Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 52

Klassifikation von Bewegungstrajektorien durch regularisierte Regression

Michael Müller

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr.-Ing. Andrés Bruhn
Betreuer/in:	Prof. Dr.-Ing. Andrés Bruhn, Dipl.-Math. Sebastian Volz
Beginn am:	20. Juni 2013
Beendet am:	20. Dezember 2013
CR-Nummer:	I.2.10, I.4.8, G.1.6

Kurzfassung

Die Extraktion von Bewegungsinformation aus Bildfolgen ist eines der zentralen Probleme im Bereich der Bild- und Videoverarbeitung. Oft ist man dabei an dem Verschiebungsvektorfeld zwischen zwei aufeinander folgenden Bildern einer Bildfolge interessiert, welches in der Literatur als optischer Fluss bezeichnet wird. Da er keine reell zu erfassende Größe ist, werden verschiedene mehr oder weniger aufwändige Verfahren entwickelt, um die Größe des optischen Flusses möglichst genau bestimmen zu können.

Während die meisten Verfahren lediglich zwei aufeinanderfolgende Bilder betrachten und aus diesen auf den optischen Fluss schließen, wird im Falle des TC-Flow Verfahrens von Volz et al. (2011) eine Bildfolge über fünf Bilder betrachtet. Die Bestimmung des optischen Flusses erfolgt bei der TC-Flow Methode in drei Schritten. Zuerst wird eine Schätzung des optischen Flusses ohne trajektorielle Regularisierung, aber unter Berücksichtigung räumlicher Glattheitsannahmen durchgeführt. Anschließend erfolgt eine Klassifikation der in Schritt eins berechneten Bewegungstrajektorien in konstante, lineare oder polynomielle Bewegungsmodelle. Zuletzt erfolgt eine erneute Schätzung des optischen Flusses, bei dem die Regularisierung der Bewegungstrajektorien in die Berechnung mit einbezogen wird und somit der optische Fluss unter räumlichen und zeitlichen Glattheitsannahmen geschätzt wird.
Wenngleich die entstehenden Klassifikationskarten sehr verrauscht sind, lässt sich der Erfolg der TC-Flow Methode anhand des anerkannten Middlebury-Benchmarks nachvollziehen, bei dem das Verfahren einen der vorderen Plätze belegt.

Da beim Klassifikationsschritt des ursprünglichen TC-Flow Verfahrens die entstehenden Klassifikationskarten teilweise stark verrauscht waren, wird in dieser Bachelorarbeit neben einem zeitlichen auch ein räumlicher Zusammenhang der Bewegungstrajektorien geschaffen. Dazu wird der ursprüngliche Klassifikationsschritt aus der TC-Flow Methode zuerst nachimplementiert und anschließend dahingehend erweitert, dass räumliche Glattheitsannahmen in der Berechnung der Klassifikationskarten berücksichtigt werden.

Abschließend erfolgt eine Evaluation der neuen Klassifikationsverfahren. Diese findet einerseits visuell anhand der Klassifikationskarten, andererseits durch Integration in die TC-Flow Methode von Volz et al. statt.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Als optischer Fluss wird per Definition ein Vektorfeld bezeichnet, welches die auftretenden Verschiebungen zwischen zwei Bildern beschreibt. Eben jenes Vektorfeld, auf dessen Eigenschaften im folgenden Kapitel genauer eingegangen wird, spielt heutzutage bei der Bild- und Videoverarbeitung eine große Rolle. Allerdings ist der optische Fluss auch eine Größe, die nicht direkt aus den gegebenen Bilddaten eindeutig berechnet werden kann. Vielmehr wurden verschiedene, unterschiedlich aufwändige Verfahren entwickelt, die eine Schätzung des optischen Flusses erzeugen können.

Ein solches Verfahren wurde zum Beispiel im Rahmen der Arbeit [VBVZ11] entwickelt, TC-Flow genannt. Dieses Verfahren wird als Basis für diese Arbeit verwendet. Während die meisten anderen Verfahren lediglich zwei Frames zur Schätzung des optischen Flussfeldes benutzen, berechnet das TC-Flow Verfahren die Schätzung des optischen Fluss unter Zuhilfenahme mehrerer Frames. Es lässt sich dabei in drei Schritte gliedern. Zunächst erfolgt eine erste Berechnung der optischen Flussfelder zwischen den Bildpaaren. Anschließend wird eine Trajektorienklassifikation für diese Berechnung erstellt. Dafür fordert man Glattheit entlang von Trajektorien. Eine Trajektorie enthält dabei den Bewegungspfad eines Pixels über mehrere Frames. Anhand dieser Forderung werden die Trajektorien schlussendlich in konstante, lineare oder polynomielle Bewegungsmodelle klassifiziert. Abschließend wird der optische Fluss erneut unter Beachtung der zeitlichen Glattheit, die durch die Trajektorienklassifikation forciert wird, geschätzt.

In dieser Bachelorarbeit wird der Teil des TC-Flow Verfahrens, in dem die Klassifikation der einzelnen Trajektorien erstellt wird, zuerst nach der von Volz et al. dargelegten Vorgehensweise nachimplementiert und anschließend erweitert. Interessant ist in diesem Zusammenhang die Frage, ob es sich bei der Klassifizierung der Trajektorien ähnlich wie bei der Bestimmung des optischen Flussfeldes lohnt, den Nachbarschaftskontext, in dem sich eine Trajektorie befindet, zu betrachten. Dabei ist vor allem die Frage von Bedeutung, ob eine Einbeziehung des Umgebungskontextes zu einer Verbesserung des TC-Flow Verfahrens führt. In diesem Fall würde die beim Middlebury-Benchmark auf den vorderen Plätzen geführte TC-Flow Methode eine nochmals verbesserte Schätzung des optischen Flusses ermöglichen.

1.2 Ziel der Arbeit / Aufgabenstellung

Das Ziel dieser Arbeit ist weiterführende Verfahren zu entwickeln, die auf dem in [VBVZ11] eingesetzten TC-Flow Verfahren zur Regularisierung von Trajektorien basieren. Das TC-Flow Verfahren arbeitet ursprünglich mit robuster Regression und soll hinsichtlich der verrauschten Klassifikationskarten verbessert werden. Dazu wird das auf lokaler Ebene arbeitende Verfahren zur Klassifikation von Trajektorien im Rahmen der TC-Flow Methode zuerst nachimplementiert und anschließend um eine globale Komponente erweitert.

Im Anschluss daran soll die Implementierung und Anwendung weiterentwickelter Verfahren, die im Folgenden beschrieben sind, stattfinden. Diese Verfahren erweitern dabei das ursprüngliche Klassifikationsverfahren dahingehend, dass der Umgebungskontext, in dem sich eine Trajektorie befindet, in die Berechnung mit einfließt.
Abschließend wird eine Evaluation der Verfahren durchgeführt. Dabei soll neben den Vor- und Nachteilen der einzelnen Klassifikationsverfahren auch eine quantitative Evaluierung mithilfe eines zur Verfügung gestellten Evaluierungstools durchgeführt werden.

Unter Beachtung der oben genannten Aspekte, wird diese Bachelorarbeit folgende drei Kerngebiete umfassen:

1. **Nachimplementierung und Analyse des bestehenden Verfahren**
   Im ersten Teil dieser Bachelorarbeit wird das Regressionsverfahren zur Klassifikation von Trajektorien im Rahmen des von [VBVZ11] entwickelten TC-Flow Verfahren implementiert. Zudem wird in diesem Zusammenhang ein Programmgrundgerüst entwickelt, in dessen Rahmen die auf dem Klassifikationsverfahren aufbauenden, neu zu entwickelnden Verfahren leicht eingesetzt werden können. Abschließend wird das bestehende Klassifikationsverfahren evaluiert, um Verbesserungsansätze herausarbeiten zu können.

2. **Entwicklung und Implementierung weiterentwickelter Verfahren zur Klassifikation**
   In diesem Teil der Bachelorarbeit steht die Entwicklung neuer Verfahren zur Trajektorienklassifikation im Vordergrund, die auf dem zuvor evaluierten Klassifikationsverfahren aufbauen und das dort entwickelte lokale Klassifikationsmodell um eine globale Komponente erweitern.

3. **Evaluierung der weiterentwickelten Verfahren**
   Im letzten Teil dieser Bachelorarbeit werden die entwickelten Verfahren in den gesamten Prozess, wie er im Rahmen des TC-Flow Verfahrens durchgeführt wird und zur Schätzung des optischen Flusses dient, eingebettet. Abschließend erfolgt eine quantitative Bewertung mittels eines Evaluierungstools, um zu zeigen, ob und wie die einzelnen Verfahren die Abschätzung des optischen Flusses verbessern. Zudem sollen die Vor- und Nachteile der einzelnen Verfahren herausgearbeitet werden.

Zu beachten ist, dass diese Kerngebiete nicht streng sequenziell im Verlauf der Bachelorarbeit bearbeitet werden, sondern das jeweilige Verfahren nach dessen Implementierung zumindest in kurzer Ausführung evaluiert wird.

1.3 Verwandte Arbeiten

Durch die globale Erweiterung der bislang vorhandenen Trajektorienklassifizierung im TC-Flow-Verfahren kann in diesem Zusammenhang auch von einer Segmentierung der Trajektorien gesprochen werden. In der Literatur finden sich zu diesem Thema jedoch nur wenig verwandte Arbeiten.

Eine Arbeit, die sich mit einem verwandten Thema befasst, basiert auf der Segmentierung von Videos durch Clustering von Bewegungstrajektorien [OB12]. Dort findet ein räumliches Clustering auf Hypergraphenebene statt. Dazu wird zunächst der optische Fluss mit einem Zweibildverfahren berechnet. Anschließend werden die Ergebnisse zu längeren Trajektorien zusammengefasst [SBK10] und diese schlussendlich klassifiziert. Im Gegensatz zum in dieser Bachelorarbeit durchgeführten Verfahren besteht allerdings keine Rückkopplung. Das bedeutet, dass die Klassifikation keinen Einfluss auf die Schätzung des optischen Flussfeldes hat.

1.4 Gliederung

Auf diese Einleitung folgt zuerst ein Kapitel mit den Grundlagen, die benötigt werden, um einerseits eine kurze Einführung in das Thema zu bieten, andererseits um anschließend die einzelnen Verfahren/Algorithmen beschreiben zu können. Dazu gehören eine kurze Einführung in das Thema “Optischer Fluss”, ein Überblick über das TC-Flow Verfahren und abschließend eine Erläuterung der verwendeten Ausgangsdaten inklusive Erklärung der verwendeten Beispielsequenzen.

Anschließend werden die während dieser Bachelorarbeit entwickelten Verfahren zur Klassifikation der Trajektorien mathematisch beschrieben. Zuerst geschieht dies für die beiden lokalen Verfahren, der linearen sowie der robusten Regression. Neben der mathematischen Beschreibung wird auch ein Beispiel für die Trajektorienklassifikation zu Hilfe genommen, damit ersichtlich wird, weshalb noch Verbesserungsbedarf vorhanden ist und aus welchen Gründen die Entwicklung weiterentwickelter Verfahren notwendig erscheint.
Danach werden zwei globale Ansätze eingeführt und mathematisch beschrieben. Diese haben im Vergleich zu lokalen Verfahren die zusätzliche Eigenschaft, dass sie die Umgebung der Trajektorien in die Berechnung mit einbeziehen. Hierauf folgt ein Kapitel über die während der Bachelorarbeit vorgenommenen Implementierung der einzelnen Verfahren.

Im Anschluss folgt ein Kapitel mit der Evaluierung aller Verfahren. In diesem Abschnitt wird noch einmal genauer auf die Vor- und Nachteile der einzelnen Verfahren eingegangen. Zudem werden die einzelnen Verfahren auf Basis der Testsequenzen des Middlebury Benchmarks [BSL$^+$] auch einem quantitativen Vergleich unterzogen. Dies ermöglicht, die im Rahmen dieser Arbeit entwickelten Verfahren dahingehend zu überprüfen, welches Verfahren nicht nur subjektiv, sondern auch objektiv auf Basis der Beispielsequenzen die besten Resultate liefert.

Abschließend folgt eine Zusammenfassung der gesamten Bachelorarbeit, in der die verschiedenen Lösungsansätze kurz aufgezeigt und die erzielten Ergebnisse diskutiert werden. Zusätzlich wird ein Ausblick auf mögliche Ansätze zur Weiterentwicklung gegeben.

2 Grundlagen

2.1 Optischer Fluss

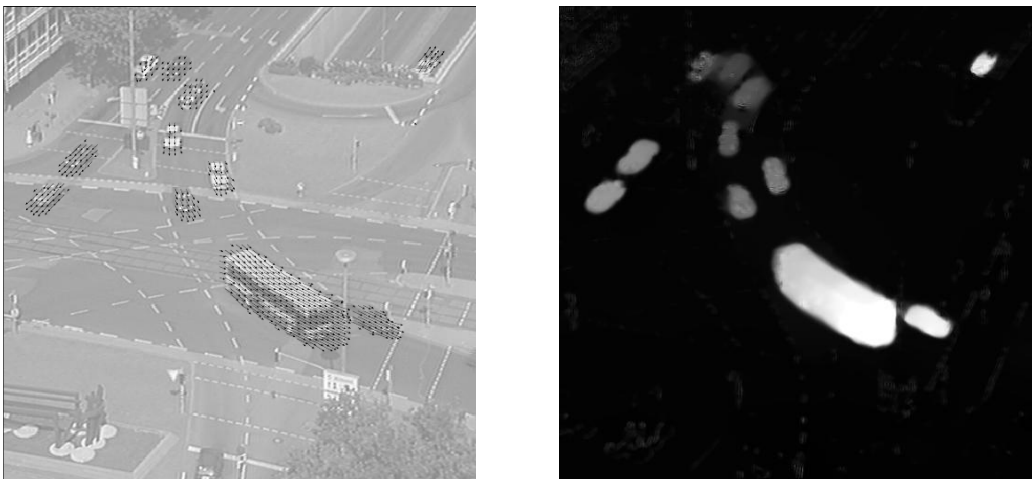
2.1.1 Beschreibung

Unter dem optischen Fluss versteht man ein Vektorfeld, welches für jeden Pixel eines Bildpaares die Bewegungsrichtung und -geschwindigkeit beinhaltet. Auf diese Weise gibt der optische Fluss an, wie sich ein Pixel vom ersten Bild des Bildpaares zum zweiten Bild hin verschiebt. In diesem Zusammenhang wird oftmals der Begriff Trajektorie verwendet, der den Bewegungspfad eines Pixels über mehrere Frames hinweg bezeichnet.

Der optische Fluss kann anhand einer gegebenen Bildfolge nicht exakt berechnet werden, sondern muss durch mehr oder weniger aufwändige Verfahren geschätzt werden. Dies ist der Tatsache geschuldet, dass in Bilddateien keinerlei Informationen über die Bewegungen von Pixel gespeichert sind. Dies führt dazu, dass aufwändige Forschungen betrieben werden müssen, um die Schätzungen für das Vektorfeld weiter zu verbessern. Der Grund dafür, dass große Anstrengungen gerade in diesem Bereich unternommen werden, ist, dass der optische Fluss in der Bild- und Videoverarbeitung eine große Rolle spielt, was im nächsten Abschnitt Anwendungen (Unterabschnitt 2.1.2) ersichtlich wird.

(a) Frame 5 der Sequenz Ettlinger Tor **(b)** Frame 6 der Sequenz Ettlinger Tor

Abbildung 2.1: Zwei Frames der Sequenz “Ettlinger Tor”[Nag]



(a) Vektorfeld des optischen Flusses **(b)** Berechneter Betrag des optischen Flusses

Abbildung 2.2: Schätzung des optischen Flusses für die Bildsequenz "Ettlinger Tor" [BBPW04]

Ein Beispiel für einen optischen Fluss stellt die Sequenz "Ettlinger Tor" in Abbildung 2.1 mit den dazugehörigen optischen Flussbildern in Abbildung 2.2 dar. Dieser Fluss wurde im Rahmen der Arbeit [BBPW04] berechnet. Zu sehen ist in der Sequenz die Aufnahme einer Verkehrskamera, die insgesamt 50 Frames umfasst, von denen in der Abbildung allerdings nur zwei dargestellt sind. Der optische Fluss repräsentiert in diesem Fall die einzelnen Autos, die sich als Objekte durch das Bild bewegen, während die Umgebung (Straße, Verkehrszeichen usw.) statisch ist und keinen optischen Fluss aufweist.

2.1.2 Anwendungen

Der optische Fluss spielt auf dem Gebiet der Informatik in der Bildverarbeitung eine große Rolle. Allerdings ist die Anwendung des optischen Flusses nicht nur auf die fachspezifische Informatik begrenzt, sondern aus den Eigenschaften des optischen Flusses werden auch in zahlreichen anderen Gebieten Vorteile gezogen. Beispiele hierfür sind unter anderem:

- **Bewegungsanalyse**

  Verkehrsflussanalyse
  Der Verkehr bewegt sich in einem statischen Umfeld vorwärts. Mithilfe des optischen Flusses können hierbei Autos als einzelne Objekte identifiziert werden, die sich fortbewegen. Demgegenüber bleibt die Straße und die Umgebung statisch, sofern es sich um eine Kamera mit festem Platz handelt. Die Bewegungen der Objekte können also in ihrer Quantität und Qualität analysiert werden, um eine genaue Analyse des Verkehrsflusses zu ermöglichen. Ein Beispiel für diesen Anwendungsfall findet sich in der im vorigen Abschnitt eingeführten Sequenz "Ettlinger Tor" (Abbildung 2.1 und Abbildung 2.2).

Bewegungsvorhersage
In diesem Fall wird der optische Fluss zur Berechnung eines neuen Frames benutzt. Beispiele für diesen Anwendungsfall sind die unabhängige Roboternavigation oder ein Kollisionsvermeidungsassistent in modernen Automobilen.

- **Videoverarbeitung Zwischenframe-Berechnung von Videosequenzen**
  Eine sehr häufig genutzte Möglichkeit den optischen Fluss einzusetzen besteht bei Videosequenzen. Möchte man beispielsweise eine aufgenommene Videosequenz in Zeitlupe abspielen lassen und soll dies mit der selben Framerate wie bei der originalen Sequenz geschehen, setzt man den optischen Fluss zur Berechnung der einzelnen Zwischenframes ein. Diese Technik findet vor allem bei Filmen Verwendung, wobei dort heutzutage oft auch sogenannte Highspeed-Kameras zum Einsatz kommen.

  Videokompression
  Bei der Videokompression werden die zuvor dargestellten Anwendungen aus der Bewegungsvorhersage oder der Zwischenframe-Berechnung direkt angewendet. Es werden dann nicht die kompletten Frames, sondern nur die Unterschiede, also per Definition der optische Fluss, des Videos gespeichert.

- **Computer-Mensch-Interaktion**

  Optische Computermäuse
  Computermäuse mit optischem Sensor benutzen entsprechend ihrem Namen den optischen Fluss für die Berechnung der Bewegung. Dabei wird aus nacheinander von einem Sensor aufgenommenen Grauwertbildern der optische Fluss bestimmt und für die Umrechnung in die entsprechenden Geschwindigkeitsdaten benutzt.

  Gestenerkennung
  Um Gesten erkennen zu können, muss herausgefunden werden, wie sich die Position einer Person, einer Hand oder allgemein eines Objekts zwischen zwei aufgenommenen Bildern verändert. Dies entspricht der Schätzung des optischen Flusses. Durch die Berechnung der Trajektorien für die einzelnen Pixel kann herausgefunden werden, wohin sich ein Objekt bewegt. Die Trajektorie gibt darüberhinaus auch Aufschluss, auf welchem Pfad diese Bewegung verläuft. Durch Abgleich mit bekannten Bewegungsmustern kann schlussendlich herausgefunden werden, welche Geste durchgeführt wurde.

- **Biologische Nutzung:**

  Navigation mithilfe des optischen Flusses
  Bienen führen ihren Tanz mithilfe des optischen Flusses auf, um ihren Artgenossen den Weg zur gefundenen Futterquelle vorzutanzen und dadurch zu beschreiben [EZST01].

Darüberhinaus besitzt der optische Fluss noch zahlreiche weitere Anwendungen, gerade in den modernen Wissenschaften, wie beispielsweise der Robotik (Beispiel: [MB04]), spielt er eine immer größer werdende Rolle. Zudem ist im Bereich der Forschung über autonomes Fahren eines Automobils der optische Fluss sehr wichtig [GMF02].

2.2 TC-Flow Verfahren

Das bereits in der Kurzfassung und im Einleitungskapitel erwähnte Verfahren, auf dem diese Bachelorarbeit aufbaut, ist das im Rahmen der Arbeit [VBVZ11] beschriebene TC-Flow Verfahren. Es handelt sich dabei um ein Verfahren, das Flussfelder entgegen den meisten bisherigen Verfahren in einen zeitlichen Zusammenhang versetzt. Dies geschieht durch die Einführung von Trajektorien, die den Bewegungspfad eines Pixels über mehrere Frames bezeichnen. Die Vorgehensweise lässt sich in drei Phasen einteilen:

1. **Berechnung des optischen Flusses ohne trajektorielle Glattheit**
   In dieser Phase wird der optische Fluss aus den einzelnen Frames abgeleitet. Es werden dabei fünf Frames betrachtet, wobei der Ansatz auf beliebig viele Frames erweiterbar ist. Im Gegensatz zu herkömmlichen, bislang verwendeten Verfahren, die bei der Minimierung wiederholt die optischen Flussfelder von verschiedenen Zeitpunkten aufeinander mappen müssen, wird in diesem Ansatz jedes Flussfeld auf ein Referenzframe parametrisiert. So wird im TC-Flow Verfahren für die einzelnen optischen Flüsse nicht die Verschiebung zweier aufeinanderfolgender Bilder angegeben, sondern diese Verschiebung mit Hinblick auf das Referenzframe ausgedrückt. Gleichzeitig wird zur Schätzung zusätzlich ein variationaler Ansatz verwendet, der räumliche Glattheit annimmt.

2. **Klassifizierung der Trajektorien in Bewegungsmodelle**
   In dieser Phase erfolgt die Klassifizierung der Trajektorien ohne Beachtung des Umgebungskontexts mit Hilfe robuster Regression. Da die einzelnen Vektoren, die zu einer bestimmten Trajektorie gehören, durch das Mapping auf einen Referenzpixel einfach zu bestimmen sind, entfällt gegenüber den meisten anderen Verfahren das Mappen der Vektoren in dieser Phase. Die Trajektorien werden hierbei in drei unterschiedliche Bewegungsmodelle klassifiziert: Gleichförmig konstante, lineare und polynomielle Bewegung.

3. **Berechnung des optischen Flusses mit trajektorialer Regularisierung**
   Nachdem jedem Pixel des Referenzframes in Schritt 2 ein Bewegungsmodell zugeordnet wurde, wird Schritt 1 wiederholt. Allerdings erfolgt im Gegensatz zu Schritt 1 eine trajektorielle Regularisierung. Das bedeutet, dass entsprechend den zuvor bestimmten Bewegungsmodellen in Schritt 2 eine diesem Bewegungsmodell entsprechende Glattheit entlang der jeweiligen Trajektorie angenommen wird. Dies führt zu Ergebnissen, die einen verbesserten Zusammenhang über die Zeit besitzen.

Wie zuvor bereits erwähnt, wird im Rahmen dieser Bachelorarbeit eine Erweiterung der zweiten Phase durchgeführt. Dazu werden die optischen Flussfelder aus dem ersten Abschnitt benutzt, um die trajektorielle Regularisierung in der zweiten Phase mit verschiedenen Verfahren durchzuführen. Wie dies genau geschieht, ist Teil der folgenden beiden Kapitel.

2.3 Ausgangsdaten

Als Ausgangsdaten dienen die als Trainingssequenzen erhältlichen Daten des Benchmarks der Universität Middlebury [BSL+]. In dieser Arbeit werden die frei verfügbaren Daten der beiden Testsequenzen "Hydrangea" und "RubberWhale" verwendet, um die im Rahmen dieser Arbeit implementierten Verfahren einerseits auf ihre richtige Funktion hin zu überprüfen, andererseits um die verschiedenen Verfahren untereinander vergleichen zu können. Zudem besteht durch die Evaluierung die Möglichkeit, die in dieser Arbeit erstellte Weiterentwicklung des TC-Flow Verfahrens mit den Originalergebnissen der TC-Flow Methode anhand dieser Daten zu vergleichen. Im Folgenden wird zuerst auf das vorliegende Format eingegangen, ehe anschließend eine Beschreibung der zugrunde liegenden Beispielsequenzen erfolgt.

2.3.1 Farbkodierung des optischen Flusses

Zur visuellen Darstellung der einzelnen optischen Flussfelder dient die in der nebenstehenden Abbildung 2.3 dargestellte Farbdarstellung. Negative Flüsse in x-Richtung werden demnach beispielsweise grün, positive rot dargestellt. Mit zunehmender Länge des Flussvektors fällt dabei die Intensität, in der der Fluss dargestellt wird, stärker aus.

