

Implementierung von Skelettierungs-Algorithmen mit dem Kinect-Sensor

— Projektbericht —

Arbeitsbereich Kognitive Systeme

Fachbereich Informatik

Fakultät für Mathematik, Informatik und Naturwissenschaften

Universität Hamburg

Vorgelegt von:

Johannes Böhler,
Christopher Kroll (6367171),
Sandra Schröder (6060939)

Hamburg, den 10.03.2013

Inhaltsverzeichnis

| | |
|---|-----------|
| 1 Einleitung | 3 |
| 1.1 Aufgabenstellung | 3 |
| 1.2 Aufbau der Projektarbeit | 4 |
| 2 Die Kinect | 5 |
| 3 Die Skelettierung | 10 |
| 3.1 Thinning | 10 |
| 3.2 Distanztransformation | 11 |
| 3.3 Weitere Verfahren | 12 |
| 4 Verwandte Arbeiten | 13 |
| 4.1 Extracting Skeletons From Distance Maps | 13 |
| 4.2 A Fast Parallel Algorithm for Thinning Digital Patterns | 17 |
| 5 Implementierung der Algorithmen | 22 |
| 5.1 Technische Umsetzung | 22 |
| 5.2 Spieler-Segmentierung | 22 |
| 5.3 Thinning | 23 |
| 5.4 Skelettierung aus der Distanztransformation | 23 |
| 6 Ergebnisse | 24 |
| 6.1 Vergleich der Algorithmen | 24 |
| 6.2 Verbesserung der Skelettqualität | 25 |
| 7 Zusammenfassung | 32 |
| 8 Fazit und Ausblick | 33 |
| Literaturverzeichnis | 34 |
| A Quellcode | 35 |
| A.1 Verbesserung der Skelettqualität | 35 |

1. Einleitung

Autor: Christopher Kroll

Im Master-Studiengang Informatik an der Universität Hamburg ist ein zweisemestriges Projekt vorgesehen. Die Autoren dieser Arbeit belegten im Sommersemester 2012 und Wintersemester 2012/2013 das Projekt 'Bildverarbeitung' unter der Leitung von Prof. Dr. Leonie Dreschler-Fischer. Der Verlauf und das Ergebnis dieses Projektes wird in dem vorliegenden Dokument dargestellt.

Die Studenten sollten in diesem Projekt die Daten der Microsoft Kinect-Kamera nutzen und diese für ein Thema ihrer Wahl verarbeiten. Nach einer Vorstellung der Kinect und der Programmiersprache Python wurden die Gruppen eingeteilt und das Thema gewählt.

Was ist Skelett? Bla bla, warum toll? Ein Skelett ist ein nutzlicher Deskriptor, um Informationen über die Region und den Rand eines Objektes kompakt und effizient zu kodieren. Es gibt die wesentlichen Grundzüge eines Objektes wieder.

1.1. Aufgabenstellung

Unsere Gruppe entschied sich für die Datenauswertung einer Person ('Spieler'). Dabei war zunächst das Ziel den Bewegungsablauf bei Sportübungen zu analysieren und eine Rückmeldung zu geben, ob diese richtig ausgeführt wurden. So soll zum Beispiel bei Kniebeugen durch eine Messung des Winkels zwischen Ober- und Unterschenkel der Person eine Hilfestellung gegeben werden.

Um dieses Ziel zu erreichen muss zunächst der Spieler von anderen Gegenständen im Raum getrennt, also herausgefiltert werden (Segmentierung). Für die Bewegungsanalyse ist es hilfreich nicht den gesamten Menschen, womöglich noch mit störender Kleidung, zu betrachten, sondern nur sein Skelett. Um das Skelett zu erhalten, bot sich die Wahl zwischen schon implementierten Skelettierungsalgorithmen zu benutzen oder dies selbst zu implementieren. Da das Thema des Projektes die Bildverarbeitung und nicht eine Anwendungsprogrammierung ist, fiel die Entscheidung auf die Konzentration auf die Skelettierung. Das Ziel war nun verschiedene Ansätze zu implementieren und hinsichtlich Qualität und Leistung zu vergleichen.

TODO skelettddefinition TODO anforderungen

1.2. Aufbau der Projektarbeit

2. Die Kinect

Autor: Johannes Böhler Die drei Haupt Hardware-Komponenten der Kinect sind ein Infrarotprojektor, eine RGB-Kamera sowie eine Infrarotkamera. Die Kombination aus Infrarotstrahler und Infrarotkamera



Abbildung 2.1.: *Links:* Infrarotprojektor *Mitte::* RGB-Kamera *Rechts::* Infrarotkamera

ermöglicht die Gewinnung von Tiefeninformationen aus der Umgebung. Im Gegensatz zu gewöhnlichen Kameras welche einem Pixel Farbinformation (z.B. über RGB-Farbkanäle) zuordnen, wird dem Pixel mit Hilfe der Infrarot Kamera eine Entfernung zugeordnet. Diese ergibt sich aus der Art und Weise wie der Infrarotstrahl von dem durch den Pixel repräsentierten Bereich eines Objektes reflektiert wurde.

Die Tiefenbilder welche man von der IR Kamera erhält, sehen aus wie komplett verrauschte Graustufenbilder. Hierbei steht jeder Grau Wert eines Pixels für die entsprechende Entfernung des korrespondierenden Objektausschnittes zur Kinect.

Hohe Grauwerte (helle Pixel) repräsentieren nahe Objekte, während niedrige Grauwerte (dunkle Pixel) weiter entfernte Objekte beschreiben. In einem Tiefenbild ist die gesamte Information über die Entfernung der im Bildausschnitt erfassten Objekte zur Kinect enthalten.

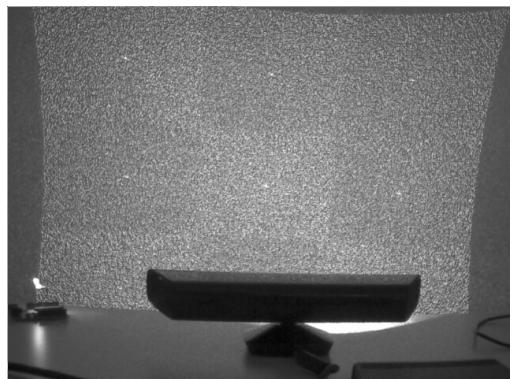


Abbildung 2.2.: Vom Infrarotprojektor ausgestrahltes strukturiertes Licht

Wird die Tiefeninformation dazu genutzt um die Pixel im dreidimensionalen Raum

anzuordnen, erhält man eine Punktwolke. Die in der 2d -Betrachtung benachbarte Pixel müssen in der 3d Repräsentation nicht miteinander verbunden da die Z-Koordinate unterschiedliche Werte aufweisen kann.

2.0.1. Funktionale Komponenten

Infrarotprojektor

Der Infrarotprojektor emittiert elektromagnetische Strahlen, deren Wellenlänge (830nm) außerhalb des für den Menschen sichtbaren Bereichs liegt (380nm-780nm). Der Projektor strahlt zur Tiefenbestimmung ein Gitter von Infrarotpunkten (strukturiertes Licht) auf die Objekte in seiner Umgebung ab. Zur Generierung dieses Musters wird ein besonderes Verfahren implementiert. Normalerweise produzieren Filter die diese Art von Muster generieren einen sehr hellen Punkt in der Mitte des Bildes, welcher die Leistungsfähigkeit des IR Projektors limitiert. Durch das hier verwendete Verfahren entstehen statt einem einzigen sehr hellen, neun helle Punkte, welche durch unvollständige Lichtfilterung zur Mustererstellung bedingt sind. Das hier eingesetzte Verfahren produziert weniger starke Artefakte und ermöglicht die Verwendung einer leistungsfähigeren Diode was eine erhöhte Genauigkeit sowie eine größere Reichweite ermöglicht. Die Reichweite des IR Projektors ist dennoch eingeschränkt, da zu hohe Intensitäten der IR Strahlen Augenschäden verursachen könnten.

Es ist wichtig dass die Wellenlänge der Infrarot strahlen konstant bleibt, was durch eine gleichbleibende Temperatur der Laserdiode und eine konstante Stromleistung (60 mW) gewährleistet wird. Problematisch sind variierende Außentemperaturen (die empfohlene Betriebstemperatur liegt zwischen 5 und 35 °C). Um diese Konstanz zu gewährleisten wird ein sogenanntes Peltier-Element verwendet, welches sowohl kühlen als auch wärmen kann.

