|  |
| --- |
| HAW Landshut |
| Faktorisierungsalgorithmus zur Affinen Rekonstruktion von Kameras und 3d-Punkten |
| Studienarbeit im Fach Bildverstehen |

|  |
| --- |
| Tobias Weiden  12.7.2019  Prof. Siebert |

# Einleitung

Eine Rekonstruktion von 3-dimensionalen Punkten aus 2D-Bildern ist ein weitgehend erforschtes Feld. Hierbei gibt es gerade in einer immer weiter automatisierten Welt immer mehr Use-Cases. Ob selbstfahrende Autos, Roboter, die mit Menschen interagieren, oder einfache Objekt-/ oder Personen-Analysen; Für immer mehr Nutzfelder ist es interessant ein dreidimensionales Modell bspw. der Umwelt zu konstruieren. Hierbei werden verschiedenste Präferenzen gesetzt: Zum einen soll ein Modell möglichst genau oder möglichst schnell erstellt werden, um darauf zu reagieren.

Die meisten Methoden zur Erstellung eines 3D-Modells werden durch eine Verarbeitung von mehreren Bildern aus unterschiedlichen Perspektiven, auf denen bestimmte und bekannte Bildpunkte vorhanden sind.

Hierbei werden Kamera-Matrizen erstellt, die 3D-Punkte in 2D-Punkte wandeln und umgekehrt.

Im Folgenden wird eine Spezialisierung von Kamera-Modellen (Affine Kamera) erläutert, mit dessen Hilfe ein einfacher Algorithmus zur Erstellung von Kamera-Matrizen durchgeführt wird. Dies ist der Faktorisierungsalgorithmus zur Affinen Rekonstruktion von Hartley & Zisserman.

Dieser wird dann mit qualitativ und zeitlich mit der Standard-Kalibrierung von OpenCV verglichen. Hierzu wird zum einen der direkte Reprojektions-Fehler verglichen, welcher die durchschnittliche Differenz zwischen den Original-Punkten und den reprojezierten Punkten darstellt, und der Reprojektions-Fehler bei verschobenen 3D-Punkten. Dies wird mit selbstaufgenommenen Bildern aus verschiedenen Distanzen verglichen.

Die Implementierung erfolgte mittels Python und auch für die OpenCV-Kalibrierung wurde die offizielle Python-Bibliothek genutzt.

# Kameras und Projektion

Durch Projektion kreiert eine projektive Kamera ein flaches Bild aus dreidimensionalen Informationen. Es werden also 3D-Punkte in 2D-Punkte umgewandelt. [1]

## Perspektivische Projektion

Um die Tiefe des dargestellten Raumes zu begreifen, vergleichen wir bei einem zweidimensionalen Bild kameranahe Objekte mit kamerafernen Objekten (im Hintergrund). Objekte mit gleicher Größe in der realen Welt werden je nach Nähe zur Kamera größer (nah der Kamera) oder kleiner (fern der Kamera) dargestellt.

Liegt nun ein Objekt „entlang“ der Tiefe, so wird dieses der Tiefe hin kleiner. Im folgenden Beispiel wird die Bande kleiner, je weiter sie von der Kamera entfernt ist:



Abbildung 1: Bande, 5 Meter entfernt der rechten senkrechten Spange  
  
Ein Bild entlang einer Bande. Der linke Teil war näher der Kamera und wird somit größer dargestellt als der rechte Teil. Dadurch lässt sich die Tiefe für den Betrachter erahnen.

In diesem Bild tritt auch die Perspektivische Projektion auf: Linien, die in der echten Welt parallel verlaufen, konvergieren im Bild. Diese laufen im Fluchtpunkt zusammen. Dies kann man am Beispiel an dem oberen und unteren Ende der Bande nachvollziehen, wie im Folgenden verdeutlicht: [1]