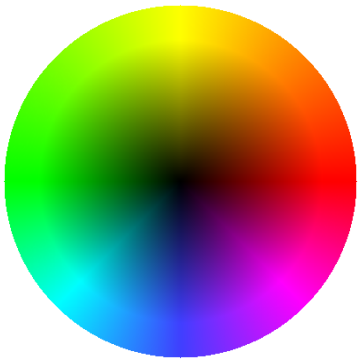


Abbildung 2.3: Farbkodierung des optischen Flusses

2.3.2 Ausgangslage vor der Klassifizierung

Nach der Berechnung des optischen Flussfeldes in Schritt 1 des TC-Flow Verfahrens erhält man als Ergebnis für fünf aufeinanderfolgende Bilder vier Flussfelder, die jeweils in einer Datei im .flo-Format abgespeichert werden. Das .flo Format wurde an der Universität Middlebury entwickelt. Jede flo-Datei ist dabei wie folgt aufgebaut:

Bytes	Inhalt
0 bis 3	Das Wort “PIEH” im ASCII-Format, das nach der Umwandlung in die Float-Zahl 202021.25 repräsentieren soll und zur Überprüfung der korrekten Umwandlung von ASCII nach Float dienen soll
4 bis 7	die Breite des Flussfeldes (Anzahl der Pixel)
8 bis 11	die Höhe des Flussfeldes (Anzahl der Pixel)
12 bis Dateiende	Daten mit folgendem Aufbau: 4 Bytes pro Flusswert horizontale x- und vertikale y-Flusskomponenten sind alternierend und zuerst spalten-, dann zeilenweise aufsteigend sortiert angeordnet: $u[0,0], v[0,0], u[0,1]$, ...

Durch das Einlesen dieser vier Dateien erhält man sowohl für die horizontale Flusskomponente u als auch die vertikale Flusskomponente v in unserem Fall vier zweidimensionale Arrays, die den obigen Frames und somit den Zeitpunkten $t_1 = -1.5, t_2 = -0.5, t_3 = 0.5$ und $t_4 = 1.5$ zugeordnet werden. Die Schätzung des optischen Flusses soll dabei für $t = 0$ geschehen. Im folgenden Text werden die zu den Zeitpunkten t_k mit $k \in \{1, 2, 3, 4\}$ gehörenden Flusskomponenten u_k und v_k mit dem zuvor definierten k bestimmt.

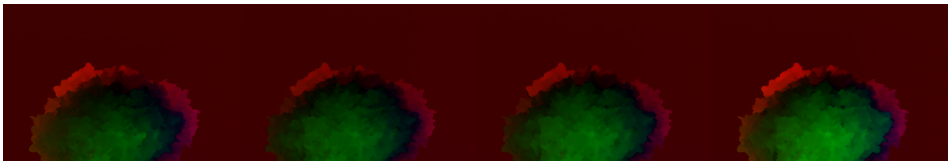
Diese vier Arrays haben dabei die Zeile x und Spalte y als Dimensionen. Wenn im Folgenden vom Fluss u_k bzw. v_k die Rede ist, ist dabei immer der pixelspezifische Fluss, also die Trajektorie $u[x][y]_k$ zum Zeitpunkt t_k gemeint.

2.3.3 Sequenz Hydrangea

(a) 1. Frame der Sequenz **(b)** 2. Frame der Sequenz **(c)** 3. Frame der Sequenz **(d)** 4. Frame der Sequenz **(e)** 5. Frame der Sequenz

Abbildung 2.4: Hydrangea Sequenz, Frames 1-5.

Die Bildsequenz Hydrangea, die als Testsequenz zur Bestimmung des optischen Flusses von der Universität Middlebury bereitgestellt wird, ist eine Bildfolge mit wenig unterschiedlichen Bewegungen, weshalb die berechneten Flussfelder relativ homogen sind. Dies wird insbesondere im Vergleich mit der Bildsequenz RubberWhale, die anschließend vorgestellt wird, deutlich.



(a) 1. Feld des optischen Flusses **(b)** 2. Feld des optischen Flusses **(c)** 3. Feld des optischen Flusses **(d)** 4. Feld des optischen Flusses

Abbildung 2.5: Hydrangea Sequenz, Optische Flussfelder 1-4.

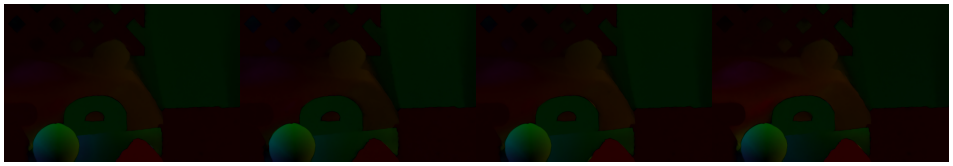
In dieser Sequenz sieht man im unteren Teil des Bildes in der Mitte eine Hortensie, die sich um die eigene Längsachse dreht, während der Hintergrund sich konstant nach rechts verschiebt.

2.3.4 Sequenz RubberWhale

(a) 1. Frame der Sequenz **(b)** 2. Frame der Sequenz **(c)** 4. Frame der Sequenz **(d)** 4. Frame der Sequenz **(e)** 5. Frame der Sequenz

Abbildung 2.6: RubberWhale Sequenz, Frames 1-5.

In der Bildsequenz “RubberWhale”, die ebenfalls aus dem Testdatensatz der Universität Middlebury stamt, finden deutlich mehr unterschiedliche Bewegungen im Vergleich zur davor gezeigten Sequenz “Hydrangea” statt.
Zu sehen ist in der Mitte ein Wal, der namensgebend für die Bildfolge steht. Im unteren rechten Teil befindet sich im Vordergrund eine gelbe Schachtel, die sich langsam nach rechts aus dem Bild bewegt. Der sich über der Schachtel befindende Vorhang bewegt sich dazu entgegengesetzt. Die Muschel, die sich in den Frames unten in der Mitte befindet, bewegt sich in einer etwas schnelleren Geschwindigkeit als die Schachtel nach rechts. Die räumlich gesehen hinter der Muschel und Schachtel liegenden Objekte, unter anderem ein Stück Pappe, bewegen sich nach links. Zudem bewegt sich das Objekt, auf dem der Wal liegt, leicht nach rechts oben.



(a) 1. Feld des optischen Flusses **(b)** 2. Feld des optischen Flusses **(c)** 3. Feld des optischen Flusses **(d)** 4. Feld des optischen Flusses

Abbildung 2.7: RubberWhale Sequenz, Optische Flussfelder 1-4.

3 Beschreibung lokaler Verfahren zur Trajektorienklassifizierung

In den folgenden Unterkapiteln werden zur Klassifikation der Bewegungstrajektorien zwei lokale, das heißt umgebungsunabhängige Verfahren detailliert beschrieben. Zuerst wird hierbei die lineare Regression erklärt. Im zweiten Schritt wird diese dann zur robusten Regression erweitert, die grundsätzlich gegenüber der linearen Regression den Vorteil hat, dass einzelne Ausreißer weniger stark berücksichtigt werden. Dieser Ansatz fand bereits bei der Trajektorienklassifikation von [VBVZ11], die die Basis dieser Bachelorarbeit darstell,t Anwendung.

In den folgenden Unterkapiteln wird immer, wenn der Fluss u als Parameter oder Index auftaucht, auf den Fluss u am Pixel (i) referenziert. Der einfacheren Schreibweise halber wird allerdings der zusätzliche Index i weggelassen. Zum Beispiel gilt für den Parameter a für den Fluss u am Pixel i folgende Kurzschreibweise a_u, während die vollständige Schreibweise $[a_u]_i$ lautet.

3.1 Lineare Regression

Unter der linearen Regression, meistens als lineare Regressionsanalyse bezeichnet, versteht man die statistische Analyse einer gegebenen Menge von Datenpunkten. Ihren Namen erhält die lineare Regression dadurch, dass die Parameter hier in linearer Form auftreten. Dabei wird je nach Regressionstyp eine Funktion von bestimmtem Grad berechnet, die von einer gegebenen Punktmenge die minimal mögliche Abweichung besitzt und somit die beste Nährung für diese darstellt:

Regressionstyp	**Gleichung der Regressionskurve**
Regressionsgerade	$ax + b$
quadratische Regressionskurve	$ax^2 + bx + c$
kubische Regressionskurve	$ax^3 + bx^2 + cx + d$
...	

Im Folgenden sei die Punktmenge P, die als Ausgangslage für die lineare Regression dient, wie folgt gegeben. Hierbei sei $f(x)$ die unbekannte Funktion, die durch die Regression approximiert werden soll.

(3.1) Für $k \in \{1, 2, ..., n\}, x_k : f(x_k) = y_k \quad P_k = (x_k, y_k) = (x_k, f(x_k))$

Darüberhinaus schränken wir uns auf Polynome vom Grad 2 ein, um $f(x)$ zu approximieren. Dies führt zu einer quadratischen Regressionskurve $ax^2 + bx + c$, die bestimmt werden muss.

3.1.1 Allgemeine Vorgehensweise

Um die Regressionskurve zu bestimmen, die sich dieser wie oben definierten Punktmenge am Besten annährt, sucht man das Minimum der folgenden Energiefunktion, welche die drei zu bestimmenden Variablen a, b und c einführt. Diese Energiefunktion entspricht dabei dem Vorgehen bei einem Least-Squares-Ansatz:

$$
\begin{aligned}
E(a,b,c) &= \sum_{k=1}^{n} (ax_k^2 + bx_k + c - f(x_k))^2 \\
E(a,b,c) &= \sum_{k=1}^{n} (ax_k^2 + bx_k + c - y_k)^2 \qquad |f(x_k) = y_k
\end{aligned} \tag{3.2}
$$

Bei Betrachten der Energiefunktion fällt auf, dass es sich um eine Funktion handelt, in der kein Maximum, sondern ein Minimum vorhanden sein **muss**. Ursache hierfür ist, dass durch das Quadrieren der einzelnen Datenterme alle ursprünglich negativen Terme nun positiv werden. Zudem werden größere Werte durch die Quadratur des Datenterms stärker wachsen als kleinere Werte. Zu beachten ist zudem, dass eine Energiefunktion, welche ein Maximum besitzt, offensichtlich nicht geeignet ist, ein Least-Squares-Problem zu lösen.

Um anschließend das Minimum der obigen Energiefunktions zu bestimmen, wird die Ableitung der Funktion gebildet, da für ein Extremum einer Funktion f immer folgende Bedingung gelten muss:

$$f'(x) = 0 \tag{3.3}$$

Die Ableitung für die zuvor eingeführte Energiefunktion wird dadurch gebildet, dass diese nach den drei zu bestimmenden Variablen a, b und c abgeleitet wird. Dies führt zu folgendem Gleichungssystem:

$$\frac{\partial E(a,b,c)}{\partial a} = \sum_{k=1}^{n} \left(2\left(ax_k^2 + bx_k + c - y_k\right) x_k^2\right) \qquad = 2\sum_{k=1}^{n} (ax_k^4 + bx_k^3 + cx_k^2 - x_k^2 y_k) \tag{3.4}$$

$$\frac{\partial E(a,b,c)}{\partial b} = \sum_{k=1}^{n} \left(2\left(ax_k^2 + bx_k + c - y_k\right) x_k\right) \qquad = 2\sum_{k=1}^{n} (ax_k^3 + bx_k^2 + cx_i - x_i y_k) \tag{3.5}$$

$$\frac{\partial E(a,b,c)}{\partial c} = \sum_{k=1}^{n} 2\left(ax_k^2 + bx_k + c - y_k\right) \qquad = 2\sum_{k=1}^{n} (ax_k^2 + bx_k + c - y_k) \tag{3.6}$$

Damit die Extremumbedingung aus Gleichung 3.3 in die Berechnung miteingebunden wird, setzt man die oben erhaltenen Ableitungsterme mit 0 gleich. Die Zwei wird als konstanter Faktor durch Division aus der Gleichung herausgekürzt.

$$\begin{aligned} \sum_{k=1}^{n} \left(a x_k^4 + b x_k^3 + c x_k^2 - x_k^2 y_k \right) &= 0 \\ \sum_{k=1}^{n} \left(a x_k^3 + b x_k^2 + c x_k - x_k y_k \right) &= 0 \\ \sum_{k=1}^{n} \left(a x_k^2 + b x_k + c - y_k \right) &= 0 \end{aligned} \tag{3.7}$$

Dieses lineare Gleichungssystem kann mit verschiedenen bekannten Methoden gelöst werden. In den folgenden beiden Abschnitten wird gezeigt, wie diese Gleichung und deren Lösung explizit in der bearbeiteten Problemstellung aussieht und wie diese effizient berechnet werden kann, indem bestimmte Eigenschaften des erhaltenen Gleichungssystems ausgenutzt werden.

3.1.2 Problemangepasste Vorgehensweise

Wird nun die allgemeine Methode auf die zuvor geschilderte Problemstellung in Abschnitt 1.2 übertragen, ergeben sich - mit den in Unterabschnitt 2.3.2 definierten Werten - für folgende Parameter die angeführten Werte, die mit einem Kommentar zur Erklärung versehen sind:

(3.8)

Ω	Dimensionen des Flussfeldes
$n = 4$	vier Datenpunkte, da fünf Frames und somit vier Flussfelder vorhanden sind.
k :	Index über aktuelle Iteration
$t_i =$	$(-1.5, -0.5, 0.5, 1.5) : k \in (1, 2, 3, 4)$
	übernommen aus Unterabschnitt 2.3.2, Zeitpunkte
$x_k = t_k$	übernommen aus Unterabschnitt 2.3.2
	Zeitpunkte entsprechen den Werten auf der x-Achse
$y_k = u_k$	übernommen aus Unterabschnitt 2.3.2
	entsprechender y-Wert zum Zeitpunkt k

Daraus ergibt sich durch Einsetzen der Parameter aus Gleichung 3.8 in Gleichung 3.2 die folgende Gleichung für die Energiefunktion E_u, welches in jedem Pixel $P = (i), i \in \Omega$ berechnet werden muss. In jedem Pixel sind zwei lineare Regressionen zu berechnen: Je eine für die Flusskomponenten in x-Richtung (u) sowie für die Flusskomponenten in y-Richtung (v). Dies kommt in allen folgenden Schritten dadurch zum Ausdruck, dass die einzelnen

Parameter, die für u und v einzeln zu betrachten sind mit den Indizes u bzw. v versehen sind.

$$
\begin{aligned}
E_u(a_u, b_u, c_u) &= \sum_{k=1}^{4} (a_u t_k^2 + b_u t_k + c_u - u_k)^2 \\
E_v(a_v, b_v, c_v) &= \sum_{k=1}^{4} (a_v t_k^2 + b_v t_k + c_v - v_k)^2
\end{aligned} \tag{3.9}
$$

Im Folgenden betrachten wir nur noch die Vorgehensweise für die gegebenen Flussfelder in x-Richtung. Dementsprechend treten ab sofort nur noch die Flusskomponenten u auf. Die Vorgehensweise für die Flussfelder in y-Richtung ist analog und kann durch einfaches Ersetzen von u durch v in den folgenden Schritten erreicht werden.

Bildet man nun analog zum allgemeinen Fall die Ableitung der Energiefunktion, ergeben sich für den problemangepassten Fall folgende Ableitungsgleichungen:

$$
\begin{aligned}
\frac{\partial E_u(a_u, b_u, c_u)}{\partial a_u} &= \sum_{k=1}^{4} 2 \left(a_u t_k^2 + b_u t_k + c_u - u_k\right) t_k^2 \\
&= \sum_{k=1}^{4} 2(a_u t_k^4 + b_u t_k^3 + c_u t_k^2 - t_k^2 u_k) \\
\frac{\partial E_u(a_u, b_u, c_u)}{\partial b_u} &= \sum_{k=1}^{4} 2 \left(a_u t_k^2 + b_u t_k + c_u - u_k\right) t_k \\
&= \sum_{k=1}^{4} 2(a_u t_k^3 + b_u t_k^2 + c_u t_k - t_k u_k) \\
\frac{\partial E_u(a_u, b_u, c_u)}{\partial c_u} &= \sum_{k=1}^{4} 2 \left(a_u t_k^2 + b_u t_k + c_u - u_k\right)
\end{aligned} \tag{3.10}
$$

Äquivalent zur allgemeinen Vorgehensweise folgt ein Gleichsetzen der Ableitung mit 0. Wiederum wird durch eine Division der Faktor 2 als konstanter Faktor eliminiert. Zudem wird im zweiten Teil der durch die Regression anzunährende Fluss auf die andere Seite gezogen, sodass das zu ermittelnde Ergebnis auf einer seperaten Seite des Gleichungssystems steht.

$$
\begin{aligned}
\sum_{k=1}^{4} a_u t_k^4 + b_u t_k^3 + c_u t_k^2 - t_k^2 u_k &= 0 \quad \Rightarrow \quad \sum_{k=1}^{4} a_u t_k^4 + b_u t_k^3 + c_u t_k^2 &= \sum_{k=1}^{4} t_k^2 u_k \\
\sum_{k=1}^{4} a_u t_k^3 + b_u t_k^2 + c_u t_k - t_k u_k &= 0 \quad \Rightarrow \quad \sum_{k=1}^{4} a_u t_k^3 + b_u t_k^2 + c_u t_k &= \sum_{k=1}^{4} t_k u_k \\
\sum_{k=1}^{4} a_u t_k^2 + b_u t_k + c_u - u_k &= 0 \quad \Rightarrow \quad \sum_{k=1}^{4} a_u t_k^2 + b_u t_k + c_u &= \sum_{k=1}^{4} u_k
\end{aligned} \tag{3.11}
$$

3.1.3 Problemangepasste Lösung des Gleichungssystems

Um das Gleichungssystem effizient für jeden Pixel lösen zu können, werden zuerst einige Anpassungen, die eine übersichtlichere Lösung gewährleisten, durchgeführt.

Zunächst werden die Vektoren $\vec{P_u}$ und $\vec{Z_k}$ wie folgt definiert:

$$
\vec{P_u} = \begin{pmatrix} a_u \\ b_u \\ c_u \\ 1 \end{pmatrix} \qquad \vec{Z_k} = \begin{pmatrix} {t_k}^2 \\ t_k \\ 1 \\ -u_k \end{pmatrix} \tag{3.12}
$$

Setzen wir diese eben definierten Vektoren aus Gleichung 3.12 in die Energiefunktion in Gleichung 3.9 ein, ergibt sich folgende durch die Vektoren ausgedrückte Gleichung:

$$
E(a_u, b_u, c_u) = \sum_{k=1}^{4} (\vec{P_u}^T * \vec{Z_k})^2 = \sum_{k=1}^{4} \vec{P_u}^T * \vec{Z_k} * \vec{Z_k}^T * \vec{P_u} \tag{3.13}
$$

Diese Gleichung kann dahingehend vereinfacht werden, dass das Summationszeichen in die Gleichung hineingezogen wird, da sowohl $\vec{P_u}$ als auch $\vec{P_u}^T$ Vektoren sind, die unabhängig vom Summationsindex k gegeben sind:

$$
E(a_u, b_u, c_u) = {\vec{P_u}}^T * \sum_{k=1}^{4} (\vec{Z_k} * {\vec{Z_k}}^T) * \vec{P_u} \tag{3.14}
$$

$\sum_{i=1}^{4}(\vec{Z}_i * \vec{Z}_i^{\,T})$ wird im Folgenden durch die Matrix J repräsentiert, die wie folgt aussieht:

$$
\begin{aligned}
J &:= \sum_{k=1}^{4}(\vec{Z}_k * \vec{Z}_k^{\,T}) \\
&= \sum_{k=1}^{4}\begin{pmatrix} t_k^4 & t_k^3 & t_k^2 & -t_k^2 u_k \\ t_k^3 & t_k^2 & t_k & -t_k * u_k \\ t_k^2 & t_k & 1 & -u_k \\ -t_k^2 u_k & -u_k t_k & -u_k & -u_k^2 \end{pmatrix} =: \begin{pmatrix} J_{u,11} & J_{u,12} & J_{u,13} & J_{u,14} \\ J_{u,21} & J_{u,22} & J_{u,23} & J_{u,24} \\ J_{u,31} & J_{u,32} & J_{u,33} & J_{u,34} \\ J_{u,41} & J_{u,42} & J_{u,43} & J_{u,44} \end{pmatrix}
\end{aligned} \tag{3.15}
$$

Folgende Eigenschaften der Matrix J sind nun feststellbar:

- J ist eine positiv semidefinite, symmetrische Matrix.
- Die Matrixeinträge von J stimmen mit den Koeffizenten vor den Variablen a_u, b_u und c_u überein.
- Die letzte Zeile der Matrix J ist in diesem Zusammenhang irrelevant, da sie keine Zeile des linearen Gleichungssystems repräsentiert.
- Die Einträge der Matrix J im ersten 3x3 Quadrat hängen lediglich vom Parameter t ab und können deshalb bereits im Voraus für alle Pixel bestimmt werden. Nur die letzte Spalte muss in jedem Pixel aktualisiert werden, da ein Zusammenhang mit den Flusskomponenten u bzw. v vorhanden ist und diese sich in jedem Pixel ändern.

Da die Matrixeinträge von J mit den Koeffizenten des Gleichungssystems der Ableitung vor den Variablen a_u, b_u und c_u übereinstimmen, lässt sich das Gleichungssystem der Ableitung unter Beachtung der Symmetrie der Matrix J auch wie folgt darstellen:

$$
\begin{aligned}
J_{u,11}a_u + J_{u,12}b_u + J_{u,13}c_u &= -J_{u,14} \\
J_{u,12}a_u + J_{u,22}b_u + J_{u,23}c_u &= -J_{u,24} \\
J_{u,13}a_u + J_{u,23}b_u + J_{u,33}c_u &= -J_{u,34}
\end{aligned} \tag{3.16}
$$

Dies entspricht einem allgemeinen Gleichungssystem der Form $A\vec{x} = \vec{b}$ mit Koeffizentenmatrix $A = J$, Lösungsvektor $\vec{x} = \begin{pmatrix} a_u \\ b_u \\ c_u \end{pmatrix}$ und Ergebnisvektor $\vec{b} = \begin{pmatrix} -J_{u,14} \\ -J_{u,24} \\ -J_{u,34} \end{pmatrix}$.

Effizente Lösung des Gleichungssystems

Zum Lösen eines derartigen Gleichungssystems können verschiedene Lösungsverfahren angewendet werden. Das bekannteste Lösungsverfahren ist das gaußsche Eliminierungsverfahren, welches im Normalfall auch einfach zu implementieren ist. Dies ist für die gegebene Problemstellung allerdings zu ineffizient, da für jede Trajektorie zwei Gleichungssysteme (jeweils eins in u- und v-Flussrichtung) gelöst werden müssen und hierfür ein hoher Aufwand

betrieben werden muss. Deshalb wird im Folgenden die Cramersche Regel erläutert, die auf der Berechnung von Determinanten basiert, und ebenfalls eine Methode zur Lösung von linearen Gleichungssystem darstellt. Die Regel von Sarrus ermöglicht eine besonders effiziente Lösung für $2x2$ und $3x3$ Matrizen, allerdings nicht für Matrizen höherer Ordnung. Das Schema der Regel von Sarrus wird daher im Anschluss an die Cramersche Regel erläutert.

Cramersche Regel

Damit die Cramersche Regel [Cra50] auf eine Matrix A angewendet werden kann, muss die Matrix A regulär und invertierbar sein. Eine äquivalente Bedingung hierzu ist, dass die Determinante der Matrix A ungleich 0 ist. Erfüllt eine Matrix diese Bedingung, ist ein Gleichungssystem, welches durch die Matrix A repräsentiert wird, eindeutig lösbar.

Mit Hilfe der Cramerschen Regel lässt sich der Lösungsvektor $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ eines Gleichungssystems $A * \vec{x} = \vec{b}$ mit $3x3$ Koeffizentenmatrix A sowie Ergebnisvektor $\vec{b}$ bestimmen.

$$\left(\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right) \tag{3.17}$$

Aus dem in Matrixschreibweise vorliegenden Gleichungssystem $A\vec{x} = \vec{b}$ (Gleichung 3.17) lässt sich mit der Cramerschen Regel die Variable x_i bestimmen, indem der Lösungsvektor b in Spalte i der Matrix A eingesetzt wird. Im Folgenden wird die neu entstandene Matrix als Matrix A_i bezeichnet. Für $i = 2$ ergibt sich entsprechend der obigen Definition folgende Matrix:

$$A_2 = \begin{pmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{pmatrix} \tag{3.18}$$

Anschließend erfolgt die Berechnung von x_i durch Division der Determinante der neu entstandenen Matrix A_i durch die Determinante der Ursprungsmatrix A:

$$x_i = \frac{\det(A_i)}{\det(A)} \tag{3.19}$$

Für das oben gegebene Beispiel mit $i = 2$ folgt durch Einsetzen entsprechend:

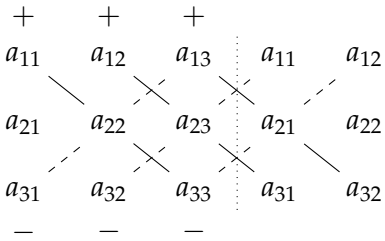
$$x_2 = \frac{\det(A_2)}{\det(A)} = \frac{\begin{vmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}}$$

Die Lösung kann nun aufgrund der zwei vorhandenen 3x3 Matrizen mit der Regel von Sarrus (Abschnitt 3.1.3) effizient berechnet werden.