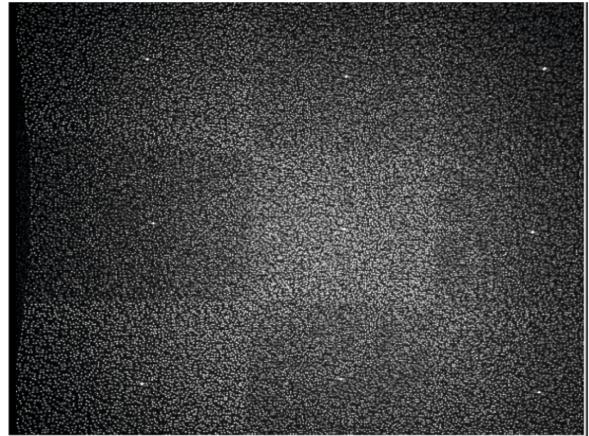


Abbildung 2.3.: Generierung des Infrarot Random-Patterns mit Hilfe von 9 Bereichen

Infrarotkamera

Die IR Kamera nimmt Bilder mit einer nativen Auflösung von 1280x1024 Pixeln, bei einer Bildwiederholrate von 30 Hz auf. Weitergeleitet werden allerdings nur Bilder mit einer Auflösung von 640x480 Pixeln da der USB Datenbus eine Limitierung bezüglich der Datenübertragungsrate darstellt. Das Blickfeld der IR Kamera beträgt in der Horizontalen 58 °, in der Vertikalen 45 °. Damit sich die Mustererkennung funktioniert ist eine Mindestabstand von 0,8 Metern

erforderlich, ab einer Entfernung von 3,5 Metern wird die Intensität der reflektierten Strahlen zu gering um Tiefeninformationen mit ausreichender Präzision zu erhalten. Bei einem optimalen Abstand von 2 Metern zum Objekt beträgt die Auflösung in der XY Ebene 3mm, in der Z Ebene 1cm. Die Quantisierungsauflösung liegt bei (2048) Bit. Es muss sichergestellt werden dass die IR Kamera nur die erwünschte elektromagnetische Strahlung im 830 nm Bereich aufnimmt und nicht von Strahlungen anderer Wellenlänge gestört wird . Dies wird durch einen Filter, welcher auf dem IR Kameraobjektiv angebracht ist realisiert. Trotz des Filters sollte die Kinect in abgedunkelten Innenräumen verwendet werden, da Sonnenlicht auch elektromagnetische Wellen im Infrarotbereich enthält.

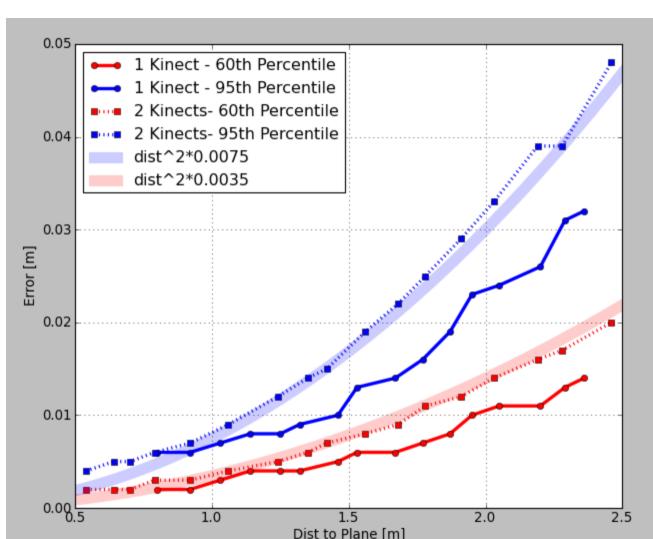


Abbildung 2.4.: Genauigkeit der Tiefenmessung in Abhängigkeit von der Objektentfernung zur Kinect

RGB Kamera

Die RGB Kamera nimmt bei einer Wiederholrate von 30 Hz ebenfalls mit einer Auflösung von 640x480 Pixeln auf, könnte jedoch auch mit einer Auflösung von 1280x1024 Pixeln bei einer reduzierten Wiederholrate von 15 Bildern pro Sekunde angesteuert werden. Die Quantisierungsauflösung liegt hier bei (256)Bit.

Mikrofon Array

Die Kinect beinhaltet vier Mikrofone, welche verteilt verbaut sind. Sie dienen sowohl der Erfassung von Ton, als auch der Lokalisierung und Unterscheidung von Soundquellen. Jedes der Mikrofone tastet mit einer Quantisierungsauflösung von (65536) Bit und einer Abtastrate von 16 KHz ab.

2.0.2. Tiefenberechnung

Die Berechnung der Objektentfernungen (Tiefenwerte) innerhalb der Kinect erfolgt durch Aussendung von strukturiertem Licht durch die Projektionseinheit und durch Weiterverwertung der durch die Infrarotkamera empfangen reflektierten Strahlen auf einem Chip der Firma PrimeSense (PS1080-A2-Chip). Das Muster für

das strukturierte Licht wird mit Hilfe eines Diffusors(einer Lochplatte mit fest definiertem Muster) erzeugt.

Das Prinzip des Strukturierten Lichts ist an ein Verfahren angelehnt, welches sich Streifenprojektion nennt und in dieser Anwendung abgewandelt wurde um die Erfassung von beweglichen Objekten zu ermöglichen. Statt Licht-

streifen wird eine Punktematrix verwendet, welche zufällig und fest definiert ist. Die IR Kamera sowie der Projektor müssen sich hierzu in einem definierten, gleich großen Abstand zueinander befinden. Die Umgebungsreflektionen der projizierten Punktematrix werden von der Infrarotkamera erfasst. Um aus diesem Gitter von Infrarotpunkten Tiefeninformation zu extrahieren, wird das Verfahren der aktiven Stereotriangulation verwendet.

Die Triangulation ist ein Verfahren zur optischen Abstandsmessung, welches sich hierzu trigonometrischer Funktionen innerhalb von aufgespannten Dreiecken bedient. Es wird allgemein zwischen aktiver und passiver Triangulation unterschieden. Aktive Triangulation bedeutet, dass mindestens eine strukturierte Lichtquelle zur Abstandsberechnung erforderlich ist, während dies bei passiver Triangulation nicht der Fall ist. Da der IR Projektor ein statisches Pseudozufallsmuster emittiert, ist dieser als strukturierte Lichtquelle einzurordnen. Stereotriangulation bedeutet dass zwei unterschiedliche Bildquellen benötigt werden um die Tiefe jedes Pixels eines Bildausschnittes berechnen zu können. Eine Bildquelle ist der Diffusor (das „Lochmuster“) welcher die vom Projektor emittierten Strahlen statisch definiert. Die andere Bildquelle ist die IR Kamera. Das erste Bild ist immer identisch und statisch, das zweite hingegen variiert je nach Umgebung. Diese beiden Bilder sind die Grundlage für die trigonometrischen Operationen zur Berechnung der Tiefeninformationen.

Zur Gewinnung der Tiefeninformation wird die horizontale Differenz des Punktes Y1 des von der IR Kamera aufgenommenen Bildes zum korrespondierenden Punkt Y2 des virtuellen statischen Referenzbildes des Projektors berechnet. Aus dieser Differenz lässt sich die Tiefe des betreffenden Pixels durch aufstellen der beiden Projektionslinien und Schnittpunktbestimmung derselben berechnen.

Der Grund warum die Pixel der „Schaublone“ im Projektor zufällig angeordnet sind, liegt darin, dass die unterschiedlichen lokalen Nachbarschaftsbedingungen die Pixelzuordnung zwis-

schen dem statischen und dem dynamisch veränderten Bild erleichtern.

2.0.3. Das Schattenproblem

Aufgrund der Entfernung der verbauten RGB-Kamera zur Infrarotkamera, weisen die Bilder beider Kameras einen kleinen Versatz auf. Schatten im Tiefenbild entstehen aufgrund der Entfernung des Infrarotprojektors zur Infrarotkamera. Der Schatten im Muster macht es für den Sensor unmöglich die Tiefe festzustellen. Die Pixel in diesen Bereichen werden auf den Wert 0 gesetzt. Das Objekt blockiert die Strahlen des Lasers. Da für die Tiefenberechnung das vom IR-Projektor emittierte Muster benötigt wird, ist es für die Kinect unmöglich die Distanz in Bereichen zu berechnen, welche außerhalb der Erreichbarkeit des IR Strahlenmusters liegen.

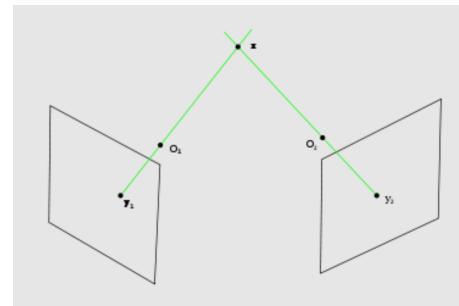


Abbildung 2.5: aktive Stereotriangulation



Abbildung 2.7.: Da der Infrarot Sensor links des Projektors positioniert ist, treten die Schatten auch linksseitig der Objekte auf.

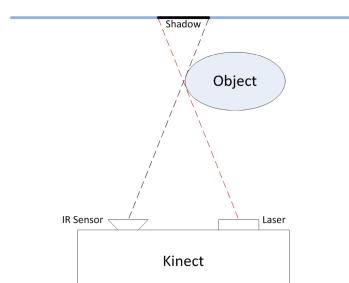


Abbildung 2.6.: Blockierte IR Strahlen

3. Die Skelettierung

Autor: Sandra Schröder

TODO: Genaue Theorie zur Skelettierung. Eigenschaften. Abhängig von Verfahren blaaaaaaaaaaa Dieses Kapitel beschreibt bekannte Konzepte und Verfahren zur Skelettierung im Bereich der Bildverarbeitung.