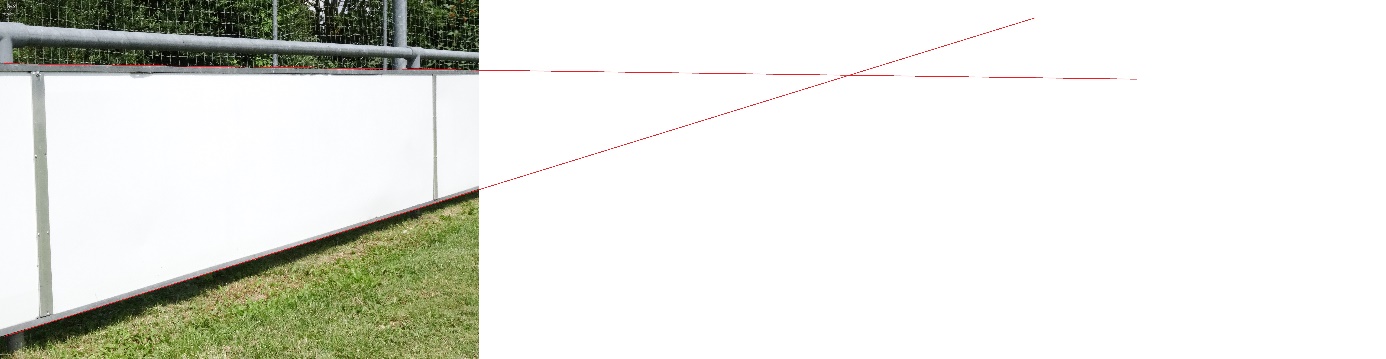


Abbildung 2: Bande mit eingezeichnetem Fluchtpunkt, 5 Meter Entfernung

## Lochkamera

Wir gehen von einer zentralen Projektion von Punkten im Raum auf eine Ebene aus. Wenn das Zentrum der Projektion der Ursprung des euklidischen Koordinatensystems ist, so gibt die Brennweite dem Abstand zwischen Ursprung und Bildebene auf der Ebene wieder, wie in Abbildung 3: Skizze einer Lochkamera [2] zu sehen. [2]

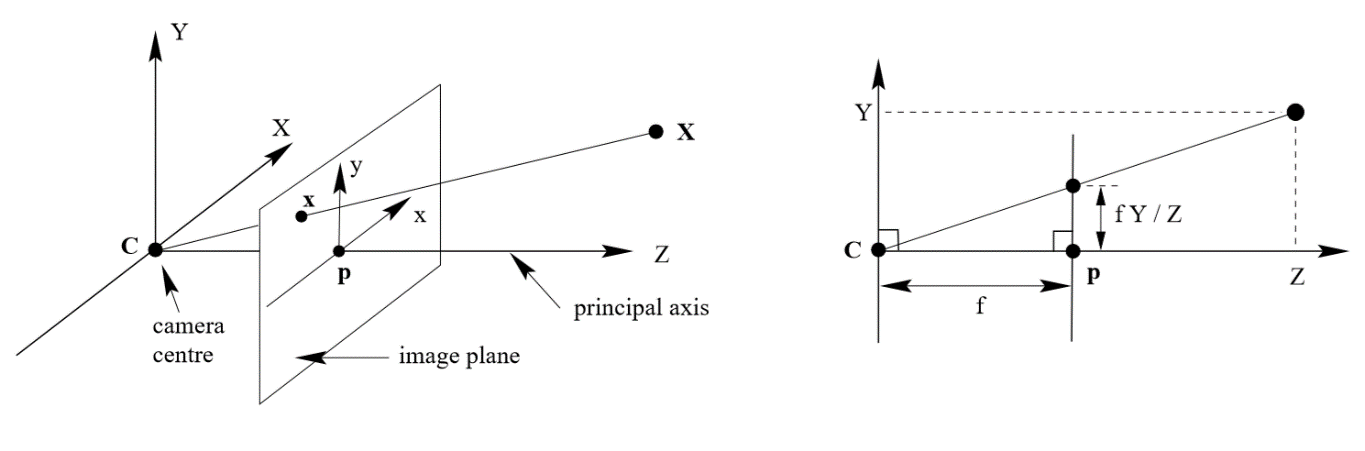


Abbildung : Skizze einer Lochkamera [2]

## Orthogonalprojektion

Wir betrachten nun folgende Bilder:

