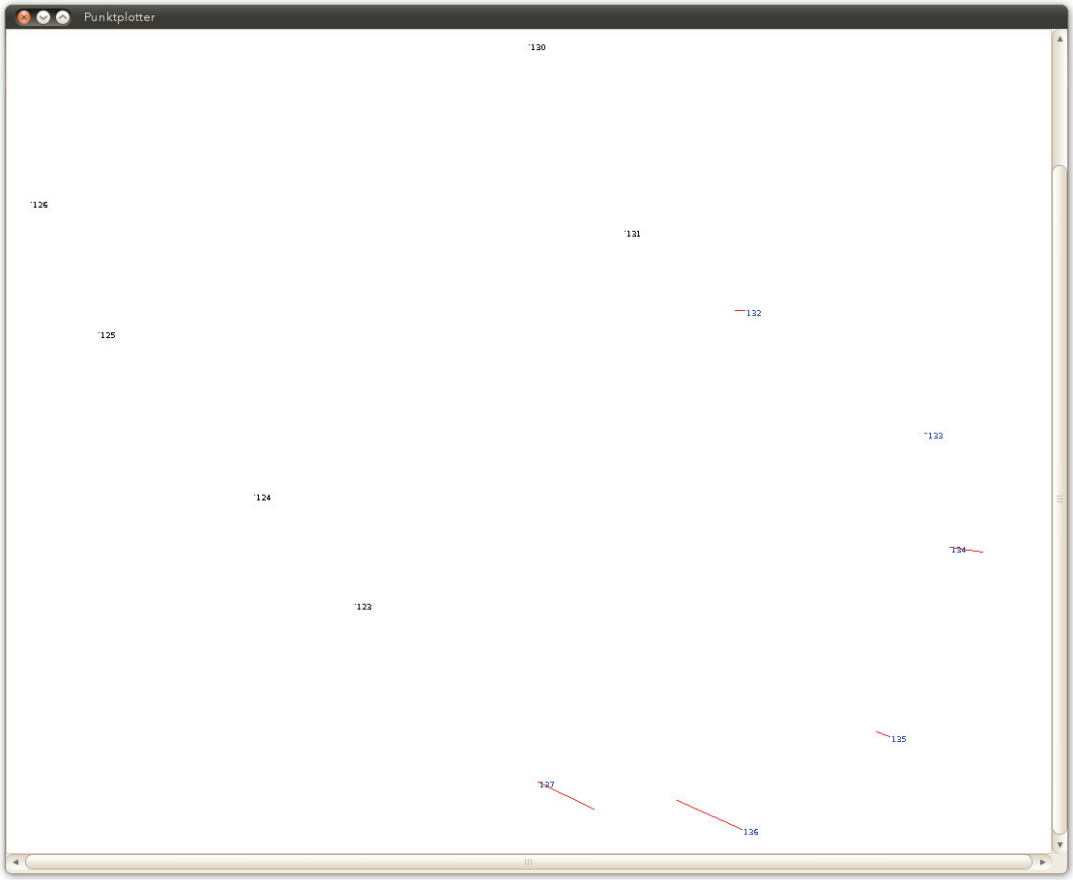

Abbildung C.1.: Messung mit 360°-Trimble-Prisma und Festzeit 1 s

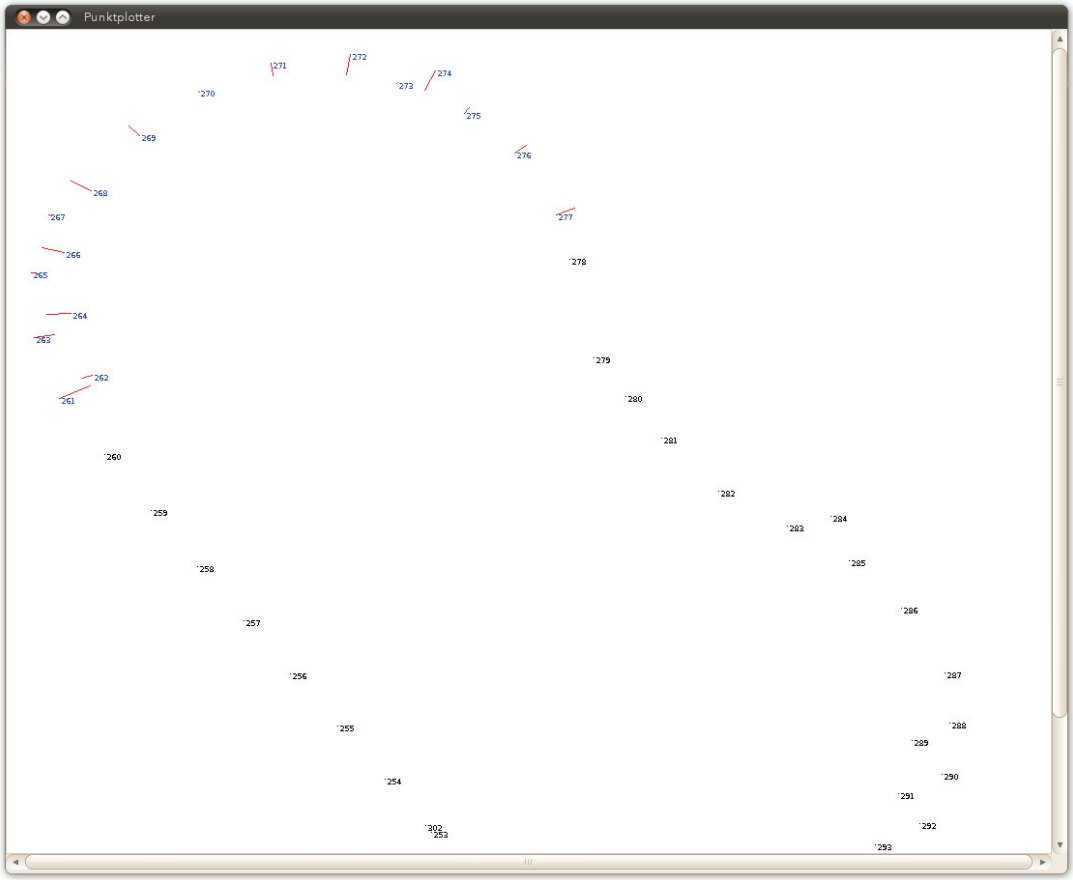

Abbildung C.2.: Messung mit 360°-Trimble-Prisma und Feststrecke 10 cm