Zwei Verfahren sind im Rahmen der Projektarbeit besonders wichtig: Das *Thinning* (deutsch: Verdünnung) und die *Distanztransformation*.

Um die Übersicht über die weiteren grundlegenden Verfahren und Konzepte zu vervollständigen, werden diese in einem separaten Abschnitt kurz vorgestellt.

3.1. Thinning

Autor: Johannes Böhler

Das Thinning bezeichnet eine Kategorie von Methoden zur Skelettierung von 2D sowie 3D Objekten. In dieser Projektarbeit ist der Fokus ausschliesslich auf die 2D Skelettierung gerichtet, da die Kinect kein vollkommenes 3D Modell eines Objektes liefert. Sie erfasst das Objekt lediglich aus einem Blickwinkel, desshalb erhält man nur ein 2,5 dimensionales Modell. Es werden nur Tiefeninformationen bezüglich der Seite des Objektes, welche der Kinect zugewendet ist bereitgestellt. Die Tiefeninformationen der Rückseite bleiben verborgen. Um ein vollständiges 3D Modell zu erhalten müssten mindestens 2 Kinects verwendet werden und die Informationen beider Geräte zusammengeführt und vereinheitlicht werden.

Alle Thinning Algorithmen verbindet das iterative Abtragen des Musters oder der Oberfläche.

3.2. Distanztransformation

Autor: Sandra Schröder

Die Distanztransformation eines Binärbildes enthält Informationen über den Abstand der Objektpixel zum Hintergrund. Der Abstand zum Hintergrund wird für jeden Objektpixel bestimmt und in einem weiteren Bild (gleiche Dimension und Größe wie das Binärbild) als Grauwert an der Stelle des Objektpixels gespeichert. Das Ergebnis ist die sogenannte *Distance Map* (Distanzkarte).

Zur Bestimmung des Abstands benötigt man Metriken. Eine gängige Metrik - die für den Algorithmus im Rahmen dieser Arbeit auch genutzt wurde - ist die euklidische Metrik d_2
TODO Quelle (Gleichung 3.1):

$$d_2(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \quad (3.1)$$

Die Definition einer Nachbarschaft spielt ebenfalls eine Rolle. Wählt man eine 4er-Nachbarschaft, wird der Abstand vom Objektpixel zur linken und zur rechten Seite, sowie nach oben und nach unten berechnet. Man erhält dementsprechend vier Abstände. In der Distance Map wird der minimale Abstand von den vier Werten gespeichert. Bei einer 8er-Nachbarschaft kommen die diagonalen Richtungen dazu.

Die Distance Map wird zur Extraktion des Skeletts benutzt. Die Pixel die im Objekt beziehungsweise in Objektregionen mittig liegen, besitzen den größten Abstand zum Hintergrund relativ zu den unmittelbaren Pixeln. Die Pixel mit dem größten Abstand sind dementsprechend die hellsten Pixel in der Distanzmap. Betrachtet man diesen Teil der Distance Map beschreibt dieser ein potentielles Skelett.



Abbildung 3.1.: Beispiel einer Distance Map. Links: Originalbild. Dieses wurde zuerst invertiert, damit das Objekt (Person) weiß markiert ist. Rechts: Resultat der Distanztransformation. Aufällig ist das Maximum in der Mitte.

3.2.1. Extraktion des Skeletts

Es stellt sich die Frage, wie das Skelett extrahiert werden kann. Betrachtet man die Struktur einer Distance Map, wie zum Beispiel in Abbildung 3.1, erkennt man, dass die

Pixel, die in der Mitte des Objektes liegen, den größten Abstand zum Hintergrund haben. Die Distance Map bildet ein Grauwertgebirge. Diese Information wird ausgenutzt, um die Skeletlinie, das heißt die höchste Stelle im Grauwertgebirge, zu extrahieren. Die Idee ist, den Gradienten der Distance Map zu bestimmen. Der Gradient (Gleichung 3.2) ist ein Differentialoperator und liefert, angewandt auf ein Skalarfeld, die Richtung des stärksten Anstiegs, sowie die Amplitude des Anstiegs (Gradientenbetrag):

$$\text{grad}(f(x, y)) = \nabla f(x, y) = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix} \quad (3.2)$$

Ein Graubild kann als skalare Funktion $f(x, y)$ beschrieben werden mit $x, y \in \mathbb{N}$ und $0 \leq f(x, y) \leq 255$. Somit lässt sich der Gradient auch auf Graubilder übertragen. Entsprechend der Definition des Gradienten, ist an der höchsten Stelle des Grauwertgebirges der Gradientenbetrag gleich null.

Kodiert man den Gradientenbetrag ebenfalls als ein Grauwertbild, wird der Gebirgskamm eine schwarze Linie sein und das potentielle Skelett markieren. Segmentierung des Gradientenbetragbildes. Skelett schwarz, anderer Bereich ist weiß (außer Hintergrund). Dann Differenz bilden mit Distanz Map. Ergibt Skelett.

3.3. Weitere Verfahren

Autor: Christopher Kroll

4. Verwandte Arbeiten

Autor: Christopher Kroll

4.1. Extracting Skeletons From Distance Maps

Autor: Sandra Schröder

In dem Paper *Extracting Skeletons from Distance Maps* beschreibt der Autor einen Algorithmus, der effizient und schnell aus einem distanztransformierten Bild ein Skelett extrahiert [Cha07]. Der Autor legt besonders hohen Wert darauf, dass die Extraktion keine komplizierten Berechnungen beinhaltet. Er möchte vor allem auf die Berechnung von Ableitungen höherer Ordnung und die Auswertung von komplexen Gleichungen verzichten. Das Verfahren, welches der Autor zur Extraktion der Skelettlinien eines Objekt vorstellt, ist die sogenannte *Ridge Point Detection* (deutsch: *Gebirgskammdetektion*). Dabei nutzt der Autor eine grundlegende Eigenschaft der Distance Map. Wie in Abschnitt 3.2 beschrieben, ist die Distance Map ein Grauwertgebirge, wobei der Gebirgskamm zentriert im Objekt liegt. Betrachtet man nur diesen Teil der Distance Map und projiziert ihn auf das Originalbild, lässt sich eine skelett-artige Beschreibung des Objekts erkennen.

Die Gebirgskammdetektion ist ein gradientenbasiertes Verfahren. Liegt ein Punkt auf den Gebirgskamm, so hat er in seiner unmittelbaren Umgebung den größten Abstand zum Objektrand und den größten Grauwert relativ zu seinen Nachbarn. Dieser Punkt ist somit ein lokales Maximum. Aufgrund dieser Tatsache eignet sich der Gradient am besten, um solch einen Punkt zu detektieren. Der Gradient zeigt nach Definition in die Richtung des stärksten Anstiegs. Dies bedeutet, dass der Gradient eines Punktes, der nicht auf dem Kamm liegt, in die Richtung des Kammes zeigt. Wählt man nun Punkte näher am Kamm, so wird der Anstieg geringer, da die Differenz zwischen dem Grauwert des betrachteten Gebirgspunktes und dem aktuell gewählten Punkt kleiner wird. Überquert man den Kamm, kehrt sich die Richtung des Gradienten um und zeigt wieder zum Kamm. Hier findet ein Vorzeichenwechsel des Gradientenbetrags statt. Der Punkt auf dem Gebirgskamm bildet eine *sign barrier* zwischen den Gradientenbeträgen der sich gegenüberliegenden Punkte, wobei sich der Gebirgspunkt zwischen diesen beiden Punkten befindet. Diese Beobachtung ist in Abbildung 4.1 dargestellt. Es ist ein Querschnitt eines Grauwertgebirges abgebildet. Legt man nun eine Projektionslinie genau durch das lokale Maximum (roter Punkt), und projiziert die Gradienten (violette Pfeile) der

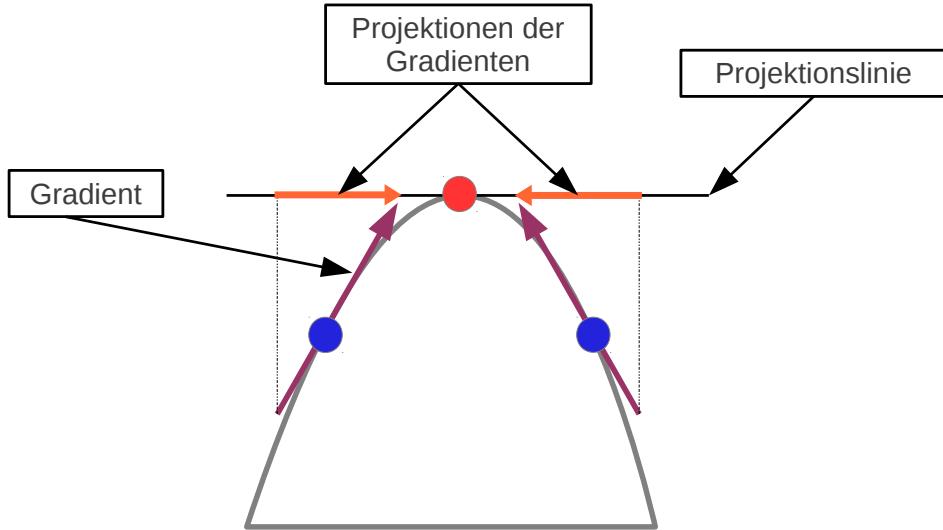


Abbildung 4.1.: Ridge Point Detection. Der rote Punkt ist ein lokales Maximum auf dem Gebirgskamm. Die blauen Punkte liegen sich gegenüber und umschließen den roten Punkt. Ihre auf die Projektionlinie projizierten Gradientenrichtungen zeigen in entgegengesetzte Richtungen. Der rote Punkt bildet somit eine *sign barrier* für die beiden Richtungen.

sich gegenüberliegenden Punkte (blau) auf diese Linie, so kann man erkennen, dass die Projektionen der Gradienten (orangene Pfeile) genau in die entgegengesetzte Richtung zeigen. Die Idee des Algorithmus ist, Projektionslinien durch die Distance Map zu legen und das Verhalten der Gradientenbeträge auf diesen Linien zu beobachten. Der Autor hat festgestellt, dass sich dabei mehrere Muster von Vorzeichenwechsel der Gradientenbeträge erkennen lassen. Diese Muster können dabei ein Indiz für einen Gebirgspunkt und somit für einen Punkt des Skeletts sein.