Regel von Sarrus

Die Regel von Sarrus [Fis85], die die einfache Berechnung der Determinante einer 2x2 beziehungsweise passend zur Problemstellung einer 3x3 Matrix erlaubt, ergibt sich aus folgendem grafischen Schema einer Matrix A mit Koeffizenten a_{ij} mit Zeilenindex $i \in \{1, 2, 3\}$ und Spaltenindex $j \in \{1, 2, 3\}$. Sie ermöglicht eine einfache Lösung für die Cramersche Regel (Abschnitt 3.1.3), wenn diese auf ein Gleichungssystem mit einer Matrix A angewendet wird, deren Dimension maximal $3x3$ beträgt.

$$det(A) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} =$$



Daraus ergibt sich für die Determinante von A:

$$det(A) = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{31}a_{22}a_{13} - a_{32}a_{23}a_{11} - a_{33}a_{21}a_{12} \tag{3.20}$$

Bestimmung der Koeffizenten

Wendet man die Cramersche Regel auf die gegebene Problemstellung an, ergeben sich für die einzelnen Koeffizenten folgende Berechnungsvorschriften:

$$a_u = \frac{\det(J_1)}{\det(J)} = \frac{\begin{vmatrix} -J_{u,14} & J_{u,12} & J_{u,13} \\ -J_{u,24} & J_{u,22} & J_{u,23} \\ -J_{u,34} & J_{u,23} & J_{u,33} \end{vmatrix}}{\begin{vmatrix} J_{u,11} & J_{u,12} & J_{u,13} \\ J_{u,12} & J_{u,22} & J_{u,23} \\ J_{u,13} & J_{u,23} & J_{u,33} \end{vmatrix}}$$

$$b_u = \frac{\det(J_1)}{\det(J)} = \frac{\begin{vmatrix} J_{u,11} & -J_{u,14} & J_{u,13} \\ J_{u,12} & -J_{u,24} & J_{u,23} \\ J_{u,13} & -J_{u,34} & J_{u,33} \end{vmatrix}}{\begin{vmatrix} J_{u,11} & J_{u,12} & J_{u,13} \\ J_{u,12} & J_{u,22} & J_{u,23} \\ J_{u,13} & J_{u,23} & J_{u,33} \end{vmatrix}} \tag{3.21}$$

$$c_u = \frac{\det(J_1)}{\det(J)} = \frac{\begin{vmatrix} J_{u,11} & J_{u,12} & -J_{u,14} \\ J_{u,12} & J_{u,22} & -J_{u,24} \\ J_{u,13} & J_{u,23} & -J_{u,34} \end{vmatrix}}{\begin{vmatrix} J_{u,11} & J_{u,12} & J_{u,13} \\ J_{u,12} & J_{u,22} & J_{u,23} \\ J_{u,13} & J_{u,23} & J_{u,33} \end{vmatrix}}$$

Wendet man die Regel von Sarrus auf die einzelnen Matrizen an, ergeben sich zur Berechnung der Koeffizenten einfache Additions- beziehungsweise Subtraktionsrechnungen.

(3.22)

$$a_u = \frac{(-J_{u,14})J_{u,22}J_{u,33}+J_{u,12}J_{u,23}(-J_{u,14})+J_{u,13}(-J_{u,14})J_{u,23}-(-J_{u,14})J_{u,22}J_{u,13}-J_{u,23}J_{u,23}(-J_{u,14})-J_{u,33}(-J_{u,14})J_{u,12}}{J_{u,11}J_{u,22}J_{u,33}+J_{u,12}J_{u,23}J_{u,13}+J_{u,13}J_{u,12}J_{u,23}-J_{u,13}J_{u,22}J_{u,13}-J_{u,23}J_{u,23}J_{u,11}-J_{u,33}J_{u,12}J_{u,12}}$$

$$b_u = \frac{J_{u,11}-J_{(u,24)}J_{u,33}+(-J_{u,14})J_{u,23}J_{u,13}+J_{u,13}J_{u,12}(-J_{u,24})-J_{u,13}(-J_{u,24})J_{u,13}-(-J_{u,24})J_{u,23}J_{u,11}-J_{u,33}J_{u,12}(-J_{u,14})}{J_{u,11}J_{u,22}J_{u,33}+J_{u,12}J_{u,23}J_{u,13}+J_{u,13}J_{u,12}J_{u,23}-J_{u,13}J_{u,22}J_{u,13}-J_{u,23}J_{u,23}J_{u,11}-J_{u,33}J_{u,12}J_{u,12}}$$

$$c_u = \frac{J_{u,11}J_{u,22}(-J_{u,34})+J_{u,12}(-J_{u,24})J_{u,13}+(-J_{u,14})J_{u,12}J_{u,23}-J_{u,13}J_{u,22}(-J_{u,14})-J_{u,23}(-J_{u,24})J_{u,11}-(-J_{u,34})J_{u,12}J_{u,12}}{J_{u,11}J_{u,22}J_{u,33}+J_{u,12}J_{u,23}J_{u,13}+J_{u,13}J_{u,12}J_{u,23}-J_{u,13}J_{u,22}J_{u,13}-J_{u,23}J_{u,23}J_{u,11}-J_{u,33}J_{u,12}J_{u,12}}$$

Diese aufgestellten Formeln lösen das vorgegebene Gleichungssystem und stellen somit eine Lösung für die lineare Regression dar. Abschließend wird das Schwellwertverfahren für die Koeffizenten, wie im folgenden Abschnitt 3.2 beschrieben, durchgeführt. Eine Beispielklassifizierung, die anhand des in diesem Kapitel beschriebenen Vorgehens durchgeführt wurde, ist in Abbildung 3.1 zu sehen.

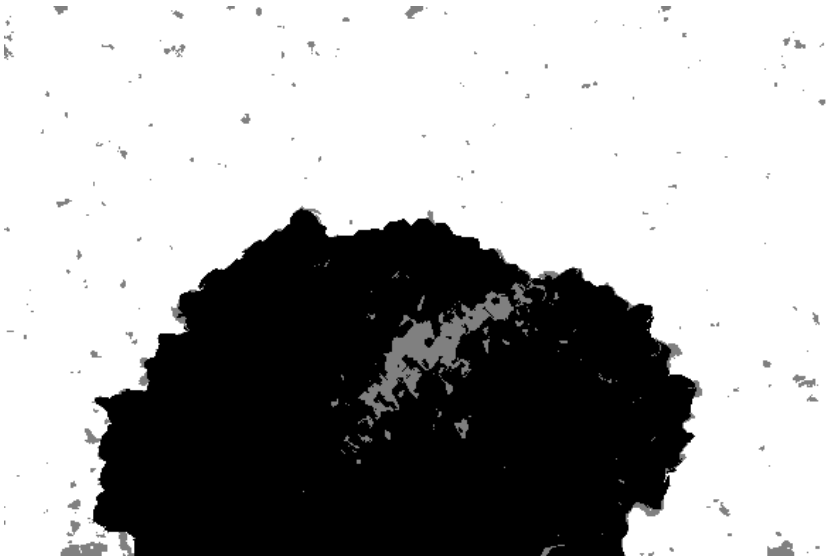


Abbildung 3.1: Trajektorienklassifizierung der Hydra-Sequenz (Unterabschnitt 2.3.3) mit polynomieller Regression vom Grad 2 und den Parametern $T_a = 0.1, T_b = 0.05$

3.2 Klassifizierung der Trajektorien

Dieses wie folgt beschriebene Schwellwertverfahren wird stets im Anschluss an eine mathematisch durchgeführte Methode ausgeführt, nachdem in jedem Pixel (i) die Koeffizenten $a_{u/v}, b_{u/v}$ und $c_{u/v}$ bestimmt worden sind.
Zuerst wird hierzu für jeden der beiden Koeffizienten a und b das Maximum aus den beiden Koeffizenten für die Flusskomponenten u und v am Pixel (i) ermittelt. Dies sorgt dafür, dass immer das Bewegungsmodell höherer Ordnung gewählt wird und damit eine trajektorielle Überregularisierung vermieden wird.

$$a(\vec{x}) = max(|a_u|, |a_v|) b(\vec{x}) = max(|b_u|, |b_v|)$$

Die endgültige Klassifizierung erfolgt anschließend über ein Schwellwertverfahren für die Koeffizenten $a(\vec{x})$ und $b(\vec{x})$. Es werden hierzu zwei Parameter T_a und T_b eingeführt, die für die Koeffizenten $a(\vec{x})$ respektive $b(\vec{x})$ zuständig sind. Die Klassifikation der Bewegungstrajektorien erfolgt nach den selben Regeln, die bereits im TC-Flow Verfahren [VBVZ11] eingeführt wurden, und läuft nach folgendem Schema ab:

Kriterium	**Klassifikation**
$a(\vec{x}) > T_a$	keine
$a(\vec{x}) \leq T_a \wedge b(\vec{x}) > T_b$	2.te Ordnung
$a(\vec{x}) \leq T_a \wedge b(\vec{x}) \leq T_b$	1.te Ordnung

3.3 Robuste Regression

Wie am Ende des Kapitels der Beschreibung der linearen Regression in Abbildung 3.1 zu sehen ist, weisen die durch das Verfahren der linearen Regression entstehenden Klassifikationskarten ein relativ starkes Rauschen auf. Dies kann auf das Vorhandensein einzelner Ausreißer zurückzuführen sein, die im Folgenden durch das Einführen des robusten Regressionsverfahrens eliminiert werden sollen.

Im Zuge der robusten Regression werden hierfür im Gegensatz zur linearen Regressionsanalyse den einzelnen Messwerten eine Gewichtungsfunktion zugeteilt. Durch diese Gewichtung werden insbesondere Ausreißer weniger stark berücksichtigt.

3.3.1 Allgemeine Vorgehensweise

Für jeden einzelnen Messpunkt zum Datenpunkt x_k wird eine Gewichtungsfunktion ψ eingeführt, während der grundsätzliche allgemeine Ansatz aus dem vorherigen Kapitel 3.1 beibehalten wird. Dementsprechend sieht die allgemeine Energiefunktion E folgendermaßen aus:

$$E(a,b,c) = \sum_{k=1}^{n} \psi((a * x_k^2 + b * x_k + c - y_k)^2) \tag{3.23}$$

Es wird nun die Variable s_k^2 für den inneren Teil der Energiefunktion eingeführt:

$$s_k^2 = (a * x_k^2 + b * x_k + c - y_k)^2 \qquad \Rightarrow E(a,b,c) = \sum_{k=1}^{n} \psi(s_k^2) \tag{3.24}$$

Bestimmt man analog zur linearen Regression die Ableitung, um das Minimum der Energiefunktion bestimmen zu können, erhält man folgende Gleichungen:

$$\begin{aligned} \frac{\partial E(a,b,c)}{\partial a} &= \sum_{k=1}^{n} 2\psi'(s_k^2)(ax_k^4 + bx_k^3 + cx_k^2 - x_k^2 y_k) \\ \frac{\partial E(a,b,c)}{\partial b} &= \sum_{k=1}^{n} 2\psi'(s_k^2)(ax_k^3 + bx_k^2 + cx_k - x_k y_k) \\ \frac{\partial E(a,b,c)}{\partial c} &= \sum_{k=1}^{n} 2\psi'(s_k^2)(ax_k^2 + bx_k + c - y_k) \end{aligned} \tag{3.25}$$

Im Zuge der linearen Regression ist keine Gewichtung erfolgt, weshalb für die ψ-Funktion folgendes gilt:

$$\psi(s^2) = s^2 \tag{3.26}$$

3.3.2 Problemangepasste Vorgehensweise

Nachfolgend wird wie bereits bei der linearen Regression die Vorgehensweise für die u-Flussrichtung betrachtet. Die Vorgehensweise für die Flussrichtung v ist analog und kann durch einfaches Ersetzen von u durch v in allen Variablen und Indizes erreicht werden.
Die Energiefunktion im Falle des zu lösenden Problemes sieht nun unter Einbeziehen der Gewichtungsfunktion ψ wie folgt aus:

$$E_u(a_u, b_u, c_u) = \sum_{k=1}^{n} \psi\left((a_u t_k^2 + b_u t_k + c_u - u_k)^2\right) \tag{3.27}$$

Betrachtet man nun analog zum linearen Fall die Voraussetzung, dass für ein Minimum die Ableitung gleich 0 sein muss, (also: $f'(x) = 0$ für eine beliebige Funktion f) ergeben sich daraus im Vergleich zur Ableitung in Gleichung 3.4 mit $s_k = (a_u t_k^2 + b_u t_k + c_u - u_k)$ folgende Ableitungen:

$$\begin{aligned}
\frac{\partial E(a_u, b_u, c_u)}{\partial a_u} &= \sum_{k=1}^{4} \psi'(s_k^2) * 2(a_u t_k^4 + b_u t_k^3 + c_u t_k^2 - t_k^2 u_k) \\
\frac{\partial E(a_u, b_u, c_u)}{\partial b_u} &= \sum_{k=1}^{4} \psi'(s_k^2) * 2(a_u t_k^3 + b_u t_k^2 + c_u t_k - t_k u_k) \\
\frac{\partial E(a_u, b_u, c_u)}{\partial c_u} &= \sum_{k=1}^{4} \psi'(s_k^2) * 2(a_u t_k^2 + b_u t_k + c_u - u_k)
\end{aligned} \tag{3.28}$$

Für die Ableitung der Gewichtungsfunktion mit Hilfe derer die einzelnen relevanten Gewichte zum Lösen des Gleichungssystems bestimmt werden, gilt im Falle der linearen Regression mit $\psi(s_k^2) = s_k^2$ also folgendes:

$$\psi'(s_k^2) = \frac{\partial s_k^2}{\partial \psi} = 1 \tag{3.29}$$

Aus diesem Grund schlägt sich die Gewichtungsfunktion in der linearen Regression nicht nieder.

Um die robuste Regression auf das Problem der Trajektorienklassifizierung anwenden zu können, ist es notwendig, die Gewichte für die einzelnen Messwerte mit Hilfe einer berechenbaren Funktion zu bestimmen. Im ersten Schritt wird die Ableitung der Gewichtungsfunktion ψ für alle Werte auf einen konstanten Wert ungleich 0 (normalerweise 1) gesetzt. In der Folge wird die erste Regressionskurve bestimmt, die der Lösung der linearen Regression entspricht. Während die lineare Regression nun bereits abgeschlossen ist, wird bei der robusten Regression anschließend der Abstand der einzelnen Messwerte zur berechneten Regressionskurve bestimmt. Dieser entspricht der zuvor eingeführten Variable s_k^2.

Anhand des Abstandes errechnet man die Gewichtung der einzelnen Datenpunkte für die folgenden Iterationen, die das oben beschriebene Vorgehen wiederholen. Da Ausreißer weniger stark gewichtet werden sollen, wird die Gewichtung mit der Charbonnier-Bestrafungsfunktion, bei der ein Kontrastparameter λ eingeführt wird, bestimmt. Der Term

$-2 * \lambda^2$ trägt dabei lediglich zur Normalisierung der Funktion sowie deren Ableitung bei. Der Parameter λ bestimmt, wie stark Ausreißer abgewertet werden: Große λ führen zu einer größeren Gewichtung, während kleinere λ zu größeren Bestrafungen und dementsprechend zu kleineren Gewichtungen für einzelne Terme führen:

$$\psi(s_k^2) = 2\lambda^2\sqrt{1+\frac{s_k^2}{\lambda^2}} - 2\lambda^2 \tag{3.30}$$

Bestimmt man nun die Ableitung dieser Gewichtungsfunktion, erhält man folgendes Ergebnis:

$$\psi'(s_k^2) = \frac{\delta\psi}{\delta s_k^2} = 2\lambda^2 \frac{1}{2\sqrt{1+\frac{s_k^2}{\lambda^2}}}\frac{1}{\lambda^2} = \frac{1}{\sqrt{1+\frac{s_k^2}{\lambda^2}}} \tag{3.31}$$

Betrachtet man diese Ableitungsfunktion, lassen sich einige Eigenschaften der Funktion erkennen:

- **streng monoton fallend**
  Dies entspricht der geforderten Eigenschaft, dass für größere Abweichungen von der berechneten Regressionskurve der relevante Datenpunkt eine geringere Gewichtung erhalten soll. Die strenge Monotonie ergibt sich aus der Tatsache, dass der Datenterm s_k^2 im Nenner des zu berechnenden Bruches unter einer Wurzel steht.
- **konvergent gegen 0**
  Dadurch, dass s_k^2 einen quadratischen Term darstellt, wird sichergestellt, dass die Ableitungsfunktion keine negativen Werte annehmen kann. Durch die strenge Monotonie ergibt sich für große s_k^2 die Konvergenz der ψ-Funktion gegen 0: $\lim\limits_{s_k^2 \to \infty} \psi(s^2) \to 0$.
- **positive Funktionswerte**
  Diese Eigenschaft ist eine offensichtliche Schlussfolgerung der beiden zuvor gezeigten Eigenschaften.

Die Ableitungsfunktion ergibt durch Einsetzen des Wertes von s_k^2 den Faktor, mit dem der berechnete Wert von s_k^2 in die Regressionsformel einfließt. Eingesetzt bedeutet das für die Ableitung folgendes Gleichungssystem:

$$\begin{aligned}
\sum_{k=1}^{4}\left(\frac{1}{\sqrt{1+\frac{s_k^2}{\lambda^2}}} * 2(a_u t_k^4 + b_u t_k^3 + c_u t_k^2 - t_k^2 u_k)\right) &= 0 \\
\sum_{k=1}^{4}\left(\frac{1}{\sqrt{1+\frac{s_k^2}{\lambda^2}}} * 2(a_u t_k^3 + b_u t_k^2 + c_u t_k - t_k u_k)\right) &= 0 \\
\sum_{k=1}^{4}\left(\frac{1}{\sqrt{1+\frac{s_k^2}{\lambda^2}}} * 2(a_u t_k^2 + b_u t_k^1 + c_u - u_k)\right) &= 0
\end{aligned} \tag{3.32}$$

Diese Gleichungen lassen sich durch das Einsetzen von $\psi_k = \frac{1}{\sqrt{1+\frac{s_k^2}{\lambda^2}}}$ weiter vereinfacht darstellen. Zudem lässt sich wie bei der linearen Regression der konstante Faktor 2 durch Division herauskürzen. Man erhält folgende Gleichungen:

$$\begin{aligned} \sum_{k=1}^{4} \psi_k * (a_u t_k^4 + b_u t_k^3 + c_u t_k^2 - t_k^2 u_k) &= 0 \\ \sum_{k=1}^{4} \psi_k * (a_u t_k^3 + b_u t_k^2 + c_u t_k - t_k u_k) &= 0 \\ \sum_{k=1}^{4} \psi_k * (a_u t_k^2 + b_u t_k^1 + c_u - u_k) &= 0 \end{aligned} \tag{3.33}$$

Schreibt man die Energiefunktion analog zur Energiefunktion in Gleichung 3.13 mithilfe der in Gleichung 3.12 definierten Vektoren $\vec{P_u}$ und $\vec{Z_i}$ um, erhält man folgende gewichtete Ableitung der Energiefunktion aus Gleichung 3.27 für die robuste Regression:

$$E(a_u, b_u, c_u) = \sum_{i=k}^{4} \psi_i * (P_u^T * Z_k)^2 = \sum_{k=1}^{4} \psi_k * ({P_u}^T * Z_k * {Z_k}^T * P_u) \tag{3.34}$$

Vereinfacht man diese Gleichung analog zur linearen Regression, und zieht die Faktoren ψ_k ebenfalls "nach innen" erhält man folgende Gleichung, aus der sich anschließend leicht die Matrix J auf die selbe Art und Weise wie bei der linearen Regression bestimmen lässt.

$$E(a_u, b_u, c_u) = {P_u}^T * (\sum_{k=1}^{4} \psi_k * Z_k * {Z_k}^T) * P_u \tag{3.35}$$

Die Matrix J repräsentiert dabei die Summenformel $\sum_{k=1}^{4} (\psi_k * Z_k * {Z_k}^T)$. Die einzelnen Einträge sehen folgendermaßen aus:

$$J := \sum_{k=1}^{4} \psi_k * (\vec{Z_k} * {\vec{Z_k}}^T) = \sum_{k=1}^{4} \psi_k * \begin{pmatrix} t_k^4 & t_k^3 & t_k^2 & -t_k^2 u_k \\ t_k^3 & t_k^2 & t_k & -t_k u_k \\ t_k^2 & t_k & 1 & -u_k \\ -t_k^2 u_k & -u_k t_k & -u_k & -u_k^2 \end{pmatrix} \tag{3.36}$$

Diese Matrix kann wie im vorangegangen Kapitel lineare Regression (Abschnitt 3.1.3) benutzt werden, um das Gleichungssystem mit Hilfe der Cramerschen Regel zu lösen. Im Gegensatz zur linearen Regression muss die Matrix J für jede zu berechnende Trajektorie aktualisiert werden, da es sich aufgrund der Gewichtungsfunktion ψ und deren Ableitung ψ' um ein nichtlineares Gleichungssystem handelt. Dieses Gleichungssystem kann durch eine sogenannte Fixpunktiteration gelöst werden, indem man eine Serie von linearen Gleichungssystemen löst. Dies entspricht einem iterativen Vorgehen, wenn man die erhaltenen Gewichtungen aus der Funktion ψ' aus dem vorherigen Iterationsschritt in der aktuellen Iteration benutzt. In diesem Falle startet man mit der Gewichtung $\psi'(s_k^2) = 1$ für jeden

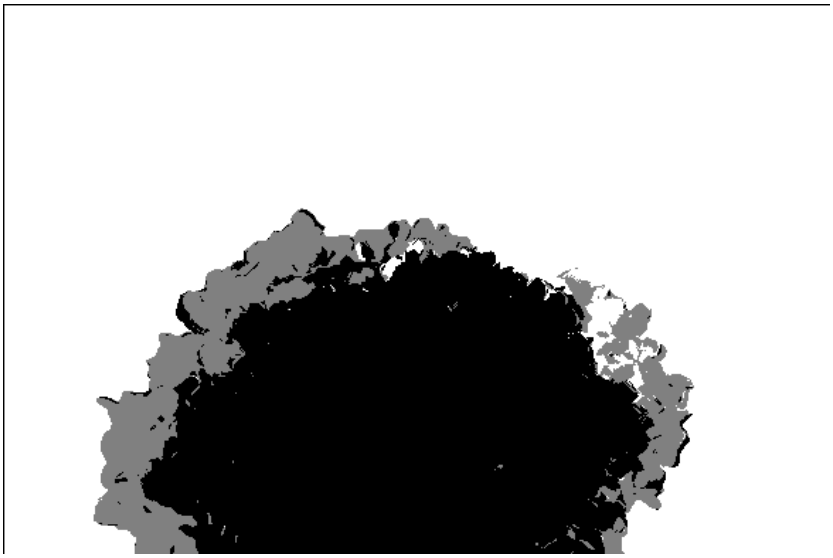


Abbildung 3.2: Klassifikationsmap für die Beispielsequenz Hydrangea

Datenterm s_k^2. Somit entspricht das Ergebnis der ersten Iteration dem Ergebnis der linearen Regression. Abschließend wird im Anschluss an die Berechnung der Koeffizienten das in Abschnitt 3.2 erläuterte Schwellwertverfahren durchgeführt. Wie in Abbildung 3.2 zu sehen ist, sind allerdings immer noch vereinzelte Ausreißer in der Klassifikationskarte erkennbar. Aus diesem Grund werden im folgenden Kapitel Methoden eingeführt, die den globalen Kontext, in dem sich eine Trajektorie befindet, in die Klassifizierungsberechnung mit einbeziehen.

4 Beschreibung globaler Verfahren zur Trajektorienklassifizierung

Im Gegensatz zu den beiden vorgestellten lokalen Verfahren, zum einen der linearen Regression (Abschnitt 3.1), zum anderen der robusten Regression (Abschnitt 3.3), bei denen die zu klassifizierende Trajektorie jeweils isoliert von ihrer Umgebung betrachtet wird, wollen wir nun globale Verfahren modellieren, bei denen die umgebenden Trajektorien ebenfalls berücksichtigt werden.

Dies macht vor Allem vor dem Hintergrund Sinn, dass sich in einem Bild meist mehrere benachbarte Pixel, die zusammen ein Objekt in der realen Welt repräsentieren(vgl. Beispielsequenzen in Unterabschnitt 2.3.3 und Unterabschnitt 2.3.4), in einer ähnlichen Form verändern, und damit die Bewegung des repräsentierten Objekts widerspiegeln.
Zudem zeigt die robuste Regression nicht die erwünschte Wirkung, da in der Klassifikationskarte in Abbildung 3.2 immer noch einzelne Ausreißer vorhanden sind.