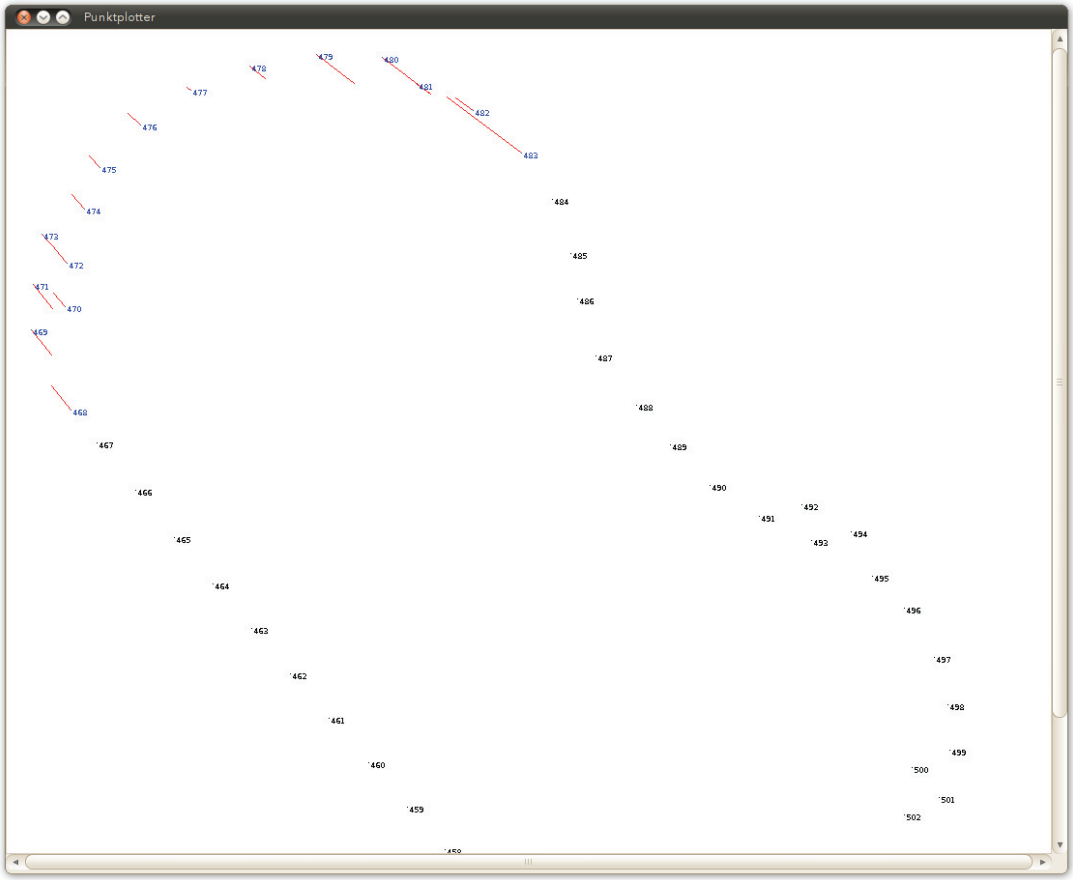

Abbildung C.3.: Messung mit *GRZ4*-Prisma von Leica und Feststrecke 10 cm

Abbildung C.4.: Ausgeglichene Strecke bei DGPS-Messung mit Traktor (Verbesserungen fünffach vergrößert)

