

Lokalisierung von Asterismen mithilfe von Sternkatalogen

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2015

Autor: Max Obrist
Betreuer: Prof. Dr. Andreas Müller

Abstract

Die Disziplin der Astrofotografie befasst sich mit der Aufnahme von Sternen und anderen Himmelskörpern. Mit einer an einem Teleskop befestigten Kamera wird ein Ausschnitt des Sternenhimmels fotografiert. In unserem Fall befindet sich das Teleskop dabei jeweils auf der Erdoberfläche, es könnte sich jedoch grundsätzlich auch im Erdorbit befinden. Die Ausrichtung des Teleskops bestimmt den Mittelpunkt des fotografierten Ausschnittes, während die Brennweite und Winkelauflösung des Teleskops die Grösse des Ausschnittes bestimmt.

Nur mit diesen Informationen kann man bestimmen, welcher Ausschnitt sich genau auf einer Astroatnahme befindet. Wenn diese Informationen fehlen, und es sich nicht um eine Aufnahme eines wohl bekannten Ausschnittes handelt, ist es sehr schwierig herauszufinden, was fotografiert wurde.

In dieser Arbeit wurde ein Algorithmus entwickelt, welcher die Sterne einer solchen Astroatnahme analysiert und versucht, diese mithilfe eines Sternenkataloges am Himmel zu lokalisieren.

Das Problem gleicht entfernt anderen Feature-Detection Problemen, wie Fingerabdruck- oder Gesichtserkennung. Solche Features – eigentlich drei zu Dreiecken kombinierte Sterne – müssen zuerst aus den Punktwolken aus Sternen extrahiert werden. Sind diese berechnet, können mit geschickter Vorselektion und Vergleich Bild- und Katalogfeatures zur Deckung gebracht werden.

Wurde der entsprechende Ausschnitt im Katalog gefunden, wird daraus der Transformationsvektor berechnet, mit welchem der Astrofotograf die Koordinaten von Sternen auf den Katalog transformieren kann.

Ein grösseres Problem stellt die enorme Datenmenge dar. Bei der Umsetzung des Algorithmus musste deshalb auch auf schnelle Berechenbarkeit Rücksicht genommen werden. Der Algorithmus wurde aber nicht besonders auf Parallelität getrimmt oder anderweitig optimiert. Mit entsprechenden Anpassungen könnten die Berechnungen wahrscheinlich stark beschleunigt werden.

Auch nicht Teil dieser Arbeit ist die eigentliche Extraktion der Helligkeit und Positionen der Sterne aus der Astroatnahme. Diese muss vorgängig durchgeführt werden. Der Algorithmus muss ebenfalls noch auf ein sphärisches Koordinatensystem angepasst werden, da in der Arbeit ausschliesslich Berechnungen mit simulierten Daten in einem ebenen Koordinatensystem durchgeführt wurden. Die meisten verwendeten Techniken lassen sich jedoch ohne konzeptionelle Änderungen auch in sphärischen Koordinatensystem einsetzen.

Inhaltsverzeichnis

Abstract	2
Management Summary	4
Lokalisierung von Asterismen	5
1 Einleitung	5
1.1 Übersicht über die Astrofotografie	5
1.1.1 Atmosphärische Fehler	6
1.1.2 Instrumentell bedingte Störungen	6
1.2 Übersicht über den UCAC4-Sternenkatalog	8
2 Ergebnisse	9
2.1 Sterne	10
2.2 Features	11
2.2.1 Eigenschaften	12
2.2.2 Extraktion	12
2.2.3 Qualität	13
2.3 Matching	14
2.3.1 Relative Nähe	14
2.3.2 Feature-Score	16
2.3.3 Matching	17
2.3.4 Transformationsvektor berechnen	19
2.4 Zwischenanalyse Performance	21
2.4.1 Laufzeit Teilschritt 1	22
2.4.2 Laufzeit Teilschritt 2	22
2.4.3 Laufzeit Teilschritte 3 und 4	22
2.4.4 Fazit	22
2.5 Vorselektion	23
2.5.1 Vorselektion mit <i>Ähnlichkeitsinvariante</i>	23
2.5.2 Vorselektion mit Bildmittelpunkt	24
3 Fazit	27
3.1 Simulation	27
3.2 Finalisierungsarbeiten	28
3.3 Optimierungspotential	29
3.4 Mögliche Erweiterungen	30
Literaturverzeichnis	31

Appendix	32
A Baryzentrische Koordinaten	32
B Satz des Heron und <i>Ähnlichkeitsinvariante</i>	34
C Symboltabelle	36
D Anpassbare Parameter	37
E Selbstreflexion	38
F Aufgabenstellung	40
G Eigenständigkeitserklärung	41
H Einverständniserklärung Publikation	42

Management Summary

Fehlen bei einer Astroaufnahme wichtige Informationen über die Ausrichtung und Brennweite des Teleskops sowie der Pixelgrösse der Kamera, ist es fast unmöglich, genau herauszufinden, welcher Ausschnitt des Himmels in einer Astroaufnahme dargestellt wird.

Der in dieser Arbeit vorgestellte Algorithmus löst dieses Problem und versucht, die Aufnahme mithilfe eines Sternenkataloges am Himmel zu lokalisieren. Ist der Ausschnitt lokalisiert, werden dem Anwender Mittel zur Verfügung gestellt, wie er die Koordinaten der Sterne in der Aufnahme in Katalogkoordinaten umwandeln kann, sodass jeder Bildstern eindeutig einem Katalogstern zugewiesen werden kann.

Diese Arbeit befasst sich primär mit der Lösbarkeit des Problems. Bevor der Algorithmus im realen Umfeld eingesetzt werden kann, sind damit noch einige Anpassungen nötig. Besonders wurde der Algorithmus nur für ein ebenes Koordinatensystem umgesetzt, Anpassungen an ein sphärisches Koordinatensystem sind somit noch notwendig.

Es wurde zwar Rücksicht auf gute Berechenbarkeit der einzelnen Teilschritte gelegt, es wurde aber keine eigentliche Optimierung auf Performance durchgeführt. Ebenfalls ist Extraktion der Positions- und Helligkeitsdaten der Sterne einer Astroaufnahme nicht Teil dieser Arbeit.

Lokalisierung von Asterismen

1 Einleitung

Die Astrofotografie ist jene Disziplin der Fotografie, welches sich mit dem Festhalten von Sternen und anderen Himmelskörpern befasst. Dabei wird mit einem meist auf der Erdoberfläche positionierten Teleskop ein Himmelsbereich aufgenommen. Es gibt natürlich auch Teleskope wie das Hubble Space Telescope oder das Kepler Teleskop, welche sich in einem Orbit um die Erde befinden. Diese Arbeit befasst aber sich ausschliesslich mit Astroatnahmen, welche von der Erde erzeugt wurden.

Der Fotograf bestimmt während dem Aufnahmeprozess diverse Faktoren, welche Einfluss auf die erstellten Aufnahmen haben. Mit der Brennweite bestimmt er die Vergrösserung der Aufnahme, was zusammen mit der Pixelgrösse des CCD-Chips der Kamera die Auflösung des Bildes bestimmt. Der Blickwinkel bestimmt den aufgenommenen Bereich und mit einer längeren Belichtungszeit lassen sich Objekte fotografieren, die normalerweise nicht sichtbar wären.

Ein sorgfältiger Astrofotograf speichert all diese Informationen gemeinsam mit seinen Aufnahmen, zusammen mit der Ausrichtung, also Rektaszension und Deklination, welche das Teleskop während der Aufnahme hatte.

Trotz aller Vorsicht kann es aber passieren, dass diese Informationen verloren gehen. Bei Amateuraufnahmen ist auch denkbar, dass die Informationen gar nie vorhanden waren. Ziel dieser Arbeit ist es, einen Algorithmus zu entwickeln, welcher mithilfe der aus einer Astroatnahme extrahierten Positions- und Helligkeitsinformation der Sterne herausfinden kann, mit was für Einstellungen eine Aufnahme erzeugt wurde, bzw. mit welchen Transformationen man die Sterne der Aufnahme, respektive deren Bildkoordinaten, auf den Katalog transformieren könnte.

Ziel dieser Arbeit ist nicht, ein fertiges Produkt zu entwickeln, welches der angesprochene interessierte Astrofotograf nehmen und direkt einsetzen könnte. Inhalt dieser Arbeit ist lediglich die Analyse des Problems sowie die Erarbeitung eines solchen Algorithmus, sofern dies denn möglich ist.

Es ist auch nicht Bestandteil dieser Arbeit, die Positions- und Helligkeitsdaten aus einer Astroatnahme zu extrahieren. Im Zuge der Analyse werden simulierte Daten eingesetzt, um den Algorithmus zu testen.

1.1 Übersicht über die Astrofotografie

Während dem Aufnahmeprozess einer solchen Aufnahme können diverse Phänomene auftreten, welche die Präzision der Aufnahme mindern. Diese müssen berücksichtigt werden, wenn eine Astroatnahme auf welche Art und Weise auch immer in einem Sternenkatalog lokalisiert werden soll.

1.1.1 Atmosphärische Fehler

Viele Probleme entstehen bereits dadurch, dass die von berücksichtigten Astrofotografien von der Erde aus aufgenommen werden. Im Gegensatz zu einem Bild von einem im Erdbereich stationierten Weltraumteleskop, wie dem Hubble Space Telescope, befindet sich zwischen der Kamera und dem Bild die Erdatmosphäre. Diese hat diverse Auswirkungen auf die Genauigkeit des Bildes. In diesen Bereich der atmosphärischen Störungen fallen die Folgenden:

Luftunruhe Die Atmosphäre, respektive die sich darin befindliche Luft ist nie ganz still. Luftunruhen entstehen durch Turbulenzen in der unteren Erdatmosphäre. Dies kann sich im Winter auch von Auge durch ein „Flimmern der Sterne“ bemerkbar machen, es entsteht somit eine Helligkeitsverzerrung der aufgenommenen Sterne, auch Szintillation genannt. Zusätzlich wird auch eine Bildbewegung erzeugt, die zwar nicht von Auge, aber durchaus mit einem Fernrohr bemerkbar sein kann. Diese Bildbewegung kann im Extremfall 5 bis 7 Gradsekunden ausmachen.

Refraktion Refraktion beschreibt den Effekt, dass Licht, welches von aussen auf die Erde trifft, an der Erdatmosphäre gebrochen wird. Licht, welches knapp über dem Horizont auf die Erde trifft, muss längeren Weg gehen, als solches, das vom Zenit her auf uns zukommt. Deshalb wird horizontnahes Licht stärker gebrochen als Licht vom Zenit her. Sterne in Horizontnähe erscheinen aus diesem Grund höher über dem Horizont als sie tatsächlich sind. Auch die Farbe des Lichtes hat einen Einfluss auf die Stärke der Brechung, was einem Regenbogen auch immer wieder schön vor Augen führen.

Streulicht Ein weiterer Effekt, den wir hier den atmosphärischen Störungen zuordnen, auch wenn die Ursache für dieses Problem eher ein zivilisatorisches als ein atmosphärisches ist. Störungen durch Streulicht und Lichtverschmutzung erhellen das gesamte oder Teile des Bildes. Dies kann dazu führen, dass der Hintergrund des Bildes (Optimalerweise ein perfektes Schwarz) in einem Teil des Bildes heller ist, als im Rest des Bildes. Ist dies der Fall, kann dies beim Extrahieren der Sterne aus dem Bild zu Ungenauigkeiten führen.

1.1.2 Instrumentell bedingte Störungen

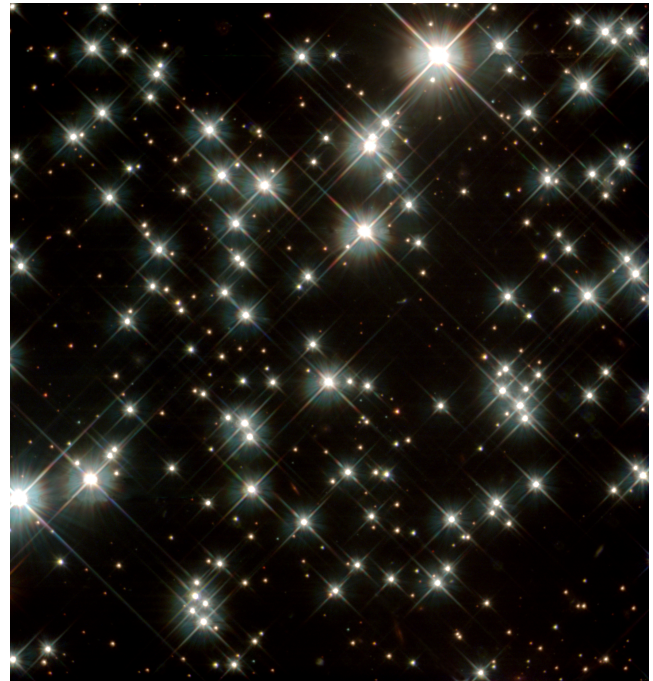
Neben den atmosphärischen Störungen kommt noch eine Reihe weiterer Störungen, die auf technische Ursachen im Teleskop und der Kamera zurückzuführen sind.

Reflexionen in der Optik Lichtreflexionen entstehen an allen Übergängen zwischen optisch unterschiedlich dichten Materialien, so auch an den Linsen und Spiegeln des Teleskops. An jeder Linse und jedem Spiegel entsteht somit Streulicht im Teleskop. Einerseits wird somit nicht das gesamte auftreffende Licht bis zur Kamera weitergeleitet, andererseits können dadurch Artefakte im Bild entstehen, bekannt als Halo um den Himmelskörper herum erscheint. Abbildung 1a zeigt eine Aufnahme des Mondes, indem dieser Haloeffekt besonders stark auftritt.

Halterung der Optik Auch die Halterung der Optik kann einen Einfluss auf das Bild haben. Dadurch entsteht ein Sterneffekt, wie in Abbildung 1b sehr schön ersichtlich. Die Aufnahme zeigt einen Ausschnitt der Milchstrasse, aufgenommen mit dem Hubble Space Telescope. Die vier Teleskophalterungen des Teleskops erzeugen durch Beugung des Lichtes den Sterneffekt.



(a) Haloeffekt[2]



(b) Sterneffekt[1]

Abbildung 1: Fehler durch Reflexionen in der Optik

Verzerrungen in der Optik Bei der Betrachtung des Himmels mit einem Teleskop, bzw. mit jeglichen bildvergrößernden Hilfsmitteln, wie Kamera, Feldstecher, Lupe, o.ä. entstehen vom Bildmittelpunkt ausgehend zu den Rändern eine immer stärkere Verzerrungen des Bildes. Dieser Fehler hat einen Einfluss bei der Positionsbestimmung der Sterne und wird grösser, je näher man sich am Rand des Teleskopes befindet.

Farbempfindlichkeit CCD-Chip Vielfach sind die in Kameras verwendeten CCD-Chips unterschiedliche empfindlich auf verschiedene Farben, was wiederum bei der Analyse der Helligkeit der Sterne zu Problemen führen kann. Leuchtet ein Stern besonders stark in einem Rotton, während die Kamera nur über einen unterproportional empfindlichen Rot-Sensor verfügt, erscheint der Stern im Bild zu dunkel. Während der Aufnahme eingesetzte Spektralfilter können dieses Problem verschärfen.

Stern auf Pixelgrenze Ein weiteres Phänomen ist auf die Art und Weise zurückzuführen, wie moderne Digitalkameras eine Aufnahme erzeugen. Eine solche entsteht, indem das Licht, welches durch das Objektiv eintritt, mittels Spiegel und Linsen auf den CCD-Chip fokussiert wird. Jeder Pixel, welcher am Ende im Bild auftaucht, hat einen physikalischen Pixel dem CCD-Chip (beziehungsweise für jede Farbe einen). Zwischen zwei Pixeln befindet sich immer ein kleiner Abstand, eine Art toter Punkt des CCD-Chips. Befindet sich nun ein Stern genau auf der Pixelgrenze zwischen zwei Pixeln, erscheint er in der finalen Aufnahme dunkler als er tatsächlich ist. Ein Teil des Lichtes ging zwischen den Pixeln verloren. Aus diesem Grund ist zum Beispiel das Kepler Teleskop defokussiert. Sterne werden dadurch über mehrere Pixel verschmiert, sodass die korrekten Daten besser extrahiert werden können.