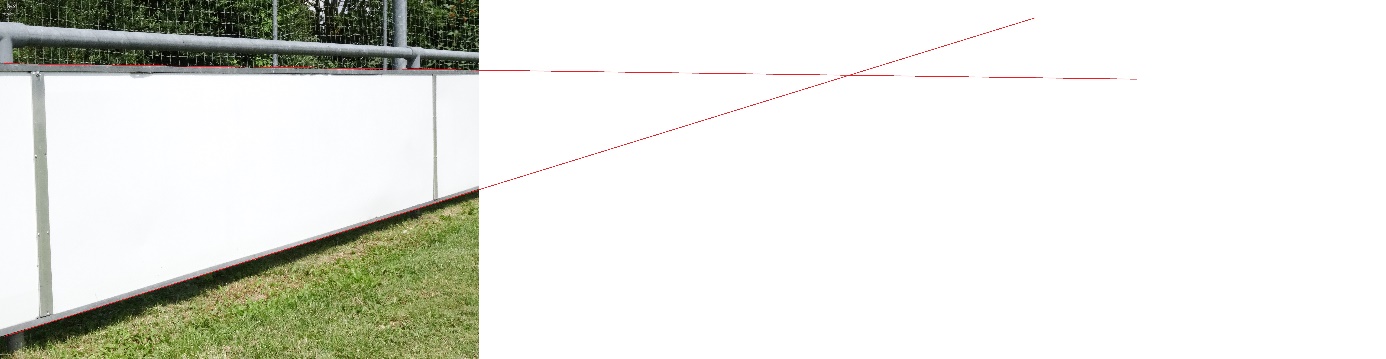


Abbildung 4: Bande mit Fluchtpunkt, 5 Meter Entfernung, 11mm Brennweite

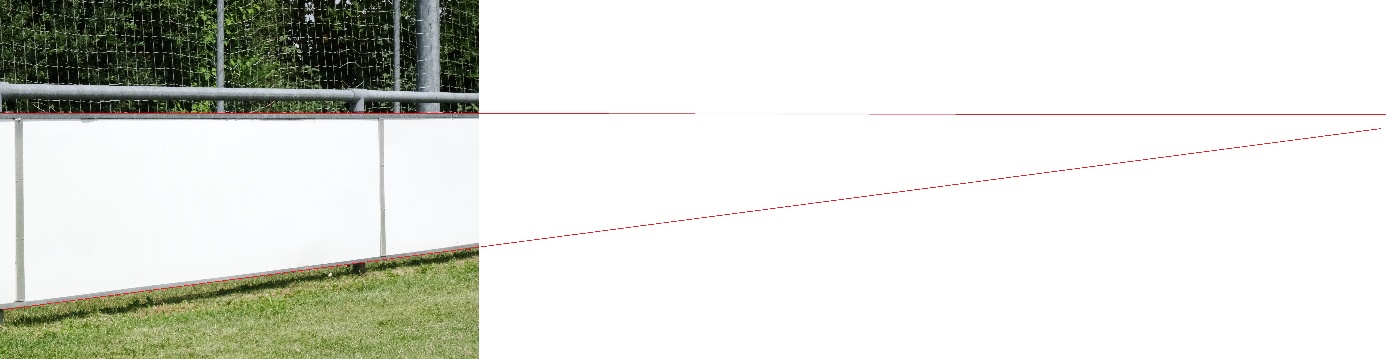


Abbildung 5: Bande mit Fluchtlinien, 10 Meter Entfernung, 26mm Brennweite

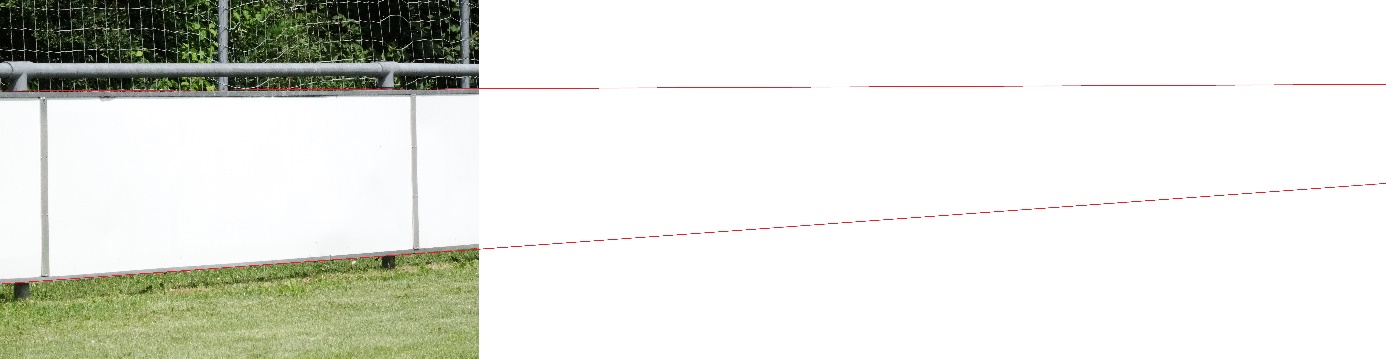


Abbildung 6: Bande mit Fluchtlinien, 20 Meter Entfernung, 55mm Brennweite

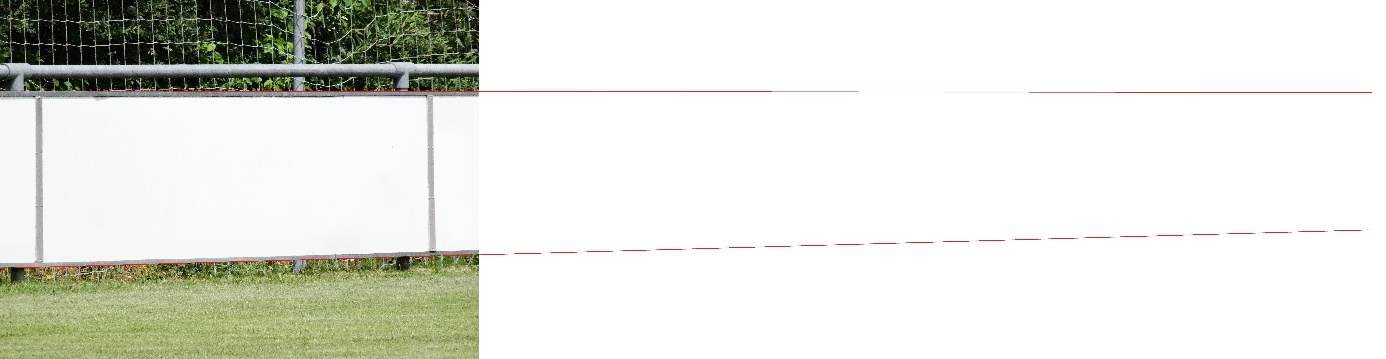


Abbildung 7: Bande mit Fluchtlinien, 50 Meter Entfernung, 124mm Brennweite

Bei den Bildern wird graduell die Brennweite (und die Distanz zum Objekt, damit dieses im gleichen Ausschnitt auf den Aufnahmen bleiben) erhöht. Wie zu erkennen ist, entfernt sich der Fluchtpunkt von der Bildmitte. Die Bilder verschieben sich von einer starken zu einer schwachen Perspektive.

Es kann also theoretisch angenommen werden, dass bei einer unendlich großen Brennweite bzw. Distanz zwischen Kamerazentrum und Bildebene in der realen Welt parallele Linien auch im Bild parallel bleiben. Dies wird als Orthogonalprojektion bezeichnet. [1] [2]