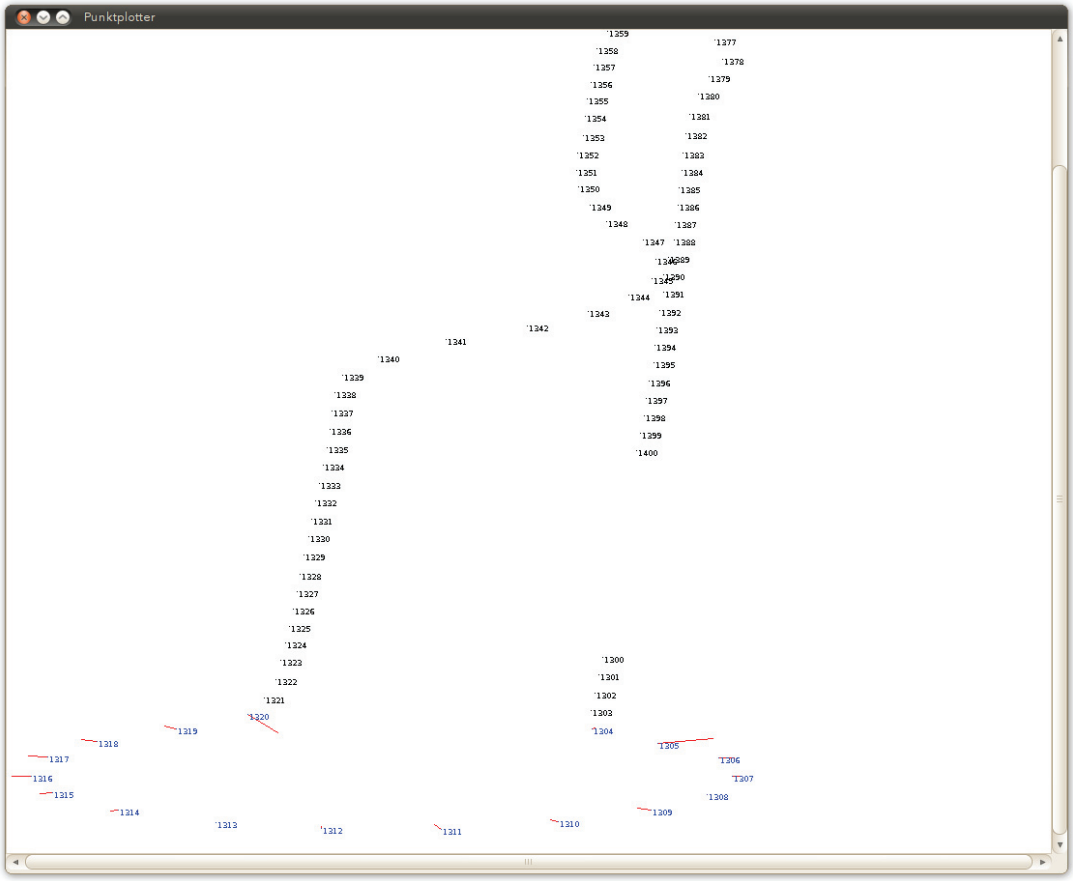

Abbildung C.5.: Ausgeglichener Kreisbogen bei DGPS-Messung mit Traktor (Verbesserungen zweifach vergrößert)