Dabei stellt sich die Frage, wieviele Richtungen mit diesen Linien untersucht werden sollen. Man kann beobachten, dass es einen Vorzeichenwechsel in den Gradientenbeträgen gibt, wenn die Projektionslinie den Gebirgskamm schneidet. Dementsprechend gibt es keinen Vorzeichenwechsel, wenn die Projektionslinie parallel zum Kamm verläuft. Findet man also in einer Richtung keinen Gebirgskamm, so muss einer in der orthogonalen Richtung liegen. Deshalb genügt es, zwei zueinander senkrechte Richtungen zu untersuchen. Dabei wählt man die eine Richtung parallel zur x-Achse und die andere Richtung parallel zur y-Achse des untersuchten Bildes. Es existieren insgesamt vier Muster, die auf einen Gebirgskamm deuten. Zwei davon sind in Abbildung 4.2 zu sehen. Die Muster beschreiben, in welcher Weise Vorzeichenwechsel zwischen zwei benachbarten Pixeln auftreten können. Die Symbole + und – die Richtungen der Gradienten. + ist eine positive Richtung

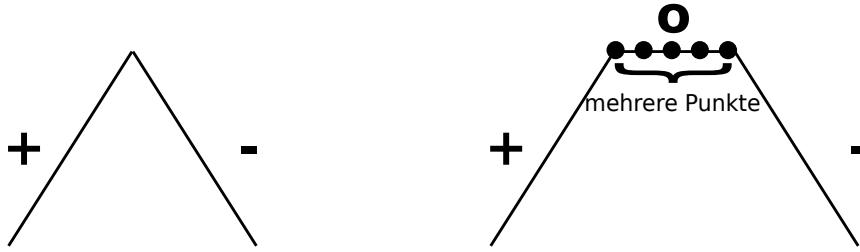


Abbildung 4.2.: Muster für Hinweise auf einen Gebirgskamm. Diese beiden Muster sind ein starkes (*strong*, linke Abbildung) und ein gutes (*good*, rechte Abbildung) Indiz für einen Punkt auf dem Gebirgskamm. **TODO die Abbildung ist doof**

(bergauf), – eine negative Richtung (bergab) auf einer Projektionslinie. Das Symbol \circ besagt, dass sich in diesem Bereich der Gradient nicht ändert, da der Grauwert im nächsten Nachbarpunkt gleich ist. Die linke Abbildung entspricht genau der Beobachtung, wie sie anhand Abbildung 4.1 beschrieben wurde. Schneidet die Projektionslinie genau einen Punkt auf dem Gebirgskamm, erzeugt dieser Punkt einen Vorzeichwechsel zwischen den beiden Punkten, die den Punkt auf dem Kamm umschließen. Die rechte Abbildung zeigt, dass es mehrere Punkte hintereinander in der Distance Map mit dem gleichen Grauwert geben kann, aber auch ein Hinweis für einen Gebirgskamm sind. Dies entspricht im Grauwertgebirge einem Plateau.

Der Algorithmus sucht nun in x -und in y Richtung - von oben nach unten und von links nach rechts - in der Distance Map nach diesen Mustern und markiert die Punkte nach den Eigenschaften *strong*, *good*, *weak* und *none*. Diese Markierung gibt die Stärke der Sicherheit des Punktes wieder, ein Punkt auf dem Gebirgskamm zu sein.

Wurden alle Punkte in beide Richtungen untersucht, hat jeder Punkt ein passendes Label. Diese Labels werden weiter benutzt, um eine Graphenrepräsentation des Skeletts zu erstellen.

Der Algorithmus findet zu jedem Punkt das richtige Label und alle Punkte, die zu einem Gebirgskamm gehören [Cha07]. Abbildung 4.3 zeigt ein Ergebnis des Algorithmus. Die gestrichelte Linie in der rechten Abbildung beschreibt das theoretische Skelett, die grau unterlegten Punkte sind die vom Algorithmus gewählten Punkte, die zu einem Gebirgskamm gehören und einem Skeletpunkt entsprechen.

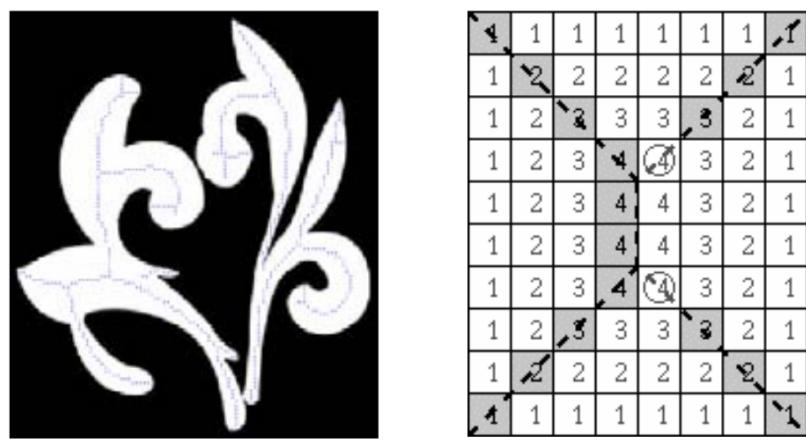
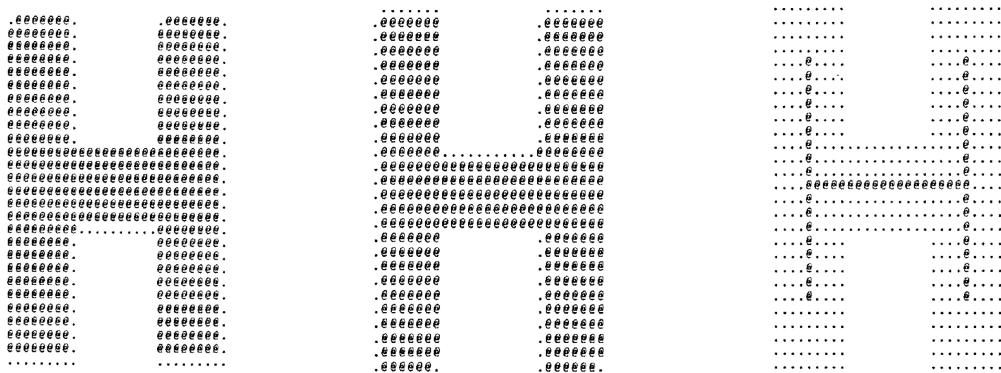


Abbildung 4.3.: Ergebnis der Ridge Point Detection [Cha07]. Links: Graphische Darstellung. Rechts: Darstellung als Bildmatrix.

4.2. A Fast Parallel Algorithm for Thinning Digital Patterns

Autor: Johannes Böhler

Der gesamte Algorithmus erstreckt sich über mehrere Iterationen. Die Randpixel des Musters werden Schicht für Schicht abgetragen. Die Iterationen selbst, sind wiederum in zwei Subiterationen unterteilt. Das Abtragen der „Schichten“ wird somit in zwei unterschiedliche Phasen aufgespalten. Mit Hilfe der ersten Subiteration werden sowohl Süd- und Ostgrenzpunkte als auch Nordwest Eckpunkte entfernt. Das entfernen von Nord- und Westgrenzpunkten sowie von Südosteckpunkten erfolgt in der zweiten Subiteration.