4.1 Variationaler Ansatz

Im Fall eines variationalen Ansatzes wird versucht mit einer Funktion $u(x, y)$ beziehungsweise $v(x, y)$ die Datenmesspunkte des optischen Flusses zu approximieren. Im Gegensatz zur linearen Regression werden die Koeffizienten a, b und c nicht nur lokal für jeden Pixel seperat betrachtet, sondern gleichzeitig global für alle Pixel bestimmt.

Hierzu wird bei dem allgemeinen variationalen Ansatz die bislang lokale Energiefunktion $E(a, b, c) = \int_\Omega F(x, a, b, c)dx$ um einen Glattheitsterm erweitert. Dieser Glattheitsterm besteht darin, dass der Wert von $a_u(a_v)$, $b_u(b_v)$ und $c_u(c_v)$ an der Stelle $u(i)$ $(v(i))$ im Vergleich zu der Umgebung sich möglichst wenig ändert. Dies ist genau dann der Fall, wenn der Gradient des jeweiligen Parameters $(\nabla a_{u/v}, \nabla b_{u/v}, \nabla c_{u/v})$ gering ist. Keine Änderung besteht genau dann, wenn $\nabla x = 0$ für $x \in \{a_u, a_v, b_u, b_v, c_u, c_v\}$. Diese Vorüberlegung führt zu den beiden folgenden zweidimensionalen, problemspezifischen, kontinuierlichen Energiefunktionen für E:

$$\begin{aligned} E(a_u, b_u, c_u) &= \int_\Omega F(x, y, a_u, b_u, c_u, \nabla a_u, \nabla b_u, \nabla c_u)dxdy \\ E(a_v, b_v, c_v) &= \int_\Omega F(x, y, a_v, b_v, c_v, \nabla a_v, \nabla b_v, \nabla c_v)dxdy \end{aligned} \tag{4.1}$$

Im zweidimensionalen Fall ergibt sich für die Ableitung der Gradient, der geometrisch interpretiert einen Vektor darstellt, der in die Richtung der größten Steigung zeigt. Die lokale

Komponente wird dabei aus dem zuvor besprochenen Regressionskapitel übernommen. Der globalen Komponente mit den drei Gradienten für a, b und c wird ein Gewichtungsparameter α vorangestellt, um die Abhängigkeit von der Umgebung verändern zu können. Damit die Darstellung des Problems einfacher gestaltet werden kann, wird im weiteren Verlauf auf die zweidimensionale Notation verzichtet und auf eine eindimensionale Darstellung zurückgegriffen. Der aktuelle Pixel, der betrachtet wird, befindet sich hierbei an Stelle (i), während die Zeitschritte den Index k zugewiesen bekommen.

$$
\begin{aligned}
E(a_u, b_u, c_u) &= \int_\Omega \sum_{k=1}^{4} (a_u t_k^2 + b_u t_k + c_u - u_k)^2 + \alpha(|\nabla a_u|^2 + |\nabla b_u|^2 + |\nabla c_u|^2) dxdy \\
E(a_v, b_v, c_v) &= \int_\Omega \sum_{k=1}^{4} (a_v t_k^2 + b_u t_k + c_u - u_k)^2 + \alpha(|\nabla a_v|^2 + |\nabla b_v|^2 + |\nabla c_v|^2) dxdy
\end{aligned} \tag{4.2}
$$

Um das Minimum dieser Energiefunktion bestimmen zu können, benötigt man analog zum lokalen Verfahren der (robusten) Regression die Ableitung der Energiefunktion, welche gleich Null gesetzt wird und die im globalen Fall ausgedrückt durch Euler-Lagrange Gleichungen folgendermaßen aussieht:

$$
\begin{aligned}
0 &= F_{a_u} - \frac{\partial}{\partial x} F_{[a_u]_x} - \frac{\partial}{\partial y} F_{[a_u]_y} \\
0 &= F_{b_u} - \frac{\partial}{\partial x} F_{[b_u]_x} - \frac{\partial}{\partial y} F_{[b_u]_y} \\
0 &= F_{c_u} - \frac{\partial}{\partial x} F_{[c_u]_x} - \frac{\partial}{\partial y} F_{[c_u]_y} \\
0 &= F_{a_v} - \frac{\partial}{\partial x} F_{[a_v]_x} - \frac{\partial}{\partial y} F_{[a_v]_y} \\
0 &= F_{b_v} - \frac{\partial}{\partial x} F_{[b_v]_x} - \frac{\partial}{\partial y} F_{[b_v]_y} \\
0 &= F_{c_v} - \frac{\partial}{\partial x} F_{[c_v]_x} - \frac{\partial}{\partial y} F_{[c_v]_y}
\end{aligned} \tag{4.3}
$$

Für die einzelnen Euler-Lagrange-Funktionale gelten dabei folgende Werte, die im Folgenden für a_u aufgelistet sind und ebenso für die anderen Variablen b_u, c_u, a_v, b_v, c_v bestimmt werden können, in der problemspezifischen Lösung:

$$
\begin{aligned}
F_{a_u} &= \sum_{k=1}^{4} (a_u t_k^4 \quad + b_u t_k^3 + \quad c_u t_k^2 - u_k t_k^2) \\
-\frac{\partial}{\partial x} F_{[a_u]_x} &= -\alpha [a_u]_{xx} \\
-\frac{\partial}{\partial y} F_{[a_u]_y} &= -\alpha [a_u]_{yy}
\end{aligned} \tag{4.4}
$$

Die erste Gleichung entspricht dabei der bekannten lokalen Komponente aus der Regression, während die zweite und dritte Gleichung die Ableitung der Gradienten darstellen. Es ergibt sich also folgendes Gleichungssystem, das minimiert werden soll:

$$
\begin{aligned}
\sum_{k=1}^{4}(a_u t_k^4 &+ b_u t_k^3 + c_u t_k^2 - u_k t_k^2) - \alpha[a_u]_{xx} - \alpha[a_u]_{yy} &= 0\\
\sum_{k=1}^{4}(a_u t_k^3 &+ b_u t_k^2 + c_u t_k - u_k t_k) - \alpha[b_u]_{xx} - \alpha[b_u]_{yy} &= 0\\
\sum_{k=1}^{4}(a_u t_k^2 &+ b_u t_k + c_u - u_k) - \alpha[c_u]_{xx} - \alpha[c_u]_{yy} &= 0\\
\sum_{k=1}^{4}(a_v t_k^4 &+ b_v t_k^3 + c_v t_k^2 - v_k t_k^2) - \alpha[a_v]_{xx} - \alpha[a_v]_{yy} &= 0\\
\sum_{k=1}^{4}(a_v t_k^3 &+ b_v t_k^2 + c_v t_k - v_k t_k) - \alpha[b_v]_{xx} - \alpha[b_v]_{yy} &= 0\\
\sum_{k=1}^{4}(a_v t_k^2 &+ b_v t_k + c_v - v_k) - \alpha[c_v]_{xx} - \alpha[c_v]_{yy} &= 0
\end{aligned}
\tag{4.5}
$$

Die in jeder Gleichung vorkommenden Faktoren $-\alpha * a_{u,xx} - \alpha * a_{u,yy}$ (hier für die Variable a_u) können zusammengefasst und durch den Laplace-Operator ausgedrückt werden: $-\alpha * a_{u,xx} - \alpha * a_{u,yy} =: -\alpha * \triangle a_u$.

$$
\begin{aligned}
\sum_{k=1}^{4}(a_u t_k^4 &+ b_u t_k^3 + c_u t_k^2 - u_k t_k^2) - \alpha \triangle a_u = 0\\
\sum_{k=1}^{4}(a_u t_k^3 &+ b_u t_k^2 + c_u t_k - u_k t_k) - \alpha \triangle b_u = 0\\
\sum_{k=1}^{4}(a_u t_k^2 &+ b_u t_k + c_u - u_k) - \alpha \triangle c_u = 0\\
\sum_{k=1}^{4}(a_v t_k^4 &+ b_v t_k^3 + c_v t_k^2 - v_k t_k^2) - \alpha \triangle a_v = 0\\
\sum_{k=1}^{4}(a_v t_k^3 &+ b_v t_k^2 + c_v t_k - v_k t_k) - \alpha \triangle b_v = 0\\
\sum_{k=1}^{4}(a_v t_k^2 &+ b_v t_k + c_v - v_k) - \alpha \triangle c_v = 0
\end{aligned}
\tag{4.6}
$$

Für Randpunkte gelten reflektierende Neumann Randbedingungen, sodass kein Einfluss auf die Berechnung genommen wird:

$$n^T \nabla a_u = 0 \tag{4.7}$$

Da die Ableitung in diskretem Kontext nicht durch kontinuierliche Verfahren ausgedrückt werden kann, erfolgt eine Diskretisierung der Gleichung. Ein Beispiel, auf welche Art und Weise dies möglich ist und bei dem in dieser Arbeit bearbeiteten Problem umgesetzt wird, wird im nächsten Unterkapitel vorgestellt.

4.1.1 Diskretisierung

Diskretisiert man diese Energiefunktion auf die im Problem vorhandenen vier Messpunkte, ergibt sich neben der bereits aus der Regression bekannten, diskretisierten Gleichung der lokalen Komponente (ausgedrückt mit Hilfe der Matrix J) noch die Fragestellung, wie der Laplace-Operator diskret bestimmt werden kann.

Hierzu wird im zweidimensionalen Raum, der in der Problemstellung gegeben ist, folgende Approximationsmaske für den Laplace-Operator angewendet, wobei h_x und h_y Gewichte für Nachbarpunkte in x bzw. y-Richtung darstellen. Üblicherweise wird sowohl h_x als auch h_y der Wert 1 zugeteilt.

	$\frac{1}{h_y}$	
$\frac{1}{h_x}$	$-2(h_x + h_y)$	$\frac{1}{h_x}$
	$\frac{1}{h_y}$	

Da die Nachbarfunktion die in Gleichung 4.7 festgelegten Randbedingungen beachten muss, darf sie einen Punkt, der sich außerhalb des Flussfeldes (Ω) befindet und Nachbar eines Punktes (i) ist, nicht als Nachbarn zurückgeben, sondern dieser muss in der Berechnung unberücksichtigt bleiben. Somit haben Punkte entlang einer Kante drei, Eckpunkte lediglich zwei Nachbarn. Dies bedeutet konkret für die Berechnung des Laplace-Operators folgende Berechungsvorschrift, die beispielhaft für den Koeffizient a_u am Punkt i, also $[a_u]_i$, aufgestellt wird:

$$[\triangle\, a_u]_i \approx \sum_{l \in x,y} \sum_{j \in N_l(i)} \frac{[a_u]_j - [a_u]_i}{h_l} \tag{4.8}$$

Zu beachten ist hierbei die Terminologie der Nachbarfunktion N_x beziehungsweise N_y. Diese gibt für den jeweiligen Pixel an der Stelle i für N_x die Nachbarn in x-Richtung sowie für N_y die Nachbarn in y-Richtung des Pixels zurück. In einem zweidimensionalen Bild kann die Anzahl zwischen insgesamt vier Nachbarn und zwei Nachbarn an den Ecken des Bildes variieren, was bei der Implementierung berücksichtigt werden muss.

Führt man die Approximation des Laplaceoperators auch auf den anderen Koeffizienten aus, so erhält man folgendes zu lösendes Gleichungssystem im Punkt (i), bei dem zusätzlich für

die lokale Komponente durch die Matrix J, die bereits in der linearen/robusten Regression eingeführt wurde, ausgedrückt wird:

$$
\begin{aligned}
0 &= [a_u]_i[J_{u,11}]_i + [J_{u,12}]_i[b_u]_i + [J_{u,13}]_i[c_u]_i + [J_{u,14}]_i - \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{[a_u]_j - [a_u]_i}{h_l} \\
0 &= [b_u]_i[J_{u,22}]_i + [J_{u,12}]_i[a_u]_i + [J_{u,23}]_i[c_u]_i + [J_{u,24}]_i - \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{[b_u]_j - [b_u]_i}{h_l} \\
0 &= [c_u]_i[J_{u,33}]_i + [J_{u,13}]_i[a_u]_i + [J_{u,23}]_i[b_u]_i + [J_{u,34}]_i - \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{[c_u]_j - [c_u]_i}{h_l} \\
0 &= [a_v]_i[J_{v,11}]_i + [J_{v,12}]_i[b_v]_i + [J_{v,13}]_i[c_v]_i + [J_{v,14}]_i - \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{[a_v]_j - [a_v]_i}{h_l} \\
0 &= [b_v]_i[J_{v,22}]_i + [J_{v,12}]_i[a_v]_i + [J_{v,23}]_i[c_v]_i + [J_{v,24}]_i - \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{[b_v]_j - [a_v]_i}{h_l} \\
0 &= [c_v]_i[J_{v,33}]_i + [J_{v,13}]_i[a_v]_i + [J_{v,23}]_i[b_v]_i + [J_{v,34}]_i - \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{[c_v]_j - [c_v]_i}{h_l}
\end{aligned}
\tag{4.9}
$$

4.1.2 Lösung des Gleichungssystems

Aufgrund der Abhängigkeit der Lösung am Pixel (i) von globalen Komponenten ist eine lokale Lösung des Gleichungssystems unmöglich. Vielmehr muss auf numerische Methoden zur Approximation der Lösung zurückgegriffen werden. In den folgenden Unterkapiteln werden mit dem Gauß-Seidel Verfahren und darauf aufbauend mit dem SOR (Successive Overrelaxation) Verfahren zwei Algorithmen vorgestellt, die diese Art von Gleichungssystemen lösen können, und im Rahmen dieser Bachelorarbeit auch eingesetzt werden.

Gauß-Seidel Verfahren

Das Gauß-Seidel Verfahren [Bar94] ist ein numerischer Algorithmus, der eine Lösung von linearen Gleichungssystemen iterativ berechnet.

Der allgemeine Iterationsschritt des Gauß-Seidel Verfahrens mit gegebenem Nährungsvektor $x^{(m)}$ und Faktoren $a_{11}...a_{1n}...a_{n1}...a_{nn}$ sowie Lösungsvektor $b_1...b_n$ heißt:

$$
x_k^{(m+1)} := \frac{1}{a_{kk}}(b_k - \sum_{i=1}^{k-1} a_{ki}x_i^{(m+1)} - \sum_{i=k+1}^{n} a_{ki}x_i^m) \tag{4.10}
$$

Zu beachten ist hierbei, dass auch die Variablen iterativ durchlaufen werden, sodass zuest die Variable x_k berechnet wird, ehe x_{k+1} berechnet werden kann. Dies ist offensichtlich ein Nachteil des Gauß-Seidel Verfahrens gegenüber dem Jacobi-Verfahren [Bar94], welches auch

im Bereich des Parallel Computing genutzt werden kann. Demgegenüber konvergiert das Gauß-Seidel Verfahren schneller als das Jacobi-Verfahren.

Wird der allgemeine Iterationsschritt auf das Gleichungssystem in Gleichung 4.5 beziehungsweise 4.6 angewendet, erhält man für die Koeffizienten a_u, b_u, c_u, a_v, b_v und c_v die folgenden Gleichungen für den Pixel (i):

$$
\begin{aligned}
[a_u]_i^{m+1} &= \frac{\left(-[J_{u,12}]_i[b_u]_i^m - [J_{u,13}]_i[c_u]_i^m - [J_{u,14}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{[a_u]_j^m}{h_l}\right)}{[J_{u,11}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{1}{h_l}} \\
[b_u]_i^{m+1} &= \frac{\left(-[J_{u,12}]_i[a_u]_i^m - [J_{u,23}]_i[c_u]_i^m - [J_{u,24}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{[b_u]_j^m}{h_l}\right)}{[J_{u,22}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{1}{h_l}} \\
[c_u]_i^{m+1} &= \frac{\left(-[J_{u,13}]_i[a_u]_i^m - [J_{u,23}]_i[b_u]_i^m - [J_{u,34}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{[c_u]_j^m}{h_l}\right)}{[J_{u,33}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{1}{h_l}} \\
[a_v]_i^{m+1} &= \frac{\left(-[J_{v,12}]_i[b_v]_i^m - [J_{v,13}]_i[c_v]_i^m - [J_{v,14}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{[a_v]_j^m}{h_l}\right)}{[J_{v,11}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{1}{h_l}} \\
[b_v]_i^{m+1} &= \frac{\left(-[J_{v,12}]_i[a_v]_i^m - [J_{v,23}]_i[c_v]_i^m - [J_{v,24}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{[b_v]_j^m}{h_l}\right)}{[J_{v,22}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{1}{h_l}} \\
[c_v]_i^{m+1} &= \frac{\left(-[J_{v,13}]_i[a_v]_i^m - [J_{v,23}]_i[b_v]_i^m - [J_{v,34}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{[c_v]_j^m}{h_l}\right)}{[J_{v,33}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{1}{h_l}}
\end{aligned}
\tag{4.11}
$$

Abbruchkriterium
Da der Gauß-Seidel Algorithmus ein iterativer Algorithmus ist, der für unterschiedliche Gleichungssysteme unterschiedlich schnell konvergiert, benötigt man ein Abbruchkriterium für den Algorithmus. Ansonsten kann es durch zu wenig Iterationen zu schlechten nährungsweisen Lösungen kommen, die für die Problemstellung wenig plausibel sind und zudem einen zu großen Restfehler aufweisen. Deshalb wird das globale Residual eingeführt, das die Residuuen (=euklidische Norm des Fehlers zur exakten Lösung) der einzelnen Gleichungen zusammenaddiert. Die Residuuen der einzelnen Gleichungen erhält man, indem in das

Gleichungssystem in Gleichung 4.6 eine zusätzliche Variable $res_{[x]_i}$ eingeführt wird.i steht dabei für den Pixel, an dem man sich momentan befindet, und $x \in \{a_u, a_v, b_u, b_v, c_u, c_v\}$ für den jeweiligen Koeffizienten, der aktuell betrachtet wird. Anschließend wird das Gleichungssystem nicht mehr nach den Koeffizienten, sondern nach den Residuuen umgestellt.

$$
\begin{aligned}
\sum_{k=1}^{4}(a_u t_k^4 + \quad b_u t_k^3 + c_u t_k^2 \quad - u_k t_k^2) + res_{a_u} - \alpha \triangle a_u = 0 \\
\sum_{k=1}^{4}(a_u t_k^3 + \quad b_u t_k^2 + c_u t_k \quad - u_k t_k) + res_{b_u} - \alpha \triangle b_u = 0 \\
\sum_{k=1}^{4}(a_u t_k^2 + \quad b_u t_k + c_u \qquad - u_k) + res_{c_u} - \alpha \triangle c_u = 0 \\
\sum_{k=1}^{4}(a_v t_k^4 + \quad b_v t_k^3 + c_v t_k^2 \quad - v_k t_k^2) + res_{a_v} - \alpha \triangle a_v = 0 \\
\sum_{k=1}^{4}(a_v t_k^3 + \quad b_v t_k^2 + c_v t_k \quad - v_k t_k) + res_{b_v} - \alpha \triangle b_v = 0 \\
\sum_{k=1}^{4}(a_v t_k^2 + \quad b_v t_k + c_v \qquad - v_k) + res_{c_v} - \alpha \triangle c_v = 0
\end{aligned}
\tag{4.12}
$$

Durch Umstellen der Gleichungen und anschließender Diskretisierung ergeben sich folgende Berechnungsvorschriften für das Residuum des Gleichungssystems aus Gleichung 4.11:

(4.13)

$$
\begin{aligned}
res^{(m)}_{[a_u]_i} &= -[J_{u,11}]_i[a_u]_i^m - [J_{u,12}]_i[b_u]_i^m - [J_{u,13}]_i[c_u]_i^m - [J_{u,14}]_i + \alpha \sum_{l \in x,y} \sum_{k \in N_l(i)} \frac{[a_u]_j^m - [a_u]_i^m}{h_l} \\
res^{(m)}_{[b_u]_i} &= -[J_{u,22}]_i[a_u]_i^m - [J_{u,12}]_i[a_u]_i^m - [J_{u,23}]_i[c_u]_i^m - [J_{u,24}]_i + \alpha \sum_{l \in x,y} \sum_{k \in N_l(i)} \frac{[b_u]_j^m - [b_u]_i^m}{h_l} \\
res^{(m)}_{[c_u]_i} &= -[J_{u,33}]_i[a_u]_i^m - [J_{u,13}]_i[a_u]_i^m - [J_{u,23}]_i[b_u]_i^m - [J_{u,34}]_i + \alpha \sum_{l \in x,y} \sum_{k \in N_l(i)} \frac{[c_u]_j^m - [c_u]_i^m}{h_l} \\
res^{(m)}_{[a_v]_i} &= -[J_{v,11}]_i[a_v]_i^m - [J_{v,12}]_i[b_v]_i^m - [J_{v,13}]_i[c_v]_i^m - [J_{v,14}]_i + \alpha \sum_{l \in x,y} \sum_{k \in N_l(i)} \frac{[a_v]_j^m - [a_v]_i^m}{h_l} \\
res^{(m)}_{[b_v]_i} &= -[J_{v,22}]_i[a_v]_i^m - [J_{v,12}]_i[a_v]_i^m - [J_{v,23}]_i[c_v]_i^m - [J_{v,24}]_i + \alpha \sum_{l \in x,y} \sum_{k \in N_l(i)} \frac{[b_v]_j^m - [b_v]_i^m}{h_l} \\
res^{(m)}_{[c_v]_i} &= -[J_{v,33}]_i[a_v]_i^m - [J_{v,13}]_i[a_v]_i^m - [J_{v,23}]_i[b_v]_i^m - [J_{v,34}]_i + \alpha \sum_{l \in x,y} \sum_{k \in N_l(i)} \frac{[c_v]_j^m - [c_v]_i^m}{h_l}
\end{aligned}
$$

Für das globale Residuum in der Iteration m gilt dann:

$$(4.14)\quad res^{(m)} = \sum_{i\in\Omega} \sqrt{\left(res^{(m)}_{[a_u]_i})^2 + res^{(m)}_{[b_u]_i})^2 + (res^{(m)}_{[c_u]_i})^2 + (res^{(m)}_{[a_v]_i})^2 + (res^{(m)}_{[b_v]_i})^2 + (res^{(m)}_{[c_v]_i})^2\right)}$$

Als Abbruchkriterium dient der Vergleich, ob das Residuum in der aktuellen Iteration im Vergleich zum Residuum zu Beginn der Berechnung (res^0) genau genug ist. Hierzu wird ein Parameter ϵ eingeführt, der zu Beginn festgelegt wird und angibt, ab welchem Bruchteil des ursprünglichen Residuums eine Lösung als genau genug gilt. Normalerweise besitzt dieser Parameter einen Wert im Bereich zwischen 10^{-5} und 10^{-3}.

$$(4.15)\quad \frac{res^{(m)}}{res^{(0)}} < \epsilon \quad \text{(Abbruchkriterium)}$$

Startbedingung
Die Startbedingungen für den Gauß-Seidel Algorithmus sind für positiv semidefinite Systemmatrizen für dessen Konvergenz unerheblich. Das heißt, der Gauß-Seidel Algorithmus konvergiert in für solche Matrizen für jede beliebige Startbelegung der Variablen $a_{u/v}, b_{u/v}$ und $c_{u/v}$ an jedem Pixel (i).

Als Startbelegung wird im in dieser Bachelorarbeit beschriebenen Fall folgende Belegung gewählt, sodass keine Vorberechnung für die Koeffizienten erfolgen muss:

$$(4.16)\quad \vec{u}_i = \vec{v}_i = \begin{pmatrix} a_u \\ b_u \\ c_u \end{pmatrix}_i = \begin{pmatrix} a_v \\ b_v \\ c_v \end{pmatrix}_i = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \text{ für alle } i \in \Omega$$

SOR (Successive Overrelaxation)

Neben dem Gauß-Seidel Algorithmus betrachten wir mit dem SOR-Algorithmus [Bar94] ein zweites Verfahren, welches bei "richtiger" heuristischer Parameterauswahl schneller konvergiert als der zuvor besprochene Gauß-Seidel Algorithmus und gleichzeitig auf dessen allgemeiner Vorgehensweise aufbaut.