## Affine Kamera

Eine Affine Kamera hat ihren Mittelpunkt auf der Ebene in der Unendlichkeit. Die Annahme einer Affine Kamera hat Vorteile, da es Berechnungen vereinfacht und stabilisiert. Natürlich gibt es keine solche Kamera in der Realität. Um ein annehmbares Bild unter solchen Bedingungen zu erhalten reicht es in der Regel, Bildpunkte auf der gleichen Ebene für Berechnungen zu nutzen (also nicht gleichzeitig ein Punkt im Vorder- und Hintergrund des Bildes) und Bilder mit einer höheren Brennweite zu nutzen.

Inwiefern der Faktor der Entfernung/Brennweite das Ergebnis von Algorithmen, die mit Hilfe von Affinen Kameras arbeiten, wird im Folgenden betrachtet.

# Affine Rekonstruktion – Faktorisierungsalgorithmus

Ziel ist es, mittels Bilder ( entspricht der Anzahl der Bilder) von Affinen Kameras Kameras und 3D-Objekt-Punkte zu ermitteln, sodass der geometrische Fehler minimal ist:

Hier wird eine Affine Kamera wie folgt beschrieben:

ist eine -Matrix und t ein -Vektor. wird hierbei als weiter behandelt und ist ein inhomogener Bildpunkt, entsprechend ist der inhomogene Weltpunkt .