(a) Nach erster Subiteration (b) Nach zweiter Subiterati- (c) Skelett des Ursprungs-
(zu Beginn) on (zu Beginn) muster

Abbildung 4.4.: Zustände des Algorithmus

4.2.1. Anforderungen an den Algorithmus

- Das Rauschen, welches der Algorithmus verursacht soll so gering wie möglich gehalten werden.
- Das Skelett des Ursprungsmusters soll die Endpunkt- und Pixelverbundenheit erhalten. Endpunktverbundenheit bedeutet, dass sich zwischen zwei Endpunkten eines Skeletts keine unverbundenen Stellen befinden.
- Das Skelett soll nach Durchlaufen des kompletten Algorithmus in einheitlicher Dicke von einem Pixel vorliegen.
- Der Algorithmus soll möglichst schnell und effizient arbeiten um Echtzeitfähigkeit gewährleisten zu können.

| | | |
|---------------------------|-----------------------|---------------------------|
| P_9 $(i - 1, j - 1)$ | P_2 $(i - 1, j)$ | P_3 $(i - 1, j + 1)$ |
| P_8 $(i, j - 1)$ | P_1 (i, j) | P_4 $(i, j + 1)$ |
| P_7 $(i + 1, j - 1)$ | P_6 $(i + 1, j)$ | P_5 $(i + 1, j + 1)$ |

Abbildung 4.5.: Betrachteter Pixel P1 und Nachbarumgebung

4.2.2. Ablauf des Algorithmus

Es wird davon ausgegangen, dass zu Beginn ein binär digitalisiertes Bild vorliegt. Die Pixel werden mit Hilfe einer zweidimensionalen Matrix IT durchlaufen, deren Wert an der jeweiligen Stelle $IT(i,j)$ entweder 0 oder 1 ist. Mit Muster ist die Menge an Pixeln gemeint, welche den Wert eins haben. Es werden in Abhängigkeit von den 8 Nachbarpixeln (siehe Abbildung 4.5), Transformationen auf den betrachteten Pixel P1 angewendet. Dieser Vorgang wird iterativ auf die Matrix IT angewendet.

Der neue Wert eines Pixels während der n-ten Iteration hängt von dem eigenen Wert während der (n-1)ten Iteration und den Werten der acht Nachbarn während der (n-1)ten Iteration ab. Dies ermöglicht paralleles Transformieren mehrerer Bildpunkte. Die Bedingungen, welche zum Ausführen der Transformation erfüllt sein müssen werden über ein 3x3 Pixel Fenster abgefragt. Der Punkt P1 über dessen Transformation entschieden wird, ist mit allen acht Nachbarn direkt verbunden.

Der Algorithmus entfernt alle Randpunkte des Musters, außer den Pixeln welche Bestandteil des Skeletts sind. Um die Verbundenheit des Skeletts zu gewährleisten wird ein Iterationsschritt in zwei Subiterationen aufgeteilt.

In der ersten Subiteration wird der Punkt P1 aus dem Muster gelöscht, wenn er folgende Bedingungen erfüllt:

- a) $2 \leq B(P1) \leq 6$ B entspricht der Anzahl der Nachbarn von P1 $\neq 0$. Die Anzahl der Nachbarn von P1 welche den Wert 1 haben, muss somit zwischen 2 und 6 liegen.
- b) $A(P1)=1$ AAnzahl der „01“-Folgen Die Anzahl der 01 Folgen in der geordneten Folge $P_2, P_3 \dots P_9$ muss genau eins betragen.
- c) $P_2 * P_4 * P_6 = 0$ Mindestens ein Pixel der Pixelmenge P_2, P_4, P_6 muss den Wert Null haben.
- d) $P_4 * P_6 * P_8 = 0$ Mindestens ein Pixel der Pixelmenge P_4, P_6, P_8 muss den Wert Null haben.

Sind alle Bedingungen a, b ,c und d erfüllt so wird der Wert des Pixels auf 0 gesetzt. Dies bedeutet dass er kein Teil des Skelett-Musters mehr ist. Wird eine der Bedingungen

nicht erfüllt, so bleibt der Pixelwert bei 1.

In der zweiten Substitution wird P1 gelöscht falls folgende Bedingungen gelten:

| | | |
|---|-------|---|
| 0 | 0 | 1 |
| 1 | p_1 | 0 |
| 1 | 0 | 0 |

Abbildung 4.6.: Anzahl 01 folgen in zyklischer Reihenfolge

- a) $2 \leq B(P1) \leq 6$
 - b) $A(P1) = 1$
 - c) $P2^*P4^*P8 = 0$
 - d) $P2^*P6^*P8 = 0$

Nur die Bedingungen c und d haben sich geändert.

Um die Bedingungen der ersten Subiteration zu erfüllen, muss $P4=0$ oder $P6=0$ oder $(P2=0 \text{ und } P8=0)$ erfüllt sein. Dies impliziert dass $P1$ entweder Süd- oder Ost-Grenzpunkt, oder Nordwesteckpunkt ist. Um die Bedingungen der zweiten Subiteration zu erfüllen muss $P2=0$ oder $P8=0$ oder $(P4=0 \text{ und } P6=0)$ sein. $P1$ ist Nord- oder West-Grenzpunkt oder Südosteckpunkt. Während mit Bedingung A ($2 \leq B(P1) \leq 6$) die Endpunkte des

| North | | |
|-------|-------|-------|
| West | P_2 | |
| | P_8 | P_1 |
| | P_6 | P_4 |
| South | | |
| East | | |

Abbildung 4.7.: Betrachteten Nachbarpunkte in den Bedingungen c und d

Skeletts erhalten werden, so wird mit Bedingung B ($A(P1)=1$) die Auslöschung von Punkten zwischen den Endpunkten der Skelettlinie verhindert.

In der Matrix Search M befinden sich während der ersten Iteration alle Pixel die gelöscht werden dürfen, da sie den Bedingungen der ersten Iteration genügen. Ist dies nicht der Fall, so ist der Counter=0 und der Algorithmus beendet, da es keine zu löschen Pixel mehr gibt. Die Skelettierung ist somit beendet.

Falls der Counter ungleich null ist, werden die Pixel welche den Bedingungen genügen von der Matrix IT (Skelett-Muster) abgezogen, der Counter wird null gesetzt und

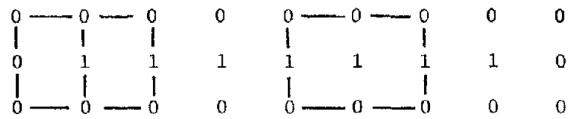


Abbildung 4.8.: Gewährleistung der Pixelverbundenheit

es wird zur zweiten Iteration fortgeschritten. Dort findet der Ablauf mit veränderten Bedingungen c und d wiederholt statt. Ist der Counter auch nach dem Durchlaufen der zweiten Subiteration ungleich null so wird der Vorgang iterativ fortgeführt.

4.2.3. Resultate

Der Algorithmus erzielt sehr gute Ergebnisse im Bezug auf Verbundenheit, und Rauschsicherheit bei Randpunkten. Die Bedingungen welche zum Auffinden der zu löschen Randpunkte führen sind sehr simpel. Durch den Bezug auf die n-1te Iteration zur Abfrage der Bedingungen kann der Algorithmus sehr schnell ausgeführt werden, da keine Warteabhängigkeiten bestehen.