Spiegel Abhängig von der Anzahl der Spiegel im Teleskop kann es sein, dass das Bild kopfstellt



Abbildung 2: Astroaufnahme mit diversen Fehlern. Der hellste Stern ist Deneb[5]

und somit alle Sterne seitenverkehrt sind. Aus diesem Grund ist es nötig, dass der Vergleich zwischen Aufnahme und Katalog unabhängig von Ausrichtung und Spiegelung der Sterne funktioniert.

Abbildung 2 zeigt eine Astroaufnahme mit diversen verschiedenen Fehlern. Der am hellsten leuchtende Stern ist der Stern Deneb (grün umrandet), welcher einen sehr starken Haloefekt hat. Die weisse Fläche im Inneren des Sternes deutet auf eine starke Überbelichtung hin. Der rechte Teil des Bildes ist allgemein leicht überbelichtet, erkennbar am etwas helleren Hintergrund. Gegen den linken und unteren Rand des Bildes ist auch eine leichte Verzerrung zu erkennen. Besonders gut erkennbar ist dies an den Sternen im orange umrandeten Bereich.

1.2 Übersicht über den UCAC4-Sternenkatalog

Der The Fourth US Naval Observatory CCD Astrograph Catalog[6], oder kurz UCAC4-Katalog, ist die vierte Version des Sternenkatalog des US Naval Observatories. UCAC ist ein astrometrisches Programm, welches im Februar 1998 am Cerro Tololo Inter-American Observatory im Norden von Chile begonnen wurde.

Bis im Mai 2004 wurden die All-Sky-Surveys abgeschlossen und aus diesen im Jahre 2009

der UCAC3 Katalog erstellt. Mit weiteren Fehlerbehebungen, mit Verbesserungen in der Datenreduktion sowie mit Daten aus der APASS5 [7] Photometrien wurde im August 2012 der UCAC4 Katalog veröffentlicht.

Der UCAC4 Katalog beinhaltet eine Liste mit 113'780'093 Objekten. Wie wir sehen werden, ist die wichtigste Information für den von uns entwickelten Algorithmus die Position. Die Magnitude der Sterne wird erst später zum Verfeinern der Resultate benötigt. Aus diesem Grund werden nur diejenigen Sterne aus dem Katalog verwendet, welche als "Guter Stern" (103'080'317) oder als "überbelichtet" (2'785'787) markiert sind.

Die im Katalog zu lokalisierenden Astrofotografien müssen also in diesen rund 106 Mio. Sternen wiedererkannt werden. Wie das geht, wird im weiteren Verlauf dieser Arbeit analysiert.

Über diese Sterne sind im Katalog Informationen zur Magnitude in verschiedenen Spektren abgelegt, sowie Rektaszension und Deklination der Sterne. Dazu kommen noch Informationen über die Standardabweichung für die gemessenen Informationen, sowie weitere Daten, welche für unsere Belange nicht weiter interessant sind.

Die Helligkeit der Sterne am Himmel ist ungefähr Paretoverteilt, wobei es deutlich mehr dunkle als helle Sterne gibt. Die Helligkeit wird deshalb als Magnitude, ein logarithmisches Mass, hinterlegt. Dies hat für uns einen grossen Vorteil. Dadurch werden relative Vergleiche der Helligkeit zweier Sterne zu einer einfachen Differenzbildung. Wir müssen damit keine komplizierten Berechnungen anstellen, wenn wir den relativen Helligkeitsunterschied zweier Sterne wissen wollen.

Da die Magnitude logarithmisch ist, ist ein solcher relativer Vergleich nur noch eine Differenzbildung zweier Magnituden. Hat ein Stern eine um 1 grössere Magnitude als ein zweiter Stern, ist der Erste 2.5 ungefähr Mal Dunkler wie der zweite Stern.

2 Ergebnisse

Somit soll ein Algorithmus zur Lokalisierung von Asterismen im UCAC4-Sternenkatalog erarbeitet werden. Dieser soll die aus einer Astroatnahme extrahierten Positions- und Helligkeitsdaten im UCAC4-Katalog lokalisieren.

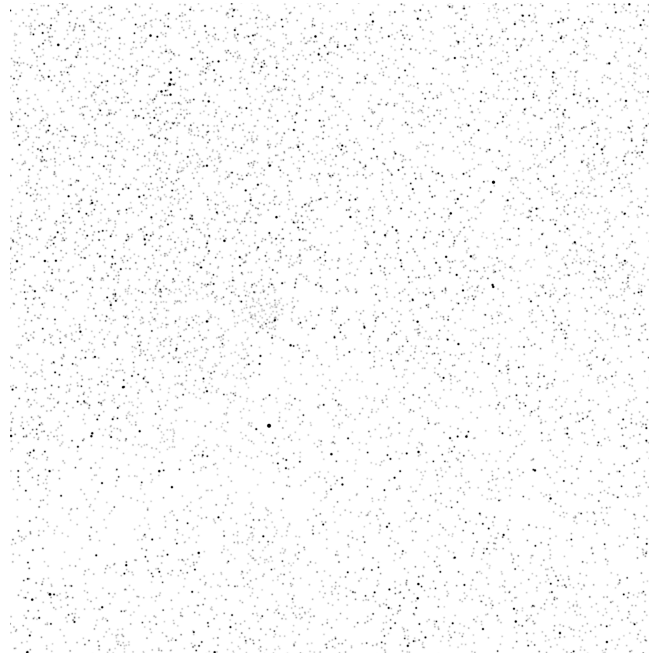
Im Folgenden soll dieser Vorgang kurz analysiert werden. In Abbildung 3a ist dasselbe Bild zu sehen wie in Abbildung 2, nur ohne den Bildmarkierungen. Dieses Bild von Deneb soll nun mit dem UCAC4-Katalog zur Deckung gebracht werden. Zur Veranschaulichung wurde die Abbildung 3b derjenigen Region im Katalog erzeugt, in welcher sich auch Deneb befindet. Deneb ist der dickste Stern im linken, unteren Quadranten.

Wir müssen jetzt herausfinden, wie die Aufnahme rotiert, gestreckt und transformiert werden muss, sodass die Sterne mit dem Katalog zur Deckung gebracht werden. Abbildung 4 zeigt das gewünschte gewünschte Ziel. Deneb wurde zwischen der Aufnahme und dem Katalog identifiziert, genauso wie einige andere Sterne. Die zur Deckung gebrachten Sterne sind schön ersichtlich an den kleinen schwarzen Punkten die von einem etwas grösseren weissen Punkt eingeschlossen sind.

Es wurde also ein Algorithmus entwickelt, welcher ein Bild von Sternen, dargestellt als die Menge S_B mit den Sternen $s_b^1, s_b^2, \dots, s_b^n \in S_B$, in einem Sternenkatalog, S_K bestehend aus den Sternen $s_k^1, s_k^2, \dots, s_k^n \in S_K$, wiederzufinden.



(a) Aufnahme von Deneb[5]



(b) Erzeugter Katalogausschnitt[5]

Abbildung 3: Die Aufnahme und der aus dem UCAC4-Katalog erzeugte Katalogausschnitt, welche zur Deckung gebracht werden sollen

2.1 Sterne

Über einen Stern können diverse Informationen gefunden werden. Einerseits hat ein Stern, ob nun in einem Sternenkatalog oder aus einer Aufnahme extrahiert, eine Position. Im Fall des Kataloges handelt es sich dabei um die Deklination und Rektaszension, bei einer Aufnahme sind nur die x und y Koordinaten des Sternes relativ zum Bild bekannt. Ausserdem hat jeder Stern eine logarithmische Magnitude, wie in Abschnitt 1.2 beschrieben. Da hauptsächlich mit simulierten Daten gearbeitet wurde, werden Koordinaten immer als $(s)_x$ und $(s)_y$ bezeichnet, die Magnitude als $(s)_{\text{mag}}$.

Das Problem ähnelt entfernt anderen Feature-Detection Algorithmen, wie Fingerabdruck- oder Gesichtserkennung. Beide Probleme werden gelöst, indem aus den Bildern des zu erkennenden Objektes Features extrahiert werden. Dabei handelt es sich um eine Kombination von auffälligen Punkten, beim Gesicht z.B. Augen, Nase Mund und Ohren, bei einem Fingerabdruck die Minuzien und verschiedenen Verästelungen der Papillarleisten des Fingers. Sind diese Elemente erkannt, lässt sich darauf die Ausrichtung des Gesichtes gegenüber der Aufnahme erkennen, bzw. die Rotation zwischen zwei Abbildung von Fingerabdrücken.

Dies ist bei einer gewichteten Punktwolke – und nichts Anderes ist eine Menge von Sternen – nicht der Fall. Für einen einzelnen Stern ist das ersichtlich. Fotografiert man dieselbe Sternformation mit unterschiedlicher Belichtungszeit und Brennweite zweimal kann diese einmal einen kleinen Bereich der Aufnahme abdecken mit kleinen, eher dunklen Sternen. Das andere Mal deckt Sie einen grossen Bereich, mit einzelnen, hellen Sternen ab. Handelt es sich nicht zufälligerweise um ein bekanntes Sternbild, erscheint eine solche Sternformation von Sternen ziemlich chaotisch und ungeordnet.

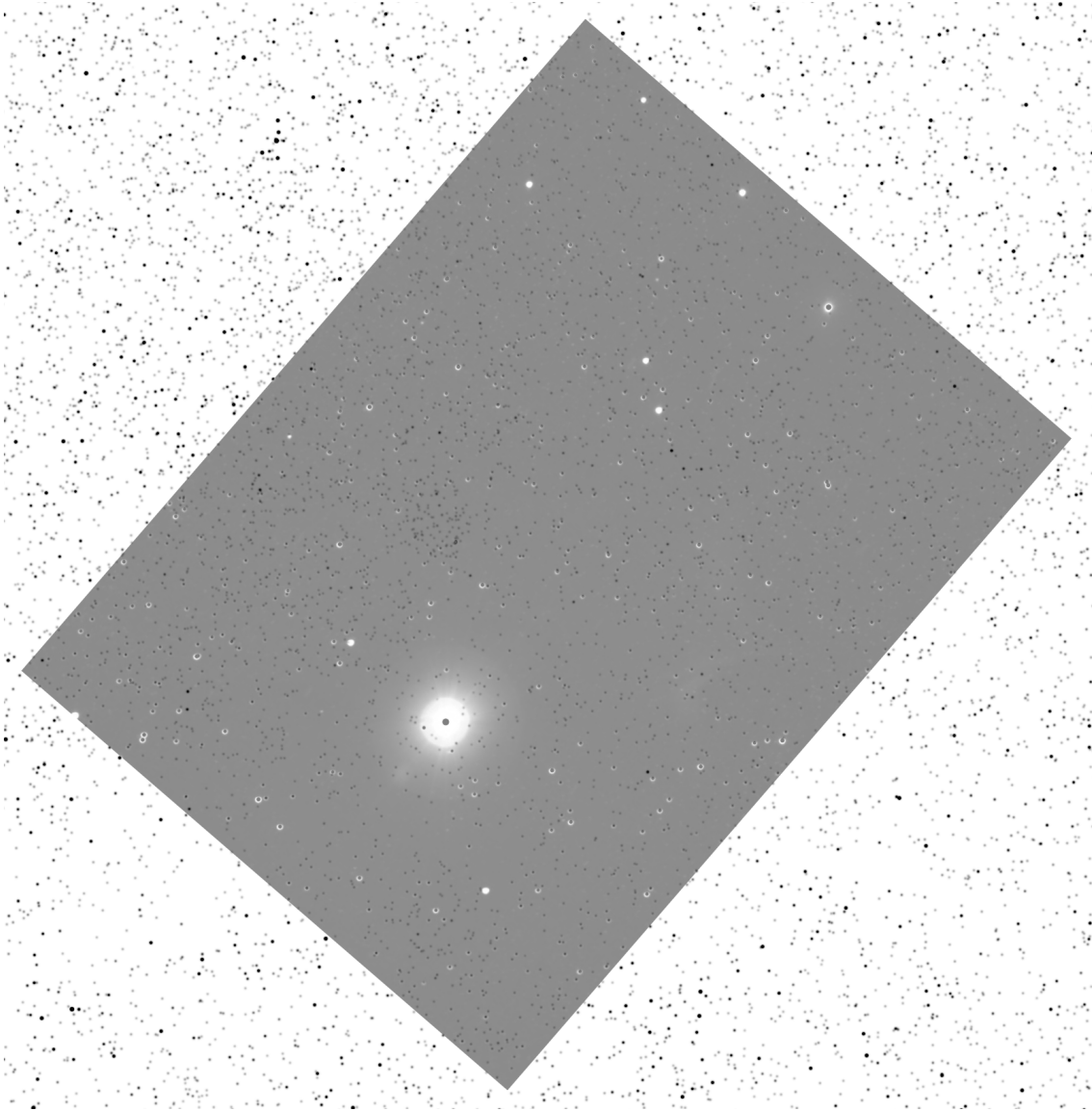


Abbildung 4: Aufnahme und Katalogbild zur Deckung gebracht[5]

2.2 Features

Um eine Sternenwolke in einem Katalog wiederzufinden muss deshalb ein Weg gefunden werden, Struktur in diese Sterne zu bringen. Dadurch erhalten wir später erst die Möglichkeit, ein Mapping einer Astroatnahme auf Sternenkatalog durchzuführen.

Diese Struktur wird mittels sogenannter Features erzeugt. Ein Feature ist eine Kombination aus mehreren Sternen. Es sollen also sowohl im Katalog wie auch in der Aufnahme aus den vorhandenen Sternen auf ganz bestimmte Art und Weise Sterne selektiert und zu Features kombiniert werden, sodass diese später zwischen Aufnahme und Katalog zur Deckung gebracht werden können.

Als Feature empfehlen sich am ehesten einfache Polygone. Polygone mit mehr Ecken haben zwei Probleme. Einerseits sind sie komplexer in der Berechnung, andererseits müssen die Polygone, welche aus S_B und S_K extrahiert werden, später zur Deckung gebracht werden. Je mehr Ecken ein Polygon hat, desto mehr Möglichkeiten gibt es, Polygone aus einer Punktmenge zu

extrahieren ($\binom{1000}{4} \gg \binom{1000}{2}$). Die Entscheidung fiel deshalb auf das einfachstmögliche Polygon als Grundlage für das Feature: Das Dreieck.

2.2.1 Eigenschaften

Ein Feature f ist also eine Kombination aus drei Sternen a , b und c . Dabei sind die Sterne so angeordnet, dass Seite bc am längsten und Seite ab am kürzesten ist.

Da die Features später zur Deckung gebracht werden müssen, müssen sie zuerst normalisiert werden. Das originale Feature darf dabei aber nicht verloren gehen, da sonst die Information über die Position des Features im Sternenkatalog verloren ginge. Ein normalisiertes Feature hat folgende zusätzliche Eigenschaften:

Ausrichtung normalisiert Da es möglich ist, dass das Bild gespiegelt ist, müssen die normalisierten Features Ausrichtungsunabhängig sein. Deshalb soll immer gelten $\overline{bc} \geq \overline{ca} \geq \overline{ab}$.

Grösse normalisiert Da die Grösse eines Features auf dem Bild vor allem von der Brennweite abhängt, müssen die normalisierten Features grössenunabhängig sein. Deshalb werden alle Seiten so genormt, dass Seite A den Wert 1 erhält.

Helligkeit normalisiert Auch die Helligkeit muss normalisiert werden, sodass die Verhältnisse zwischen den Sternen ersichtlich sind.