Iterationsschritt
Im Vergleich mit dem Gauß-Seidel Algorithmus besteht im Iterationsschritt der einzige Unterschied. Es wird ein Gewichtungsparameter ω eingeführt, der die entsprechenden Komponenten "überrelaxiert", das heißt im Vergleich zum Gauß-Seidel Verfahren auch "über das Ziel hinaus schießen" kann. Der Iterationsschritt sieht für das Successive Overrelaxation Verfahren wie folgt aus:

$$(4.17)\quad x_k^{(m+1)} = (1-\omega)x_k^{(m)} + \frac{\omega}{a_{kk}}\left(b_k - \sum_{i>k} a_{ki}x_i^{(m)} - \sum_{i<k} a_{ki}x_i^{(m+1)}\right), \quad k = 1, 2, \ldots, n.$$

Dies entspricht unter Einbeziehung des Gauß-Seidel Iterationsschritt $[x_k^m]_{GS}$ folgender Iterationsvorschrift:

$$x_k^{(m+1)} = (1-\omega)x_k^{(m)} + \omega[x_k^m]_{GS} \quad , \quad k = 1, 2, \ldots, n. \tag{4.18}$$

Angewandt auf unser Gleichungssystem bedeutet dies, dass sich die Berechnungsvorschriften der einzelnen Variablen im Pixel i folgendermaßen ändern. Zugleich wird $\overline{\omega}$ wie folgt definiert: $\overline{\omega} = 1 - \omega$.

$$\begin{aligned}
[a_u^{m+1}]_i &= \overline{\omega}[a_u]_i^m + \frac{\omega\left(-[J_{u,12}]_i[b_u]_i^m - [J_{u,13}]_i[c_u]_i^m - [J_{u,14}]_i + \alpha\sum_{l\in x,y}\sum_{j\in N_l(i)}\frac{[c_u]_j}{h_l}\right)}{[J_{u,11}]_i + \sum\limits_{l\in x,y}\sum\limits_{j\in N_l(i)}\frac{1}{h_l}} \\
[b_u^{m+1}]_i &= \overline{\omega}[b_u]_i^m + \frac{\omega\left(-[J_{u,12}]_i[a_u]_i^m - [J_{u,23}]_i[c_u]_i^m - [J_{u,24}]_i + \alpha\sum_{l\in x,y}\sum_{j\in N_l(i)}\frac{[c_u]_j}{h_l}\right)}{[J_{u,22}]_i + \sum\limits_{l\in x,y}\sum\limits_{j\in N_l(i)}\frac{1}{h_l}} \\
[c_u^{m+1}]_i &= \overline{\omega}[c_u]_i^m + \frac{\omega\left(-[J_{u,13}]_i[a_u]_i^m - [J_{u,23}]_i[b_u]_i^m - [J_{u,34}]_i + \alpha\sum_{l\in x,y}\sum_{j\in N_l(i)}\frac{[c_u]_j}{h_l}\right)}{[J_{u,33}]_i + \sum\limits_{l\in x,y}\sum\limits_{j\in N_l(i)}\frac{1}{h_l}} \\
[a_v^{m+1}]_i &= \overline{\omega}[a_v]_i^m + \frac{\omega\left(-[J_{v,12}]_i[b_v]_i^m - [J_{v,13}]_i[c_v]_i^m - [J_{v,14}]_i + \alpha\sum_{l\in x,y}\sum_{j\in N_l(i)}\frac{[c_u]_j}{h_l}\right)}{[J_{v,11}]_i + \sum\limits_{l\in x,y}\sum\limits_{j\in N_l(i)}\frac{1}{h_l}} \\
[b_v^{m+1}]_i &= \overline{\omega}[b_v]_i^m + \frac{\omega\left(-[J_{v,12}]_i[a_v]_i^m - [J_{v,23}]_i[c_v]_i^m - [J_{v,24}]_i + \alpha\sum_{l\in x,y}\sum_{j\in N_l(i)}\frac{[c_u]_j}{h_l}\right)}{[J_{v,22}]_i + \sum\limits_{l\in x,y}\sum\limits_{j\in N_l(i)}\frac{1}{h_l}} \\
[c_v^{m+1}]_i &= \overline{\omega}[c_v]_i^m + \frac{\omega\left(-[J_{v,13}]_i[a_v]_i^m - [J_{v,23}]_i[b_v]_i^m - [J_{v,34}]_i + \alpha\sum_{l\in x,y}\sum_{j\in N_l(i)}\frac{[c_u]_j}{h_l}\right)}{[J_{v,33}]_i + \sum\limits_{l\in x,y}\sum\limits_{j\in N_l(i)}\frac{1}{h_l}}
\end{aligned} \tag{4.19}$$

Alle anderen beim Gauß-Seidel Verfahren besprochenen Eigenschaften treffen auch auf das SOR-Verfahren zu. Insbesondere stellt man fest, dass beim Einsetzen von $\omega = 1$ der berechnete Wert aus der vorherigen Iteration mit dem Faktor 0 multipliziert wird und somit wegfällt. Man erhält damit wieder den altbekannten Iterationsschritt aus dem Gauß-Seidel Verfahren. Folglich ist das Gauß-Seidel Verfahren eine spezielle Variante des SOR-Verfahrens, in der ω auf 1 gesetzt wird.

4.2 Variationaler Ansatz mit Gewichtung der Parameter

In diesem Abschnitt wird das aus dem vorigen Kapitel bekannte Energiefunktional $E(a_u, b_u, c_u) = \int_\Omega (a_u t_k^2 + b_u t_k + c_u - u_k)^2 + \alpha(|\nabla a_u|^2 + |\nabla b_u|^2 + |\nabla c_u|^2) dxdy$ robust "gemacht", indem Bestrafungsterme sowohl für die lokale als auch die globalen Komponenten eingeführt werden. Im Folgenden werden dabei zwei verschiedene Varianten eingeführt, die folgendermaßen lauten:

(4.20a)

$$E(a_u, b_u, c_u) = \int_\Omega \sum_{k=1}^{4} \psi_l(a_u t_k^2 + b_u t_k + c_u - u_k)^2 + \alpha \psi_g(|\nabla a_u|^2 + |\nabla b_u|^2 + |\nabla c_u|^2) dxdy$$

(4.20b)

$$E(a_u, b_u, c_u) = \int_\Omega \sum_{k=1}^{4} \psi_l(a_u t_k^2 + b_u t_k + c_u - u_k)^2 + \alpha(\psi_a(|\nabla a_u|^2) + \psi_b(|\nabla b_u|^2) + \psi_c(|\nabla c_u|^2)) dxdy$$

Die oben genannten Energiefunktionale für $u_{x,y}$ gelten ebenso für die v-Komponente in jedem Pixel $(i) \in \Omega$ und können durch einfache Ersetzung von u durch v erhalten werden. In beiden Fällen wird der Bestrafungsterm analog zur robusten Regression durch die Charbonnier-Bestrafungsfunktion eingeführt, die in Gleichung 3.30 definiert wurde und deren Ableitung folgendermaßen lautet:

(4.21) $$\psi'(s^2) = \frac{1}{\sqrt{(1 + \frac{s^2}{\lambda^2})}}$$

4.2.1 zu Gleichung 4.20a

Bildet man die Euler-Lagrange Gleichungen analog zum Fall ohne Gewichtung der Parameter, setzt die entsprechenden Werte ein und setzt diese gleich o erhält man das Gleichungssystem

aus Gleichung 4.22. Dabei wird der Term $a_u t_k^2 + b_u t_k + c_u - u_k$ durch s_k^2 ersetzt (siehe auch Abschnitt 3.3).

$$
\begin{aligned}
&\int_\Omega \sum_{k=1}^{4} \psi_l'(s_k^2)(a_u t_k^4 + b_u t_k^3 + c_u t_k^2 - u_k t_k^2) \\
&\qquad - \alpha * div(\psi_g'(|\nabla a_u|^2 + |\nabla b_u|^2 + |\nabla c_u|^2) * \nabla a_u) dxdy = 0 \\
&\int_\Omega \sum_{k=1}^{4} \psi_l'(s_k^2)(a_u t_k^3 + b_u t_k^2 + c_u t_k - u_k t_k) \\
&\qquad - \alpha * div(\psi_g'(|\nabla a_u|^2 + |\nabla b_u|^2 + |\nabla c_u|^2) * \nabla b_u) dxdy = 0 \\
&\int_\Omega \sum_{k=1}^{4} \psi_l'(s_k^2)(a_u t_k^2 + b_u t_k + c_u - u_k) \\
&\qquad - \alpha * div(\psi_g'(|\nabla a_u|^2 + |\nabla b_u|^2 + |\nabla c_u|^2) * \nabla c_u) dxdy = 0 \\
&\int_\Omega \sum_{k=1}^{4} \psi_l'(s_k^2)(a_v t_k^4 + b_v t_k^3 + c_v t_k^2 - v_k t_k^2) \\
&\qquad - \alpha * div(\psi_g'(|\nabla a_v|^2 + |\nabla b_v|^2 + |\nabla c_v|^2) * \nabla a_v) dxdy = 0 \\
&\int_\Omega \sum_{k=1}^{4} \psi_l'(s_k^2)(a_v t_k^3 + b_v t_k^2 + c_v t_k - v_k t_k) \\
&\qquad - \alpha * div(\psi_g'(|\nabla a_v|^2 + |\nabla b_v|^2 + |\nabla c_v|^2) * \nabla b_v) dxdy = 0 \\
&\int_\Omega \sum_{k=1}^{4} \psi_l'(s_k^2)(a_v t_k^2 + b_v t_k + c_v - v_k) \\
&\qquad - \alpha * div(\psi_g'(|\nabla a_v|^2 + |\nabla b_v|^2 + |\nabla c_v|^2) * \nabla c_v) dxdy = 0
\end{aligned}
\tag{4.22}
$$

Diskretisierung

Diskretisiert man diese Gleichung nach dem selben Schema, wie dies beim variationalen Ansatz ohne Gewichtung der Parameter passierte, erhält man folgendes Gleichungssystem für den Pixel (i).

(4.23)

$$\sum_{k=1}^{4} \psi'_{l_u}([s_k]_i^2)([a_u]_i t_k^4 + [b_u]_i t_k^3 + [c_u]_i t_k^2 - u_k t_k^2) - div(\psi'_g(|\nabla a_u|^2 + |\nabla b_u|^2 + |\nabla c_u|^2)\nabla a_u) = 0$$

$$\sum_{k=1}^{4} \psi'_{l_u}([s_k]_i^2)([a_u]_i t_k^3 + [b_u]_i t_k^2 + [c_u]_i t_k - u_i t_k) - div(\psi'_g(|\nabla a_u|^2 + |\nabla b_u|^2 + |\nabla c_u|^2)\nabla b_u) = 0$$

$$\sum_{k=1}^{4} \psi'_{l_u}([s_k]_i^2)([a_u]_i t_k^2 + [b_u]_i t_k + [c_u]_i - u_k) - div(\psi'_g(|\nabla a_u|^2 + |\nabla b_u|^2 + |\nabla c_u|^2)\nabla c_u) = 0$$

$$\sum_{k=1}^{4} \psi'_{l_v}([s_k]_i^2)([a_v]_i t_k^4 + [b_v]_i t_k^3 + [c_v]_i t_k^2 - v_k t_k^2) - div(\psi'_g(|\nabla a_v|^2 + |\nabla b_v|^2 + |\nabla c_v|^2)\nabla a_v) = 0$$

$$\sum_{k=1}^{4} \psi'_{l_v}([s_k]_i^2)([a_v]_i t_k^3 + [b_v]_i t_k^2 + [c_v]_i t_k - v_k t_k) - div(\psi'_g(|\nabla a_v|^2 + |\nabla b_v|^2 + |\nabla c_v|^2)\nabla b_v) = 0$$

$$\sum_{k=1}^{4} \psi'_{l_v}([s_k]_i^2)([a_v]_i t_k^2 + [b_v]_i t_k + [c_v]_i - v_k) - div(\psi'_g(|\nabla a_v|^2 + |\nabla b_v|^2 + |\nabla c_v|^2)\nabla c_v) = 0$$

Bei diesem Gleichungssystem fehlt allerdings die Diskretisierung des Terms $div(\psi'_g(|\nabla a_u|^2 + |\nabla b_u|^2 + |\nabla c_u|^2)\nabla a_u)$. Dieser Term kann im variationalen Verfahren ohne Gewichtung der Parameter sehr leicht diskret berechnet werden, da dort keine Gewichtsfunktion ψ_g eingeführt wurde und somit gilt: $div(\nabla c_v) = \triangle c_v$, was dem Laplace-Operator entspricht, der durch eine einfache Approximation ermittelt werden kann.

Die Diskretisierung des oben genannten Terms ist im Folgenden erklärt. Diese Erklärung findet analog auch für die Ableitungen nach a_v, $b_{u/v}$ und $c_{u/v}$ Verwendung, die auf die selbe Art und Weise diskret ermitellt werden können.

Der zu diskretisiernde Term wird so umgeschrieben, dass eine Ableitung sowohl nach der x- als auch der y-Komponente erfolgt. Zur Vereinfachung der Gleichung wird dabei der Funktionsterm der Funktion ψ'_g nicht mehr explizit aufgelistet:

$$(4.24)\quad div(\psi'_g \nabla a_u) = \partial_x \left(\psi'_g(|\nabla a_u|^2 + |\nabla b_u|^2 + |\nabla c_u|^2)\partial_x \nabla a_u\right) + \partial_y \left(\psi'_g(...)\partial_y \nabla a_u\right)$$

Die Approximation der beiden Ableitungsterme nach x und y ist im Folgenden für die Ableitung in x-Richtung dargestellt und wird in y-Richtung analog vollzogen (nach [Bru13b]). Diese Approximation wird im Folgenden im zweidimensionalen Raum dargestellt.

$$(4.25)\quad \partial_x(\psi'_g \partial_x \nabla a_u) \approx \frac{1}{h_x}\left(\psi'_{g_{i+1/2,j}}\left(\frac{[a_u^k]_{i+1,j} - [a_u^k]_{i,j}}{h_x}\right) - \psi'_{g^k_{i-1/2,j}}\left(\frac{[a_u^k]_{i,j} - [a_u^k]^k_{i-1,j}}{h_x}\right)\right)$$

$\psi'_{g^k_{i+1/2,j}}$ ist hierbei das arithmetische Mittel der beiden berechenbaren Funktionen $\psi'_{g_{i,j}}$ und $\psi'_{g_{i+1,j}}$. Analog dazu werden auch die anderen ψ' Funktionen bestimmt, die jeweils das arithmetische Mittel der beiden benachbarten ganzzahligen ψ'-Funktionen darstellen.

Die Berechnung der Länge des Gradienten von a_u, sprich $|\nabla a_u|$, erfolgt folgendermaßen:

$$|\nabla a_u| = \sqrt{(\delta_x a_u)^2 + (\delta_y a_u)^2} \approx \sqrt{\left(\frac{[a_u^k]_{i+1,j} - [a_u^k]_{i-1,j}}{2h_1}\right)^2 + \left(\frac{[a_u^k]_{i,j+1} - [a_u^k]_{i,j-1}}{2h_2}\right)^2} \tag{4.26}$$

Somit lässt sich mithilfe der Charbonnier-Funktion auch die ψ' Funktion für alle Nachbarn von dem Punkt (i,j) berechnen. Die Maske, die zur Berechnung verwendet wird, sieht folgendermaßen aus:

	$\frac{\psi_{i,j-1}+\psi_{i,j}}{2h_y^2}$	
$\frac{\psi_{i-1,j}+\psi_{i,j}}{2h_x^2}$	$-\sum$	$\frac{\psi_{i+1,j}+\psi_{i,j}}{2h_x^2}$
	$\frac{\psi_{i,j+1}+\psi_{i,j}}{2h_y^2}$	

Durch die oben durchgeführten Rechnungen ergibt sich ein diskretes Gleichungssystem. Zu beachten ist hierbei, dass die ψ' in allen Punkten vor dem nächsten Iterationsschritt zur Approximation der Lösung des Gleichungssystems neu berechnet werden müssen. Dadurch ergibt sich unter Benutzung der Nachbarfunktion $N_l(i), l \in \{x,y\}$ am Pixel (i) folgende Beispielgleichung für die Variable $[a_u]_i$:

$$\sum_{k=1}^{4} \psi_{l_u} \left([a_u]_i t_k^4 + [b_u]_i t_k^3 + [c_u]_i t_k^2 - u_k t_k^2\right) - \sum_{l\in x,y} \sum_{j\in N_l(i)} \left(\frac{\psi_j + \psi_i}{2h_l}\left([a_u]_j - [a_u]_i\right)\right) = 0 \tag{4.27}$$

Lösen des komplett diskretisierten Gleichungssystems

Die Lösung erfolgt analog zur variationalen Methode mithilfe des Gauß-Seidel bzw. SOR-Algorithmus. Dazu wird in der jeweiligen Ableitung die Gleichung nach der jeweiligen abgeleiteten Variable aufgelöst und man erhält beispielsweise für die Variable $[a_u]_i$ folgende Berechnungsvorschrift für den Gauß-Seidel Algorithmus. Zu beachten ist hier, dass wiederum die Matrix J benutzt wurde, um die lokale Komponente einfacher ausdrücken zu können.

$$[a_u]_i^{m+1} = \frac{\left(-[J_{u,12}]_i [b_u]_i^m - [J_{u,13}]_i [c_u]_i^m - [J_{u,14}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{(\psi_j+\psi_i)[a_u]_j}{2h_l}\right)}{[J_{u,11}]_i + \alpha \sum_{l\in x,y} \sum_{j\in N_l(i)} \frac{\psi_j+\psi_i}{h_l}} \tag{4.28}$$

Die Berechnungsvorschriften für die restlichen zu bestimmenden Variablen können analog gebildet werden. Eine Lösung des Gleichungssystems kann iterativ mit dem Gauß-Seidel oder SOR-Algorithmus gebildet werden.

4.2.2 zu Gleichung 4.20b

Auch hier wird die Ableitung des Energiefunktionals benötigt, um das Minimum des Energiefunktionals bestimmen zu können. Ebenfalls wird der Term s_k^2 eingeführt, wie dies bereits in Unterabschnitt 4.2.1 der Fall war.

$$
\begin{aligned}
&\sum_{k=1}^{4} \psi_l'(s_k^2)\left(a_u t_k^4 + b_u t_k^3 + c_u t_k^2 - u_k * t_k^2\right) && -div\left(\psi_{g_a}'(|\nabla a_u|^2)\nabla a_u\right) = 0\\
&\sum_{k=1}^{4} \psi_l'(s_k^2)\left(a_u t_k^3 + b_u t_k^2 + c_u t_k - u_k * t_k\right) && -div\left(\psi_{g_b}'(|\nabla b_u|^2)\nabla b_u\right) = 0\\
&\sum_{k=1}^{4} \psi_l'(s_k^2)\left(a_u t_k^2 + b_u t_k + c_u - u_k\right) && -div\left(\psi_{g_c}'(|\nabla c_u|^2)\nabla c_u\right) = 0
\end{aligned}
\tag{4.29}
$$

$$
\begin{aligned}
&\sum_{k=1}^{4} \psi_l'(s_k^2)\left(a_v t_k^4 + b_v t_k^3 + c_v t_k^2 - v_k t_k^2\right) && -div\left(\psi_{g_a}'(|\nabla a_v|^2)\nabla a_v\right) = 0\\
&\sum_{k=1}^{4} \psi_l'(s_k^2)\left(a_v t_k^3 + b_v t_k^2 + c_v t_k - v_k t_k\right) && -div\left(\psi_{g_b}'(|\nabla b_v|^2)\nabla b_v\right) = 0\\
&\sum_{k=1}^{4} \psi_l'(s_k^2)\left(a_v t_k^2 + b_v t_k + c_v - v_k\right) && -div\left(\psi_{g_c}'(|\nabla c_v|^2)\nabla c_v\right) = 0
\end{aligned}
$$

Die Diskretisierung des Gleichungssystems erfolgt analog zum Vorgehen in Abschnitt 4.2.1. Es gilt lediglich für die ψ'-Funktionen, dass hier nicht der ganze umgebungsabhängige Term eine Rolle spielt, sondern lediglich der Gradient des aktuell abzuleitenden Koeffizienten.

5 Implementierung

Die Implementierung basierte grundsätzlich auf dem Programm, welches im Rahmen der Vorlesung "Correspondence Problems in Computer Vision" [Bru13a] an der Universität Stuttgart im Sommersemester 2013 im Rahmen der Übungen eingeführt wurde. Dieses wurde Schritt für Schritt um die in den beiden vorangegangen Kapiteln (Kapitel 3 und Kapitel 4) beschriebenen Verfahren erweitert. Darüberhinaus wurden nocht weitere Anpassungen an die Darstellung und die Bedienung des Programmes durchgeführt, die im Folgenden zuerst chronologisch dargestellt werden. Anschließend erfolgt noch jeweils ein kurzer Überblick über die implementierte Benutzeroberfläche sowie die Ausgabemöglichkeiten des während der Bachelorarbeit entstandenen Programms.

5.1 Voraussetzungen

5.1.1 Basisprogramm

Der als Basis der Implementierung benutzte Programmcode aus der Vorlesung "Correspondence Problems in Computer Vision" diente ursprünglich dazu, im Rahmen der zu der Vorlesung gehörenden Übungen mithilfe des Horn-Schunck Algorithmus das optische Flussfeld zu berechnen. Dies sollte durch die Übungsteilnehmer in Eigenarbeit geschehen. Der Code des Programms ist dabei in C geschrieben.Die Ausgabe des Programms erfolgt sowohl auf einer grafischen Oberfläche als auch in der Konsole. Auf der grafischen Oberfläche wird dabei neben dem Ursprungsbild daneben das optische Flussfeld, welches berechnet werden sollte, visualisiert. Auf der Konsole können hingegen in einer einfach gehaltenen Menüstruktur die Parameter, die bei der Berechnung verwendet werden, durch Selektieren und erhöhen/erniedrigen des Wertes mit der Tastatur verändert werden.

Vor- und Nachteile

Aus einigen Eigenschaften des Programms konnten daher im Rahmen dieser Bachelorarbeit ein Vorteil gezogen werden:

- **OpenGL-Schnittstelle**
  Der Programmcode verfügte bereits in der zur Verfügung gestellten Version über eine OpenGL-Schnittstelle, mit Hilfe derer die grafische Ausgabe gesteuert wird. Im Verlauf der Bachelorarbeit musste daher lediglich die Größe des Fensters angepasst sowie die

Daten, die ausgegeben werden sollten, angepasst werden. Jedoch musste nichts an der abstrakten Implementierung der OpenGL-Schnittstelle an sich verändert werden.

- **Funktionen zum Allokieren von Speicher**
  Das Ursprungsprogramm besitzt bereits durch Einbinden einer Bibliothek Routinen zum (De-)Allokieren von Speicher, sodass diese im Rahmen dieser Bachelorarbeit weiterbenutzt werden konnten.
- **Methode zur farblichen Darstellung von Flussfeldern**
  Durch eine Bibliothek besitzt das Basisprogramm bereits die Möglichkeit, optische Flussfelder farblich darzustellen. Diese Routine musste im Verlauf der Bachelorarbeit lediglich dahingehend angepasst werden, dass die maximale Länge eines Vektors des Flussfeldes, die im Basisprogramm auf 1 beschränkt war, erhöht wurde.
- **Tastaturhandler**
  Das Programm besitzt bereits die Möglichkeit Parameter mittels eines einfachen Handlers für Tastatureingaben zu verändern.
- **Ausgabe von .ppm Dateien**
  Das Programm besaß durch Einbindung einer Bibliothek bereits die Möglichkeit .ppm Dateien auf dem Dateisystem zu speichern. Dies wurde im Rahmen der Speicherung der Klassifikationskarten genutzt, die in .ppm Dateien abgespeichert werden.

Neben dieser ganzen Reihe von Vorteilen, gibt es auch einige wenige Nachteile, die bei der Implementierung zu beachten waren:

- **Globale Parameter**
  Zu Beginn existierten zahlreiche globale Parameter, die für die Implementierung des in dieser Bachelorarbeit zu bearbeitenden Problems von keiner Bedeutung waren. Es war daher eine umfangreiche Anpassung notwendig.
- **Monolithisches Grundgerüst**
  Mit Ausnahme der in der Vorlesungsübung zu implementierenden Methoden befinden sich alle Funktionen des Programms in einer einzigen Quellcodedatei. Dies führte dazu, dass der Überblick über den Quellcode gerade zu Beginn schnell verloren ging. Abhilfe wurde dadurch geschaffen, dass Funktionen, die einer bestimmten Aufgabe zugeordnet werden konnten, in seperate Pakete ausgelagert wurden.
- **Einlesen von Flussdaten**
  Das Einlesen von Flussdaten war nicht mit dem .flo-Format, sondern nur mit dem .F-Format auf Anhieb möglich. Hierzu musste erst eine zur Verfügung gestellte Bibliothek in den Code eingebunden werden.
- **Menüsteuerung**
  Die Menüsteuerung ist nur möglich, wenn das OpenGL-Fenster fokussiert ist. Dies ist besonders bei kleineren Bildschirmen ein Problem, da hier möglicherweise “blind” durch das Menü navigiert werden muss. Grund hierfür ist, dass die Menüdarstellung in der Konsole geschieht, die Steuerung jedoch wie gesagt über das OpenGL-Fenster

geschieht. Dieser Aspekt wurde in der vorgenommen Veränderung des Basisprogramm nicht verändert.