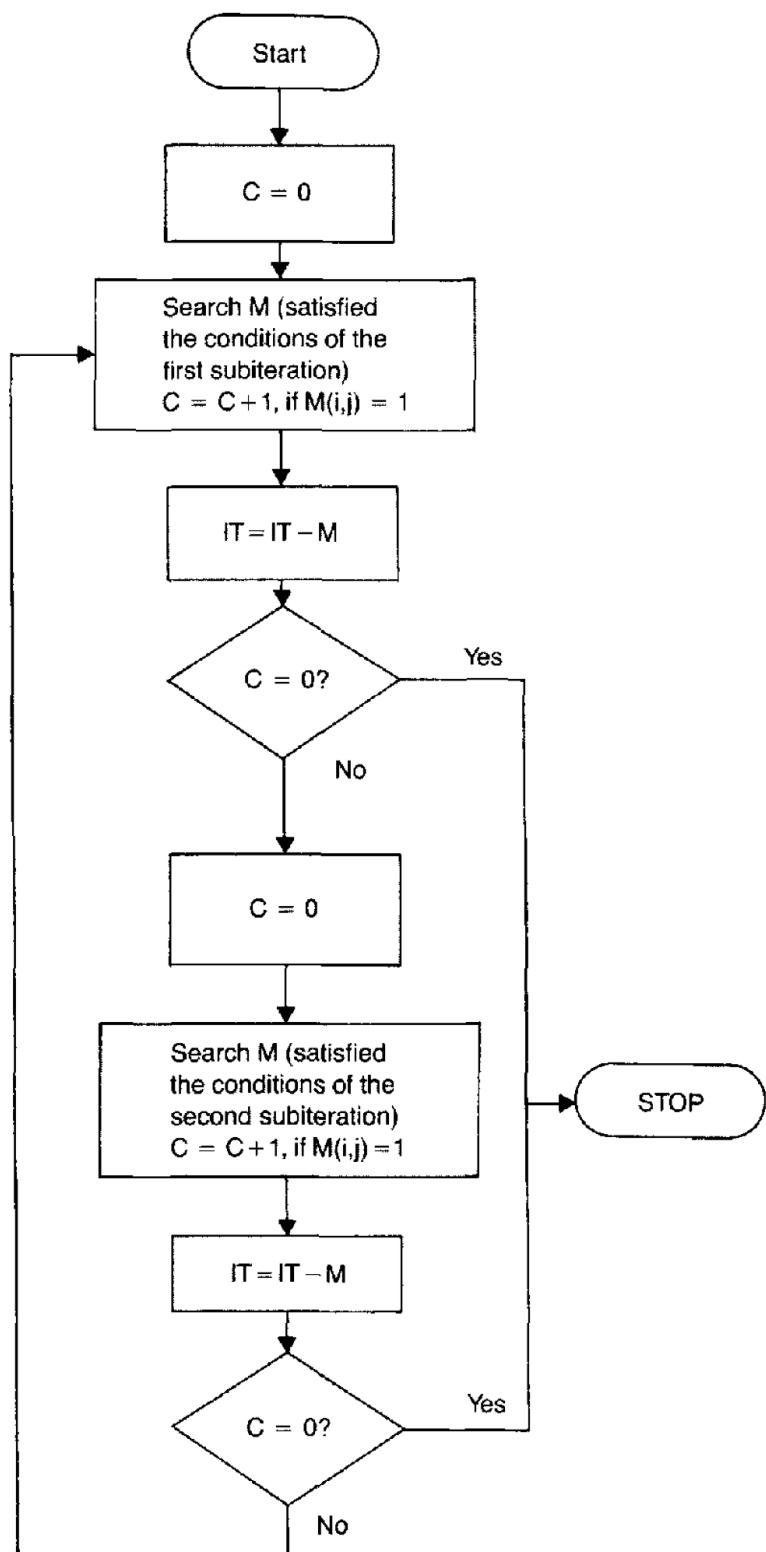


Abbildung 4.9.: Gesamter Algorithmus in der Übersicht.

5. Implementierung der Algorithmen

Autor: Sandra Schröder

Dieses Kapitel beschreibt die Algorithmen für Thinning und die Distanztransformation.

Der Algorithmus erhält einen Videostream von Tiefenbilder der Kinect. Diese werden für die Skelettierung weiterverarbeitet.

Bevor Skelettierungsalgorithmen angewendet werden können, muss ein Binärbild des Bildes erzeugt werden, das die Kinect liefert. Dafür wurde eine Segmentierung implementiert, die Objekte, die sich vor der Kinect befinden, weiß markiert. Für die Segmentierung wurde nur die Tiefeninformation der Kinect genutzt. Anschließend werden die Algorithmen angewandt.

```
1: procedure DO_SKELETONIZATION(Image)
2:   while True do
3:     Segmentiere Spieler
4:     Wende Thinning oder Distanztransformation an
5:   end while
6: end procedure
```

5.1. Technische Umsetzung

Autor: Christopher Kroll Die meisten Programme wurden mit der Programmiersprache *Python* umgesetzt. Aus Performanzgründen erfolgten einige Umsetzungen in *C++*. -> oder in arbeitsumgebung rein?

5.2. Spieler-Segmentierung

Autor: Sandra Schröder

Algorithmus oder nur beschreiben??

Es wird nur anhand der Tiefe segmentiert. Um nicht für jeden einzelnen Pixel die Bedingung zu überprüfen, ob er den Schwellwert für die Segmentierung überschreitet beziehungsweise unterschreitet, wird die effiziente Numpy-Funktion `logical_and` benutzt,

die global auf dem Bild arbeitet und für das gesamte Bild die Schwellwertbedingung prüft. Für den Schwellwert werden zwei Werte definiert, um ein Intervall festzulegen, in dem sich das Objekt befinden darf.

5.3. Thinning

5.4. Skelettierung aus der Distanztransformation

Autor: Sandra Schröder

Die Skelettierung anhand der Distanztransformation läuft folgendermaßen ab:

- Bestimmen der Distanztransformation des Binärbildes
- Berechne den Gradientenbetrag auf der Distance Map
- Differenz zwischen dem Gradientenbild und der Distance Map bilden

6. Ergebnisse

Autor: Sandra Schröder

Verfahren zur Skelettierung von Bildobjekten fordern bestimmte Eigenschaften an das Skelett **todo Quelle**. Dies ist zum einen die *Konnektivität* des Skeletts. Dies bedeutet, dass es keine Lücken und Unterbrechungen im Skelett gibt. Ein zusammenhängendes Objekt sollte nämlich ein zusammenhängendes Skelett besitzen. Viele Skelettierungsverfahren fordern, dass ein Skelett genau 1 Pixel breit ist und *zentriert* im ursprünglichen Objekt liegt. Zentriert bedeutet, dass der Abstand des Skeletts zum ursprünglichen Rand zu allen Seiten gleich groß ist.

Das das Skelett als Deskriptor für die Form eines Objektes dient, sollte es die geometrischen Eigenschaften und die Topologie des Objektes gut wiedergeben können.

Bilder sind immer mit Rauschen behaftet. Eine Skelettierung sollte, unabhängig davon, wie stark das Rauschen ausgeprägt ist, robust gegenüber solchen Störungen sein.

In den folgenden Abschnitten werden unsere Ergebnisse für die beiden Skelettierungsalgorithmen, die im Rahmen dieser Projektarbeit entwickelt wurden, vorgestellt. Zur Bewertung der Ergebnisse greifen wir auf die gewünschten Eigenschaften eines Skeletts zurück und überprüfen, ob die Algorithmen sie erfüllen.

6.1. Vergleich der Algorithmen

6.1.1. Thinning

6.1.2. Distanztransformation

Screenshots und Laufzeitmessungen

6.2. Verbesserung der Skelettqualität

Das Skelett, welches mit der Methode der Distanztransformation bestimmt wurde, weist Lücken zwischen den Skelettteilen auf. Um die Topologie und geometrische Eigenschaften des Objekts gut wiederzugeben, ist ein lückenloses Skelett wünschenswert.

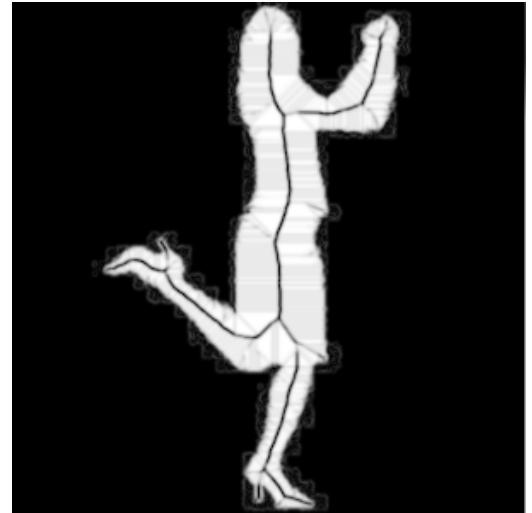
Die Ursache der Lücken sind der Gradientenbetrag, der für die Extraktion der Skelettlinie berechnet wurde und die Segmentierung des Gradientenbildes. Abbildung 6.1 zeigt ein Beispiel dazu. Abbildung (a) ist das Originalbild mit dem Objekt, von dem das Skelett, wie in Abbildung (d) zu sehen, bestimmt wurde. Da der Algorithmus auf der Grundlage von Bildern arbeitet, bei denen das Objekt weiß und der Hintergrund schwarz markiert sind, wurde das Bild zuvor invertiert. Abbildung (b) zeigt das Gradientenbetragsbild der Distance Map des Originalbildes. Die schwarze Linie markiert die potentielle Skelettlinie des Objektes. Der Gradientenbetrag ist hier gleich null, da dies die höchste Stelle (lokales Maximum) im Grauwertgebirge ist. Das Gradientenbetragsbild wurde segmentiert, um ein Binärbild zu erhalten. Jedoch treten bei der Segmentierung bereits Lücken auf, die bei dem Resultat der Differenzbildung zwischen der Distance Map und dem segmentierten Gradientenbild auch erhalten bleiben (Abbildung (d)).

In den folgenden Abschnitten werden Methoden zur Verbesserung des Skeletts vorgestellt. Dies umfasst zum einen die Herstellung Konnektivität der Skelettkomponenten beziehungsweise dem Füllen der Lücken zwischen den Skelettlinien. Zum anderen ist es wünschenswert, dass die Skelettlinie des Objektes nicht breiter als ein Pixel ist. Zuerst wurden markante Punkte auf dem Skelett bestimmt. Diese wurden genutzt, um mittels Breitensuche und Tiefensuche Pfade zwischen ihnen zu finden und sie zu verbinden. Punkte werden miteinander verbunden, wenn sie nah genug beieinander sind. Das Ergebnis ist ein Skelett zusammenhängenden Skelettlinien. Die genaue Umsetzung der Algorithmen wird in den Abschnitten 6.2.2 und 6.2.3 beschrieben.