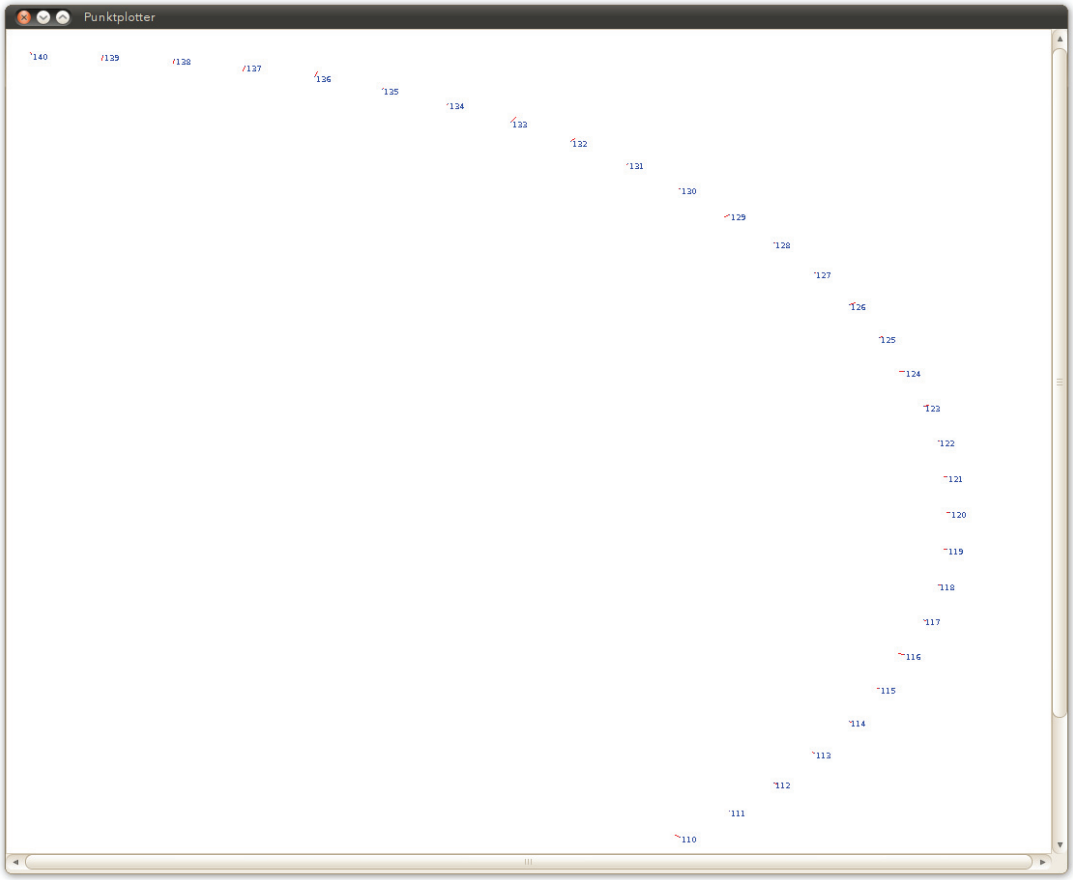

Abbildung C.6.: Simulierte Tachymeter-Messung (Verbesserungen 50-fach vergrößert)