Da wir Sterne im Feature so angeordnet haben, dass sie nach den Längen der Seiten angeordnet sind, ist die Ausrichtung bereits soweit normalisiert, dass wir diese nicht mehr anpassen müssen. Werden einzelne Sterne eines Features referenziert, werden diese als f^a , f^b und f^c bezeichnet. Wie alle Indexe in dieser Arbeit, werden auch die Sternindexe hochgeschrieben.

Betrachtet man die normalisierte Grösse, stellt man fest, dass es sich hier eigentlich nur um zwei neue Werte handelt. Seite \overline{bc} soll immer 1 sein. Damit müssen nur \overline{ab} und \overline{ca} gespeichert werden müssen. In Appendix B wird dies etwas genauer angeschaut und die beiden Werte dort als *Ähnlichkeitsinvariante* bezeichnet. Analog den Koordinaten bei Sternen, wird für diese die Notation $(f)_{sb}$ und $(f)_{sc}$ verwendet.

Mit diesen Informationen können die Features später zur Deckung gebracht werden.

2.2.2 Extraktion

Die beiden Mengen S_B und S_K müssen also überführt werden in die beiden Featuremengen $f_B^1, f_B^2, \dots, f_B^n \in F_B$ und $f_K^1, f_K^2, \dots, f_K^n \in F_K$, welche später partiell zur Deckung gebracht werden können. Um zu gewährleisten, dass sowohl im Katalog wie auch im Bild Features aus denselben zueinander gehörenden Sternen generiert werden, muss der Algorithmus die Features auf eine Art selektieren, die sowohl im Katalog wie auch in der Aufnahme zu ähnlichen Features führen. Features sollten deshalb nur Sterne beinhalten, welche eine ähnliche Magnitude haben. Ausserdem sollten die Sterne der Features möglichst nahe beieinander liegen.

Diese Kombination von Eigenschaften können wir uns zunutze machen. Wie in Abschnitt 1.2 beschrieben, ist die Magnitude der Sterne im Katalog ungefähr Pareto-Verteilt. Dies bedeutet, dass es deutlich mehr dunkle Sterne gibt als helle. Dies bedeutet auch, dass die mittlere Distanz zwischen Sternen im Katalog mit einer hohen Magnitude (dunkel) geringer ist, als zwischen Sternen mit einer tiefen Magnitude (hell). Daraus ergibt sich, dass die Magnitude der Sterne die Grösse der Features vorgibt: Sucht man für einen hellen Stern s die beiden nächsten Sterne,

deren Magnitude sich mehr als um Δb von s unterscheidet, wird eine grössere Fläche abgedeckt, als wenn dies für einen dunklen Stern durchgeführt wird.

In Abschnitt 1.1 haben wir mögliche Fehler in Astroaufnahmen analysiert. Wir haben herausgefunden, dass es möglich ist, dass die Helligkeit der Sterne im Bild falsch wiedergegeben wird. Somit kann es sein, dass ein Stern, dessen Magnitude sich eigentlich um weniger als Δm von $(s)_{\text{mag}}$ unterscheidet, auf der Aufnahme so dargestellt wird, als wäre die Magnitudenunterschied grösser. In diesem Fall erhalten wir bei der Suche nach ähnlich hellen Sternen eine andere Menge zurück. Es kann also sein, dass wir in der Aufnahme ein Feature extrahieren, welches sich so gar nicht im Katalog befindet. Auch für Features am Rande der Aufnahme kann es vorkommen, dass der Stern, welcher im Katalog zur Extraktion des Features verwendet wurde, im Bild gar nicht sichtbar ist.

Solche Situationen müssen während der Extraktion berücksichtigt werden. Ein Ansatz ist, für jeden Stern mehrere Features zu extrahieren. So wird die Wahrscheinlichkeit, dass sowohl im Katalog wie auch in der Aufnahme dieselben Features extrahiert wurden, bedeutend grösser. Zumindest wenn es denn überhaupt sinnvolle andere Sterne gibt, um ein Feature zu generieren. Man kann sich zum Beispiel vorstellen, dass Sirius, der hellste Stern am Nachthimmel, ein äusserst schlechter Kandidat ist, da er beinahe doppelt so hell, wie der zweithellste Stern Canopus ist.

2.2.3 Qualität

Bisher haben wir nur analysiert, wie wir aus der Menge von Sternen auf eine sinnvolle Art und Weise Features extrahieren können. Wir haben aber überhaupt nicht berücksichtigt, wie gut sich gewisse Features für das spätere Matching eignen. Wir wissen, dass beim Aufnahmeprozess diverse Störungen entstehen. Problematisch sind in diesem Schritt vor allem die Positionsverschiebungen. Die Features sollen später benutzt werden, um auch die Position anderer Sterne zu vergleichen (befindet sich relativ zum Bildfeature ein Stern, muss an selber relativer Position zum Katalog-Feature ebenfalls ein Stern zu finden sein).

Das grosse Problem sind nicht zwingend die Positionen der Sterne selbst, doch je spitzer der einzelnen Winkel des Feature im geometrischen Sinn sind, desto stärker wird die Koordinatenrechnung von einer Positionsverschiebung eines der Bildsterne beeinflusst. Es sollen also nur Features extrahiert werden, bei welchen alle Winkel einen gewissen Wert nicht unterschreiten.

Da es sich bei trigonometrischen Funktionen allerdings um sehr berechnungsintensive Funktionen handelt, haben wir uns entschieden, hierfür nicht direkt Winkel zu vergleichen, sondern ein anderes Mass zu wählen. In Appendix B wird erläutert, wie basierend auf dem Satz des Heron ein Wert berechnet werden kann, wie Spitz ein Dreieck ist. Mit der Formel $\text{Area}/\overline{bc}^2$ kommt ein Wert heraus, der bei spitzen Winkeln immer kleiner wird. Als Grenzwert wurde in diesem Abschnitt der Wert 0.15 erarbeitet.

Auch die Berechnung der Fläche benötigt normalerweise eine teure Wurzeloperation, und wurde deshalb durch die Formel

$$\frac{1}{2} \left| (f^b)_x ((f^c)_y - (f^a)_y) + (f^c)_x ((f^a)_y - (f^b)_y) + (f^a)_x ((f^b)_y - (f^c)_y) \right|$$

ersetzt. Es werden somit nur Features verwendet, welche die folgende Gleichung erfüllen:

$$0.15 \leq \frac{1}{2(\overline{bc})^2} \left| (f^b)_x ((f^c)_y - (f^a)_y) + (f^c)_x ((f^a)_y - (f^b)_y) + (f^a)_x ((f^b)_y - (f^c)_y) \right| \quad (1)$$

Der Algorithmus iteriert somit durch alle Sterne s^i , und selektiert jeweils die nächsten 2 Sterne, deren Magnitude sich nicht mehr als Δm von $(s^i)_{\text{mag}}$ unterscheidet und noch nicht versucht wurde, mit diesen beiden Sternen ein Feature zu erzeugen. Dann wird aus s^i und den beiden gefundenen Sternen ein Feature erzeugt und mittels Gleichung 1 überprüft, ob es sich um ein geeignetes Feature handelt.

Dies wird solange fortgesetzt, bis entweder keine passenden Sterne mehr gefunden werden, um damit ein Feature zu erzeugen, oder bis eine gewisse Zahl Features erzeugt wurden.

Dieser Parameter ist wie einige andere anpassbar. Je genauer die aus der Astroatnahme extrahierten Daten und die im Katalog hinterlegten Sterne, desto weniger Features müssen für eine erfolgreiche Durchführung extrahiert werden. Für die Simulation wurden jeweils 3 Features pro Stern erzeugt.

Die Feature-Extraktion für den Katalog kann und sollte vorgängig durchgeführt werden, sodass beim eigentlichen Matching die Features nur noch abgefragt und nicht mehr extrahiert werden müssen. Die Extraktion der zu vergleichenden Bildfeatures erfolgt auf die exakt selbe Weise, sodass auch sichergestellt ist, dass die Features zur Deckung gebracht werden können.

2.3 Matching

Wir haben nun die benötigte Strukturinformation im Katalog gefunden, sodass wir nun ein richtiges Matching zwischen den extrahierten Bild- und Katalogfeatures durchführen können.

Die Grundidee ist, dass wir für ähnliche Features in Bild und Katalog jeweils die Übereinstimmung der relativ nahen Sterne überprüfen. So ist anzunehmen, dass in der Nähe der meisten Features weitere Sterne zu finden sind. Ist ein solcher auf dem Bild erkennbar, können wir im Katalog überprüfen, ob sich an der entsprechenden Position im Katalog (oder zumindest in akzeptabler Nähe) ebenfalls ein Stern befindet, und ob dieser zusätzlich ungefähr die richtige Magnitude hat.

Führen wir das für alle Sterne in relativer Nähe zum Bildfeature durch, können wir für jedes Feature-Tupel (f_B^i, f_K^j) einen Wert, den FeatureScore berechnen. Dieser zeigt auf, wie gut die zwei Features tatsächlich zueinander passen.

Aus diesen Erkenntnissen ergeben sich zwei Fragen:

1. Was heisst relative Nähe?
2. Wie wird FeatureScore berechnet?

2.3.1 Relative Nähe

Im Kapitel Abschnitt 1.1 haben wir herausgefunden, dass es aufgrund des Aufnahmeprozesses zu diversen Verzerrungen auf den Fotografien kommen kann. Diese Verzerrungen sind für grosse Bildausschnitte oder Ausschnitte am Rand des Bildes problematisch, da die Verzerrungen in diesen Bereichen vergleichsweise gross sind. In einem kleineren Bildausschnitt in der Mitte des Bildes sind diese Verzerrungen hingegen vernachlässigbar.

Wir wollen deshalb nur Sterne zwischen Bild und Katalog vergleichen, die relativ Nahe beim Feature liegen. Für die Definition von Nahe eignen sich baryzentrischen Koordinaten hervorragend. Eine kurze Einführung dazu findet sich in Appendix A. Es handelt sich bei diesen um ein normiertes, homogenes Koordinatensystem. Die Koordinaten eines Punktes werden anhand der normierten Abstände zu den Eckpunkten des Dreiecks definiert, das heisst, je weiter vom Dreieck ein Punkt entfernt ist, desto grösser, bzw. kleiner werden die einzelnen Koordinatenkomponenten.

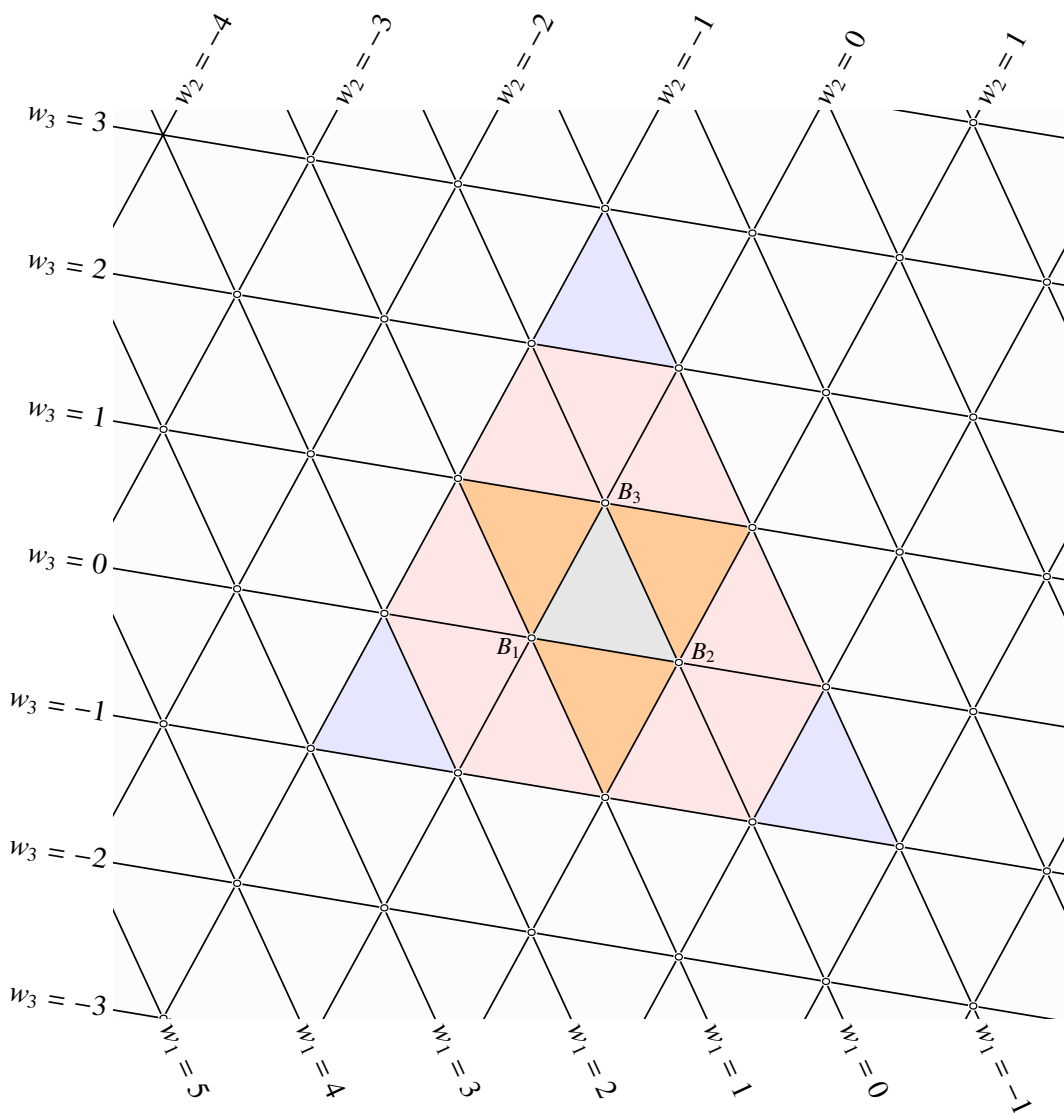


Abbildung 5: Definon der relativen Nähe mittels baryzentrischen Koordinaten[3]

Dies können wir uns zunutze machen, indem wir eine obere und untere Schranke für die Koordinatenelemente definieren. Diese Schranke bestimmt, welche Sterne nahe genug beim Feature liegen. In Abbildung 5 hat das Ausgangsdreieck in der Mitte einen grauen Farbton. Mit verschiedenen Farben schraffiert sind unterschiedliche untere und obere Schranken für die einzelnen Koordinaten. Die untere Koodinate ist jeweils -1 . Mit Orange schraffiert ist die obere Schranke 1, mit Rosa die obere Schranke 2 und mit Violett 3.

Je nach Definition der unteren und oberen Schranke für die einzelnen Koordinaten gelten somit mehr oder weniger Sterne als nah. Dieser Parameter ist grundsätzlich anpassbar. In Gebieten mit eher wenigen Sternen könnte es Sinn machen, einen grösseren Ausschnitt zu analysieren. Im Zuge dieser Arbeit wurde als untere Grenze -1 und als obere 2 gewählt, sodass der maximale Abstand zum Feature in alle Richtungen ungefähr gleich ist.

Solche nahen Sterne werden mit der Notation $\text{Close}_B(s_B^i)$ im Bild, bzw. $\text{Close}_K(s_K^i)$ im Katalog bezeichnet.

2.3.2 Feature-Score

Auf diese Art und Weise wird zu jedem Bildfeature $f_B^i \in F_B$ alle nahen Sterne $\text{Close}_B(f_B^i) \in S_B$ gefunden. Analog werden auch alle $\text{Close}_K(f_K^j) \in S_K$ für alle Katalogfeatures $f_K^j \in F_K$ gesucht.