Wie üblich in solchen Minimierungsproblemen, wird der Translationsvektor eliminiert, indem dieser als Zentrum zwischen den Objektpunkten definiert wird. Bei der Affinen Kamera wird hierbei das Zentrum zwischen den Objektpunkten auf das Zentrum zwischen den Bildpunkten abgebildet. Daher ist nun kein Translationsvektor notwendig, weshalb gilt: . Damit dies gilt, müssen allerdings alle Objektpunkte für jede Kamera auf vorhandene Bildpunkte abgebildet werden. Somit ist nun folgende Formel gegeben:

Dies kann nun einfach als Matrix dargestellt werden: Die Messmatrix ist eine -Matrix aus den zentrierten Bildpunkten:

Da jeder Bildpunkt ist, kann diese auch umgeschrieben werden:

Da Rauschen vorhanden ist, kann die Gleichung nicht exakt erfüllt sein, also wird eine Matrix gesucht, die möglichst ähnlich zu in der Frobenius-Form ist. Diese Matrix kann letztendlich per SVD mit Rang 3 erstellt werden. Hierbei gibt es keine eindeutige Lösung, da:

Somit gibt es je zwei Möglichkeiten und zu definieren:

oder

Damit ergibt sich folgender Algorithmus: [2]

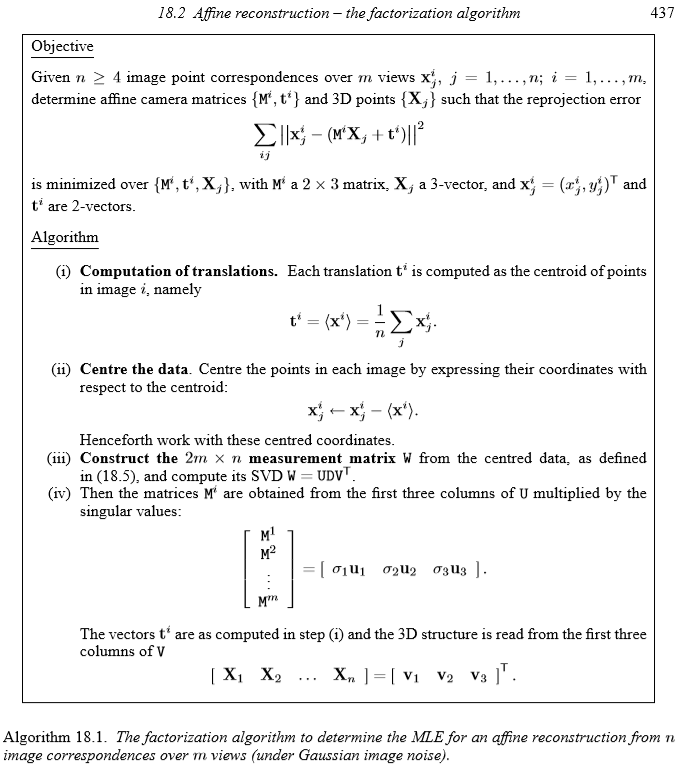


Abbildung : Faktorisierungs-Algorithmus von Hartley & Zisserman [2]

# Implementierung

Zur Implementierung des folgenden Codes wurde Python genutzt. Zunächst werden die Bilder eingelesen, also eindeutige Bildpunkte bestimmt. Für diese wird dann mit Hilfe der OpenCV-Bibliothek eine klassische Kamera-Matrix bestimmt. Dann werden Kamera-Matrizen mit dem Faktorisierungs-Algorithmus von Hartley & Zisserman bestimmt. Die Ergebnisse werden dann zunächst via geometrische Fehler verglichen. Zusätzlich werden dann die 3D-Punkte verschoben und mit den Matrizen abgebildet, um diese dann mit weiteren bestimmten Bildpunkten zu vergleichen.

## Vorbereitung

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | M. Obeysekera, *Affine Reconstruction from Multiple Views using Singular Value Decomposition,* The University of Western Australia, 2003. |
| [2] | R. Hartley und A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge: Cambridge University Press, 2003. |