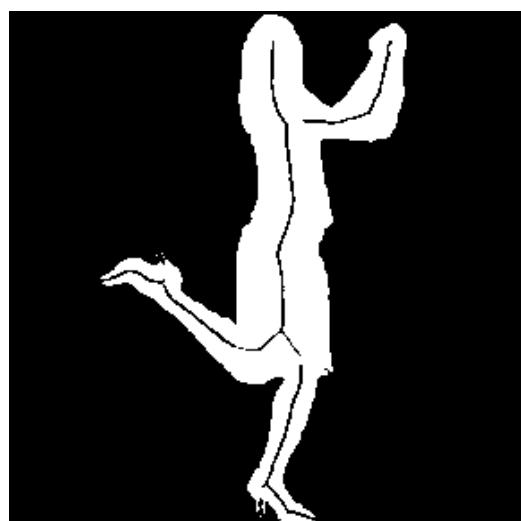
Breiten -und Tiefensuche liefern aufgrund ihrer algorithmischen Eigenschaften jeweils unterschiedliche Ergebnisse. Die Unterschiede werden gezeigt und es wird diskutiert, inwieweit sich die Verfahren mit der Kinect anwenden lassen.



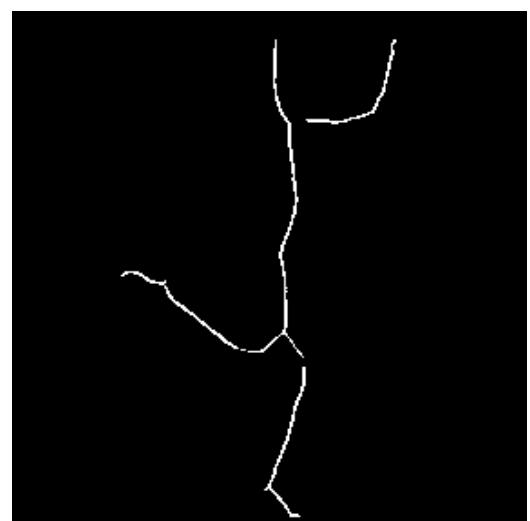
(a)



(b)



(c)



(d)

Abbildung 6.1.: Bestimmung des Skelett mittels Distanztransformation. (a) Originalbild
(b) Gradientenbild (c) Gradientenbild segmentiert(d) Skelett

6.2.1. Berechnung von Features auf dem Skelett

Die Bildverarbeitungsbibliothek *OpenCV* bietet Funktionen zur Bestimmung von markanten Punkten, sogenannten *Features*, in einem Bild. Die Funktion `goodFeaturesToTrack` bestimmt die am stärksten auftretenden Ecken in einem Bild oder Bildausschnitt nach einem Algorithmus von [ST94]. Mittels eines Qualitätsmaßes wird entschieden, ob die Stärke einer Ecke an einem bestimmten Pixel ausreicht, um in die Featuremenge aufgenommen zu werden. Für die Funktion können der Wert des Qualitätsmaßes, den eine Ecke erfüllen muss, die Anzahl der Ecken, die gefunden werden sollen und die minimale Distanz zwischen den stärksten Ecken übergeben werden.

Die Qualität einer Ecke wird anhand von Eigenwerten bestimmt. Die Eigenwerte beziehen sich dabei auf die Kovarianzmatrix von Ableitungen einer festgelegten Umgebung eines Pixels. Es wird minimale Eigenwert für die Eckendetektion weiterverwendet. Ist der minimale Eigenwert einer Ecke kleiner als das gewünschte Qualitätsmaß, wird diese Ecke verworfen. Die verbleibenden Ecken werden nach ihrer Qualität absteigend sortiert. Anschließend wird überprüft, ob es in der spezifizierten Distanz Ecken gibt, die stärker sind. Abbildung 6.2 zeigt das Ergebnis der Berechnung. Die Kreise markieren die Feature-

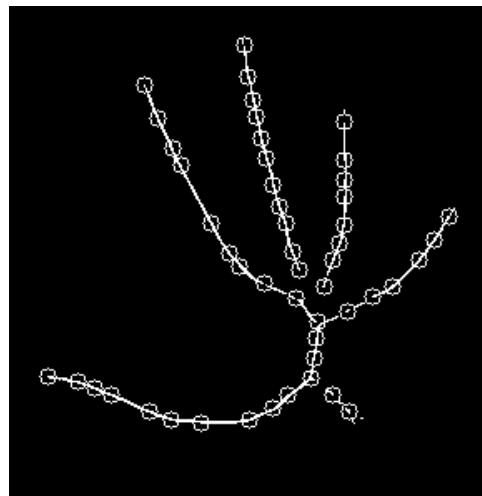


Abbildung 6.2.: Ergebnis der Funktion `goodFeaturesToTrack`. Qualitätslevel: 0.1 - gewünschte Anzahl von Ecken: 50 - minimale Distanz zwischen den Ecken: 10

Punkte. Wie man erkennen kann, befinden sich die Features auf der Skeletlinie. Dies ist hilfreich für die weiteren Verbesserungen des Skeletts. Befinden sich Features außerhalb der Skeletlinien könnten die ursprüngliche Form des Skeletts und die Topologie des Objekts verfälscht werden.

6.2.2. Breitensuche

Der Breitensuche-Algorithmus durchsucht einen Graphen ausgehend von einem Startpunkt nach weiteren Knoten. Der Algorithmus sucht zunächst nur nach direkt nach-

folgenden Knoten und somit in die Breite des Graphen. Knoten, die bereits besucht wurden, werden markiert. Wurde ein Knoten noch nicht besucht, wird er in eine *Queue* (Warteschlange) aufgenommen.

Mittels Breitensuche sollen Pfade zwischen den markanten Punkten auf dem Skelett gefunden werden. Die markanten Punkte entsprechen Knoten in einem Graphen. Für die Nachbarschaftsbeziehung zwischen zwei Knoten wird ein Nachbarschaftsmaß definiert. Abbildung 6.3 zeigt, wie überprüft wird, ob ein Punkt in einem festgelegten Intervall des Punktes (x, y) liegt. Es wird eine Suchdistanz für beide Richtungen festgelegt. Sie wird entsprechend der minimalen Distanz, die zwischen markanten Punkten erlaubt ist, gewählt (Abschnitt 6.2.1).

Erst wird in x-Richtung gesucht, dann in y-Richtung. Fällt der Punkt in das Intervall, wird er als besucht markiert und mit dem Punkt (x, y) verbunden. In Anhang A befindet sich der Quellcode zur Breitensuche.

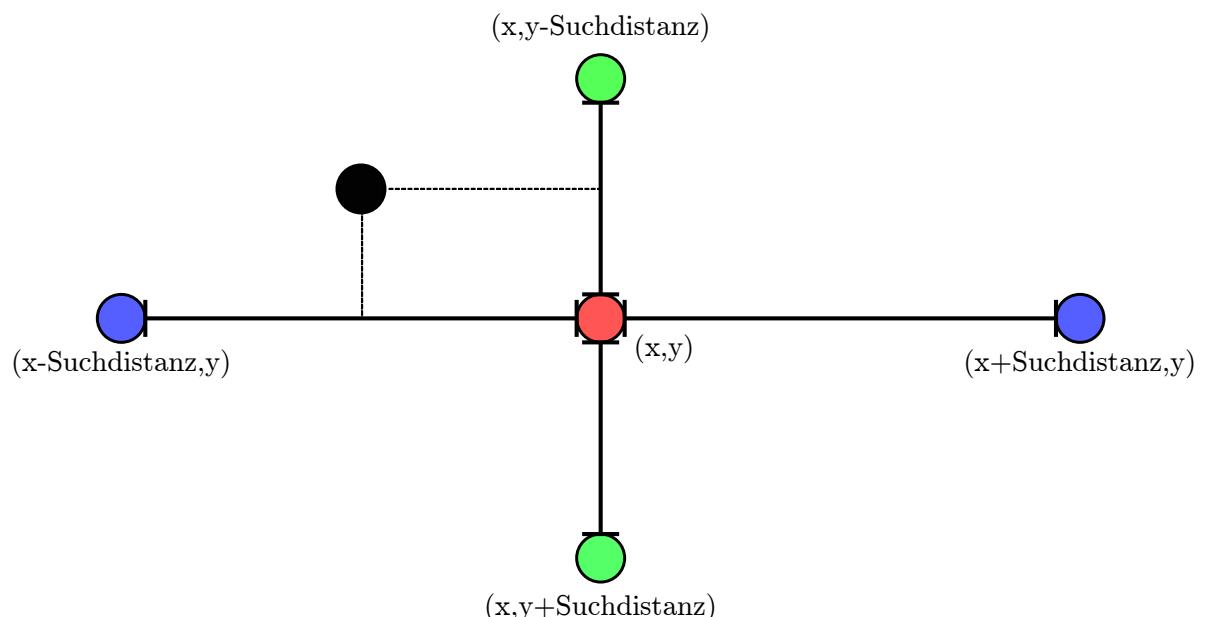


Abbildung 6.3.: Das aufgespannte Suchkreuz vom Punkt (x, y) (rot) aus. Für den schwarzen Punkt wird überprüft, ob er in den festgelegten Suchintervallen des Punktes (x, y) liegt.

Das Ergebnis des Algorithmus ist in Abbildung 6.4 im Vergleich zum vorigen Skelett zu sehen. Das Skelett besitzt nun zusammenhängende Komponenten. Auffällig jedoch sind die Ausläufer, die auf die Vorgehensweise des Breitensuchealgoritmus zurückzuführen sind. Wird die Suchdistanz zu groß gewählt, findet der Algorithmus mehrere Nachfolger, die er dann mit dem aktuellen Punkt verbindet. Das Ergebnis sind fächerartige Ausläufer, da der Algorithmus in die Breite sucht.