Mit diesen Information kann überprüft werden, wie gut zwei Feature-Tupel (f_B^i, f_K^j) übereinstimmen. Dazu wird für alle $\text{Close}_B(f_B^i)^p$ überprüft, ob sich dazu in $\text{Close}_K(f_K^j)^q$ ein Stern befindet, der sich relativ zu den Features gesehen, ungefähr an derselben Position befindet. Dies wird überprüft, in dem die Koordinaten von $\text{Close}_B(f_B^i)^p$ über die baryzentrischen Matrizen mittels der Transformation $L(f_K^j)^{-1} L(f_B^i) \text{Close}_B(f_B^i)^p$ in Katalogkoordinaten umgewandelt werden, wie genauer in Appendix A beschrieben. Auch die Magnitude, welche der Stern im Katalog hätte, wird berechnet. Das Verhältnis zwischen der Magnitude des hellsten Sternes im Bildfeature $(f_B^i)_{\text{bsm}}$ und $(\text{Close}_B(f_B^i)^p)_{\text{mag}}$ soll ähnlich sein wie zwischen $(f_K^j)_{\text{bsm}}$ und $(\text{Close}_K(f_K^j)^q)_{\text{mag}}$. Daraus ergibt sich ein neuer Stern s'_K mit den Koordinaten und der Magnitude, welche der gesuchte Katalogstern idealerweise hätte.

Mithilfe dieses berechneten Sternes lässt sich jedem Stern $\text{Close}_K(f_K^j)^1$ nahe dem gerade verglichenen Feature f_K^j ein Wert zuweisen, der anzeigt wie nahe am gesuchten Stern s'_K er sich in Bezug auf Position und Magnitude befindet. Position und Magnitude werden dabei einzeln Berechnet. Ausserdem müssen die Werte für Position und Magnitude unterschiedlich gewichtet werden können, da sie nur indirekt verglichen werden können. Als allgemeine Formel für Formel für den Ähnlichkeits-Score zweier Sterne im Katalog aus $\text{Close}_K(f_K^j)^1$ und $\text{Close}_B(f_B^i)^p$, der Einfachheit halber hier einfach als s_B und s_K geschrieben, ist somit

$$\text{Score}_{\text{Star}}(s_B, s_K) = \alpha \text{Score}_{\text{Pos}}(s_B, s_K) + \beta \text{Score}_{\text{Mag}}(s_B, s_K).$$

Um $\text{Score}_{\text{Pos}}(s_B, s_K)$ zu berechnen wird die auf die quadrierte euklidische Distanz zurückgegriffen. Dabei sollen aber nicht direkt die absoluten Katalogkoordinaten von s_B und s_K verglichen werden, sondern deren baryzentrische Koordinaten $(s_B)_{\text{bar}}$ und $(s_K)_{\text{bar}}$. Dies hat den Vorteil, dass bei einem stark verzerrten Bild die Koordinaten besser vergleichbar sind. Ausserdem wird ein Umrechnungsschritt weniger benötigt, da die Umrechnung in Katalogkoordinaten weggelassen werden kann. Damit ist

$$\text{Score}_{\text{Pos}}(s_B, s_K) = \|(s_B)_{\text{bar}} - (s_K)_{\text{bar}}\|^2 \quad (2)$$

Die Berechnung von $\text{Score}_{\text{Mag}}(s_B, s_K)$ ist dank der logarithmischen Magnitude eine einfache quadrierte Differenz zwischen $(s_B)_{\text{mag}}$ und $(s_K)_{\text{mag}}$.

$$\text{Score}_{\text{Mag}}(s_B, s_K) = ((s_B)_{\text{mag}} - (s_K)_{\text{mag}})^2 \quad (3)$$

Damit wird $\text{Score}_{\text{Star}}$ zu

$$\text{Score}_{\text{Star}}(s_B, s_K) = \alpha \|(s_B)_{\text{bar}} - (s_K)_{\text{bar}}\|^2 + \beta ((s_B)_{\text{mag}} - (s_K)_{\text{mag}})^2. \quad (4)$$

Ist auf diese Art und Weise der $\text{Score}_{\text{Star}}$ zwischen den aktuellen Sternen in $\text{Close}_B(f_B^i)^p$ und allen $\text{Close}_K(f_K^j)^q$ berechnet worden, wird für diesen Bildstern nur der beste Katalogstern behalten, also derjenige mit den tiefsten Score. Für jeden nahen Stern eines Bildfeatures gibt es somit zu jedem Katalogfeature ein $\text{Score}_{\text{FeaturePartial}}$, einen Teilscore pro nahen Stern:

$$\text{Score}_{\text{FeaturePartial}}(s_B, f_K) = \text{Min}(\text{Score}_{\text{Star}}(s_B, \text{Close}_K(f_K)^q)) \quad (5)$$

Ist dies für alle $\text{Close}_B(f_B^i)^p$ durchgeführt worden, lässt sich der $\text{Score}_{\text{Feature}}$ für ein Feature-Tupel (f_B^i, f_K^j) ausrechnen. Es ist zu berücksichtigen, dass die Anzahl naher Sterne pro Bildfeature stark variieren kann. Würde man den Gesamtscore einfach als Summe aller Scores berechnen, könnte ein Feature mit wenigen nahen Sternen einen besseren Score haben, als ein Feature mit vielen nahen Sternen, obwohl die einzelnen Sterne des Features viel schlechter abschneiden.

Für den Gesamtscore des Features muss deshalb der Durchschnitt der einzelnen Scores verwendet werden. Somit ist

$$\text{Score}_{\text{Feature}}(f_B, f_K) = \frac{1}{n} \sum_{j=1}^p \text{Score}_{\text{FeaturePartial}}(\text{Close}_B(f_B, f_K)^p) \quad (6)$$

wobei n der Anzahl nahen Sterne des Bildfeatures entspricht.

Es kann vorkommen, dass nahe Sterne in der Aufnahme im Katalog gar nicht gefunden werden – sei dies, weil wir zwei verschiedene Features miteinander vergleichen, weil es sich um ein Artefakt im Bild handelt oder gar um einen nicht im Katalog hinterlegten Stern.

Was würde dies bedeuten? Es wurde festgelegt, dass jeweils der Katalogstern mit dem besten Score für einen Bildstern behalten wird. Haben wir auf der Astrofotografie ein Artefakt, welches wir als Stern interpretieren, erhalten wir für diesen Stern einen äusserst hohen (also schlechten) Score. Im Falle eines Artefaktes möchten wir diesen Ausreisser am liebsten ignorieren, oder auf jeden Fall nicht gleicht Stark wie die anderen Werte.

Vergleichen wir hingegen zwei nicht passende Features, so haben gibt es für dieses Feature diverse hohe Scores. In diesem Fall möchten wir die hohen Werte nicht ignorieren.

Das Problem am Durchschnitt in Gleichung 6 ist, dass Ausreisser zu viel Gewicht zugewiesen wird. Ein besserer Wert wäre deshalb der Median. Sind alle Scores hoch, oder sehr weit verteilt, erhalten wir mit dem Median einen höheren Score, als mit dem Durchschnitt. Gleichzeitig haben auch mehrere Ausreisser nur einen geringen Einfluss auf den Score.

Es hat sich aber herausgestellt, dass die Medianberechnung in vielen Fällen zu komplex ist. In dieser Arbeit wurde schlussendlich auf den Durchschnitt gesetzt. Mit dem Median wären bessere Resultate möglich. Wird der hier entwickelte Algorithmus umgesetzt, sollte dies berücksichtigt werden und wieder der Median verwendet werden.

2.3.3 Matching

Nach diesem Schritt haben wir also für jedes Feature-Tupel zwischen Bild und Katalog einen Score, wobei die tiefsten Werte am besten passen.

Bei so vielen Sternen kann es unter Umständen vorkommen, dass ein Ausschnitt fotografiert wurde, für welchen es mehrere Bereiche im Katalog gibt, die dem fotografierten Bereich sehr ähnlich sind. Es wird also für beide ein guter Score berechnet. Wie findet man die verschiedenen Bereiche, welche passen?

Ein Ansatz ist, zu überprüfen, wohin denn die verschiedenen Features zeigen. Der Score wurde ja jeweils für ein Feature-Tupel berechnet. Wir können also mithilfe der baryzentrischen Koordinaten der beiden Features Punkt vom Bild auf den Katalog mappen. Wir können nun

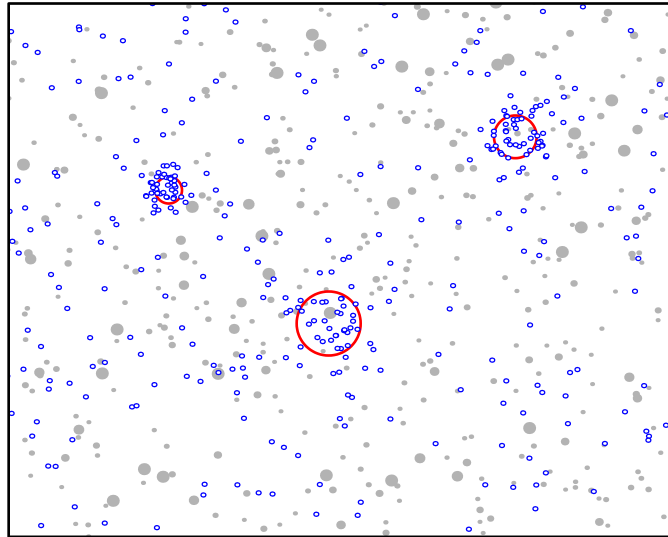


Abbildung 6: *Projektionsrauschen*: Darstellung der Bildmittelpunkte, die über die verschiedenen Features auf den Katalog gemappt wurden. [4]

jeweils den selben Punkt vom Bild auf den Katalog mappen. Tun wir das für den Bildmittelpunkt, können wir bestimmen, wo sich der Bildmittelpunkt im Katalog befinden würde, sofern es sich beim Bild- und Katalogfeature um zwei Features handelt, die aus denselben Sternen in Bild und Katalog extrahiert wurden.

Und genau das tun wir. Was wir aus dem Verfahren erhalten, sind eine Menge von Koordinaten im Katalog. Abbildung 6 zeigt ein Beispiel, wie die Bildmittelpunkte auf den Katalog gemappt worden sein könnten. Die gemappten Punkte werden als kleine, blaue Kreise dargestellt. Wenn wir die Verteilung dieser Punkte betrachten, stellen wir fest, dass die meisten gemappten Punkte sich wie Rauschen relativ gleichmässig über den Katalog verteilen. Wir nennen dies *Projektionsrauschen*. Es gibt allerdings drei dichtere Ansammlungen von Punkten, die rot umkreist sind. Diese können unterschiedlich dicht gepackt sein, was durch die Grösse des roten Kreises dargestellt wird. Diese dichtgepackten Cluster sind für uns besonders interessant, denn was diese bedeuten, ist nichts anderes, als dass Feature-Tupel, die den Bildmittelpunkt in diesen Cluster projizieren, sehr wahrscheinlich die richtigen sind. Jedem Punkt ist ausserdem ein Wert zugewiesen (In Abbildung 6 nicht ersichtlich), der $\text{Score}_{\text{Feature}}$. Zusammen lässt sich aus diesen Informationen herausfiltern, bei welchen Mappings es sich um die richtigen handeln könnte.

Dafür muss das *Projektionsrauschen* analysiert werden. Wir wollen die Rot umkreisten Bereiche im Katalog finden. Gibt es wie in Abbildung 6 mehrere Bereiche, welche dichter gepackt sind als der Rest, müssen wir den korrekten Mittelpunkt anhand dem der Feature-Tupel zugewiesenen $\text{Score}_{\text{Feature}}$ von den falschen trennen.

Es gibt im Bereich der Statistical Analysis diverse Cluster-Analyse und -Detection Verfahren. Da man über ein solches Thema aber eine eigene Semesterarbeit schreiben könnte, und solche Algorithmen eher teuer sind, wenn Sie genau arbeiten sollen, wurde entschieden, einen eigenen Ansatz zu finden.

Wenn der Katalog mit den gemappten Mittelpunkten in verschiedene, gleich grosse Sektoren aufgeteilt wird, kann jedem Sektor wiederum ein $\text{Score}_{\text{Sector}}$ zugewiesen werden, welcher dem Durchschnitt des summierten $\text{Score}_{\text{Star}}((f_B, f_K)^i)$ aller Feature-Tupel entspricht, die den Bildmit-

telpunkt in diesen Sektor projizieren, also

$$\text{Score}_{\text{Sector}}((f_B, f_K)^n) = \frac{1}{n} \sum_{i=1}^n \text{Score}_{\text{Feature}}((f_B, f_K)^i). \quad (7)$$

Einer der Sektoren, welche dabei tiefe $\text{Score}_{\text{Sector}}$ zugewiesen erhalten, ist mit hoher Wahrscheinlichkeit der richtige.

Bei der Berechnung der analysierten Sektoren ist zu beachten, dass sich diese überlappen sollten. Sonst bestünde die Möglichkeit, dass die richtig projizierten Bildmittelpunkte sich in mehrere Sektoren verteilen und somit alle betroffenen Sektoren einen zu hohen Score zugewiesen erhalten.

Die Grösse der Sektoren selbst ist wiederum ein anpassbarer Parameter. Die analysierten Sektoren können kleiner werden, sofern die vorhandenen Daten genauer sind. Während der Entwicklung wurde mit unterschiedlichen Werten experimentiert, wobei sich mit den Simulierten Daten einige Gradsekunden als optimal herausgestellt haben.

Aus diesen Sektoren werden dann alle Feature-Tupel, deren $\text{Score}_{\text{Feature}}$ den $\text{Score}_{\text{Sector}}$ nicht überschreiten, dem nächsten Schritt zur Berechnung des Transformationsvektors übergeben. Ausserdem erfahren wir daraus, wo im Katalog sich der Mittelpunkt des Bildes befindet, nämlich ungefähr im Schwerpunkt der projizierten Mittelpunkte.

Es ist zu beachten, dass in diesen Tupeln immer noch einige falsche Mappings vorhanden sein können. Sofern aber der grösste Teil der Mappings passt, wird das im nächsten Schritt verwendete Least-Square-Verfahren davon nur geringfügig beeinflusst.

Bei diesem Schritt handelt es sich aktuell noch um den Schwachpunkt des Algorithmus. Es sollte ein besseres Cluster-Detection Verfahren eingesetzt werden, um die wichtigen Cluster von projizierten Bildmittelpunkten zu identifizieren.

2.3.4 Transformationsvektor berechnen

Nach dem zuvor beschriebenen Verfahren kennen wir mehrere Bereiche, wo sich der Bildmittelpunkt auf dem Katalog befinden könnte. Diese Information ist zwar auch alleine schon hilfreich – Der Astrofotograph könnte damit im Sternenkatalog nachschauen, welche Sterne sich dort befinden und selbst versuchen, herauszufinden, welche Sterne sich nun genau auf der Aufnahme befinden.