5.2 Anforderungen an die Implementierung

In diesem Abschnitt werden die Anforderungen an die Implementierung getrennt nach funktionalen und nicht-funktionalen Anforderungen aufgelistet. Auf Grundlage der Anforderungen erfolgt anschließend eine detaillierte Ausarbeitung der erforderlichen Anwendungsfälle (Use Cases). Zu dieser Ausarbeitung zählt dazu neben einer Übersicht in Diagrammform die detaillierte Beschreibung der einzelnen Use Cases.

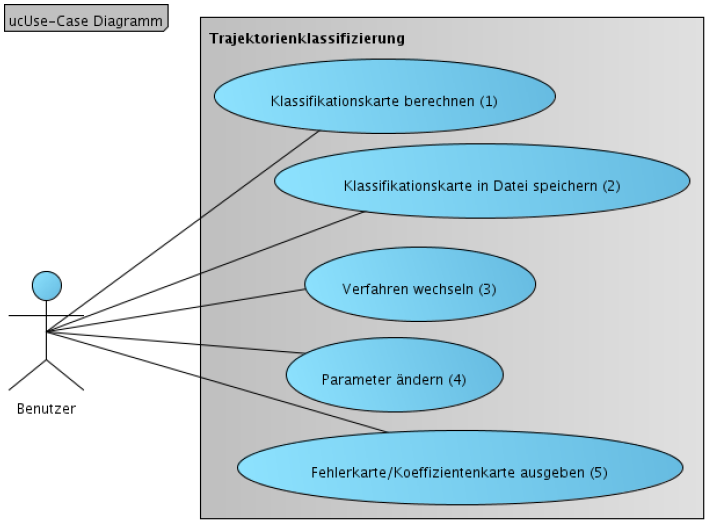
5.2.1 Funktionale Anforderungen

- Im Verlauf der Implementierung sollen verschiedene Verfahren (lineare/robuste Regression, variationale Ansätze) zur Berechnung der Trajektorienklassifikation implementiert werden.
- Anzeige der berechneten Klassifikationskarten auf der grafischen Oberfläche.
- Um die weitere Verarbeitung im Rahmen des TC-Flow Verfahrens zu ermöglichen, soll das Programm die Funktion bieten, Regularisierungskarten in einer Datei zu speichern.
- Es muss die Möglichkeit geschaffen werden, optische Flussfelder aus .flo-Dateien einzulesen.

5.2.2 Nichtfunktionale Anforderungen

- Die Benutzerfreundlichkeit des erstellten Programms soll dadurch erhöht werden, dass zur Berechnung erforderliche Parameter nicht als Konstanten im Quellcode festgelegt sind, sondern zur Laufzeit veränderbar sind.
- Die Erweiterbarkeit des Programmes soll dahingehend gewährleistet sein, dass neue Verfahren einfach zur bestehenden Implementierung hinzugefügt werden können.
- Die Korrektheit der Berechnungen soll durch entsprechende Debugausgaben gewährleistet werden können.
- Das Programm muss lediglich unter Linux in der 64-bit Version lauffähig sein. Es ist keine ausdrückliche Portierbarkeit in andere Umgebungen gewünscht.
- Die Menüführung des Programmes soll ohne große Einarbeitung verständlich und intuitiv sein.

5.2.3 Use Case Diagramm



5.2.4 Use Cases

1. Klassifikationskarte berechnen

Ziel	Der Benutzer möchte eine Klassifikationskarte mit einem zuvor ausgewählten Verfahren berechnet haben.
Beschreibung	Hauptfunktionalität
Akteure	Benutzer, System
Ebene	Benutzersicht
Priorität	Hoch
Normalablauf	
Vorbedingung	Der Benutzer hat das Programm gestartet und befindet sich im Hauptmenü
1 Benutzer	Der Benutzer geht in das Menü zur Auswahl der Verfahren
2 System	Das System stellt das Menü zur Auswahl der Verfahren dar.
3 Benutzer	Der Benutzer wählt das gewünschte Verfahren aus.
4 System	Speichert gewähltes Verfahren.
5 Benutzer	Startet Berechnung durch Drücken der "." Taste.
6 System	Nach beendeter Berechnung erfolgt die grafische Ausgabe der Klassifikationskarte auf dem Bildschirm.
Nachbedingung	Das System gibt nach der Berehcnung der Klassifikationskarte automatisch die berechnete Klassifikationskarte aus.

2. Klassifikationskarte in Datei speichern

Ziel	Der Benutzer möchte zur weiteren Verwendung die Klassifkationsdatei in eine Datei ausgeben lassen.
Akteure	Benutzer, System
Ebene	Benutzersicht
Priorität	Hoch
Normalablauf	
Vorbedingung	Die auszugebende Klassifikationskarte wird grafisch in der Oberfläche angezeigt.
1 Benutzer	Drückt in der grafischen Oberfläche "F2".
2 System	Speichert Klassifikationskarte in Datei "classification.ppmïm Programmordner.
Nachbedingung	Die Klassifikationskarte ist in der Datei gespeichert worden.

3. Berechnungsverfahren wechseln

Ziel	Der Benutzer will ein gewünschtes Verfahren zur Berechnung der Klassifikationskarte einsetzen.
Akteure	Benutzer, System
Ebene	Benutzersicht
Priorität	Hoch
Normalablauf	
Vorbedingung	Der Benutzer hat das Programm gestartet und befindet sich im Hauptmenü.
1 Benutzer	Wählt das Untermenü für die Auswahl der Verfahren aus.
2 System	Zeigt das Menü zur Auswahl der Verfahren an.
3 Benutzer	Wählt gewünschtes Verfahren aus.
4 System	Markiert das gewählte Verfahren als das Verfahren, das zur Berechnung der Klassifikationskarte benutzt wird.
Nachbedingung	Das gewünschte Verfahren wird nun bei der Klassifikationsberechnung benutzt.

4. Parameter ändern

Ziel	Der Benutzer will den Wert eines Parameters ändern ohne das Programm jedes Mal neu kompilieren zu müssen.
Akteure	Benutzer, System
Ebene	Benutzersicht
Priorität	Mittel
Normalablauf	
Vorbedingung	Der Benutzer hat das Programm gestartet und befindet sich im Hauptmenü.
1 Benutzer	Wählt je nach Parametertyp das entsprechende Untermenü aus.
2 System	Zeigt entsprechendes Untermenü an.
3 Benutzer	Wählt gewünschten Parameter aus.
4 System	Markiert ausgewählten Parameter.
5 Benutzer	Verändert Wert des Parameters.
6 System	Speichert neuen Wert des Parameters.
Nachbedingung	Der Wert des gewünschten Parameters wurde geändert und wird entsprechend bei der Berechnung benutzt.

5. Fehlerkarte/Koeffizientenkarte ausgeben

Ziel	Der Benutzer will eine Fehler- bzw. Koeffizientenkarte auf der grafischen Oberfläche angezeigt bekommen, um weiteres Feedback zu den einzelnen Verfahren zu bekommen.
Beschreibung	Hauptfunktionalität
Akteure	Benutzer, System
Ebene	Benutzersicht
Priorität	Mittel
Normalablauf	
Vorbedingung	Der Benutzer hat das Programm gestartet und befindet sich im Hauptmenü
1 Benutzer	Wählt Untermenü für die Visualisierungsparameter aus.
2 System	Zeigt Untermenü an.
3 Benutzer	Wählt anzuzeigende Option aus.
4 System	Gibt auf der grafischen Oberfläche die gewünschte Karte aus.
Nachbedingung	Die gewünschte Ausgabekarte wird angezeigt.

5.3 Verlauf der Implementierung

Bevor mit der Implementierung der einzelnen zuvor beschriebenen Methoden zur Trajektorienklassifizierung begonnen werden konnte, musste das Ausgangsprogramm entsprechend

den Anforderungen angepasst werden. Hierzu wurde der Code dahingehend verändert, dass mittels einer zur Verfügung gestellten Bibliothek Dateien im .flo Format, die Informationen über die Flussfelder beinhalteten, eingelesen werden konnten. Zudem musste jetzt durch Angabe eines Parameters bei Start des Programms angegeben werden, welche vier Flussfelder, die am Ende des Dateinamens mit den Nummern 1-4 nummeriert sein müssen, eingelesen werden sollten.

Anschließend begann die Arbeit an der Implementierung der Regression. Neben dem Verfahren wurden hier im Übersichtsmenü, das in der Konsole dargestellt wird, überflüssige Parameter gelöscht, während neue Parameter wie beispielsweise die beiden Schwellenwerte T_a und T_b für die zu bestimmenden Variablen a und b hinzugefügt wurden, sodass diese zur Laufzeit verändert werden können. Zudem wurde die Benutzeroberfläche dahingehend verändert, dass neben der Klassifikationsmap auch alle vier Flussfelder in der grafischen Oberfläche zu sehen waren. Dies geschah zunächst auf Basis nur einer Flussrichtung, die in Grautönen dargestellt wurde, wobei der Wert 0 dem Grauwert 127,5 entsprach.

Durch die Implementierung der robusten Regression, die auf der Implementierung der linearen Regression aufbaute und diese erweiterte, stieg die Anzahl der Menüpunkte weiter, da zum einen die Methodenauswahl zur Laufzeit eingeführt wurde, zum anderen zusätzliche Parameter (Beispiel: λ) veränderbar sein mussten. Aus diesem Grund wurde vor der Implementierung der variationalen Modelle entschieden, dass das Menü in drei Untermenüs aufgegliedert werden sollte. Diese dienten 1) zur Methodenauswahl, 2) zur Veränderung numerischer Parameter (Beispiel: λ) 3) zur Veränderung visueller Parameter. Darüberhinaus wurde die Ausgabe der Flussfelder dahingehend angepasst, dass diese mittlerweile eine farbliche Kodierung aufwiesen und sowohl die u- als auch v- Komponente der einzelnen Trajektorien berücksichtigten. In der Konsole wird dabei während der Berechnung mittels der robusten Regression die aktuelle Iterationsanzahl ausgegeben.

Abschließend wurde die variationale Methode implementiert, die auf den vorherigen Implementierungen aufbaute und um die Einführung der ψ-Funktionen (sowohl lokal als auch global) ebenfalls erweitert wurde. Es wurde zudem die Möglichkeit geschaffen, verschiedene grafische Ausgaben (siehe auch Abschnitt 5.6) neben den Klassifikationsmaps ausgeben zu lassen. Zudem wurden für den Gauss-Seidel/SOR-Algorithmus eine Ausgabe hinzugefügt, die sowohl das aktuelle als auch das zu Beginn der Berechnung berechnete Residuum auf der Konsole ausgibt.

5.4 Programm-Architektur

Die im Verlauf der Bachelorarbeit implementierte Programmarchitektur kann mit Hilfe von Abbildung 5.1 nachvollzogen werden.

Die **Frontend**-Komponente ist sozusagen das Herzstück des Programms und beinhaltet den Main-Loop, in dem die Programminstanz ausgeführt wird. Hier werden zu Programmstart die globalen Variablen entsprechend initialisiert, die Flussfeldbilder, zu denen im

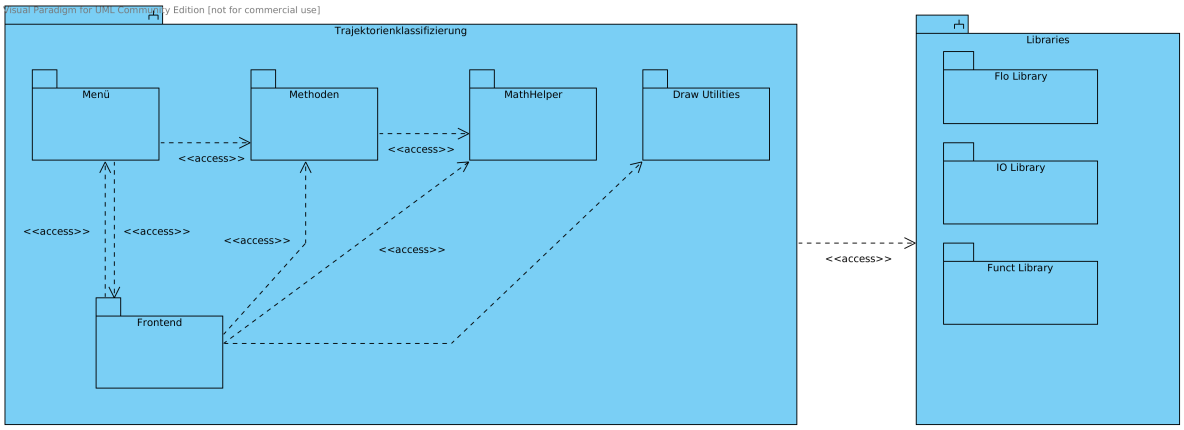


Abbildung 5.1: Paketdiagramm des implementierten Programms

weiteren Verlauf die Trajektorienklassifizierung erstellt werden soll, eingelesen, die OpenGL-Schnittstelle inklusive der Ausgabe auf dem Bildschirm initialisiert und insbesondere auch die anderen Komponenten - soweit nötig - eingebunden.

Dies ist zum Programmstart insbesonders die **Menü**-Komponente, deren Handler für Maus- und Tastaturaktionen auf der OpenGL-Schnittstelle registriert werden und mit deren Hilfe die Steuerung der Menüs durchgeführt wird. Die Menükomponente dient vor allem dazu, dem Benutzer die Möglichkeit zu geben, Parameter und das genutzte Klassifikationsverfahren zur Programmlaufzeit zu ändern. Will der Benutzer eine neue Berechnung der Trajektorienklassfikation durchführen, teilt er dies der Menükomponente per Tastendruck mit.

Diese wiederum ruft eine Methode im Frontend auf, die entsprechend der aktuellen Parameterkonfiguration die entsprechende Klassifikationsmethode aus der **Methoden**-Komponente aufruft. Die **Methoden**-Komponente wiederum greift auf mathematische Hilfsmethoden, die von der **Math Helper**-Komponente bereitgestellt werden, zurück.

Zur Visualisierung der Flussbilder/Klassifikationen/... dient die **DrawUtils**-Komponente, die direkt von der **Frontend**-Komponente angesprochen wird. Alle Komponenten können dabei auf in der **Frontend**-Komponente importierten Bibliotheken zurückgreifen.

In Abbildung 5.2 ist ein Beispiel neben der Interaktion mit dem Benutzer der interne Ablauf für den Use-Case “Klassifikationskarte berechnen” in einem UML-Sequenzdiagramm dargestellt. Anhand dieser Abbildung lässt sich das Zusammenwirken der einzelnen Programmkomponenten für diesen Ablauf nachvollziehen.

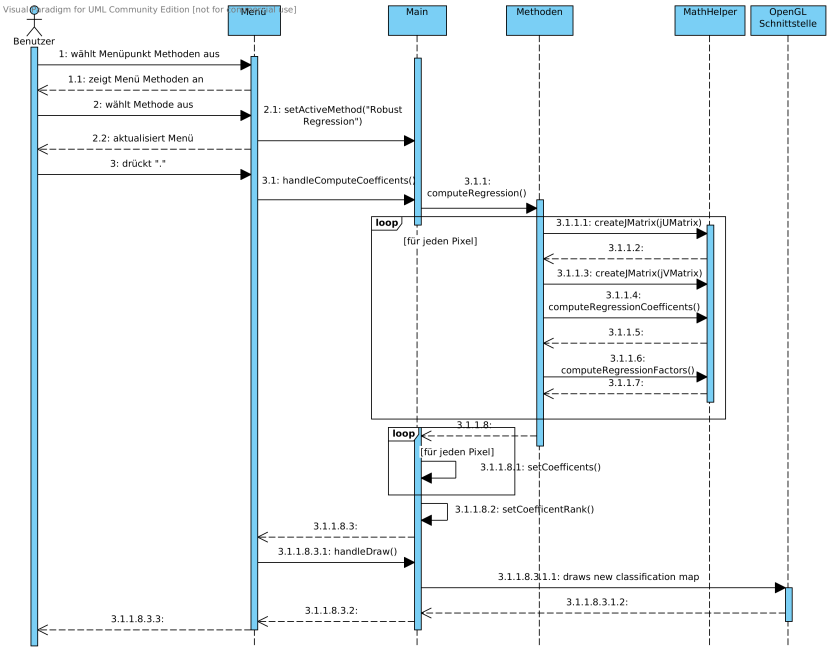


Abbildung 5.2: Sequenzdiagramm für den Use Case "Klassifikationskarte berechnen" (Abschnitt 5.2.4)

5.5 Benutzeroberfläche

Die Benutzeroberfläche gliedert sich in zwei Teile: Dies ist zum einen die grafische Ausgabe, bei der in der Version bei Abgabe der Arbeit neben den vier Flussfeldern die aktuell berechnete Klassifikationsmap sowie eine der in Abschnitt 5.6 dargestellten Ausgabemöglichkeiten gleichzeitig dargestellt werden. Zum anderen befindet sich die Menükomponente des Programms in der Konsole, über die die Auswahl der Methode zur Trajektorienklassifizierung sowie die Veränderung von verschiedenen Parametern erfolgen kann.

Zudem wurde ein Maushandler implementiert, der bei Klick auf einen Pixel der grafischen Ausgabe diesen auf eine bestimmte Trajektorie mappt und Informationen zu dieser Trajektorie in der Konsole ausgibt. Hierzu gehört unter anderem die einzelnen Datenwerte für die Flüsse u_i zu den Zeitpunkten t_i ($i \in \{1, 2, 3, 4\}$) sowie die aktuell berechneten Werte für die Koeffizienten $a_{u/v}$, $b_{u/v}$ und $c_{u/v}$. Ein Screenshot mit Informationen zu einem Pixel ist in Abbildung 5.4b gegeben.

Ein Screenshot der erstellten GUI findet sich in Abbildung 5.3. Ein Screenshot der Menüstruktur in der Konsole befindet sich in Abbildung 5.4a.

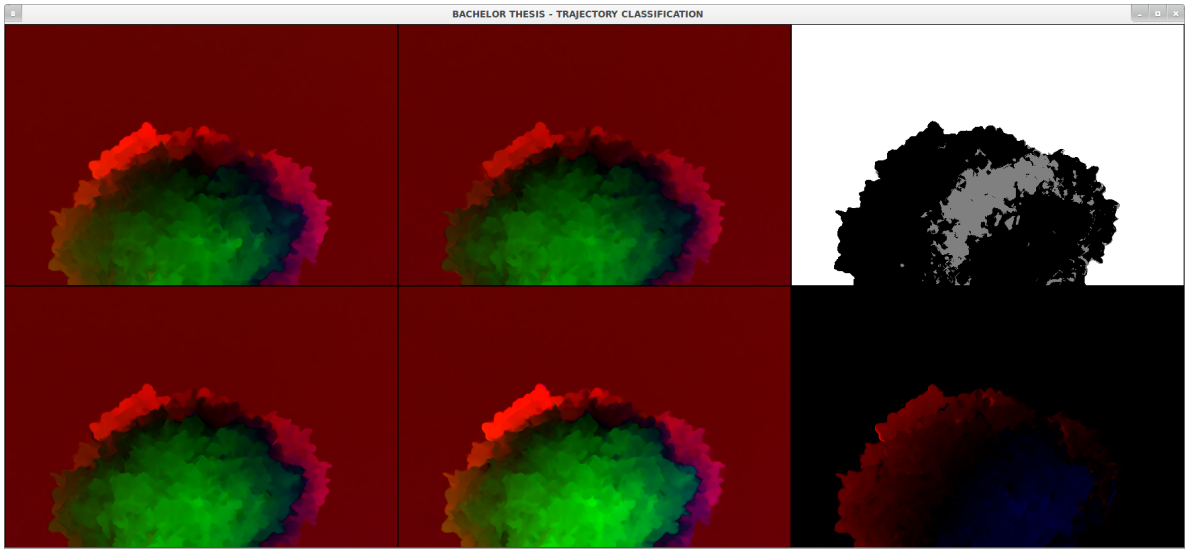
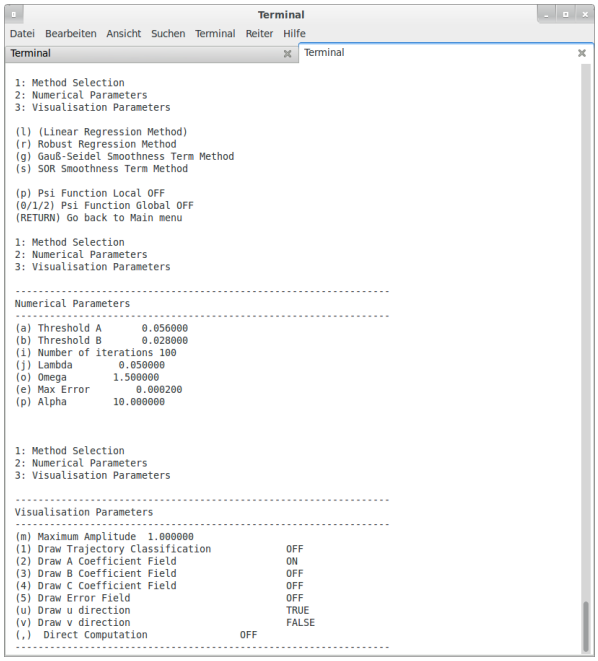
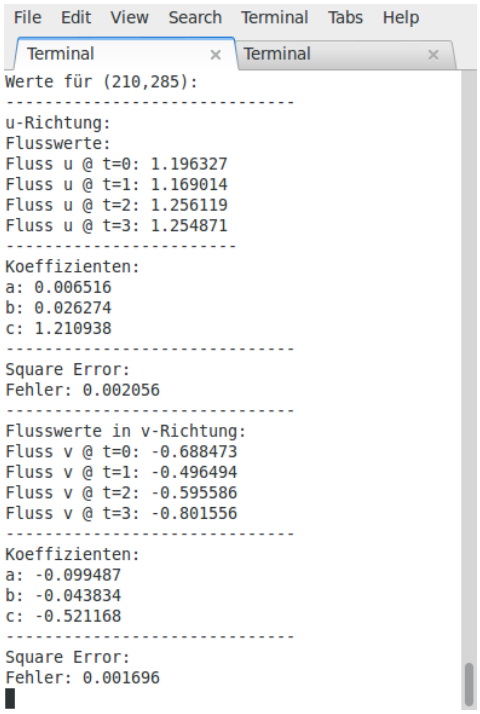


Abbildung 5.3: GUI



(a) Menüs



(b) Trajektorieninformationen nach Mausklick

Abbildung 5.4: Konsolenfenster des Programms

5.6 Ausgabemöglichkeiten

Neben der Ausgabe der Klassifikationsmaps auf dem Bildschirm besitzt das implementierte Programm zusätzliche Möglichkeiten der Ausgabe, die im Folgenden alle aufgelistet sind:

- **Ausgabe der Klassifikationskarte**
  Dies entspricht der wichtigsten Ausgabefunktion des Programmes. Klassifikationskarten können sowohl auf dem Bildschirm angezeigt als auch als Datei exportiert werden. Durch den Export als Datei ist es möglich, die Klassifikationskarte im Rahmen des Verfahrens, das in [VBVZ11] beschrieben ist, zu verwenden. Dies geschieht auch in Kapitel 6 mit Hilfe des Evaluierungstools (Abschnitt 6.1).
- **Ausgabe der Koeffizientenkarte**
  Diese Karten können nach der Berechnung einer Klassifikationskarte innerhalb des implementierten Programms zur Trajektorienklassifizierung betrachtet werden. Mit Hilfe dieser Karten können Aussagen zu den Auswirkungen der einzelnen Verfahren getroffen werden.
- **Ausgabe Fehlerkarte**
  Diese Ausgabe in der grafischen Oberfläche wurde nach Implementierung der linearen Regression geschaffen. Mittlerweile ist sie durch die globalen Verfahren wenig aussagekräftig geworden. Die Möglichkeit, sich den lokalen Fehler in einem Punkt anzuschauen, besteht dennoch.

6 Bewertung und Vergleich der Resultate der Klassifizierungsverfahren

Zuerst erfolgt eine kurze Einführung in das Tool, welches zur Evaluierung der einzelnen Verfahren genutzt wurde, ehe dann die zuvor vorgestellten Klassifizierungsverfahren zuerst einzeln charakterisiert und bewertet werden. In diesem Fall wird auch die Bedeutung der einzelnen Parameter für das jeweilige Klassifizierungsverfahren und dessen Berechnung genauer untersucht. Anschließend findet ein quantitativer Vergleich aller Klassifizierungsverfahren statt.

Die Klassifikationskarten werden mithilfe der drei RGB-Farben schwarz, weiß und grau dargestellt.

Farbe	Klassifizierungsgrad
Schwarz	kein
Grau	2. Ordnung
Weiß	1. Ordnung

6.1 Evaluierungstool

Die Evaluierung wurde mithilfe eines am Institut für Visualisierung und Interaktive Systeme (VIS) entstandenen Programms mit grafischer Oberfläche durchgeführt. Als Ausgangsdaten dienen hierbei die einzelnen Frames der jeweiligen Originalsequenz. Anschließend werden die einzelnen Schritte des TC-Flow-Verfahrens von [VBVZ11] ausgeführt. Um eine Evaluierung der im Rahmen dieser Bachelorarbeit entstandenen Klassifizierungskarten durchzuführen, muss der Benutzer den Dateinamen der einzulesenden .ppm Datei relativ zum “temp” Verzeichnis des Programms angeben. Ein Screenshot der grafischen Oberfläche inklusive dem Eingabefeld für die Klassifizierungskarten ist in Abb. 6.1 zu sehen.