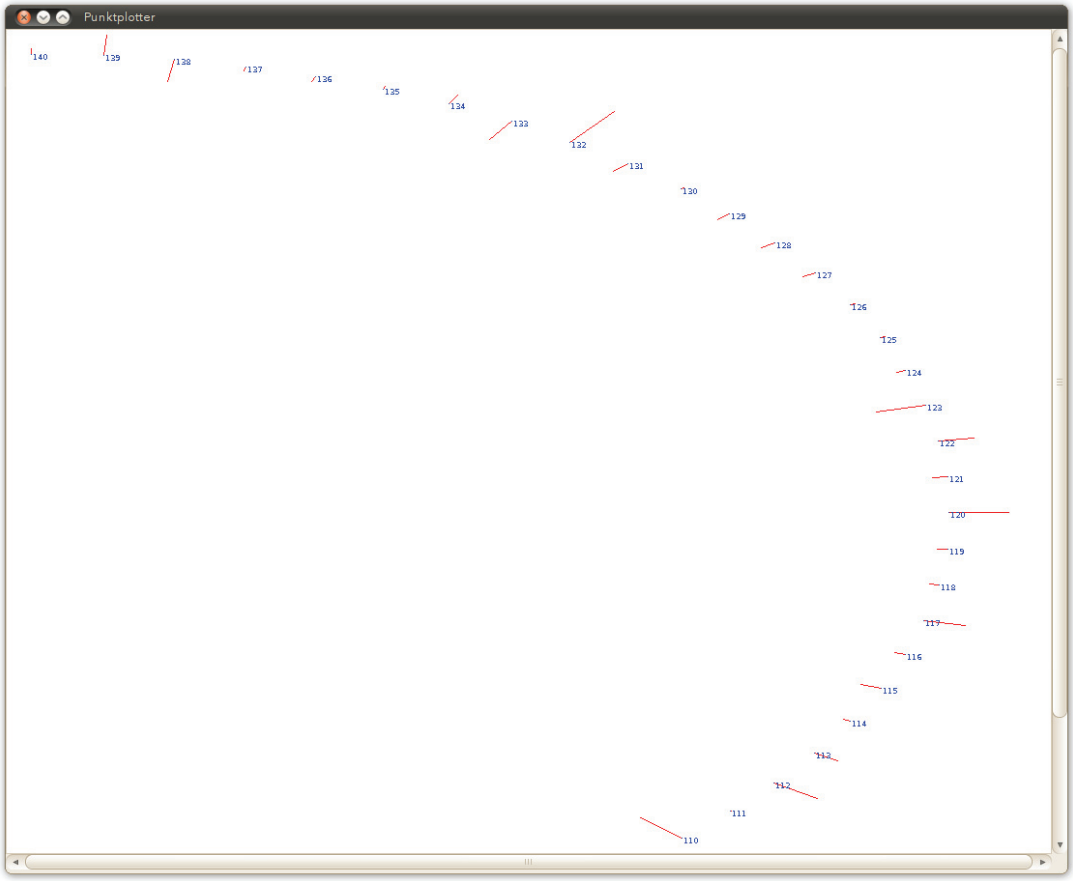

Abbildung C.7.: Simulierte DGPS-Messung mit SAPOS (Verbesserungen 50-fach vergrößert)