Nützlicher wäre ihm wohl, wenn wir ihm ein Mittel zur Verfügung stellen, die Koordinaten der einzelnen Bildsterne in Katalogkoordinaten umzurechnen. Wir möchten also die Translation, Rotation und Streckung berechnen, um $(s_B^i)_x$ und $(s_B^i)_y$ in $(s_K^i)_x$ und $(s_K^i)_y$ umzurechnen. In Vektorform geschrieben möchten wir also

$$\begin{pmatrix} (s_B^i)_x \\ (s_B^i)_y \end{pmatrix} \rightarrow \begin{pmatrix} (s_K^i)_x \\ (s_K^i)_y \end{pmatrix}$$

Beginnen wir mit der Translation. Diese kann ebenfalls als Vektor

$$\vec{t} = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}$$

dargestellt werden. Da die Translation rein linear ist, ergibt sich als Gleichungssystem

$$\begin{pmatrix} (s_B^i)_x \\ (s_B^i)_y \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} (s_K^i)_x \\ (s_K^i)_y \end{pmatrix}.$$

Nachdem die Koordinate transformiert wurde, muss auch die Rotation um den Winkel α berücksichtigt werden. Die Rotation kann mit der Matrix

$$\begin{pmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{pmatrix}$$

beschrieben werden, woraus das Gleichungssystem zu

$$\begin{pmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} (s_B^i)_x \\ (s_B^i)_y \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} (s_K^i)_x \\ (s_K^i)_y \end{pmatrix}$$

wird. Damit fehlt nur noch die Streckung der Koordinaten um λ . Daraus ergibt sich das Gleichungssystem

$$\lambda \begin{pmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} (s_B^i)_x \\ (s_B^i)_y \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} (s_K^i)_x \\ (s_K^i)_y \end{pmatrix}.$$

Da λ Konstant ist, kann es auch in die Rotationsmatrix verschoben werden, womit folgt, dass

$$\begin{pmatrix} \lambda \cos \alpha & \lambda \sin \alpha \\ -\lambda \sin \alpha & \lambda \cos \alpha \end{pmatrix} \begin{pmatrix} (s_B^i)_x \\ (s_B^i)_y \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} (s_K^i)_x \\ (s_K^i)_y \end{pmatrix}.$$

Da sowohl λ wie auch α Konstant sind, können die Terme $\lambda \cos \alpha$ und $\lambda \sin \alpha$ in der Rotations- und Streckungsmatrix durch c und s substituiert werden. Das finale Gleichungssystem, welches für alle Sterne s_B^i und s_K^i gelöst werden muss, sieht somit wie folgt aus:

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} (s_B^i)_x \\ (s_B^i)_y \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} (s_K^i)_x \\ (s_K^i)_y \end{pmatrix} \quad (8)$$

Dieses Gleichungssystem können wir auch als normales Gleichungssystem zur Lösung der Unbekannten c , s , t_1 und t_2 schreiben:

$$\begin{aligned} c(s_B^i)_x + s(s_B^i)_y + 1t_1 + 0t_2 &= (s_K^i)_x \\ -s(s_B^i)_x + c(s_B^i)_y + 0t_1 + 1t_2 &= (s_K^i)_y \end{aligned}$$

Wir haben also einen Weg gefunden, aus zwei zwischen Bild und Katalog gemappten Sternen zwei Gleichungen zu erzeugen. Wir suchen aber die 4 Unbekannten s , c , t_1 und t_2 . Da wir aber Features und nicht Sterne mappen, können wir für jede 2 gemappten Features insgesamt 6 Gleichungen erzeugen. Dadurch erhalten wir folgendes Gleichungssystem:

$$\begin{aligned} c(s_B^1)_x + s(s_B^1)_y + 1t_1 + 0t_2 &= (s_K^1)_x \\ -s(s_B^1)_x + c(s_B^1)_y + 0t_1 + 1t_2 &= (s_K^1)_y \\ c(s_B^2)_x + s(s_B^2)_y + 1t_1 + 0t_2 &= (s_K^2)_x \\ -s(s_B^2)_x + c(s_B^2)_y + 0t_1 + 1t_2 &= (s_K^2)_y \\ c(s_B^3)_x + s(s_B^3)_y + 1t_1 + 0t_2 &= (s_K^3)_x \\ -s(s_B^3)_x + c(s_B^3)_y + 0t_1 + 1t_2 &= (s_K^3)_y. \end{aligned} \quad (9)$$

Somit haben wir nun ein Gleichungssystem mit 6 Gleichungen bei weiterhin 4 Unbekannten. Und wir haben diverse gemappte Features, die zusammengehören und alle nochmals 6 Gleichungen erzeugen. Wir können das Gleichungssystem aufgrund den zu erwartenden Ungenauigkeiten und zu wenigen Unbekannten nicht exakt lösen. Wir verwenden deshalb das Least-Square-Verfahren zur Lösungsfindung.

Das obige Gleichungssystem wird deshalb in die Least-Square Form $A\vec{T} = \vec{r}$ überführt. \vec{T} ist der gesuchte Transformationsvektor

$$\begin{pmatrix} c \\ s \\ t_1 \\ t_2 \end{pmatrix},$$

die Matrix A besteht aus den bekannten Koordinaten aus Gleichung 9, während \vec{r} der rechten Seite dieser Gleichung entspricht.

Das mit dem Least-Square Verfahren zu lösende System sieht somit wie folgt aus:

$$\begin{pmatrix} (s_B^1)_x & (s_B^1)_y & 1 & 0 \\ (s_B^1)_y & (s_B^1)_x & 0 & 1 \\ (s_B^2)_x & (s_B^2)_y & 1 & 0 \\ (s_B^2)_y & (s_B^2)_x & 0 & 1 \\ (s_B^3)_x & (s_B^3)_y & 1 & 0 \\ (s_B^3)_y & (s_B^3)_x & 0 & 1 \end{pmatrix} \begin{pmatrix} c \\ s \\ t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} (s_K^1)_x \\ (s_K^1)_y \\ (s_K^2)_x \\ (s_K^2)_y \\ (s_K^3)_x \\ (s_K^3)_y \end{pmatrix}. \quad (10)$$

Wird dieses System mit dem Least-Square Verfahren gelöst, erhalten wir als Ergebnis den gesuchten Transformationsvektor. Damit kann der Astrofotograph nun die Koordinaten aller Bildsterne in Katalogkoordinaten unwandeln oder den genauen Ausschnitt berechnen, der Dargestellt wird. Wie und wofür er diese verwendet ist für uns nicht mehr von Belang und unser Teil der Arbeit somit erledigt.

2.4 Zwischenanalyse Performance

Somit haben wir nun ein Verfahren entwickelt, wie wir für eine Astroatnahme herausfinden können, welchen Ausschnitt des Sternenhimmels darin ersichtlich ist und wie die Koordinaten entsprechend transformiert werden.

Aufgrund der enormen Datenmenge stellt sich aber schnell die Überlegung, ob das Problem überhaupt in einer sinnvollen Zeit lösbar ist. Der UCAC4 Katalog beinhaltet eine Übersicht über $n = 1.3 \cdot 10^8$ Sterne, wobei pro Stern bis zu vier Features extrahiert werden. Es gibt also maximal $m = 5.2 \cdot 10^8$ Katalog-Features. In einem Bild gibt es zwar vergleichsweise wenige Sterne j aus denen Features k berechnet werden müssen, aber auch hier muss mit einigen Tausend gerechnet werden.

Im folgenden soll die Laufzeit der verschiedenen Teilschritte des Matching-Algorithmus analysiert werden:

1. Finde der nahen Sterne eines Features, sowohl im Bild wie im Katalog.
2. Das Berechnen des Feature-Scores, basierend auf diesen nahen Sternen.

3. Das durchführen des eigentlichen Mappings, indem die verschiedenen, zum selben Cluster gehörenden Features analysiert werden.
4. Transformationsvektor berechnen

2.4.1 Laufzeit Teilschritt 1

Für die Laufzeitanalyse können wir die Bildanalyse ignorieren, da dort vergleichsweise wenige Sterne zu finden sind. Wir müssen für jedes Katalogfeature alle nahen Sterne finden. Dafür müssen für jedes Feature in Bild (k) und Katalog (n) alle Sterne (j und n) analysiert werden, also $O(jk + mn)$. Für die so gefundenen Sterne werden noch die baryzentrischen Koordinaten berechnet, was einer Multiplikation einer Matrix mit einem Vektor entspricht, was in linearer Zeit durchgeführt werden kann. Dies kann also ignoriert werden.

Somit ist die Laufzeit für Teilschritt 1: $O(nm)$. Sowohl n wie auch m sind grösser als 10^8 , es sind somit über 10^{16} Operationen durchzuführen. Das Problem ist damit zwar nicht quadratisch, aber vergleichsweise aufwändig zu berechnen.

2.4.2 Laufzeit Teilschritt 2

Nun müssen wir für alle km Feature-Tupel für alle o nahen Sterne des Bildes alle p nahen Sterne im Katalog zuerst den Einzel-Score der Sterne berechnen und daraus den Feature-Score herleiten. Dafür muss für jedes Feature-Tupel und jeden nahen Stern zuerst der Stern-Score berechnet werden, und dann das kleinste Element gefunden werden. Danach wird der Median über alle Einzel-Score gebildet und danach der Median

Die Berechnung für alle Stern Kombinationen zwischen Bild und Stern hat eine Laufzeit von $O(op)$. Das Resultat muss für alle Bildsterne noch in $O(p \log p)$ sortiert und das kleinste Element gefunden werden.

Die Berechnung des Durchschnitts benötigt nur $O(o)$. Somit ist die Gesamtlaufzeit für diesen Teilschritt $O(o^2 p^2 \log p)$

2.4.3 Laufzeit Teilschritte 3 und 4

Das Finden der Feature-Mappings, die zum selben Cluster gehören ist aufgrund der Implementieren relativ schnell. Am längsten geht das Sortieren nach der Menge der beinhalteten Feature-Mappings, was aber in $O(km \log km)$ durchgeführt werden kann.

Auch die Berechnung des Transformationsvektors kann mittels des Least-Square-Verfahrens sehr effizient implementiert werden.

2.4.4 Fazit

Es ist erkennbar, dass der Algorithmus mit der zu erwartenden hohen Datenmenge ziemlich langsam arbeiten wird, wenn er überhaupt zu unseren Lebzeiten fertig wird. Wir müssen also einen Weg finden, die durchzuführenden Vergleiche soweit zu reduzieren, dass die Berechnung in einer sinnvollen Zeit durchgeführt werden kann.

An den Teilschritten des Algorithmus lässt sich allerdings nicht viel optimieren. Einige Teilschritte können zwar sehr stark optimiert werden. Die enorme Zahl der Vergleiche bleibt allerdings bestehen.

Wie lässt sich die Laufzeit optimieren? Bisher vergleicht der Algorithmus einfach alle Bild- mit allen Katalogfeatures. Es ist erstrebenswert, im ersten Teilschritt des Algorithmus bereits eine vorselektierte Liste mitzugeben, welche Katalogfeatures allenfalls zu einem Bildfeature passen könnten. Da sowohl die Berechnung der relativen Nähe wie auch von $\text{Score}_{\text{Feature}}$ immer mit Feature-Tupeln arbeitet, werden diese Schritte durch eine Vorselektion nicht tangiert.

2.5 Vorselektion

Es werden zwei Ansätze verfolgt, um diese Vorselektion durchzuführen. Der erste Ansatz stützt sich auf die Form der Features. Dazu wird die *Ähnlichkeitsinvariante* verwendet, welche bereits zur Analyse von geeigneten Dreiecken verwendet wurde und in Appendix B genauer beschrieben ist.

Der zweite Ansatz beruht darauf, dass ein über die richtigen Features auf den Katalog projizierten Bildpunkt immer ungefähr an derselben Stelle liegt. Diese beiden Ansätze werden nun genauer analysiert.

2.5.1 Vorselektion mit *Ähnlichkeitsinvariante*

Ein erster Ansatz für diese Vorselektion ist die Ähnlichkeit der Features. Es macht keinen Sinn, zwei Features zu vergleichen, die eine völlig unterschiedliche Form haben. Gleichzeitig darf eine Streckung, Rotation oder Spiegelung der Features keinen Einfluss haben, einzig und allein die Form des Dreieckes soll ähnlich sein.

In Appendix B hatten wir ein ähnliches Problem während der Herleitung des Heron-Grenzwertes, welcher in Abschnitt 2.2.3 benötigt wurde, um die Qualität der Features zu überprüfen. Wir mussten die Form von zufällig erzeugten Features analysieren, um herauszufinden, welche Arten von Dreiecken häufiger vorkommen und welche weniger. Dazu haben wir die Features normalisiert und deren Seitenverhältnisse $b : c$ als *Ähnlichkeitsinvariante* definiert. Diese Darstellung erlaubt lässt uns auch etwas über die Ähnlichkeit zweier Features sagen. Ist die *Ähnlichkeitsinvariante* ähnlich, sind sich auch die Dreiecke ähnlich. Deshalb auch die *Ähnlichkeitsinvariante*.

Um nun für ein Bildfeature f_B ähnliche Katalogfeatures zu finden, muss einfach die *Ähnlichkeitsinvariante* verglichen werden. Wird die *Ähnlichkeitsinvariante* als Vektor dargestellt, kann mithilfe der Summennorm der Abstand die Ähnlichkeit zwischen einem Bild- einem Katalogfeature berechnet werden:

$$\text{Similarity}(f_B, f_K) = \left| \begin{pmatrix} (f_B)_{sb} \\ (f_B)_{sc} \end{pmatrix} - \begin{pmatrix} (f_K)_{sb} \\ (f_K)_{sc} \end{pmatrix} \right| \quad (11)$$

Mit dieser Formel werden jetzt für alle f_B^i diejenigen f_K^i gesucht, welche einen bestimmten Werte für Similarity nicht überschreiten. Auch bei diesem Grenzwert handelt es sich um einen

f_B^i	$(f_B^i)_{sb}$	$(f_B^i)_{sc}$
f_B^1	0.9	0.4
f_B^2	0.8	0.7
f_B^3	0.7	0.7
f_B^4	0.6	0.5

Tabelle 1: Beispiel für *Ähnlichkeitsinvariante* im Bild

f_K^i	$(f_K^i)_{sb}$	$(f_K^i)_{sc}$
f_K^1	0.9	0.7
f_K^2	0.9	0.6
f_K^3	0.9	0.4
f_K^4	0.8	0.6
f_K^5	0.8	0.6
f_K^6	0.8	0.5
f_K^7	0.7	0.7
f_K^8	0.7	0.6
f_K^9	0.6	0.6
f_K^{10}	0.6	0.5

Tabelle 2: Beispiel für *Ähnlichkeitsinvariante* im Katalog

konfigurierbaren Parameter. Wieder kann dieser verkleinert werden, je genauer die vorhandenen Daten sind. Wir haben hierfür den Wert 0.001 gewählt.

Tabelle 1 und Tabelle 2 zeigen einige Beispieldaten für die *Ähnlichkeitsinvariante* in Bild und Katalog. Mit einem Grenzwert von 0.1 würden unter Verwendung der Summennorm jene Katalogfeatures gefunden, welche in beiden Dimensionen des Vektors zusammen um nicht mehr als 0.1 vom Bildfeature abweichen. Somit entstehen folgende Featurezuweisungen:

$$\begin{aligned}
f_B^1 &\rightarrow (f_K^3) \\
f_B^2 &\rightarrow (f_K^1, f_K^4, f_K^5, f_K^7) \\
f_B^3 &\rightarrow (f_K^7, f_K^8) \\
f_B^4 &\rightarrow (f_K^9, f_K^{10})
\end{aligned}$$

Mit diesem Verfahren wird zwar noch die Magnitude vernachlässigt, dafür ist es aber enorm effizient. Es werden aber noch immer zu viele Features gefunden, die sich ähneln. Die Features müssen weiter eingeschränkt werden, denn nur weil sich zwei Features ähneln, passen sie noch lange nicht zusammen.

2.5.2 Vorselektion mit Bildmittelpunkt

Es gibt aber etwas, was die richtigen Feature-Tupel gemeinsam haben: Sie befinden sich alle ungefähr am selben Ort. Wir können deshalb denselben Weg gehen wie in Abschnitt 2.3.3. Wird derselbe Bildpunkt über die korrekten Feature-Tupel mithilfe der baryzentrischen Koordinaten auf den Katalog projiziert, befindet sich der projizierte Punkt immer ungefähr an derselben Stelle, natürlich immer mit einer gewissen Ungenauigkeit. Wir wollen dieses Konzept an dieser Stelle etwas genauer anschauen, da mithilfe dieser Vorselektion der grösste Teil der Arbeit erledigt wird.

In Abbildung 7 ist ein Ausschnitt aus einem Beispielkatalog zu sehen. Darin zu sehen sind einige schwarze Sterne, sowie daraus extrahierte Features in Orange.