Abbildung 6.4.: Ergebnis der Breitensuche.

6.2.3. Tiefensuche

Wie bei der Breitensuche wird ausgehend von einem Startknoten ein Graph nach weiteren Knoten durchsucht. Im Gegensatz zur Breitensuche erfolgt das Traversieren des Graphens in die Tiefe. Wurde ein Nachfolgeknoten gefunden, wird für diesen weiter geprüft, ob er ebenfalls einen Nachfolgeknoten besitzt. Dies wird solange wiederholt (rekursiv) bis kein Nachfolgeknoten mehr gefunden werden kann. Mittels *Backtracking* wird der Pfad zurückverfolgt. Anschließend wird ein neuer Knoten als Startpunkte gesucht, der noch nicht besucht wurde und das Durchsuchen nach nächsten Nachbarn wird für diesen Knoten wiederholt.

Die Suche nach dem nächsten Nachbarn funktioniert wie bei der Breitensuche (Abbildung 6.3). Der Quellcode zum Algorithmus befindet sich im Anhang A.

Das Ergebnis ist in Abbildung 6.5 zu sehen. Es fällt auf, dass es noch Lücken im Skelett gibt, was auf die Suchdistanz zurückzuführen ist. Ist sie zu klein, können keine weiteren Punkte im Umkreis des aktuellen Punktes gefunden werden und der Algorithmus endet für diesen Pfad.



Abbildung 6.5.: Ergebnis der Tiefensuche.

Das Skelett, welches aus der Tiefensuche entsteht, bedarf demnach einer Nachbearbeitung. Die Idee ist, Punkte zu finden, die keinen Nachfolger besitzen und für diese Punkte

den nächsten Nachbarn zu bestimmen. Dies wird mittels eines Ansatzes realisiert, der zwischen allen Punkten ohne Nachfolger den euklidischen Abstand bestimmt und den Punkt als nächsten Nachbar wählt, der den minimalen Abstand zu dem fest gewählten Punkt hat. In Abbildung 6.6 wurden die Punkte ohne Nachfolger eingezeichnet. Diese befinden sich genau am Ende eines Pfades. Sie entsprechen den Blättern des Baumes, der bei der Tiefensuche für einen Startknoten entsteht. Diese Punkte wurden nun miteinander

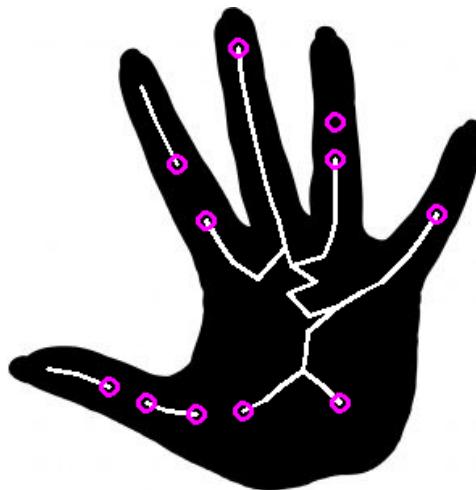


Abbildung 6.6.: Punkte ohne Nachfolger nachdem das Skelett nachgezeichnet wurde (markiert mit Kreisen).

verbunden. Wie in Abbildung 6.7 zu sehen ist, wurden die Punkte ohne Nachfolger, die am nächsten beieinander sind, miteinander verbunden (rote Linien). Obwohl zwei Punkte falsche Verbindungen erzeugen, liefert der Algorithmus ein gutes Ergebnis für das Skelett. Durch Einführung weiterer Bedingungen - beispielweise, ob ein Punkt bereits verbunden ist - könnten auch diese Ausläufer eliminiert werden.

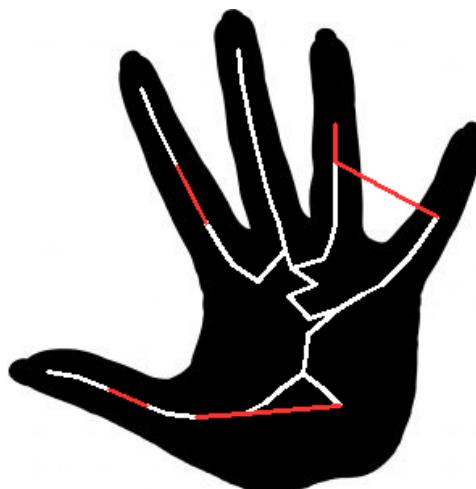


Abbildung 6.7.: Ergebnis der Tiefensuche

6.2.4. Diskussion

Beide Algorithmen liefern zufriedenstellende Ergebnisse bei der Verbesserung der Skelettqualität. In beiden Fällen besteht Pixelkonnektivität und somit ein Zusammenhang zwischen den Skelettkomponenten. Die Ergebnisse der beiden Algorithmen sind jeweils unterschiedlich aufgrund ihrer Vorgehensweise bei der Traversierung der Punkte. Die Skelette der beiden Algorithmen die geometrischen Eigenschaften des ursprünglichen Objektes gut wieder. Bei der Breitensuche sowie der Tiefensuche werden die Finger der Hand nachgezeichnet. Nur im Bereich des Handballen entstehen - besonders bei der Breitensuche - Ausläufer im Skelett. Abhängig von der Anwendung muss man entscheiden, ob diese Ausläufer stören könnten.

Ein weiterer Faktor, der die Form des modifizierten Skeletts besonders beeinflusst, ist die Suchdistanz zum Finden der nächsten Nachbarn. Wird sie zu groß gewählt, werden viele Nachbarn gefunden und somit auch mehrere Wege von einem Punkt aus (Beispiel Breitensuche). Ist sie zu klein, entstehen Lücken im Skelett und die Konnektivität zwischen den Skelettlinien ist nicht mehr gegeben. Man muss für jedes Bild beziehungsweise für jedes Bildobjekt entscheiden, welche Suchdistanz sich am besten eignet. Dies kann bei der Anwendung der beiden Algorithmen auf den Videostream der Kinect hinderlich sein, da die optimale Suchdistanz von Bild zu Bild unterschiedlich sein kann.

7. Zusammenfassung

8. Fazit und Ausblick

Literaturverzeichnis

- [Cha07] Sukmoon Chang. Extracting Skeletons from Distance Maps. *International Journal of Computer Science and Network Security*, 7, 2007.
- [ST94] Jianbo Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593 –600, jun 1994.

A. Quellcode

A.1. Verbesserung der Skelettqualität

Listing A.1: Breitensuche

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Jan 16 12:27:19 2013
4
5 @author: 8schroed
6 """
7 import cv
8 import numpy
9
10 filename = "hand.jpg"
11 image = cv.LoadImage(filename, cv.CV_LOAD_IMAGE_GRAYSCALE)
12
13 def calcGoodFeatures(image):
14     #grayImage = cv.LoadImage(image,2)
15     eigenvalueImage =
16         → cv.CreateImage(cv.GetSize(image),cv.IPL_DEPTH_32F,1)
17     tempImage =
18         → cv.CreateImage(cv.GetSize(image),cv.IPL_DEPTH_32F,1)
19
20     corners = []
21     cornerCount = 30
22     qualityLevel = 0.1
23     minDistance = 5
24
25     corners =
26         → cv.GoodFeaturesToTrack(image,eigenvalueImage,tempImage,
27                               cornerCount,qualityLevel,minDistance)
28
29     return corners
30
31 def drawFeatures(features,image):
```

```

31     for point in features:
32         center = int(point[0]), int(point[1])
33         cv.Circle(image,(center),5,(255,0,0))
34
35 def startConnect(features,minDistance,epsilon,img):
36     print features
37     searchDistance = minDistance + epsilon
38     startpoint = features.pop()
39
40     neighbours =
41         → connectFeatures(startpoint,features,searchDistance,img)
42     return neighbours
43
43 def connectFeatures(startpoint,features,searchDistance,img):
44
45     neighbours = []
46     #Besuchte Knoten
47     visited = []
48     neighbours.append(startpoint)
49     while len(neighbours) != 0:
50         current = neighbours.pop()
51         visited.append(current)
52         x = current[0]
53         y = current[1]
54         for f in features:
55             #print startpoint
56             if f not in visited:
57                 x1 = f[0]
58                 y1 = f[1]
59                 #Befuellen der temporaeren Liste. Haelt die
59                 → Punkte, die am naechsten dran vom
59                 → aktuellem Punkt sind
60                 if x1>=x-searchDistance and x1<=x or
60                 → x1<=x+searchDistance and x1>=x:
61                     if y1>=y-searchDistance and y1<=y or
61                     → y1<=y+searchDistance and y1>=y:
62                         neighbours.append((x1,y1))
63                         cv.Line(img,
63                             → (int(current[0]),int(current[1])),
63                             → (int(f[0]),int(f[1])),
63                             → (255,255,255), thickness=2,
63                             → lineType=8, shift=0)
64                         visited.append(f)
65

```

```
66 |
67 |
68     return visited
```