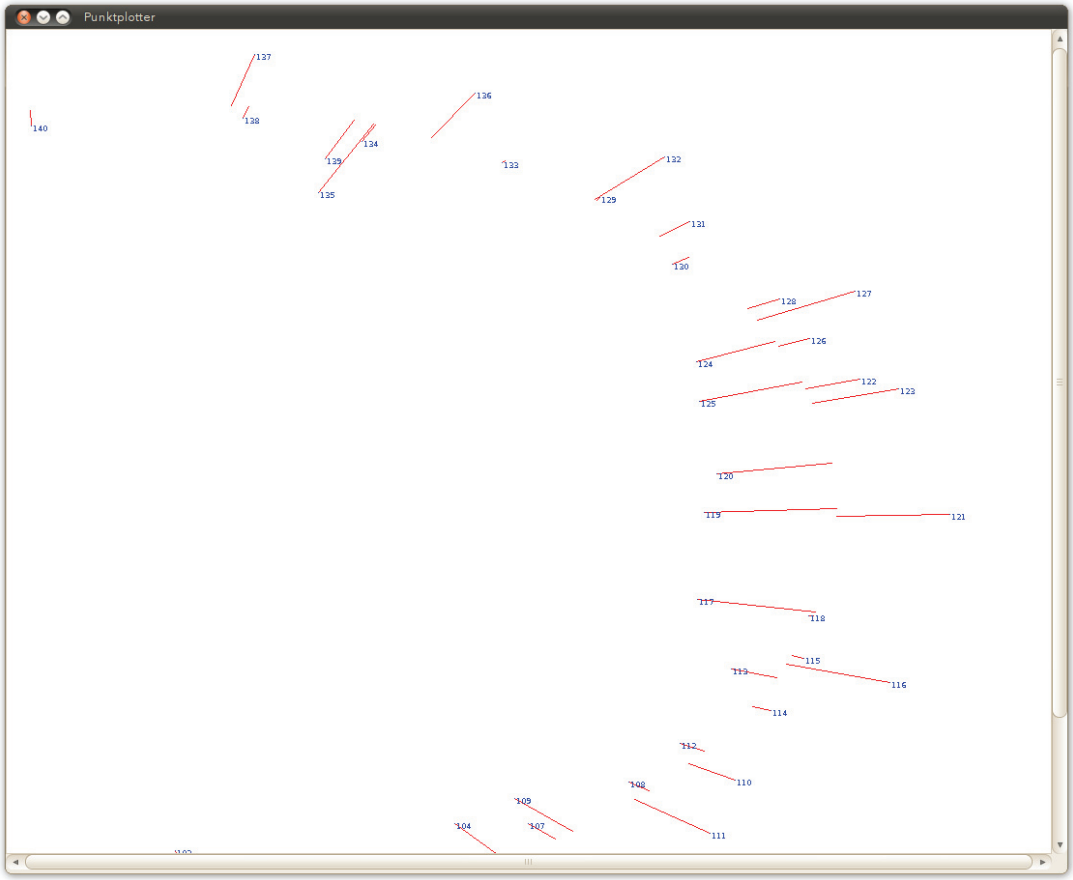

Abbildung C.8.: Simulierte GPS-Messung ohne Korrekturdaten

Wahre Beob.		**Tachymetrie**		**SAPOS**		**GPS**	
Y	X	Y	X	Y	X	Y	X
125,0000	100,0000	124,9997	99,9990	124,9997	100,0040	124,9358	100,4115
124,9229	101,9615	124,9228	101,9623	124,9248	101,9397	124,6097	104,7514
124,6922	103,9109	124,6914	103,9161	124,6921	103,9155	124,4160	105,7767
124,3092	105,8361	124,3090	105,8380	124,3148	105,8172	124,9108	101,9399
123,7764	107,7254	123,7753	107,7301	123,7832	107,7092	121,5927	113,0583
123,0970	109,5671	123,0973	109,5677	123,0990	109,5671	123,1763	109,8403
122,2752	111,3498	122,2773	111,3471	122,2908	111,3239	124,1161	107,0294
121,3160	113,0625	121,3164	113,0634	121,3116	113,0747	120,3786	114,9263
120,2254	114,6946	120,2278	114,6929	120,2236	114,7021	117,3769	118,3832
119,0101	116,2362	119,0106	116,2373	119,0134	116,2374	119,8899	115,5847
117,6777	117,6777	117,6780	117,6790	117,6852	117,6752	115,6825	119,8612
116,2362	119,0101	116,2349	119,0129	116,2324	119,0184	117,8967	117,8715
114,6946	120,2254	114,6930	120,2283	114,7059	120,2222	114,9403	120,4290
113,0625	121,3160	113,0604	121,3190	113,0858	121,3066	110,4071	123,0758
111,3498	122,2752	111,3495	122,2770	111,3340	122,2881	112,1794	122,1918
109,5671	123,0970	109,5666	123,0990	109,5536	123,1074	109,0581	123,6364
107,7254	123,7764	107,7269	123,7777	107,7161	123,7843	109,5632	123,4373
105,8361	124,3092	105,8372	124,3108	105,8542	124,3097	106,3307	124,5023
103,9109	124,6922	103,9154	124,6933	103,8982	124,6989	106,5941	124,4334
101,9615	124,9229	101,9619	124,9247	101,9746	124,9266	99,9535	125,2855
100,0000	125,0000	99,9980	125,0018	99,9790	125,0046	97,1856	125,1172
98,0385	124,9229	98,0361	124,9245	98,0426	124,9278	100,4702	125,2832
96,0891	124,6922	96,0919	124,6944	96,0900	124,6968	92,6057	124,1481
94,1639	124,3092	94,1654	124,3114	94,1603	124,3128	93,4878	124,4047
92,2746	123,7764	92,2728	123,7776	2,2666	123,7781	89,6482	123,0214
90,4329	123,0970	90,4373	123,1006	90,4381	123,1033	92,1297	123,9949
88,6502	122,2752	88,6524	122,2781	88,6567	122,2825	89,9506	123,1564
86,9375	121,3160	86,9406	121,3197	86,9370	121,3195	88,3455	122,3833
85,3054	120,2254	85,3005	120,2237	85,2912	120,2188	87,6480	122,0024
83,7638	119,0101	83,7643	119,0123	83,7426	118,9955	80,9221	116,4585
82,3223	117,6777	82,3185	117,6755	82,3347	117,6933	84,4440	119,8469
80,9899	116,2362	80,9895	116,2375	80,9716	116,2177	83,1683	118,7680
79,7746	114,6946	79,7738	114,6953	79,7689	114,6894	80,8626	116,3886
78,6840	113,0625	78,6837	113,0636	78,6885	113,0720	78,4898	113,0822
77,7248	111,3498	77,7233	111,3483	77,7333	111,3679	76,2787	108,3362
76,9030	109,5671	76,9033	109,5693	76,9106	109,5860	76,2535	108,2623
76,2236	107,7254	76,2232	107,7256	76,2299	107,7441	77,0879	110,3915
75,6908	105,8361	75,6904	105,8359	75,6979	105,8628	75,0909	103,0541
75,3078	103,9109	75,3084	103,9155	75,3098	103,9162	75,1694	103,6822
75,0771	101,9615	75,0769	101,9596	75,0808	101,9888	76,0239	107,5518
75,0000	100,0000	75,0002	99,9950	75,0021	100,0240	75,3869	95,6520

Tabelle C.1.: Wahre Beobachtungen und ausgeglichene Beobachtungen der einzelnen simulierten Messreihen