In Abbildung 8 ist ein Bild zu sehen, welches in diesem Katalog wiedergefunden werden soll. Es sind wiederum einige schwarze Sterne sowie zwei grüne daraus extrahierte Features zu

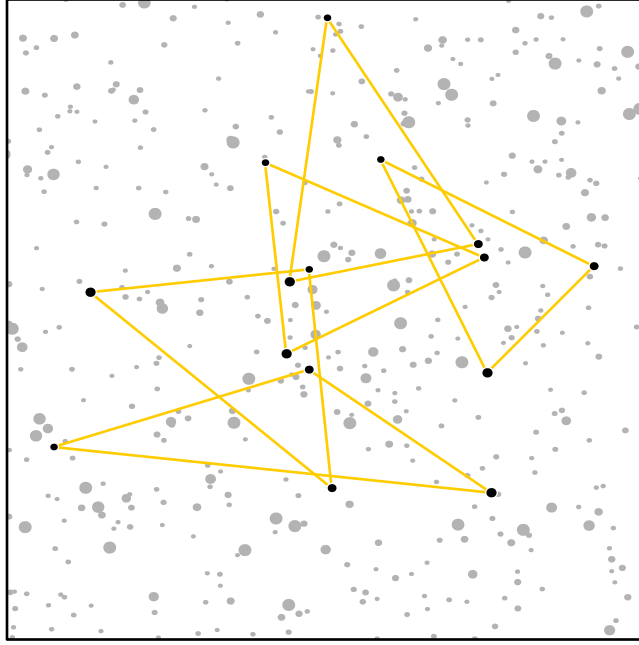


Abbildung 7: Beispielkatalog mit schwarzen Sternen und orangen Features[4]

sehen.

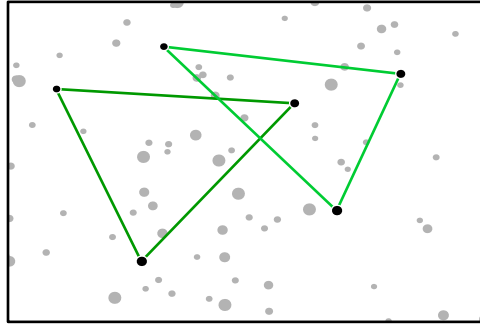


Abbildung 8: Beispielbild mit schwarzen Sternen und grünen Features[4]

Im nächsten Schritt wird für jedes Bildfeature mittels der baryzentrischen Matrix $L(f_B^i)$ die baryzentrischen Koordinaten \vec{b}^i der absoluten Koordinaten des Bildmittelpunktes \vec{m}_B berechnet, also $\vec{b}^i = L(f_B^i)\vec{m}_B$. In Abbildung 9 wird dies dargestellt, indem jeweils vom der längsten Seite gegenüberliegenden Stern ein grüner Vektor eingetragen wird, der zum Bildmittelpunkt zeigt (blaues Kreuz). Dies ist einfacher in der Veranschaulichung, während das eigentliche Prinzip dahinter, die relative Position des Bildmittelpunktes verglichen mit dem Feature, dasselbe bleibt.

Damit erhalten wir eine Sammlung von baryzentrischen Koordinaten, wo sich der Bildmittelpunkt relativ zu einem bestimmten Bildfeature befindet. Wird eine solche baryzentrische Koordinate \vec{b}^i für ein beliebiges Katalogfeature mit der inversen baryzentrischen Matrix $L(f_K^j)^{-1}$ wieder in eine absolute Katalogkoordinate \vec{m}_K^i umgewandelt, wissen wir, wo sich die Bildmittelpunkt im Katalog befinden würde, wenn die entsprechenden Bild- und Katalogfeatures in Bild und Katalog tatsächlich aus denselben drei Sternen gebildet worden wären.

Wird für jedes Katalogfeature f_K^j jeder berechnete \vec{b}^i mit $\vec{m}_K^i = L(f_K^j)^{-1}\vec{b}^i$ erhalten wir eine

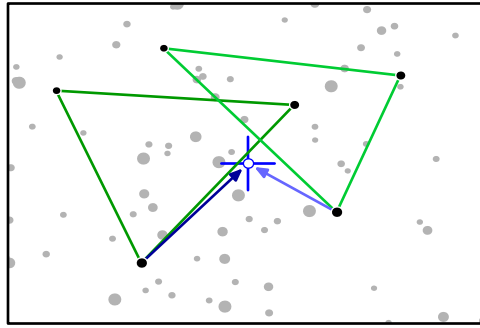


Abbildung 9: Gleiches Beispielbild. Die Position des Mittelpunktes relativ zu den Features ist mit einem blauen Kreuz bezeichnet[4]

Sammlung von Punkten im Katalog, welche die möglicherweise richtigen Bildmittelpunkte im Katalog angegeben. In Abbildung 10 sind dafür die in Abbildung 9 gezeigten Vektoren auf alle Katalogfeatures projiziert worden. Die Winkelverhältnisse des Vektors zum Feature wurden dabei beibehalten. Die beiden richtig projizierten Vektoren sind fetter gedruckt, als die anderen.

Bei vielen Sternen sind diese Katalogkoordinaten relativ gleichmässig über den Katalog verteilt. Es ist jedoch erkennbar, dass die Punkte an einer Stelle deutlich dichter sind als im Rest des Kataloges. Das ist derjenige Punkt, auf welchen die Mittelpunkte mittels dem richtigen Feature-Tupel projiziert haben.

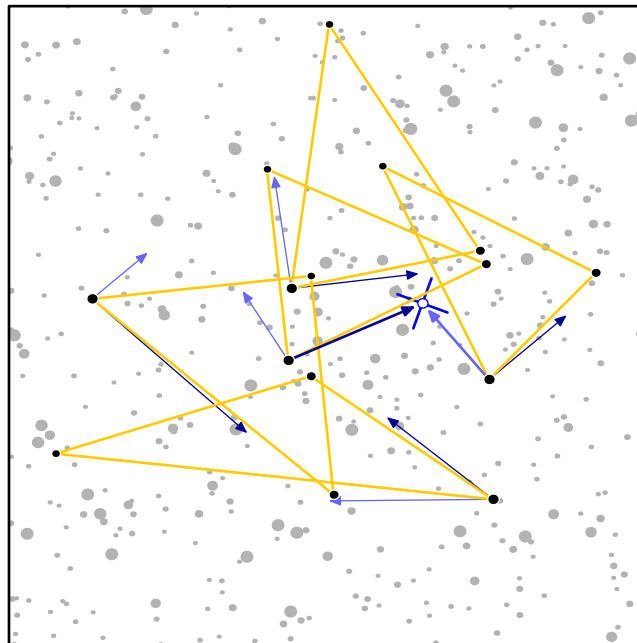


Abbildung 10: Für alle Bild- und Katalogfeatures wurde eingezeichnet, wie der Mittelpunkt mit diesen beiden Features gemappt wurde. Bei mehreren ähnlichen Mappings handelt es sich um eine gute Übereinstimmung[4]

Am Ende werden dem Algorithmus nur diejenigen Feature-Tupel mitgegeben, die den Bildmittelpunkt auf diese im Katalog identifizierten Mittelpunkt projizieren, oder zumindest in seiner Nähe liegen. Es kann sein, dass die projizierten Mittelpunkte mehrere sehr dichte Sektoren bilden. In diesem Fall müssen dem Algorithmus alle Feature-Tupel mitgegeben werden, die den

Mittelpunkt auf einen dieser Sektoren mappen.

Mit diesen beiden Schritten lässt sich die zu analysierende Datenmenge massiv reduzieren und es müssen nur noch wenige Features miteinander verglichen werden, welche einige wichtige Eigenschaften miteinander teilen.

3 Fazit

Es wurde ein Algorithmus entwickelt, welcher die Sterne einer Astroaufnahme mittels Feature-Extraktion und -Detektion in einem Sternenkatalog lokalisieren kann.

Der Algorithmus arbeitet zurzeit noch ziemlich langsam. Eine Beispielimplementation, die als Featuredatenbank eine PostgreSQL-Installation verwendete, benötigt noch rund 10 Minuten, um einen simulierten Ausschnitt aus dem Sternenkatalog wiederzufinden, wobei die Featuredatenbank nur einen Ausschnitt von $\frac{1}{96}$ in beide Richtungen beinhaltet. Um den Algorithmus tatsächlich einsetzen zu können, müssen vor allem die Datenzugriffszeiten massiv verbessert werden – deutlich am längsten dauert das Abfragen der ähnlichen Features, was eigentlich nichts weiterem entspricht, als einer Range-Abfrage auf zwei indexierten Spalten vom Typ double. Eine effizientere Datenbanklösung würde wohl Wunder bewirken.

Nichtsdestotrotz funktioniert der Algorithmus. Der simulierte Ausschnitt konnte im Katalog lokalisiert werden. Ein Beispiel dazu findet sich in Abschnitt 3.1.

In Abschnitt 3.2 wird noch besprochen, welche Arbeiten noch durchzuführen sind, bevor der Algorithmus auch mit sphärischen Koordinaten funktioniert, während in Abschnitt 3.3 mögliches Optimierungspotential besprochen wird. In Abschnitt 3.4 werden einnige Überlegungen getätigt, wie der Algorithmus allenfalls zur Lösung anderer Probleme eingesetzt werden könnte.

3.1 Simulation

Dieser Abschnitt soll aufzeigen, wie die Erarbeitung des Algorithmus von verschiedenen Programmierarbeiten begleitet wurde. So fand die gesamte Entwicklung des Algorithmus in einer simulierten Umgebung statt. Als Entwicklungsumgebung wurde .NET 4.6 mit C# 6 verwendet. Dies hauptsächlich aus zwei Gründen. Erstens kannte sich der Entwickler mit dieser Technologie besonders gut aus, zweitens erlaubt es einem C#, sehr funktionalen Code zu schreiben, was die Arbeit mit vielen Daten doch einiges erleichtert.

In einem ersten Schritt ging es vor allem um die grundsätzliche Analyse einiger Probleme. Schwerpunkt dabei war das Erarbeiten eines Feature-Extraktors, was Analysen über gute Dreiecke und verschiedenen Möglichkeiten zur Feature-Extraktion beinhaltete. Die Ergebnisse aus dem Schritt sind vor allem in Appendix B und den finalen Feature-Extraktor in Abschnitt 2.2.2 übergegangen. Der grösste Teil dieses Codes wurde wieder verworfen, da er nur für einige vorgängige Analyse Zwecken benötigt wurde.

Aus diesen Analysen ging eine Simulationsumgebung hervor, welche einen zufällig generierten Sternenkatalog erzeugen kann. Basierend darauf wurde einerseits ein Bildgenerator aufgebaut, welcher aus diesem Katalog Bilder erzeugt. Den sich darin befindlichen Sternen wurde ein normalverteilter Fehler sowohl auf die Position wie auch auf die Magnitude addiert.

Mit diesen Daten konnten dann ein Feature-Extraktor sowie ein Matcher geschrieben werden, mit welchen die verschiedenen im Algorithmus verwendeten Techniken überprüft werden konnten. So konnten schwierige Stellen in den Berechnungen gefunden werden, die frühzeitig optimiert werden mussten. Zudem konnten einfach verschiedene Methoden verglichen werden,

wie die einzelnen Probleme zu lösen sind. Zudem kristallisierte sich in diesen Arbeitsschritten heraus, welche Daten exakt benötigt werden, sodass die verschiedenen Datenstrukturen optimiert werden können.

Dies half, den Simulator im letzten Schritt so zu erweitern, dass er mit realen Sterndaten umgehen konnte. Die eigentlichen Anpassungen waren hier vergleichsweise gering. Statt mit x - und Y -Koordinaten wurde einfach die Rektaszension und Deklination verwendet, die Helligkeit wurde zur Magnitude. Da der UCAC4-Katalog ausserdem mit Ganzzahl- statt Fließkommadata-typen arbeitet, mussten einige Datentypen angepasst werden.

Ausserdem musste in diesem Schritt eine Featuredatenbank aufgebaut werden, wofür PostgreSQL 9.4 verwendet wurde. Diese Entscheidung stellte sich nachträglich als nicht ganz optimal heraus, da besonders das Selektieren der ähnlichen Features aus der Featuredatenbank extrem lange dauert.

Die Featuredatenbank wurde mit einem Ausschnitt des UCAC4-Kataloges befüllt. Dieser Ausschnitt ging in der Rektaszension von 19^h bis 22^h und in der Deklination von 0° bis 25° . In diesem Bereich befinden sich total $12'642'944$ Sterne. Mit diesen Sternen sollten nun der Feature-Extraktor sowie der Matcher überprüft werden. Es stellte sich heraus, dass besonders die Feature-Extraktion in der aktuellen Form noch sehr langsam ist. Am Ende konnte die Features aufgrund Zeitdruck nur für $\frac{3}{13}$ dieses Bereiches berechnet werden. Nach rund drei Tagen Berechnungen wurden die bis zu diesem Zeitpunkt erzeugten Features für die weiteren Tests verwendet. Es wurden insgesamt $6'551'978$ Features aus $3'328'509$ Sternen in diesem Bereich extrahiert.

Interessant ist in diesem Zusammenhang möglicherweise auch die Verteilung der Ähnlichkeitsinvariante der aus realen Daten erzeugten Katalogfeatures, wie in Abbildung 11 ersichtlich.

Der Algorithmus konnte auch in der Umgebung mit echten Daten grundsätzlich überprüft werden. Die am Ende vorhandene, sehr schwache Implementation benötigte dafür rund 15 Minuten, wobei ein guter Teil währenden mit Datenbankabfragen zugebracht wurde. In einem vollständigen Katalog würde diese Zeit aber stark ansteigen.

Der mitgegebene Code ist zwar soweit vollständig und auch kommentiert, wurde aber explizit nicht mit der Idee einer Beispielimplementation entwickelt. Der Code war ein Hilfsmittel zur Erarbeitung des Algorithmus, nicht umgekehrt, und das sieht man ihm auch an. Ergebnis dieser Arbeit ist ausdrücklich der beschriebene Algorithmus, nicht dessen umgesetzte Form.

3.2 Finalisierungsarbeiten

Aktuell rechnet der gesamte Algorithmus noch ausschliesslich mit Koordinaten in der Ebene. Am Himmel muss man aber mit sphärischen Koordinaten arbeiten. Die einzelnen Teilschritte müssen deshalb modifiziert werden, um auch im realen Umfeld zu funktionieren.

Die meisten Schritte bedürfen nur minimaler Anpassungen. In Abschnitt 2.2.2 muss dabei berücksichtigt werden, dass die Ecken eines Dreiecks in einem sphärischen Koordinatensystem mehr als 180° Winkel umschliessen. Die Definition für gute Features muss deshalb angepasst werden. Aber solange die längste Seite als Seite a und die Kürzeste als Seite c definiert sind, kann die *Ähnlichkeitsinvariante* genau gleich verwendet werden.

Die für verschiedene ausrichtungsunabhängigen Vergleiche und Projektion von Bild- auf Katalogkoordinaten verwendeten baryzentrischen Koordinaten werden von den sphärischen Koordinaten hingegen nicht beeinflusst, die Matrizen und deren Erzeugung funktioniert analog wie in der Ebene. Bei diesen Umrechnungen müssen allerdings Polregionen und die Ränder des Sternenkatalogs berücksichtigt werden, da hier einige Koordinatentransformationen nötig sein werden.