Ausgabedaten des Programmes nach der Evaluierung, die etwas Zeit für jede Sequenz in Anspruch nimmt, sind die auch im Benchmark der Universität Middlebury verwendeten Kennwerte für den “Average Angle Error”(AAE, durchschnittlicher Fehler für den Trajektorienwinkel) sowie der “Average Endpoint Error” (AEE, durchschnittlicher Endwertfehler), auf deren Basis eine qualitative Evaluierung der Klassifizierungsverfahren erfolgen kann.

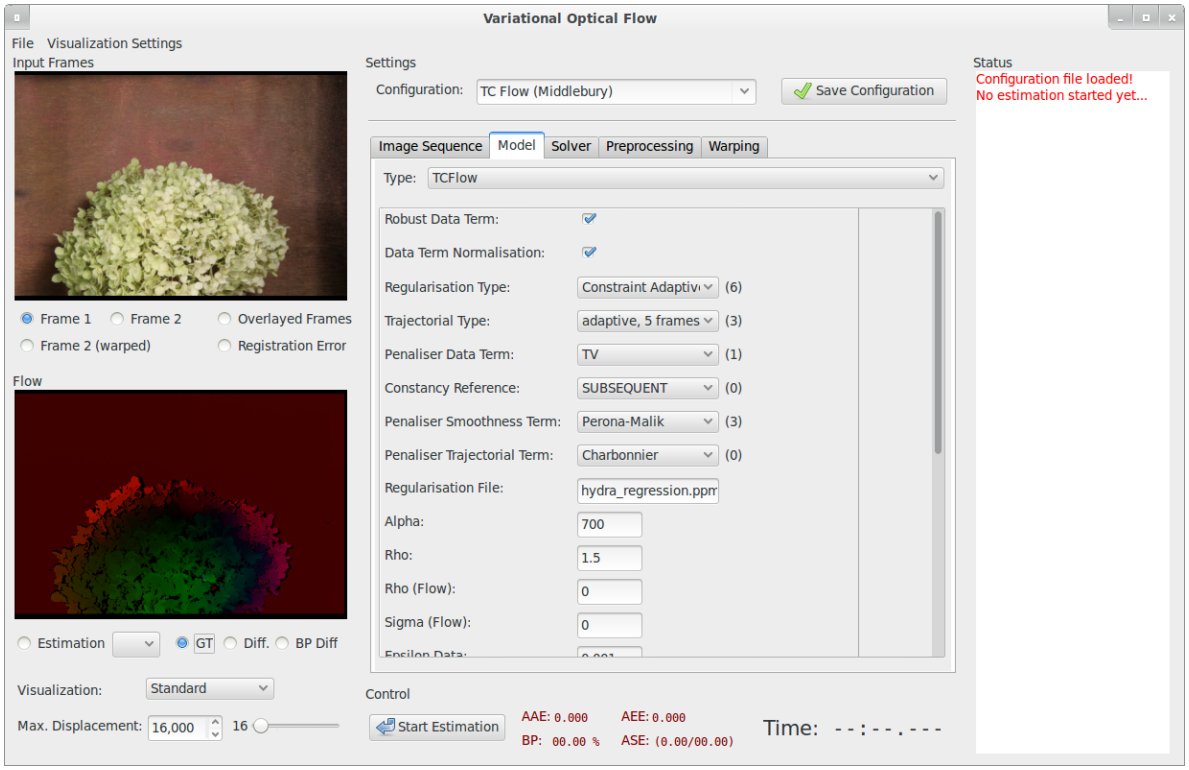


Abbildung 6.1: Screenshot des Evaluierungstools

6.2 Bewertung der Klassifizierungsverfahren

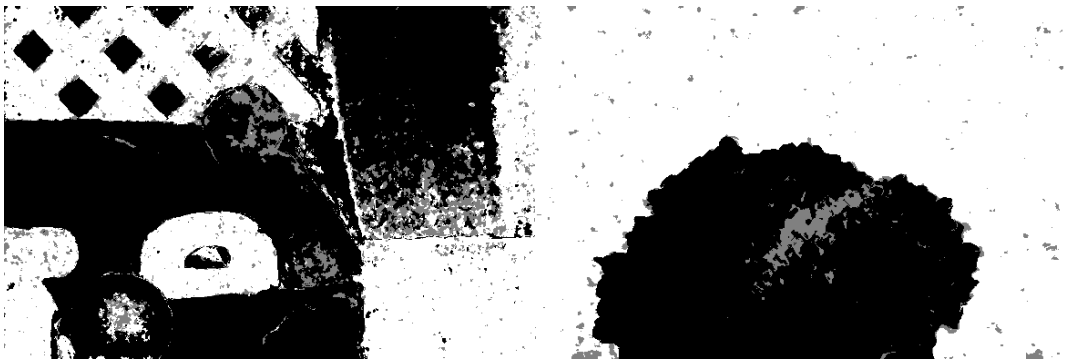
Im Folgenden sollen die einzelnen Verfahren zunächst für sich bewertet werden. Dabei wird auch genauer auf die möglichen Parameter, die von Hand verändert werden können, eingegangen und deren Auswirkungen auf das Ergebnis der Klassifikation aufgezeigt.

6.2.1 Regression

Als erste Methode wurde die gewöhnliche Regressionsmethode für Funktionskurven mit dem Grad 2 implementiert. Diese berechnet für jeden Pixel die optimale Kurve $ax^2 + bx + c$, die den gemessenen Daten am Nächsten kommt. Dies entspricht einem Least-Squares Ansatz. Beispiele für die Klassifikation mithilfe der normalen Regression sind in Abbildung 6.2 für die Sequenz Hydrangea sowie für die Sequenz RubberWhale zu sehen.

Anhand der auf diesen Bildern zu sehenden Klassifikationskarten erkennt man schnell die Probleme dieses Ansatzes. Dies sind hauptsächlich die beiden folgenden:

- **“Verrauschte” Klassifikationskarten**
  Die Klassifikationskarten für die Trajektorien sind für einzelne Sequenzen sehr unregelmäßig. Allerdings ist auf Grund der Bildstrukturen, die hauptsächlich Objekte



(a) Sequenz RubberWhale, $T_a = 0.028$ und $T_b = 0.014$

(b) Sequenz Hydrangea, $T_a = 0.1$ und $T_b = 0.05$

Abbildung 6.2: Lineare Regression

abbilden, eher davon auszugehen, dass benachbarte Pixel/Trajektorien mit dem selben Grad klassifiziert werden, sofern sie zum selben Objekt gehören.

- **Ausreißer führen schnell zu höherem Klassifizierungsgrad**
  Wie in Abb. 6.3 illustriert, können Ausreißer in den Eingabedaten schnell zu einer für den Betrachter offensichtlich falschen Regressionskurve führen. Der Ausreißer nimmt durch die lineare Gewichtung aller Punkte erheblichen Einfluss auf die berechnete Regressionskurve.

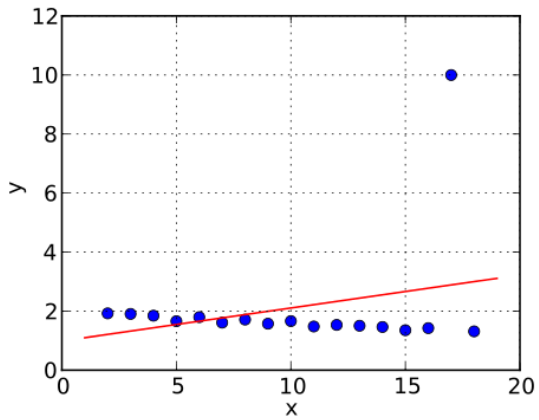
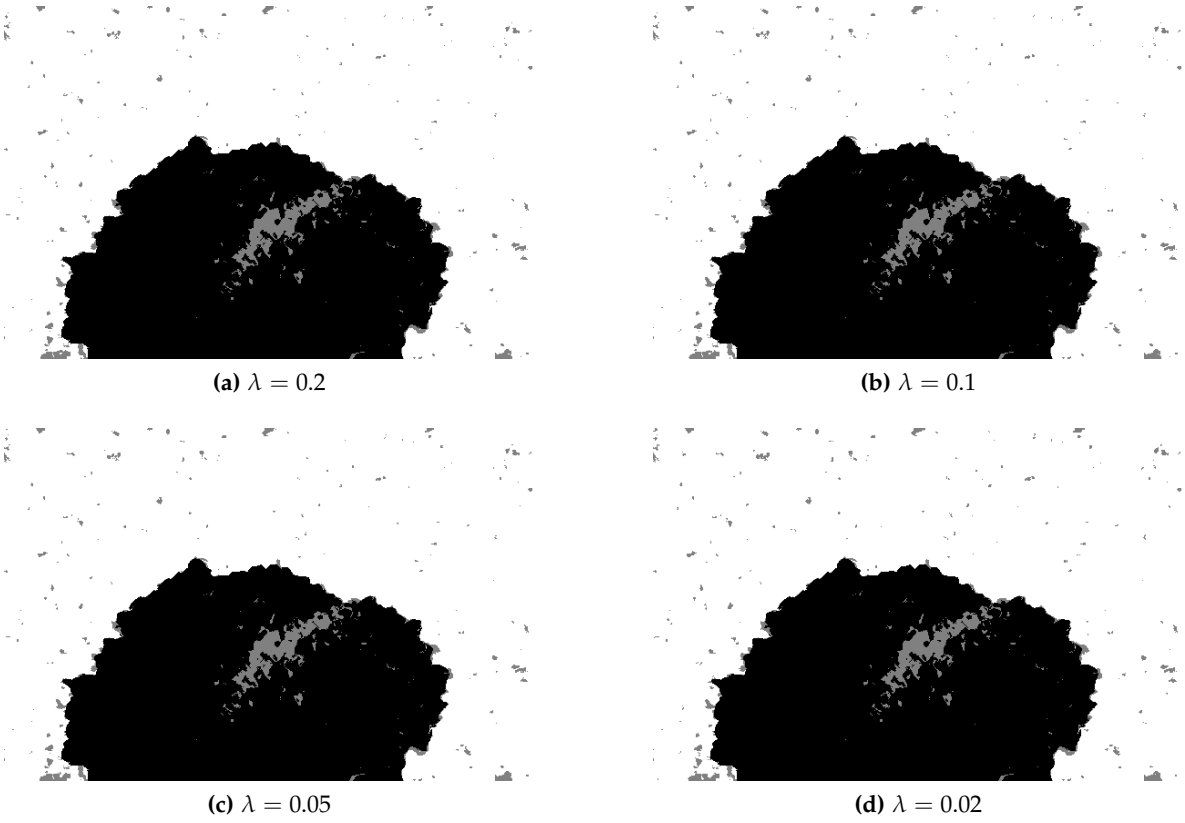


Abbildung 6.3: Beispiel für eine Regressionsgerade mit Ausreißer[1]

[1] http://commons.wikimedia.org/wiki/File:MDKQ4.svg



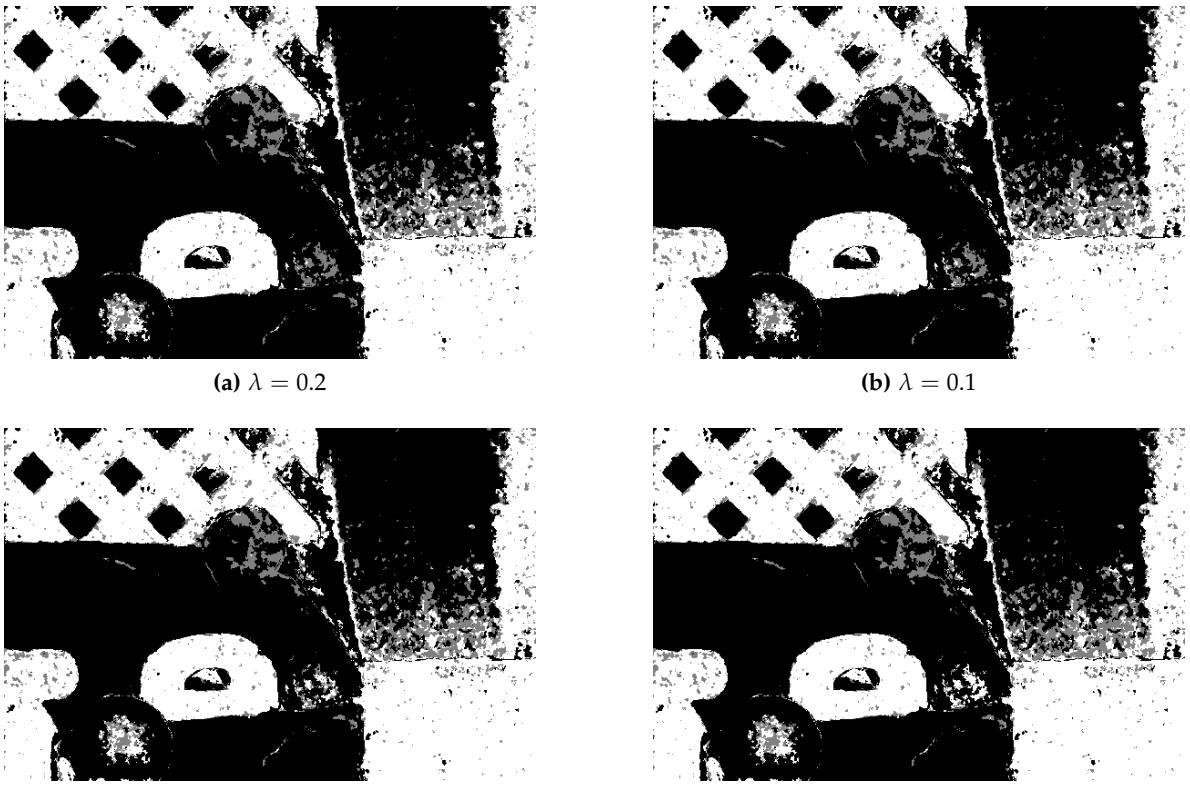
(a) $\lambda = 0.2$ **(b)** $\lambda = 0.1$

(c) $\lambda = 0.05$ **(d)** $\lambda = 0.02$

Abbildung 6.4: Robuste Regression für die Sequenz Hydrangea,$T_a = 0.01$ und $T_b = 0.05$

6.2.2 Robuste Regression

Das Problem, dass Ausreißer bei der linearen Regression die Berechnung der Regressionskurve maßgeblich beeinflussen, sollte in dieser Bachelorarbeit mithilfe der robusten Regression gelöst werden. Die robuste Regression führt hierzu nach jedem Durchgang eine Bewertung der Datenpunkte durch, bei der die Entfernung von der Regressionskurve ermittelt wird. Ausreißer werden also weniger stark gewichtet, da sie weiter von der berechneten Kurve entfernt sind als Punkte, die nahe an der Regressionskurve liegen. Beispiele für mit der robusten Regression berechneten Klassifikationskarten finden sich in Abbildung 6.4 für die Sequenz Hydrangea und Abbildung 6.5 für die Sequenz RubberWhale. Im Falle der robusten Regression wird im Rahmen der Gewichtungsfunktion ψ ein Parameter λ eingeführt (detaillierte Erklärung in Abschnitt 3.3), der als Kontrastparameter dient. Ist der Fehler der berechneten Kurve am Datenpunkt deutlich größer als λ, so erhält der Punkt nur eine sehr geringe Gewichtung und ist weniger entscheidend für die Berechnung. Die Auswirkungen des Parameters λ auf die Trajektorienklassifikation zeigen die Klassifikationskarten in Abbildung 6.5 und Abbildung 6.4. Allerdings sind die Unterschiede für verschiedene Werte des Parameters λ nur sehr gering bis kaum sichtbar, sodass vermutet werden kann, dass auch



(a) $\lambda = 0.2$ **(b)** $\lambda = 0.1$

(c) $\lambda = 0.05$ **(d)** $\lambda = 0.02$

Abbildung 6.5: Robuste Regression für die Sequenz RubberWhale, $T_a = 0.028, T_b = 0.014$

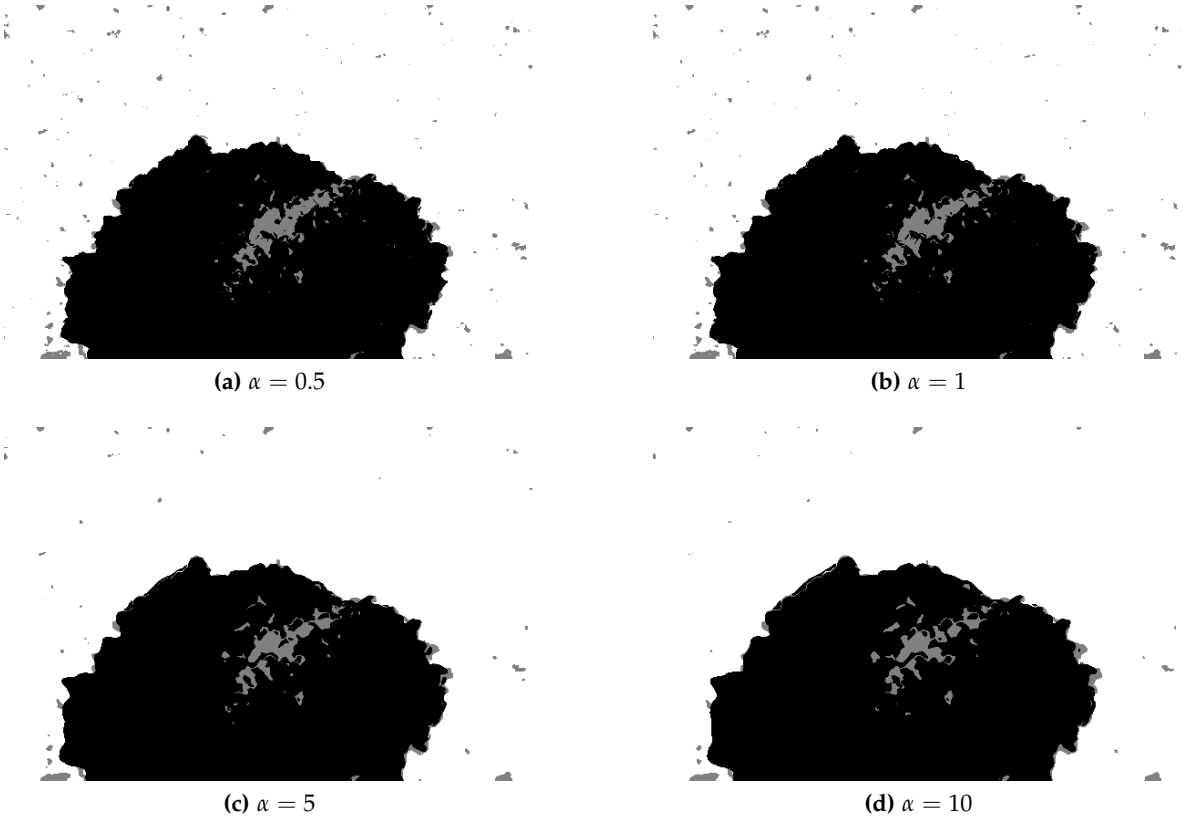
der Einsatz der robusten Regression im Vergleich zur linearen Regression nur zu marginal verbesserten Ergebnissen führen wird.

Im Rahmen dieser Bachelorarbeit zeigt die robuste Regression nicht die gewünschte Wirkung und die Ergebnisse unterscheiden sich nicht erwähnenswert von den Klassifizierungskarten, die durch das Durchführen der lineare Regression entstehen. Dies spiegelt sich auch in den zuvor erwähnten und abgebildeten Klassifizierungskarten im Vergleich zur Klassifikationskarte in Abbildung 6.2 wider. Grund hierfür ist sicherlich auch die geringe Anzahl an Eingabedaten, die lediglich aus vier Messwerten bestehen.

6.2.3 Variationale Methode

Ohne Gewichtung der Parameter

Die variationale Methode, die auch die Umgebung einer Trajektorie in die Berechnung mit einbezieht, führt zu homogeneren Klassifizierungen als dies bei der linearen Regression der Fall ist. Dies bedeutet vor allem vor dem Hintergrund, dass bei der linearen Regression die



(a) $\alpha = 0.5$ **(b)** $\alpha = 1$

(c) $\alpha = 5$ **(d)** $\alpha = 10$

Abbildung 6.6: Gradient-Methode für die Sequenz Hydrangea, $T_a = 0.028$, $T_b = 0.014$

(a) Lineare Regression **(b)** Variationaler Ansatz, $\alpha = 5$

Abbildung 6.7: Variationaler Ansatz in Vergleich zur linearen Regression

Klassifikationskarten sehr verrauscht sind, dieses Rauschen durch den variationalen Ansatz verringert wird.

Dies hat insbesondere mit der Wahl des Parameters α zu tun. Wird dieser nur sehr klein gewählt (<1), so spielt die Umgebung nur eine untergeordnete Rolle bei der Bestimmung der Koeffizienten a, b und c, während bei großen α die Berechnung stark von der Umgebung abhängig ist. Wählt man sogar $\alpha = 0$, so ist man wieder bei der Berechnung der linearen Regression angekommen und die Umgebung spielt in diesem Falle gar keine Rolle. Verdeutlicht wird die Rolle von α auch durch die abgebildeten Sequenzen in Abbildung 6.6. Vorallem bei der Sequenz Hydrangea wird deutlich, dass außerhalb der sich drehenden Hortensie mit größerem α die Trajektorien höherer Ordnung abnehmen und zu einer homogeneren Klassifikationskarte führen. Andererseits wird bei der Sequenz RubberWhale im rechten mittleren Teil in einem schmalen Streifen der Klassifikationskarte deutlich, dass hier Trajektorien höherer Ordnung vorliegen, die in der linearen Regression nicht ersichtlich sind.

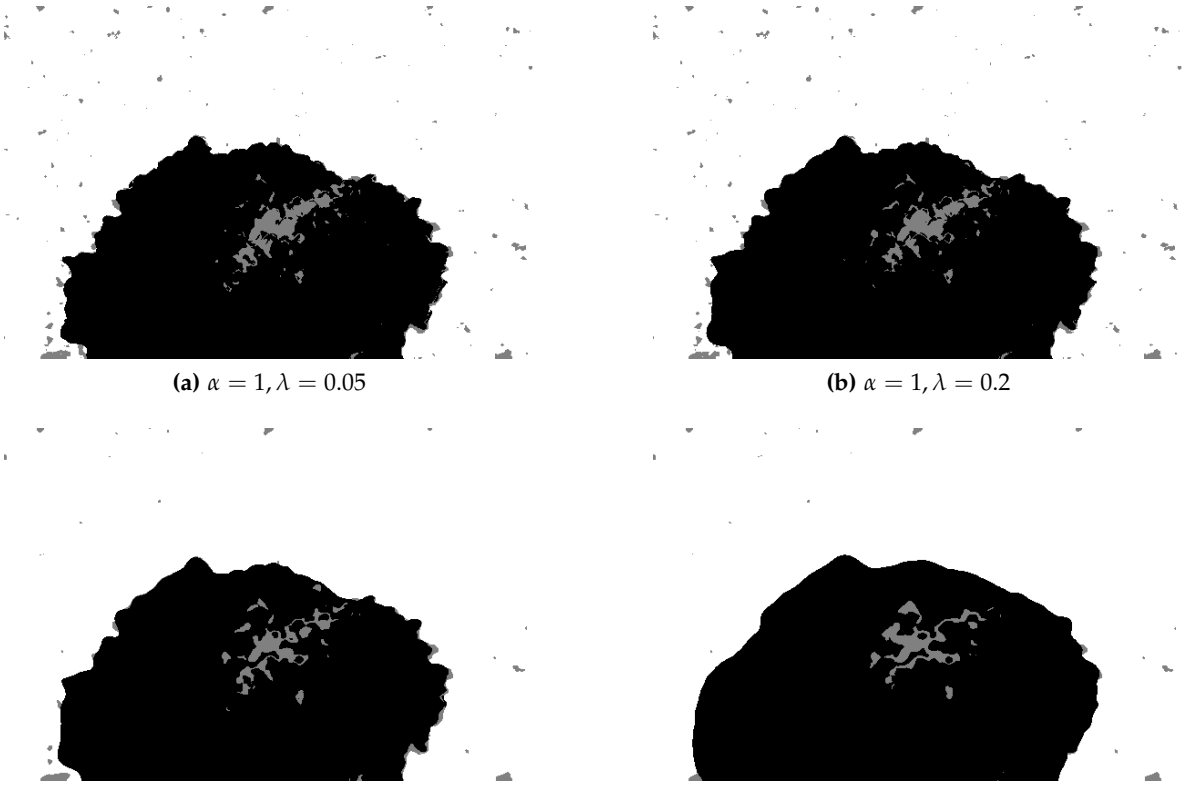
Im Gegensatz zur robusten Regression führt die variationale Methode zur Trajektorienklassifikation, die während dieser Bachelorarbeit entwickelt wurde, zu deutlich unterscheidbaren Ergebnissen im Vergleich mit Klassifikationskarten der linearen Regression. Rein optisch betrachtet lässt sich auf eine Verbesserung der Klassifikation durch die Reduktion des Rauschens in den Klassifikationskarten schließen. Ob sich dies auch positiv auf die Schätzung des optischen Flusses auswirkt, kann allerdings erst mit der Evaluierung der einzelnen Verfahren im nächsten Abschnitt beurteilt werden.