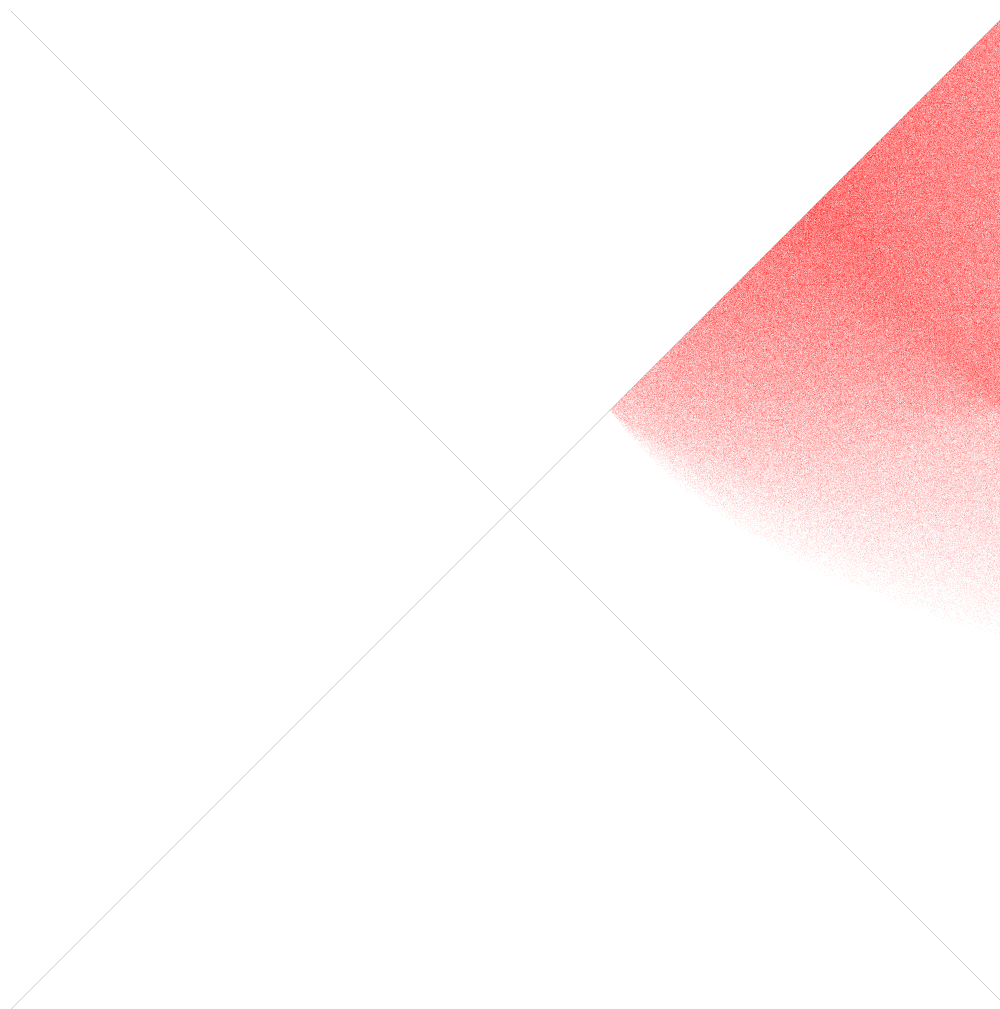


Abbildung 11: *Ähnlichkeitsinvariante* für Katalogfeatures aus echten Daten

Auch der in Abschnitt 2.3.2 beschriebene Weg mittels der euklidischen Distanz, um einem Mapping einen Score zuzuweisen, kann wiederverwendet werden. Auch wenn in dieser Situation bei grossen Ausschnitten gewisse Verzerrungen auftreten können ist das nicht weiter relevant, da bei solch grossen Distanzen der Score so oder so viel zu gross wird.

Der einzige Schritt, der noch diverser Anpassungen benötigen wird, ist die Berechnung des Transformationsvektors in Abschnitt 2.3.4. Die Transformation von Ebenen- auf andere Ebenenkoordinaten ist relativ einfach. Müssen die Koordinaten aber von einem Ebenen- auf ein Kugelkoordinatensystem umgerechnet werden, muss dies beim Berechnen des Transformationsvektors berücksichtigt werden. Dies wird im Rahmen dieser Arbeit aber nicht mehr angeschaut.

3.3 Optimierungspotential

Weiteres Optimierungspotential findet sich bei der Laufzeit des Algorithmus. In Abschnitt 3.1 war ersichtlich, dass dieser aktuell sehr langsam ist und die Laufzeit sicher noch verbessert werden könnte.

Zum einen können die meisten Schritte hochgradig parallelisiert durchgeführt werden. Z.B. könnte die Feature-Extraktion in Abschnitt 2.2.2 Sektorweise durchgeführt werden, sodass nicht alle Sterne aufs Mal geladen und verglichen werden müssen, sondern immer nur für einen Aus-

schnitt. Dabei muss aber darauf geachtet, dass die Sektoren nicht zu klein gewählt werden, da sonst die Feature-Qualität sinkt. Jeder Sektor könnte dann parallel berechnet werden.

Ähnlich sieht die Situation fast im gesamten Algorithmus aus. Die einzelnen Schritte werden jeweils für eine Menge von Sternen oder Feature-Tupeln durchgeführt, was ebenfalls gut parallelisierbar ist. Überall wo mittels der baryzentrischen Matrizen Koordinaten von Bild- auf Katalog umgerechnet werden, könnte die Matrizenmultiplikation gar auf die Grafikkarte ausgelagert werden. Es ist prüfenswert, ob allenfalls weitere Schritte auf die Grafikkarte ausgelagert werden können.

In diversen Bereichen könnten auch die Daten effizienter geladen werden. Bestes Beispiel hierfür ist wieder die Feature-Extraktion. Da Features jeweils aus den nächsten Sternen generiert werden müssen, werden hier viele Sortieroperationen durchgeführt. Da bekannt ist, dass es viel mehr dunkle wie helle Sterne gibt, könnte der Bereich, aus dem Feature-Kandidaten selektiert werden, verkleinern, je dunkler die gerade analysierten Sterne sind.

Mit weiteren Analysen lassen sich vermutlich weitere Möglichkeiten finden, den Algorithmus, bzw. dessen Umsetzung, auf Performance zu trimmen. Wie bereits erwähnt lag der Fokus dieser Arbeit aber nicht auf der Umsetzung, und schon gar nicht auf einer performanten, sondern auf dem Algorithmus selbst. Diese kurze Analyse kann somit als Anregung für weitere Verbesserungen interpretiert werden.

3.4 Mögliche Erweiterungen

Das gelöste Problem, das Finden von Asterismen in einem Sternenkatalog, kann auch ausgedrückt werden als das "lokalisieren eines Ausschnittes in einer gewichteten Punktwolke". So formuliert handelt es sich beim vorgestellten Algorithmus um die Lösung eines Spezialfalls des etwas allgemeineren Problems "lokalisieren eines Ausschnittes in einer Punktwolke".

Hat man statt Sternen einfach eine grosse Punktwolke, und will herausfinden ob sich eine kleinere Punktwolke darin lokalisieren lässt, ist dies äusserst aufwändig und wurde gemäss dem Wissen des Autors bisher nicht gelöst. Der vorgestellte Algorithmus arbeitet zwar mit gewichteten Punktwolken – also Sterne und deren Magnitude – die Gewichtung spielt aber während den Berechnungen nur eine Vergleichsweise kleine Rolle. Sie wird nämlich benötigt, um ein Ausschlusskriterium zu finden, welche Sterne mit welchen zusammen in einem Feature kombiniert werden können. Später wird sie auch noch benötigt, um den Feature-Score zweier Features zu berechnen. Setzt man den maximalen Magnitudenunterschied für die Featureselektion auf unendlich und gewichtet bei der Berechnung des Feature-Score die Magnitude mit 0, arbeitet der Algorithmus grundsätzlich ungewichtet.

In den restlichen Bereichen des Algorithmus spielt die Magnitude aber keine Rolle mehr. Man könnte versuchen, den Algorithmus mittels leichten Modifikationen so anzupassen, dass er auch mit ungewichteten Punktwolken umgehen kann. Dem Autor fallen während dem Schreiben dieser Arbeit leider keine Beispiele ein, wie das eingesetzt werden könnte, aber es gibt kaum einen Algorithmus, der keine Anwendung findet.

Literaturverzeichnis

- [1] NASA and H. Richer, *Ancient, White Dwarf Stars in the Milky Way Galaxy*. University of British Columbia, 1993,
<http://hubblesite.org/gallery/album/entire/pr2002010c/>
- [2] Earthsky, Aaron Robinson, *What makes a halo around the sun or moon?*.
<http://earthsky.org/space/what-makes-a-halo-around-the-moon>
- [3] Prof. Dr. Andreas Müller, *Linaere Algebra HS 2015 - Aufgabensammlung, Aufgabe 40000035*. Hochschule für Technik Rapperswil, 2015,
<https://github.com/AndreasFMueller/LinAlg/tree/master/aufgaben/4/40000035>
- [4] Prof. Dr. Andreas Müller, Hochschule für Technik Rapperswil, 2015
<https://github.com/AndreasFMueller/AstroPhotography/control/documents/localize/>
- [5] Private Kommunikation mit Prof. Dr. Andreas Müller.
- [6] US Naval Observatory, *The Fourth US Naval Observatory CCD Astrograph Catalog*
<http://ad.usno.navy.mil/ucac/>
- [7] American Association of Variable Star Observers, *The AAVSO Photometric All-Sky Survey*
<https://www.aavso.org/apass>

Appendix

A Baryzentrische Koordinaten

Es musste ein Verfahren entwickelt werden, wie wir möglichst Transformationsunabhängig verschiedene Punkte relativ zu einem Feature vergleichen können. Im Verlaufe der Berechnungen kommt es mehrmals vor, dass wir zwei ein Bild- und Katalogfeature f_B und f_K haben. Wir müssen dann herausfinden, welche Koordinate ein Bildpunkt \vec{p} im Katalog hätte, wenn es sich bei diesen zwei Features um dasselbe Feature handeln würde, einmal im Bild und einmal im Katalog.

Das Problem ist aber wie immer die Ungenauigkeiten, die während der Astroatnahme entstehen. Dreiecke können im Bild deshalb leicht verzerrt sein, oder gar auf den Kopf gestellt. Alle diese Effekte müssen berücksichtigt werden. Es stellt sich heraus, dass das baryzentrische Koordinatensystem perfekt für diesen Zweck geeignet ist.

Ein baryzentrisches Koordinatensystem ist ein homogenes Koordinatensystem, welches die Position eines Punktes relativ zu den drei Ecken des Dreiecks beschreibt. Es spannt also ein lokales Koordinatensystem um ein Dreieck herum auf. Die 2-Dimensionalen Stern-Koordinaten werden dadurch durch 3-Dimensionale baryzentrische Koordinaten beschrieben.

Diese Umwandlung geschieht mithilfe einer Matrixmultiplikation $\vec{w} = L(f_B)\vec{p}$. Der Resultatsvektor \vec{w} beinhaltet die 3 baryzentrischen Koordinaten. Der Vektor \vec{p} ist nichts weiter als die Koordinaten des umzuwandelnden Sternes s_B erweitert mit dem Element 1, also

$$\vec{p} = \begin{pmatrix} s_1 \\ s_2 \\ 1 \end{pmatrix}.$$

Die Spalten der Transformationsmatrix $L(f_B)$ entsprechen den Koordinaten der 3 Stern a , b und c des Ursprungsfeatures f_B , wiederum alle mit 1 erweitert:

$$L(f_B) = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 1 & 1 & 1 \end{pmatrix}. \quad (12)$$

Die Umwandlung von baryzentrischen zurück in absolute Koordinaten geschieht mit der Umkehroperationen $\vec{p} = L(f_B)^{-1}\vec{w}$, die Umwandlung von normalen in baryzentrische mithilfe der Inversen Matrix. Die vollständigen Umwandlungsoperationen sind somit

$$\begin{pmatrix} s_1 \\ s_2 \\ 1 \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} \quad (13)$$

und

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} s_1 \\ s_2 \\ 1 \end{pmatrix}. \quad (14)$$

Diese Matrix sollte ebenfalls für jedes abgelegt Feature vorberechnet werden, da sie mehrmals benötigt wird.

B Satz des Heron und Ähnlichkeitsinvariante

Der Satz des Heron beschreibt eine mathematische Formel, mit deren Hilfe der Flächeninhalt eines Dreiecks anhand der Seitenlängen berechnet werden kann.

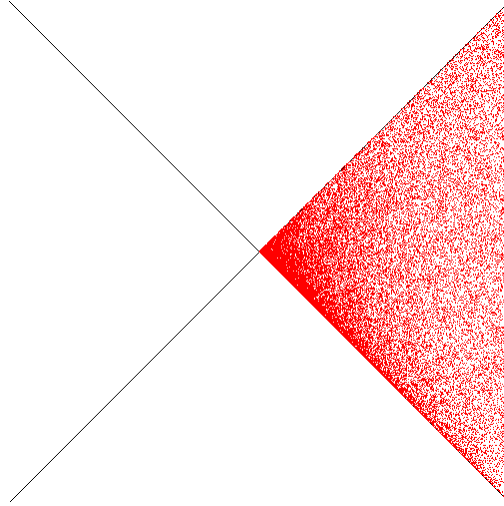


Abbildung 12: Ähnlichkeitsinvariante ohne Berücksichtigung des Heron-Wertes

Wenn der Flächeninhalt A mit den Seitenlängen a, b, c berechnet werden soll, wird der halbe Umfang der Seiten als

$$s = \frac{a + b + c}{2} \quad (15)$$

definiert. Die Fläche ist dann

$$A = \sqrt{s(s-a)(s-b)(s-c)}. \quad (16)$$

Der Satz des Heron stellt also einen Zusammenhang zwischen Fläche und Seiten eines Dreiecks her. Dieses Verhältnis machen wir uns zunutze. Denn je kleiner das Verhältnis zwischen der Fläche des Dreiecks und seiner längsten Seite, desto spitzer ist das Dreieck, was sich für unsere Zwecke schlecht eignet.

Berechnet wir für ein Dreieck mit Fläche A und längster Seite a also

$$h = \frac{A}{a^2} \quad (17)$$

erhalten wir einen Wert für die "Spitzigkeit" eines Dreiecks. Doch wie spitz darf ein Dreieck sein?

Dafür müssen wir die Verteilung von zufällig aus Punkten generierten Dreiecken analysieren. Wir nehmen denselben Algorithmus zur. Von diesen Dreiecken ist vorerst ausschliesslich das Seitenverhältnis relevant, wobei $a \geq b \geq c$ sein soll und a auf 1 normiert wird.

Die Seitenverhältnisse b und c werden dann als Vektor interpretiert. Dieser stellt die *Ähnlichkeitsinvariante* eines Dreiecks dar. *Ähnlichkeitsinvariante* deshalb, weil sich dieser Wert auch eignet, um die Ähnlichkeit zweier Features zu berechnen.

Wird die Koordinate wie in Abbildung 12 der *Ähnlichkeitsinvariante* für jedes Feature auf eine Ebene gezeichnet, ist schön ersichtlich, wie sich die Seitenverhältnisse der verschiedenen Features verteilen und somit auch, welche Arten von Features besonders häufig sind.

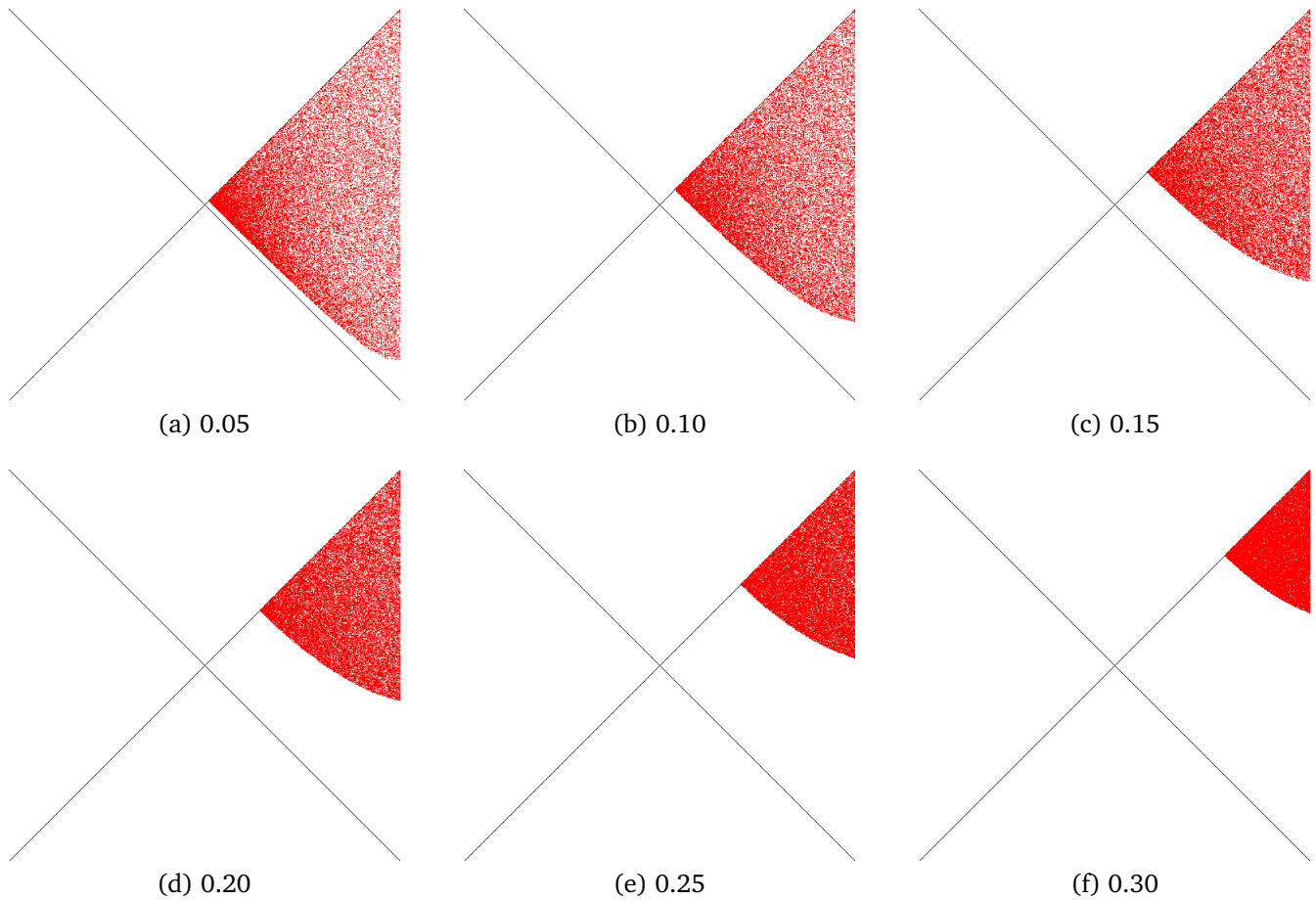


Abbildung 13: Ähnlichkeitsinvariante bei verschiedenen Heron-Grenzwerten

In Abbildung 12 entspricht die horizontale der Seite b , die Vertikale ist c , wobei die Skala jeweils von 0 bis 1 geht. Da b und c zusammen mindestens 1 ergeben müssen ist die linke Seite der Abbildung leer. Am linken, unteren Rand des gefüllten Bereiches sind jene Dreiecke, die extrem spitz sind ($b+c=1$). Dreieck am linken, oberen Rand hingegen sind alle Gleichschenkelig ($b=c$), und in der oberen, rechten Ecke sind sie gar gleichseitig ($a=b=c=1$).

Wie erwähnt sind vor allem spitzwinklige Dreiecke eher schlecht geeignet, da in solchen Fällen Verzerrungen in der Astroatnahme einen vergleichsweise grossen Einfluss haben. Wir schliessen deshalb alle Dreiecke, die einen bestimmten Heron-Wert unterschreiten, aus.

In Abbildung 13 wird der Einfluss des Grenzwertes auf die Featureextraktion gezeigt. Es ist schön zu erkennen, wie die Dreiecke bei steigendem Heron immer weniger spitz werden.

Es ist aber auch zu erkennen, dass sich die Dreiecke dafür untereinander immer mehr gleichen, was der Qualität des Algorithmus ebenso abträglich ist. Der Grenzwert darf deshalb auch nicht zu gross gewählt werden. Am Ende wurde der Wert 0.15 gewählt. Diese Entscheidung ist jedoch eher willkürlich und hängt auch stark davon ab, wie genau die Daten aus einer Astroatnahme extrahiert werden können. Je genauer die Daten, desto weniger Einfluss haben Verzerrungen und andere Fehler. Bei dieser Überlegung auch nicht zu vernachlässigen ist der Performanceeinfluss. Denn je grösser der Grenzwert, desto mehr Features müssen während der Extraktion verworfen werden, wodurch auch mehr Features extrahiert werden müssen.

C Symboltabelle

In Tabelle 3 sind die verschiedenen, in der Arbeit verwendeten Notation ersichtlich.

Objekt	Symbol	Im Bild	Im Katalog
Bild	B	-	-
Katalog	K	-	-
Stern	s	s_B	s_K
Feature	f	f_B	f_K
Sternmenge	S	S_B	S_K
Featuremenge	F	F_B	F_K

Tabelle 3: Symboltabelle

Weiter werden Indexe auf diesen Elementen hochgeschrieben. So ist der Stern b eines Katalogfeatures f_K^b und das i -te Element der Bildsternmenge S_B^i . Auf Objekten können diverse Eigenschaften vorhanden sein, wie in der nachfolgenden Aufzählung ersichtlich. Die Notation für solche Eigenschaften ist $(S_B^i)_{\text{Eigenschaft}}$. Es können folgende Eigenschaften auf den verschiedenen Elementen vorhanden sein:

1. Stern

- (a) x : X (bzw. Right Ascension)
- (b) y : Y (bzw. Declination)
- (c) mag : Magnitude
- (d) bar : Stern umgewandelt in baryzentrische Koordinaten
- (e) 1: Komponente 1 der baryzentrischen Koordinaten
- (f) 2: Komponente 2 der baryzentrischen Koordinaten
- (g) 3: Komponente 3 der baryzentrischen Koordinaten

2. Feature

- (a) sb : Ähnlichkeitsinvariante Komponente B
- (b) sc : Ähnlichkeitsinvariante Komponente C
- (c) bsm : Magnitude des Hellsten Sternes

In Kapitel Abschnitt 2.3.2 werden verschiedene Score Funktionen definiert, z.B. $\text{Score}_{\text{Feature}}$. Solche und andere Operationen werden mit $\text{Score}_{\text{Feature}}(f^i, f^j)$ bezeichnet.

D Anpassbare Parameter

Es wurden fünf Parameter gefunden, welche steuern können, wie gut der Algorithmus mit unscharfen Daten umgehen kann.

Maximaler Magnitudenunterschied zwischen Feature-Sternen Beschreib den maximalen Magnitudenunterschied, der zwischen einzelnen Sternen eines Features herrschen darf. Kann bei genauen Magnituden-Daten kleiner gewählt werden.

Heron-Grenzwert Beschreibt den minimalen Heron-Wert, der ein Feature besitzen darf, um akzeptiert zu werden. Kann bei genauen Positionsdaten kleiner gewählt werden

Baryzentrische Schranke Die untere und obere baryzentrische Schranke definieren, welche Sterne als nah bei einem Feature gelten. Anpassung hängt von der Situation ab. Genaue Positionsdaten würden zwar eine höhere, obere Schranke erlauben. Dadurch gelten aber auch viel mehr Sterne als nah, was die Performance negativ beeinflusst.

Sektorgrösse Bei der Analyse der projizierten Bildmittelpunkte im Katalog wird der Katalog in Sektoren aufgeteilt, für welche ein kombinierter $\text{Score}_{\text{Sector}}$ berechnet wird. Die Grösse dieses Sektors kann bei genauen Positionsdaten kleiner gewählt werden

Delta Ähnlichkeitsinvariante Der maximale Wert, um den sich die *Ähnlichkeitsinvariante* zweier Features unterscheiden darf, damit diese als ähnlich gelten.

E Selbstreflexion

Ich war mir von Anfang an bewusst, dass ich nicht eine typische Studienarbeit durchführen wollte. Ich wollte also nicht eine mittelgrossen Entwicklungsarbeit durchführen. Die Möglichkeit, praktische Erfahrung habe ich dank meiner Arbeit als Software Engineer in der CREALOGIX AG bereits reichlich. Da ich auch ein mathematisch interessierter Mensch bin, spielte ich schon früh mit dem Gedanken, eine Arbeit mit einem solchen Hintergrund bei Prof. Dr. Andreas Müller durchzuführen. Nach Abschluss der Hochschulausbildung bietet sich die Möglichkeit, wirklich über eine längere Zeit an einem Algorithmus arbeiten zu können, wohl nur noch selten.

Die Arbeit stellte sich dann auch von Anfang an als äusserst spannend, aber auch anspruchsvoll heraus. Meistens waren es dieselben Dinge, die sowohl zum einen, wie auch zum anderen führten.

In den ersten Wochen ging es vor allem darum, den Inhalt des Problems genau abzustecken und erste Überlegungen anzustellen, wie denn eine solche Astroaufnahme genau mit einem Sternenkatalog zur Deckung gebracht werden könnte. Dabei wurden Feature-Detection Probleme wie Fingerabdruck- und Gesichtserkennung analysiert. Mit den daraus gewonnen Erkenntnissen und den Resultaten der eigenen Analysen konnte dann begonnen werden, den Algorithmus zu entwickeln.

Ich wurde während der gesamten Arbeit immer äusserst gut betreut, worum ich auch froh war. Die mathematischen Konstrukte zu finden, mit welchen die verschiedenen Probleme lösbar wurden, ist nicht immer einfach. Bei der grundsätzlichen Erarbeitung des Algorithmus, wie also die verschiedenen Techniken eingesetzt werden konnten, war ich auf mich alleine gestellt.

Dieser Teil der Arbeit war aber auch der spannendste. Zu diesem Zeitpunkt begann die Entwicklung der Testimplementation und des Simulators. Und es erfreut doch jedes Entwicklerherz, an einem Problem herumzutüfteln, immer wieder kleine Verbesserungen oder ganz neue Ansätze für ein bekanntes Problem zu finden.

Diese Art und Weise der Studienarbeit führte aber auch Probleme mit sich. Durch den unklaren Ausgang und vielen nötigen Analysen fehlte zu Beginn ein wenig die Richtung. So habe ich mich mir viel zu früh Gedanken um kleine Details gemacht, während der Blick fürs Ganze teilweise etwas verloren ging. Nichtsdestotrotz kam im Abschluss an diese Phase, etwa in der Hälfte des Semesters, ein funktionierender Algorithmus heraus, der das Problem in den simulierten Daten lösen konnte.

Danach ging es an die Anpassung an reale Katalogdaten. Dieser Abschnitt war zwar nicht der komplizierteste, die Anpassung führte aber doch immer wieder zu kleineren Problemen. Ich hatte vor allem auch mit der plötzlich enorm gestiegenen Datenmenge zu kämpfen. Die Generierung der Features dauerte länger als gedacht und war sehr schwierig zu optimieren. Da es zu Beginn auch noch einige Probleme im Feature-Extraktor gab, musste dieser Schritt dazu auch noch mehrmals wiederholt werden.

Der Algorithmus konnte dadurch erst ganz gegen Ende der Arbeit richtig getestet werden, was natürlich ungünstig ist. Dadurch fehlte bis ganz am Ende ein Beweis, dass der Algorithmus auch tatsächlich korrekt arbeitete.

Nachträglich betrachtet, gab es zwei Abschnitte in der Arbeit, die besser hätten laufen können. Einerseits hätte ich mich während der Erarbeitung des Algorithmus besser auf die Gesamtheit des Problems fokussieren müssen. Die Entwicklung der Vorselektionsmethoden geschah erst vergleichsweise spät.

Der zweite Punkt ist war die Anpassung an reale Katalogdaten. Es hätte auf keinen Fall so lange dauern dürfen, bis die Feature-Extraktion korrekt funktionierte. Ich bin froh, dass am Ende

dann doch noch alles funktioniert hat.

Abschliessend kann ich nur sagen, dass ich während der Arbeit, wenn Sie auch anspruchsvoll war und nicht immer alles optimal geklappt hat, immer Spass hatte und vor allem äusserst viel gelernt habe.

F Aufgabenstellung

Aufgabenstellung

Lokalisierung von Asterismen mit Hilfe von Sternkatalogen

Aufnahmen mit fest montiertem Teleskop und Kameras haben eine bekannte Orientierung und Winkelauflösung/Pixel. Damit ist es einfach, die Bilder nach Norden auszurichten und den Bildmittelpunkt pixelgenau auf der Himmelskugel zu lokalisieren.

Mobil eingesetzte Amateurteleskope sind oft nicht genau aufgestellt, die Kameraorientierung ist nicht bekannt, und manchmal, zum Beispiel beim Einsatz von Barlow-Linsen mit variabler Brennweite ist nicht einmal die exakte Brennweite des Gesamtsystems bekannt.

Übliche Softwareprodukte können Bilder zur Deckung bringen, aber gegeneinander verdrehte Bilder nur in einem kleinen Winkelbereich erkennen und korrigieren. Meistens ist es unmöglich, die Nordrichtung in einem Bild zu ermitteln, oder die genauen Koordinaten des Bildmittelpunktes.

Die meisten Bildverarbeitungs-Algorithmen für solche Lokalisierungsproblem verlangen, dass die Bilder klar identifizierbare Features enthalten. Bilder von Sternfeldern enthalten aber nur Punkte. Die Lokalisierung kann sich also nur auf Gruppen von Sterne, sogenannte Asterismen stützen. Es kann also nötig sein, aus Asterismen identifizierbare Features zu erzeugen.

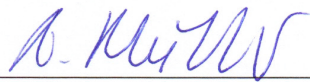
Mit den modernen, hoch aufgelösten Sternkatalogen mit sehr geringen Grenzhelligkeiten (zum Beispiel 16 Grössenklassen bei UCAC4) sollte es möglich sein, von jedem genügend grossen und tiefen Bild (tief im Sinne von ausreichender Grenzhelligkeit) die Position und die Orientierung zu ermitteln.

Ziele dieser Arbeit sind:

- Verschiedene algorithmische Möglichkeiten der Feature-Lokalisierung auf deren Eignung für das vorliegende Problem hin prüfen
- Einfluss des optischen Systems (Auflösung, point spread function, Verzerrung durch das optische System, Verzerrung grossformatiger Bilder durch atmosphärische Aberration) auf die Leistung der Algorithmen evaluieren.
- Mögliche Vereinfachungen bei bekannter Winkelauflösung, bekannter ungefährender Orientierung und weiterer Parameter bewerten.
- Anforderungen an Sternkataloge und APIs ableiten.

Betreuer: Prof. Dr. Andreas Müller

Student: Max Obrist

14.9.2015 
Datum Unterschrift

19.09.15 
Datum Unterschrift

G Eigenständigkeitserklärung



Eigenständigkeitserklärung

Semesterarbeit

Rapperswil, 18.12.2015

Titel der Arbeit: **Lokalisierung von Asterismen mithilfe von Sternenkatalogen**

Team: **Max Obrist**

Betreuer: **Prof. Dr. Andreas Müller**

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- dass ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Max Obrist

H Einverständniserklärung Publikation



Einverständniserklärung Publikation auf eprints.hsr.ch

Semesterarbeit

Rapperswil, 18.12.2015

Titel der Arbeit: **Lokalisierung von Asterismen mithilfe von Sternenkatalogen**

Team: **Max Obrist**

Betreuer: **Prof. Dr. Andreas Müller**

Wir sind mit der Publikation unserer Arbeit auf eprints.hsr.ch einverstanden, sofern für diese Arbeit keine Geheimhaltungsvereinbarung unterzeichnet wurde.

Nach Bekanntgabe der Note haben wir die Möglichkeit innert 14 Tagen Einsprache zu erheben und das Einverständnis zur Publikation der Arbeit auf eprints.hsr.ch zurückzuziehen. In diesem Falle wird nur der Abstract publiziert.

Max Obrist

A handwritten signature in black ink, which appears to be 'Max Obrist', is written over a horizontal line.