Mit Gewichtung der Parameter

Im Vergleich zum vorherigen Unterkapitel wird die variationale Methode nun dahingehend erweitert, dass zum einen eine lokale Gewichtungsfunktion analog zur robusten Regression eingeführt wurde, zum anderen einmal eine globale Gewichtungsfunktion für alle Gradienten sowie eine Gewichtungsfunktion für jeden einzelnen Gradienten hinzugefügt wurde. Dementsprechend wird dieses Unterkapitel in drei Abschnitte gegliedert.

Lokale Gewichtungsfunktion
Betrachtet man die entstehenden Klassifikationskarten in Abbildung 6.8 kann man beobachten, dass mit größerem λ für die lokale Komponente die Klassifikationskarte homogener wirkt und weniger Rauschen in dieser vorhanden ist. Betrachtet man beispielsweise in Abbildung 6.8 die Abbildungen a) und b) erkennt man, dass in Abbildung a) einzelne Trajektorien innerhalb einer homogenen Klassifizierungsumgebung herausstechen und anders klassifiziert sind. Dies ist bei der Abbildung b) mit größerem λ nicht der Fall.

Besonders zu beachten ist hierbei ein Sonderfall, bei dem für kleine Gewichtungen der lokalen Terme sowie einem großen Einfluss der globalen Komponente (großes α) die Gleichung für erlaubte Fehler $\epsilon \approx 10^{-3}$ bereits bei der ersten Iteration gelöst ist. Gleichzeitig steht man hier nun auch vor dem Problem, dass bei der Wahl eines kleineren ϵ die Berechnung sowohl mit dem Gauss-Seidel-Verfahren als auch mit dem SOR-Verfahrens deutlich verlängert wird. Während das Gauss-Seidel Verfahren beispielsweise ursprünglich mit $\approx 50 - 100$



(a) $\alpha = 1, \lambda = 0.05$ **(b)** $\alpha = 1, \lambda = 0.2$

(c) $\alpha = 10, \lambda = 0.05$ **(d)** $\alpha = 10, \lambda = 0.2$

Abbildung 6.8: Gradient-Methode für die Sequenz Hydrangea, $T_a = 0.028$, $T_b = 0.014$

Iterationen ausgekommen ist, werden in diesem Fall ungefähr 1000 Iterationen benötigt, um die Fehlerschranke zu erfüllen. Dieser Effekt ist beim SOR-Verfahren nicht so extrem ausgeprägt, wenngleich die Anzahl der Iterationen auch hier steigt. Gleichzeitig zeigt sich in diesem Fall, dass sich die entstandenen Klassifikationskarten der beiden Verfahren für $\epsilon = 2 * 10^{-5}$ merklich unterscheiden, wie sich in Abbildung 6.9 zeigt.

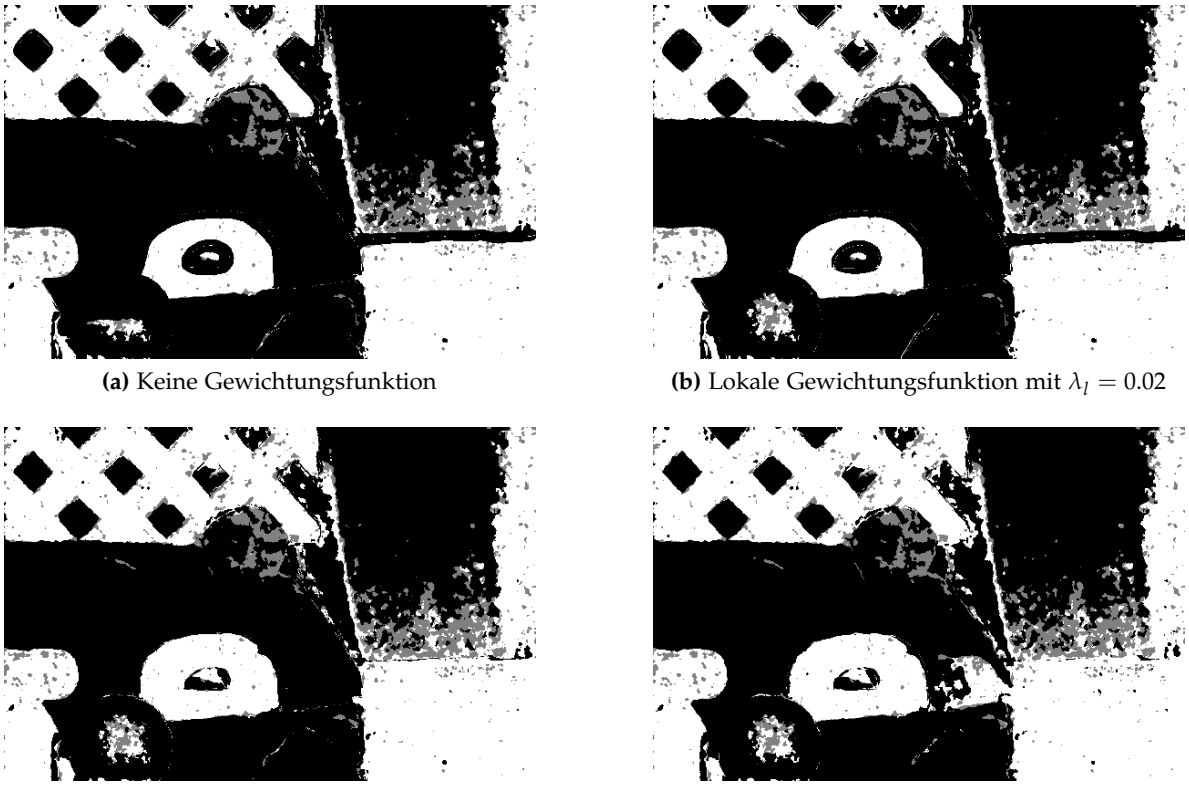
(a) $\alpha = 10, \lambda = 0.2 (SOR)$ **(b)** $\alpha = 10, \lambda = 0.2 (GS)$

Abbildung 6.9: Unterschiedliche Ergebnisse für Gauss-Seidel und SOR

Lokale und globale Gewichtungsfunktion (pro Gradient und für alle Gradienten)
Wie die berechneten Klassifikationskarten in Abbildung 6.10 zeigen, ändern sich mit der Einführung der Gewichtungsfunktionen die Klassifikationskarten recht deutlich.

Wie in Abbildung 6.10 zu sehen ist, wird durch die Einführung einer lokalen Gewichtungsfunktion das Ergebnis kaum beeinflusst, während durch die globalen Gewichtungsfunktionen das Ergebnis sichtbar beeinflusst wird. Zu sehen ist dies in der Klassifikationskarte vorallem im rechten Bereich. Wo sich in den Abbildungen a) und b) noch ein deutlich sichtbarer "Streifen", in dem laut Klassifikation eine polynomielle Bewegung stattfindet, befindet, ist dieser in Abbildung c) und d) zwar noch vorhanden, aber für den Betrachter deutlich weniger auffällig. Zudem führt unterhalb der Mitte des Bildes bei den beiden Klassifikationsverfahren, bei denen einmal globale Glattheit der Parameter und einmal Glattheit für jeden Parameter seperat gefordert ist, zu unterschiedlichen Klassifikationen der Trajektorien (vgl c) und d) in Abbildung 6.10). Abzuwarten bleibt, ob dies letztendlich zu einem Vorteil bei der Schätzung des optischen Flusses führt.

Wie allerdings in Abbildung 6.11 zu sehen ist, kann bereits durch eine geringfügige Änderung des lokalen Gewichtungsparamets λ_l erreicht werden, dass Klassifikationskarten mit deutlich schlechterer Qualität berechnet werden. In der Abbildung ist in Teilbild b) zu sehen, dass Objekte im Gegensatz zu anderen Klassifikationskarten nicht mehr zusammenhängend klassifiziert werden. Vielmehr verschwimmen hier die Abgrenzungen zwischen den Objekten, die in den bisherigen globalen Klassifikationsverfahren klar auszumachen sind und in den lokalen Verfahren zumindest erahnt werden können. Reguliert man allerdings die Gewichtungsfunktion für die Glattheit bezüglich des Koeffizienten c herunter (zu sehen in Teilbild c)), sodass hier nur eine sehr geringe Gewichtung erreicht wird, erhält man wieder eine Klassifikationskarte, die die Grenzen zwischen den einzelnen Objekten wieder erkennen lässt.



(a) Keine Gewichtungsfunktion

(b) Lokale Gewichtungsfunktion mit $\lambda_l = 0.02$

(c) Lokale + Globale Gewichtungsfunktionen mit $\lambda_l = 0.02$ und $\lambda_g = 0.1$

(d) Lokale + Globale Gewichtungsfunktion mit $\lambda_l = 0.02$ und $\lambda_a = \lambda_b = \lambda_c = 0.1$

Abbildung 6.10: Beispiele für lokale und globale Gewichtungsfunktionen

(a) Globale Gewichtungsfunktion mit $\lambda_g = 0.1$

(b) Globale Gewichtungsfunktionen per Koeffizient mit $\lambda_a = \lambda_b = \lambda_c = 0.1$

(c) Globale Gewichtungsfunktionen per Koeffizient mit $\lambda_a = \lambda_b = 0.1, \lambda_c = 0.0001$

Abbildung 6.11: Bei allen Bildern: Lokale Gewichtungsfunktion mit $\lambda = 0.05$

6.3 Quantitativer Vergleich der Klassifizierungsverfahren

Nachdem die Charakteristika der einzelnen Methoden zur Klassifikation im vorausgehenden Unterkapitel aufgezeigt wurden, sollen nun die Verfahren vorallem im Hinblick auf die Ergebnisse, die der Benchmark der Universität Middlebury für die einzelnen Klassifikationskarten liefert, untereinander verglichen werden. Hierzu wird das Programm zur Evaluierung benutzt, welches zu Beginn des Kapitels (in 6.1) vorgestellt wurde. Um platzsparende Analysen zu ermöglichen, werden im Folgenden die beiden Sequenzen mit Hydra anstatt Hydrangea und RW statt RubberWhale abgekürzt.

6.3.1 Baseline-Verfahren

Sequenz	**Verfahren**	α	ψ_l	ψ_g	**AEE**	**AAE**	**BP**
Hydrangea	Baseline	-	-	-	0.1344	1.6706	0.17%
RW	Baseline	-	-	-	0.0712	2.3476	0.09%

Zu Beginn wurden von beiden Sequenzen Ergebnisse ermittelt, wenn kein Regularisierungsverfahren im Vorfeld angewendet wurde.

6.3.2 Lokale Verfahren

RubberWhale

Verfahren	T_a	T_b	λ	**AEE**	**AAE**	**BP**
Lineare Regression	0.028	0.014	-	0.0693	2.2917	0.09%
Lineare Regression	0.05	0.025	-	0.0716	2.3466	0.08%
Lineare Regression	0.1	0.05	-	0.0779	2.4967	0.09 %
Lineare Regression	0.2	0.1	-	0.083	2.651	0.08%
Robuste Regression	0.028	0.014	0.02	0.0693	2.2918	0.09%
Robuste Regression	0.028	0.014	0.05	0.0693	2.2917	0.09%
Robuste Regression	0.028	0.014	0.1	0.0693	2.2911	0.09%
Robuste Regression	0.028	0.014	0.2	0.0693	2.2915	0.09%

Wie in der Tabelle zu sehen ist, unterscheiden sich die einzelnen Testresultate für die lineare beziehungsweise robuste Regression selbst mit unterschiedlichen Parametern nur sehr gering voneinander. Grund hierfür könnte sein, dass nur eine geringe Anzahl an Trajektorien unterschiedlich zwischen den einzelnen Verfahren klassifiziert werden. Immerhin können alle Verfahren verbesserte Werte gegenüber dem Baseline-Verfahren erzielen, bei dem keine Regularisierung vorgenommen wurde.

Aufgrund der Tatsache, dass bereits bei der linearen Regression die Verfahren mit höheren Schwellenwerten als $0.028(T_a)$ bzw. $0.014(T_b)$ deutlich schlechtere Ergebnisse aufgewiesen haben, wurden im weiteren Verlauf für diese Schwellenwerte keine weiteren Ergebnisse

gelistet. Stichproben zeigen hier, dass auch andere Verfahren bei höheren Schwellenwerten nicht an die Ergebnisse der Schwellenwerte $0.028(T_a)$ und $0.014(T_b)$ herankommen.

Hydrangea

Verfahren	T_a	T_b	λ	**AEE**	**AAE**	**BP**
Lineare Regression	0.028	0.014	-	0.1345	1.7178	0.17%
Lineare Regression	0.05	0.025	-	0.1344	1.7299	0.18%
Lineare Regression	0.1	0.05	-	0.1355	1.7556	0.18%
Lineare Regression	0.2	0.1	-	0.1431	1.8814	0.18%
Robuste Regression	0.028	0.014	0.02	0.1346	1.7184	0.17%
Robuste Regression	0.028	0.014	0.05	0.1345	1.7179	0.17%
Robuste Regression	0.028	0.014	0.1	0.1345	1.7178	0.17%
Robuste Regression	0.028	0.014	0.2	0.1345	1.7178	0.17%

Die Ergebnisse der Sequenz Hydrangea können die Eindrücke aus der Evaluierung der Sequenz RubberWhale nur bestätigen. Die Ergebnisse der lokalen Verfahren unterscheiden sich auch hier nur marginal voneinander. Auch hier werden nur wenige Trajektorien unterschiedlich klassifiziert, sodass die oben vermuteten Gründe auch auf diese Sequenz zutreffen könnten. Wenn dies der Fall ist, könnte eine allgemeine Aussage hierzu getroffen werden.

6.3.3 Globale Verfahren

Betrachtet wird im Folgenden nur noch die Sequenz RubberWhale, da aufgrund der Vielzahl an möglichen Kombinationen eine Betrachtung beider Sequenzen den Rahmen dieser Bachelorarbeit sprengen würde. Zu beachten bleibt bei der folgenden Tabelle, dass in der Tabellenspalte, die den Parameter λ wiedergibt, eine Unterteilung zum einen in das lokale, zum anderen in die bis zu drei globalen λ-Werte stattfindet.

Verfahren	T_a	T_b	α	λ(**lokal/global**)	**AEE**	**AAE**	**BP**
Var. Methode	0.028	0.014	0.5	-	0.0693	2.2914	0.09%
Var. Methode	0.028	0.014	1	-	0.0693	2.2913	0.09%
Var. Methode	0.028	0.014	5	-	0.0694	2.2921	0.09%
Var. Methode	0.028	0.014	0.5	0.1	0.0693	2.2921	0.09%
Var. Methode	0.028	0.014	1	0.1	0.0693	2.2921	0.09%
Var. Methode	0.028	0.014	5	0.1	0.0694	2.2917	0.09%
Var. Methode	0.028	0.014	0.5	(0.05)/(0.1/0.1/0.1)	0.0694	2.2950	0.09%
Var. Methode	0.028	0.014	1	(0.05)/(0.1/0.1/0.1)	0.0694	2.2935	0.09%
Var. Methode	0.028	0.014	5	(0.05)/(0.1/0.1/0.1)	0.0694	2.2924	0.09%
Var. Methode	0.028	0.014	0.5	(0.05)/(0.1/0.1/10000)	0.0693	2.2921	0.09%
Var. Methode	0.028	0.014	1	(0.05)/(0.1/0.1/10000)	0.0693	2.2895	0.09%
Var. Methode	0.028	0.014	0.5	(0.1)/(0.1)	0.0693	2.2936	0.09%
Var. Methode	0.028	0.014	1	(0.1)/(0.1)	0.0694	2.2965	0.09%
Var. Methode	0.028	0.014	5	(0.1)/(0.1)	0.0694	2.2934	0.09%

Es lässt sich auch mit unterschidlich gewählten Parametern für die globalen Verfahren kaum ein Unterschied zu den Ergebnissen der Trajektorienklassifikation aus dem TC-Flow Verfahren feststellen. Lediglich die globale Methode mit relativ konstanter Gewichtung für den Koeffizienten c (erzielt durch $\lambda_c = 10000$), kann mit einem geringeren Winkelfehler gegenüber den anderen Ergebnissen aufwarten. Ansonsten herrschen nur marginale Unterschiede zwischen den einzelnen Verfahren vor.

Auch wenn in der obigen Tabelle die Ergebnisse für die Sequenz Hydrangea nicht aufgelistet sind, zeigten Versuche, dass die Schätzung des optischen Flusses nicht von der globalen Komponente, die in diesen Versuchen hinzugefügt wurde, verbessert wurde.

Schlussendlich zeigt sich nach der Evaluation, dass trotz subjektiv verbesserter Klassifikationskarten, auf denen das Rauschen durch den global wirkenden Glattheitsterm entfernt wurde, keine Verbesserung der Schätzung des optischen Flusses erzielt werden kann. Begründet kann dies vor allem mit der Tatsache werden, dass zwar die Klassifikationskarten mit lediglich lokal wirkenden Verfahren innerhalb des TC-Flow Verfahrens berechnet wurden, die Regularisierung aber dennoch global wirkt. Grund hierfür ist das Verfahren zur Schätzung des optischen Flusses, welches bereits räumliche Glattheit annimmt, wodurch letztendlich auch die Regularisierung, wenngleich lokal berechnet, Auswirkungen auf Trajektorien in der Umgebung hat.

7 Zusammenfassung und Ausblick

In dieser Bachelorarbeit wurden verschiedene Verfahren eingeführt und implementiert, um im Rahmen des TC-Flow Verfahrens eine Klassifikationskarte zu erstellen, die entsprechend auf die Regularisierung einzelner Trajektorien einwirkt.

Hierfür wurde zuerst das in [VBVZ11] beschriebene Regressionsverfahren zum Anpassen des Grades der Regularisierung zuerst nachvollzogen, anschließend implementiert und evaluiert. Anschließend erfolgte dann eine Erweiterung des bisher rein lokalen Verfahrens auf den Umgebungskontext. Hierzu wurde ein variationaler Ansatz gewählt.
In der Betrachtung der Klassifikationskarten können dabei deutliche Unterschiede zwischen einzelnen Verfahren festgestellt werden. Während bei der linearen Regression teilweise sehr verrauschte Klassifikationskarten das Ergebnis der Trajektorienklassifizierung sind und die Erweiterung zur robusten Regression ebenfalls keine nennenswerte Veränderung bringt, eliminiert ein variationaler Ansatz mit entsprechend großer Gewichtung des Glattheitsterms dieses Rauschen und führt somit zu deutlich erkennbarer räumlicher Glattheit in der Klassifizierung.

Allerdings war diese Glattheit bei der Trajektorienklassifizierung in der Abschätzung des optischen Flussfeldes je nach Sequenz nicht hilfreich und teilweise sogar hinderlich, wie die anschließende Evaluierung mithilfe eines Evaluierungstools und Traininigssequenzen des Middlebury-Benchmarks zeigte. Begründet kann dies dadurch werden, dass eine räumliche Glattheit zwischen den Trajektorien bereits in der Berechnung der Flussfelder im Vorfeld der Trajektorienklassifizierung angenommen wird.

7.1 Ausblick

Trotz des eher mäßigen Erfolges im Hinblick auf die Benchmarkergebnisse zur Schätzung des optischen Flussfeldes können weitere Vorschläge zu einer möglichen, erfolgreichen Erweiterung dieser Bachelorarbeit gemacht werden.
Zuallererst können natürlich weitere Experimente mit veränderten Parametern auf Basis der vorgestellten Verfahren durchgeführt werden.
Darüberhinaus kann das in dieser Bachelorarbeit durchgeführte Schwellwertverfahren auf Parameterbasis dahingehend verändert werden, dass das Schwellwertverfahren auf Verfahrensbasis arbeitet. Dies ist dadurch erreichbar, dass lokale Regressionsverfahren mit aufsteigendem Grad gelöst werden, bis ein Verfahren ein Ergebnis produziert, das ein hinreichend gutes Approximationsergebnis liefert.

Außerdem kann bei den globalen Verfahren durch die Einführung von Gewichtungsparametern vor den Umgebungsabhängigkeiten(Gradienten) der einzelnen Koeffizienten zu verbesserten Ergebnissen führen, da möglicherweise die räumliche Glattheit des Koeffizienten a stärker behandelt werden muss als die räumliche Glattheit bei den Koeffizienten b und c.
Zuletzt könnten die Verfahren in dieser Bachelorarbeit dahingehend erweitert werden, dass mehr Frames als Ausgangsdaten verwendet werden. Vermutlich würde dies vorallem dem Verfahren der robusten Regression zuträglich sein, da durch mehr Eingabeframes Ausreißer zuverlässiger erkannt werden. Interessant ist hier die Frage, ob dies letztendlich auch zu besseren Ergebnissen im Rahmen des Middlebury-Benchmarks führt.

Literaturverzeichnis

[Bar94] Barrett, R.: *Templates for the solution of linear systems: building blocks for iterative methods*. Siam, 1994 (43) (Zitiert auf den Seiten 39 und 42)

[BBPW04] Brox, T. ; Bruhn, A. ; Papenberg, N. ; Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: *Computer Vision-ECCV 2004*. Springer, 2004, S. 25–36 (Zitiert auf Seite 12)

[Bru13a] Bruhn, A.: *Vorlesung Correspondence Problems in Computer Vision, Universität Stuttgart*. http://www.vis.uni-stuttgart.de/nc/lehre/details/typ/vorlesung/1908/120.html. Version: Sommersemester 2013 (Zitiert auf Seite 49)

[Bru13b] Bruhn, A.: *Vorlesung Imaging Science, Universität Stuttgart*. http://www.vis.uni-stuttgart.de/nc/lehre/details/typ/vorlesung/1768/119.html. Version: Sommersemester 2013 (Zitiert auf Seite 46)

[BSL+] Baker, S. ; Scharstein, D. ; Lewis, JP ; Roth, S. ; Black, M. ; Szeliski, R.: *Optical Flow Benchmark, University of Middlebury* (Zitiert auf den Seiten 9 und 15)

[Cra50] Cramer, G.: *Introduction à l'analyse des lignes courbes algébriques*. Chez les frères Cramer et C. Philibert, 1750 http://books.google.de/books?id=HzcVAAAAQAAJ (Zitiert auf Seite 25)

[EZST01] Esch, H. E. ; Zhang, S. ; Srinivasan, M. V. ; Tautz, J.: Honeybee dances communicate distances measured by optic flow. In: *Nature* 411 (2001), Nr. 6837, S. 581–583 (Zitiert auf Seite 13)

[Fis85] Fischer, G.: *Analytische Geometrie*. Bd. 20017. Vieweg Wiesbaden, 1985 (Zitiert auf Seite 26)

[GMF02] Gern, A. ; Moebus, R. ; Franke, U.: Vision-based lane recognition under adverse weather conditions using optical flow. In: *Proc. IEEE Intelligent Vehicle Symposium* Bd. 2, IEEE, 2002, S. 652–657 (Zitiert auf Seite 13)

[MB04] McCarthy, C. ; Barnes, N.: Performance of optical flow techniques for indoor navigation with a mobile robot. In: *Proc. IEEE International Conference on Robotics and Automation* Bd. 5, IEEE, 2004, S. 5093–5098 (Zitiert auf Seite 13)

[Nag] Nagel, H.-H. Institut für Algorithmen und Kognitive Systeme Universität K.: *Image Sequence "Ettlinger Tor"*. http://i21www.ira.uka.de/image_sequences/ (Zitiert auf Seite 11)

[OB12] Ochs, P. ; Brox, T.: Higher order motion models and spectral clustering. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012* IEEE, 2012, S. 614–621 (Zitiert auf Seite 9)

[SBK10] Sundaram, N. ; Brox, T. ; Keutzer, K.: Dense point trajectories by GPU-accelerated large displacement optical flow. In: *Computer Vision–ECCV 2010*. Springer, 2010, S. 438–451 (Zitiert auf Seite 9)

[VBVZ11] Volz, S. ; Bruhn, A. ; Valgaerts, L. ; Zimmer, H.: Modeling temporal coherence for optical flow. In: *Proc. IEEE International Conference on Computer Vision*, IEEE, 2011, S. 1116–1123 (Zitiert auf den Seiten 7, 8, 14, 19, 28, 60, 61 und 75)

Alle URLs wurden zuletzt am 05. 11. 2013 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift