UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

Tiefen-basierte Verdeckung in Augmented Reality mittels Natural Image Matting

Masterarbeit

zur Erlangung des Grades Master of Science (M.Sc.)
im Studiengang Computervisualistik

vorgelegt von
Kevin Kremer

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	Anna Katharina Hebborn, M.Sc. (Institut für Computervisualistik, AG Computergraphik)

Koblenz, im September 2017

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Inhaltsverzeichnis

Zusammenfassung

Im Bereich Augmented Reality ist es von großer Bedeutung, dass virtuelle Objekte möglichst realistisch in ein Kamerabild eingebettet werden. Nur so ist es möglich, dem Nutzer eine immersive Erfahrung zu bieten. Dazu gehört unter anderem, Verdeckung dieser Objekte korrekt zu behandeln. Während schon verschiedene Ansätze existieren, dieses Verdeckungsproblem zu beheben, wird in dieser Arbeit eine Lösung mittels Natural Image Matting vorgestellt. Mit Hilfe einer Tiefenkamera wird das Kamerabild in Vorder- und Hintergrund aufgeteilt und anschließend das virtuelle Objekt im Bild platziert. Für Bereiche, in denen die Zugehörigkeit zu Vorder- oder Hintergrund nicht eindeutig ist, wird anhand bekannter Pixel ein Transparenz-Wert geschätzt. Es werden Methoden präsentiert, welche einen Ablauf des Image Matting in Echtzeit ermöglichen. Zudem werden Verbesserungsmöglichkeiten dieser Methoden präsentiert und gezeigt, dass durch diese eine höhere Bildqualität für schwierige Szenen erreicht wird.

Abstract

For augmented reality applications, it is important to embed virtual objects realistically into a camera image for providing an immersive experience for the user. Therefore, occlusion of these objects has to be handled correctly. While there already exist a few approaches for solving the occlusion problem, this thesis covers a solution through Natural Image Matting. The image is divided into fore- and background using a time-of-flight camera. Afterwards, the virtual object is placed into the image. For image regions which are diffucult to assign to fore- or background, a transparency value is estimated. This work presents a technique which enables real-time Image Matting. Additionally, improvements are introduced for this method which lead to a higher matte quality in difficult scenes.

1 Einleitung

Die Verbindung von realer und virtueller Welt ist eine Idee, deren Umsetzung mittlerweile ein sehr großes Forschungsgebiet beschäftigt und auch in der Industrie bereits Fuß fasst. Diese Fusion beider Welten auf Basis von Kameras wird Augmented Reality (AR) genannt und ist mögliche Grundlage vieler Anwendungen, die in Zukunft unseren Alltag bestimmen könnten. Erste AR-Software ist bereits kommerziell erhältlich, zum Beispiel auf dem Smartphone, oder wird in der Industrie verwendet. Daher ist es besonders wichtig, dass solche Anwendungen eine immersive Erfahrung bieten. Dies stellt eine Herausforderung sowohl im Bereich Bildverarbeitung als auch in der Computergrafik dar.

Es existieren bereits einige Ansätze, welche versuchen, eine solche Verschmelzung beider Welten möglichst fließend wirken zu lassen. Eine wichtige Aufgabe, die für eine solche immersive Anwendung nötig ist, ist die korrekte Behandlung von Verdeckungen (siehe Abbildung 1). Standardmäßig werden virtuelle Objekte in AR-Anwendungen immer vor dem Kamerabild eingefügt. In der Realität kann es allerdings vorkommen, dass ein Objekt ein oder mehrere andere Objekte verdeckt. Daher sollte es für eine realistische Darstellung auch möglich sein, virtuelle Objekte durch echte Objekte verdecken zu können. Wird eine solche Verdeckung nicht berücksichtigt, sorgt dies für einen falschen Tiefeneindruck. Das kann sich zum Beispiel folgendermaßen äußern: Angenommen, ein Nutzer möchte mit einem virtuellen Objekt interagieren und versucht, dieses zu greifen. Ohne Verdeckungsbehandlung sieht es für den Benutzer immer so aus, als befände sich das virtuelle Objekt vor den echten Objekten, auch wenn es tatsächlich weiter weg und hinter echten Objekten liegen kann. Durch diesen verfälschten Tiefeneindruck kann es passieren, dass der Nutzer an dem virtuellen Objekt vorbeigreift. Dies wird als das Verdeckungsproblem in AR bezeichnet. Um eine Anwendung für AR-Software möglich zu machen, muss zudem gewährleistet sein, dass diese Verdeckungsberechnung in Echtzeit geschieht. Hierzu existieren bereits verschiedene Ansätze, welche beispielsweise auf Objekttracking basieren [1] oder Farb- und Tiefendaten miteinander verbinden [2].

Ein anderer, neuer Ansatz zur Behandlung des Verdeckungsproblems stellt eine Lösung durch Natural Image Matting [3] dar. Dies ist eine Technik, die normalerweise dazu dient, einen Bildbereich zu maskieren und aus dem Bild zu extrahieren. Diese Idee kann auf das Verdeckungsproblem übertragen werden. Dabei wird auf Basis des Image Mattings eine Trennung des Kamerabilds in Vorder- und Hintergrund vorgenommen und das virtuelle Objekt zwischen diesen beiden Ebenen platziert. Dies wird unter anderem durch die Verwendung von Tiefendaten ermöglicht und sorgt für eine realistisch wirkende Einbettung des Objekts im Bild. Jedoch besitzt auch dieses Verfahren noch Probleme, gerade dann, wenn Vorder- und

(a) **(b)**

Abbildung 1: Die korrekte Behandlung von Verdeckungen soll es für AR-Anwendungen möglich machen, zu entscheiden, ob sich ein virtuelles Objekt im Vordergrund (a) oder Hintergrund (b) befindet. Das in (b) gezeigte Bild, in dem ein virtuelles Objekt verdeckt wird, wurde durch das in dieser Arbeit beschriebene Matting-Verfahren erstellt.

Hintergrund visuell schwer voneinander zu unterscheiden sind. In solchen Fällen ist es für den Algorithmus schwer, zu entscheiden, zu welcher der beiden Ebenen ein Pixel im Bild zuzuordnen ist. Daraus erfolgt oft eine falsche Unterteilung in Vorder- und Hintergrund. Auch stellen detaillierte Bereiche wie große Haarregionen eine Fehlerquelle dar. Das kommt daher, dass für eine Bildsequenz einer AR-Anwendung das Image Matting für jeden Frame von Grund auf neu durchgeführt wird, ohne temporale Informationen in die Berechnung mit einzubeziehen. So kann ein Flimmern in diesen Bereichen zwischen mehreren Frames bemerkt werden.

In dieser Arbeit wird das Verdeckungsproblem in Augmented Reality als Natural Image Matting-Problem betrachtet und eine Lösung mit diesem Ansatz vorgestellt. Unter Verwendung von Informationen aus Farb- und Tiefenbild wird das Kamerabild in Vorder- und Hintergrund separiert. Dazu wird zuerst eine Trimap erstellt, welche das Bild in Vordergrund, Hintergrund und unbekannte Regionen einteilt. Für die unbekannten Bereiche wird dann ein Alpha-Matting Verfahren angewandt, um eindeutig die Zugehörigkeit zu Vorder- oder Hintergrund zu bestimmen und das Bild zu trennen. Zwischen den so entstehenden Ebenen wird dann das virtuelle Objekt eingefügt. Dabei wird das Verfahren aus [3] als Grundlage genommen. Der Schwerpunkt dieser Arbeit ist die Erweiterung des bestehenden Verfahrens, um die Qualität der Alpha Matte in den oben genannten Problemfällen zu verbessern. Hierzu wird ein Konzept vorgeschlagen, welches dafür sorgt, dass Artefakte in der Alpha Matte in solchen Szenen reduziert werden. Zusätzlich wird dessen Implementierung besprochen und evaluiert, inwieweit eine Verbesserung anhand der neuen Ansätze erreicht wird.

2 Grundlagen

Dieser Abschnitt dient dazu, einen Überblick zu schaffen, auf welchen Themengebieten diese Arbeit aufbaut. Die folgenden Grundlagen sind für den Leser notwendig, um die Funktionsweise und den Nutzen des in dieser Arbeit besprochenen Themas zu verstehen. In den anschließenden Kapiteln wird somit vorausgesetzt, dass diese Grundlagen bekannt sind. Aus diesem Grund werden sie in den folgenden Unterabschnitten zum Verständnis des Lesers erläutert.

2.1 Augmented Reality

Unter Augmented Reality (dt. *Erweiterte Realität*), stellenweise auch *Mixed Reality* genannt, wird im Allgemeinen die Verbindung von realer und virtueller Welt verstanden. Das dient dem Zweck, die Wahrnehmung der Realität durch diverse Funktionen zu erweitern. Der Fokus liegt hierbei meist darauf, die Verbindung beider Welten durch eine visuelle Darstellung durchzuführen. Allerdings kann diese Fusion theoretisch über alle Sinnesorgane erfolgen. Im Folgenden wird allerdings hauptsächlich auf die visuelle Erweiterung der Realität eingegangen. Diese erfolgt meist dadurch, dass virtuelle Objekte in ein Kamerabild eingebettet werden. Innovationen wie die Microsoft Hololens versuchen zudem bereits, anstatt einer Kamera direkt durch eine Brille virtuelle Informationen in der realen Welt erscheinen zu lassen [4]. Dies sorgt für eine höhere Immersion, da die reale Welt nicht mehr nur durch ein Kamerabild erblickt werden muss.

Eine Fusion beider Welten kann in einer Vielzahl verschiedener Bereiche Anwendung finden. Es ist z.B. vorstellbar, dass AR-Anwendungen zusätzliche Informationen über Personen, Sehenswürdigkeiten etc. direkt auf dem Bildschirm anzeigen können (siehe Abbildung 2). Auch gibt es bereits Spiele, die auf AR basieren. Für diese ist ein hoher Immersionsgrad besonders wichtig, um dem Spieler so gut wie möglich den Eindruck zu verschaffen, dass virtuelle Spielelemente in der echten Welt existieren. Eine visuelle Bedienungsanleitung, in der Schritt für Schritt eine bestimmte Vorgehensweise durch Pfeile oder andere Visualisierungen erklärt wird, wäre ein weiterer Anwendungsbereich.

Damit eine realistische Einbettung virtueller Objekte in die reale Welt durchgeführt werden kann, müssen einige Herausforderungen überwunden werden. Eine dieser Herausforderungen ist das erfolgreiche Bestimmen der Position eines virtuellen Objekts in der echten Welt. Das bereitet insbesondere dann Schwierigkeiten, wenn sich die Position des virtuellen Objekts im Laufe der Zeit ändert. Dies kann entweder durch Bewegung der Kamera oder durch die eigene Bewegung des Objekts geschehen. Das Hauptproblem besteht darin, dass die Position und Orientierung der Kamera in der Welt unbekannt ist. Dieses Problem wird mittels

Abbildung 2: Eine mögliche AR-Anwendung zur Anzeige sozialer Informationen.[5]

Tracking-Verfahren versucht zu lösen. Zwei Möglichkeiten dazu bietet das *marker-basierte* sowie das *markerlose Tracking*.

Für marker-basiertes Tracking werden spezielle, leicht durch die Kamera identifizierbare Objekte (sogenannte Marker) in der echten Szene platziert. Diese werden benutzt, um die Position der Kamera zu bestimmen und daraufhin auch die der virtuellen Objekte ermitteln zu können. Dies kann u.A. zu Problemen führen, wenn die Marker den Bildbereich verlassen. Dann ist kein Referenzobjekt mehr vorhanden, anhand welchem die Position der Kamera ermittelt werden kann. Daraus folgend ist es auch nicht möglich, die Position des virtuellen Objekts festzustellen.

Markerloses Tracking versucht daher, ganz ohne spezielle Marker die Position der Objekte zu bestimmen. Dies bildet eine deutlich schwerer zu lösende Aufgabe, da meist nur Informationen aus dem Bild vorhanden sind. Zudem kann die Umgebung, in der sich die Kamera befindet, nicht immer als bekannt vorausgesetzt werden.

Ein weiteres Problem, welches den Realismus der AR einschränkt, ist die unterschiedliche Beleuchtung von virtueller und realer Szene. Während virtuelle Objekte meist mit einem speziellen Beleuchtungsmodell gerendert werden, kann die Beleuchtung der echten Szene stets unterschiedlich sein. Wird das virtuelle Objekt nicht auf die selbe Weise beleuchtet wie die echte Szene, führt das dazu, dass sich die virtuellen Objekte stark vom Rest der Szene unterscheiden und unecht wirken. Auf diese Weise kann eine realistische Fusion beider Welten nicht bewerkstelligt werden. Um beide Beleuchtungsmodelle aneinander anzupassen, muss daher bekannt sein, wie die Beleuchtungssituation in der echten Welt ist. Diese lässt sich

allerdings oft nur aus den Bildinformationen ableiten, was eine große Herausforderung darstellt. Ein Ansatz, dieses Problem zu lösen, kann zum Beispiel unter [6] nachgelesen werden.

Neben vielen weiteren Teilgebieten der AR-Forschung ist auch die korrekte Verdeckung von virtuellen Objekten eine wichtige Vorraussetzung für ein immersives Erlebnis. Dazu muss, wie schon unter Abbildung 1 dargestellt, immer ermittelt werden, ob sich reale Objekte vor den virtuellen Objekten befinden. Werden solche Verdeckungen nicht beachtet, kann das dem Nutzer einen falschen Tiefeneindruck vermitteln. So kann es u.a. passieren, dass der Nutzer bei Interaktionen mit dem virtuellen Objekt vorbei greift, weil er den Eindruck hat, dass sich das Objekt weiter vorne befindet. Des Weiteren muss das Szenario beachtet werden, dass der Nutzer bzw. die Kamera die Position ändert. Dann kann es vorkommen, dass das virtuelle Objekt nun verdeckt wird, obwohl es vorher im Vordergrund war. Auch dieser Fall bricht den Eindruck einer Fusion von virtueller und echter Szene, wenn er nicht gesondert behandelt wird. Solche dynamischen Verdeckungen (Wechsel von Vorder- in Hintergrund oder umgekehrt) müssen daher ebenfalls beachtet werden.

2.2 Alpha Matting

Alpha Matting ist eine weit verbreitete Technik, die hauptsächlich im Bildbearbeitungsbereich angewandt wird. Dabei wird versucht, ein Bild in Vorder- und Hintergrund einzuteilen. Dahinter steht die Idee, dass sich jedes Bild I als Komposition von Vordergrund (F) und Hintergrund (B) folgendermaßen beschreiben lässt:

$$I = \alpha F + (1 - \alpha)B \tag{1}$$

Diese Gleichung ist fundamentale Basis des Alpha Mattings und wird in nahezu jeder wissenschaftlichen Arbeit zu diesem Thema referenziert (siehe u.a. [7], [8], [9]). Dabei ist $\alpha \in [0, 1]$ und gibt die Opazität des Vordergrunds an. Dieses Alpha kann daher von Pixel zu Pixel variieren. Gilt für einen Pixel $\alpha = 1$, bedeutet das, dass dieser vollständig zum Vordergrund gehört. Für $\alpha = 0$ wäre dieser dementsprechend eindeutiger Hintergrund. Jeder Wert dazwischen bedeutet, dass sich die Farbe des Pixels aus einer Kombination von Vordergrund und Hintergrund ergibt. Somit besitzt der Vordergrund an diesen Stellen eine gewisse Transparenz. Das Hauptproblem des Mattings besteht darin, dass das Lösen der obigen Gleichung äquivalent zum Lösen eines unterbestimmten Gleichungssystems ist. Für jeden Pixel müssen drei Unbekannte bestimmt werden: F, B und α. Jedoch ist nur eine Variable bekannt: Das Eingabebild I bzw. der jeweilige Pixel.

Alpha Matting kann unter Anderem dazu verwendet werden, den Vordergrund eines Bilds zu extrahieren und in ein anderes Bild einzufügen. Ein oft angewandtes und effizientes Verfahren zum Trennen der Bildbereiche ist das *Bluescreen Matting*. Bei diesem wird das Objekt oder die Person vor einem einfarbigen Hintergrund platziert (meist blau oder grün). Beim Trennvorgang werden dann alle Pixel, welche dieselbe Farbe wie der Hintergrund besitzen, herausgefiltert. Als Resultat bleibt nur noch das Vordergrundobjekt übrig und kann nun beliebig weiterverwendet werden. Dieses Verfahren wird unter anderem in der Film- und Fernsehindustrie angewandt. Hier werden z.B. Aufnahmen in einem Studio mit einfarbiger Hintergrundwand durchgeführt und dieser Hintergrund anschließend durch einen Anderen ausgetauscht. Das bietet eine einfache Möglichkeit, Personen oder Objekte für eine Videoszene in andere Umgebungen einzufügen, ohne dass sie sich tatsächlich dort befinden müssen.

(a) **(b)** **(c)**

Abbildung 3: Alpha Matting. (a): Eingabebild. (b): Trimap. Vordergrund (weiß), Hintergrund (schwarz) und unbekannter Bereich (grau). (c): Die auf Basis der Trimap entstandene Alpha Matte. Hierfür wurde für alle Pixel aus dem unbekannten Bereich der Trimap ein Alpha-Wert bestimmt. Die Bilder wurden aus der Online-Datenbank von Rhemann et al. [10] entnommen (Zuletzt abgerufen am 24.08.17).

Beim Bluescreen Matting können allerdings Schwierigkeiten entstehen, wenn das Vordergrundobjekt teilweise dieselbe Farbe wie der Hintergrund enthält oder transparent ist. Dann werden versehentlich Teile des Vordergrunds als Hintergrund interpretiert und die Trennung wird fehlerhaft. Des Weiteren ist es nicht in jedem Anwendungsbereich möglich, ein Objekt vor einem festgelegten Hintergrund zu filmen. Bei diversen Kameraanwendungen müssen Bilder in einer natürlichen Umgebung geschossen werden, da eine Studioumgebung gerade für Alltagsanwendungen nicht als gegeben vorausgesetzt werden kann. Für solche Fälle wird *Natural Image Matting* benötigt. Dieses trennt Vorder- und Hintergrund für natürliche Bilder ohne einheitlichen Hintergrund. Dafür ist es nötig, zu bestimmen, welche Bereiche des Bildes garantiert zu Vorder- oder Hintergrund gehören. Ohne eine solche Zuweisung ist es nicht möglich, das unterbestimmte Gleichungssystem aus Gleichung 1 zu lösen. Dies geschieht durch sogenannte *Trimaps*. In einer Trimap wird ein Bild in drei Bereiche einge-

teilt: Vordergrund, Hintergrund oder Unbekannt, wenn für diesen Bereich die Zugehörigkeit zu Vorder- oder Hintergrund nicht eindeutig ist. Diese Einteilung wird meist durch eine Benutzereingabe festgelegt, kann aber auch automatisch erfolgen. Um eine anschließende, definitive Trennung in Vorder- und Hintergrund zu ermöglichen, wird für die Pixel im unbekannten Bereich ein Alpha-Wert auf Basis der bekannten Bereiche geschätzt. Somit entsteht eine *Alpha Matte*. Sie enthält für jedes Pixel einen Alpha-Wert, welcher die Zugehörigkeit zur jeweiligen Ebene angibt. Ein Beispiel für eine Alpha Matte sowie eine zugehörige Trimap lässt sich in Abbildung 3 finden.

2.3 ToF-Kameras

Für viele Anwendungen, nicht nur innerhalb des Bildverarbeitungsbereichs, ist es wichtig, die Distanz zwischen dem gerade verwendeten Gerät und anderen Objekten automatisch messen zu können. Ist diese Messung in einer Echtzeitanwendung nötig, muss sie auch entsprechend schnell erfolgen. Auch für Kameraanwendungen ist es oft hilfreich, für jedes Pixel die Distanz zu anderen Objekten zu kennen. Zur Messung dieser Distanz werden meist sogenannte *Time-of-Flight*-Kameras, kurz ToF-Kameras, verwendet. Diese Kameras erstellen ein Tiefenbild, welches für jedes der Pixel die Distanz zur Kamera beinhaltet. Dazu wird für jedes Pixel ein Lichtpunkt entlang der Blickrichtung ausgestrahlt. Für diesen kann nun die Zeit gemessen werden, die er benötigt, um zum nächstgelegenen Objekt und wieder zurück zu gelangen. Aus der gemessenen Zeit lässt sich dann die Distanz ermitteln, die das Licht zurückgelegt hat. In vielen Fällen wird zusätzlich zum Tiefenbild auch ein Farbbild zum selben Zeitpunkt aufgenommen. Dadurch besitzt man gleichzeitig Informationen über RGB-Farbe und Entfernung zum Objekt.

Abbildung 4: Microsoft Kinect v2. Dieses Modell wurde auch zur Aufnahme der Testdaten dieser Arbeit verwendet. [11]

Während es viele verschiedene Modelle solcher ToF-Kameras gibt, haben sich erst in den letzten Jahren kostengünstige Kameras etabliert, wodurch die Nutzung für private Zwecke ermöglicht wurde. Ein Beispiel einer solchen kostengünstigen Kamera ist die *Microsoft Kinect* (Abbildung 4). Sie wurde ursprünglich für Spieleanwendungen entwickelt, wird nun aber sowohl in der Forschung als auch für Privatzwecke genutzt und ist weit verbreitet. Sie enthält eine RGB- sowie eine Tiefenkamera. Diese befinden sich jedoch an unterschiedlichen Stellen am Gerät, was bedeutet, dass Objekte im Farb- und Tiefenbild zueinander verschoben sind und eine leicht andere Perspektive besitzen. Dazu kommt, dass das Tiefenbild eine geringere Auflösung als das Farbbild besitzt. Diese beiden Faktoren erschweren die eindeutige Zuordnung von Farb- zu Tiefenpixel und können dafür sorgen, dass die Tiefendaten nicht pixelgenau auf das Farbbild übertragen werden können. Allerdings besitzt die Kinect-Software Möglichkeiten, diese Zuordnung automatisch und möglichst genau durchzuführen. Dennoch können hier Fehler entstehen.

Ein weiteres Problem, welches den meisten ToF-Kameras zugeordnet werden kann, ist ein hoher Grad an Rauschen, welcher durch die Aufnahme der Tiefendaten entsteht. So kann sich die Tiefeninformation eines Videos von einem zum nächsten Frame verändern, obwohl die aufgenommene Szene still geblieben ist. Dieses Rauschen erschwert es zusätzlich, pixelgenaue Aussagen anhand der Tiefendaten zu treffen und muss bei der Verwendung einer solchen Kamera immer berücksichtigt werden. Da in diesem Bezug in den letzten Jahren jedoch schon Fortschritte erzielt wurden, ist zu erwarten, dass die Fehleranfälligkeit der Tiefendaten in Zukunft weiter sinken wird.

2.4 Optical Flow

In einer Vielzahl von Anwendungsbereichen der Bildverarbeitung ist es notwendig, Objekte oder Personen in einer Bildsequenz zu verfolgen. Eine solche Verfolgung kann zum Beispiel durchgeführt werden, indem man die Bewegung der Pixel von einem zum nächsten Frame misst. Dies kann durch die Berechnung des Optical Flows (dt. *Optischer Fluss*) durchgeführt werden. Ziel ist es, für jedes Pixel des vorherigen Frames zu ermitteln, wohin sich dieses im darauffolgenden Frame bewegt hat. Dazu muss der Farb- bzw. Helligkeitswert des aktuell betrachteten Pixels in einer gewissen Nachbarschaft im nächsten Bild wiedergefunden werden. Hierbei gibt es zwei wichtige Voraussetzungen: Die Helligkeit der Bildsequenz darf sich von einem zum nächsten Bild nicht verändern, und die Bewegung des Pixels muss klein genug sein, dass dieses im nächsten Bild noch nahe genug am Ursprungspixel liegt. Sind diese Voraussetzungen gegeben, liefert der Optical Flow ein Ergebnisbild, in welchem der Verschiebungsvektor jedes Pixels enthalten ist. Für den Optical Flow ist die sogenannte *Helligkeits-*

konstanzeinschränkung (engl. *brightness constancy constraint*) vorausgesetzt, welche besagt:

$$I(\vec{x}, t) \approx I(\vec{x} + \partial\vec{x}, t + \partial t) \tag{2}$$

(siehe Horn und Schunck [12] oder Fleet und Weiss [13]). $I(\vec{x}, t)$ bezeichnet hier die Bildhelligkeit bzw. Farbe des Pixels $\vec{x} = (i, j)$ zum Zeitpunkt t. Aus der Gleichung resultiert, dass die Helligkeit an Pixel $\vec{x}$ zum Zeitpunkt t in etwa der Helligkeit des verschobenen Pixels $\vec{x}+\partial\vec{x}$ zum Zeitpunkt $t+\partial t$ entsprechen muss. Demnach gibt der Optical Flow den Verschiebungsvektor $\partial\vec{x}$ des Pixels an, die nach Ablauf der Zeit ∂t erfolgt ist.

Das Prinzip des Optical Flow ist in Abbildung 5 dargestellt. Die Pixelhelligkeiten im Eingabebild zum Zeitpunkt t (linkes Bild) verschieben sich zum nächsten Frame $t + \partial t$ (rechtes Bild) an andere Positionen im Bild. Ziel ist es, herauszufinden, welcher Pixel sich an welche Stelle im nächsten Frame bewegt hat. Dies ist durch die schwarzen Pfeile im rechten Bild angegeben, welche die Verschiebungsvektoren darstellen.

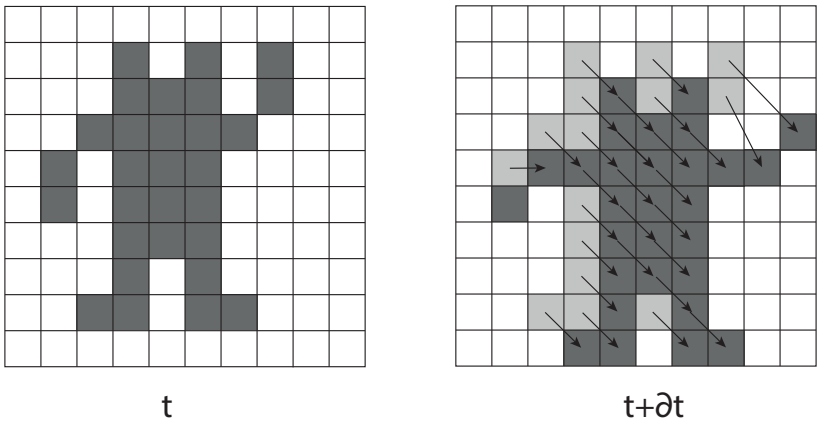


Abbildung 5: Darstellung des Optical Flows. Links: Das Eingabebild zum Zeitpunkt t. Rechts: Das Bild zum Zeitpunkt $t + \partial t$ inklusive Verschiebungsvektoren. Die ursprünglichen Pixelpositionen sind hellgrau markiert.

Bei der Berechnung des Optical Flows kann sich das *Aperturproblem* ergeben. Dieses besagt, dass eine genaue Ermittlung des Flows nur dann möglich ist, wenn sich die Pixel entlang des Helligkeitsgradienten bewegen. Dies ist jedoch nicht immer der Fall. Die Bestimmung des Flows ist somit mehrdeutig, was eine potentielle Fehlerquelle darstellen kann. Je kleiner das Fenster ist, in dem der Flow berechnet wird, desto höher ist die Chance, dass sich das Aperturproblem ereignet.

Unter den vielen möglichen Varianten und Algorithmen, den Optical Flow zu berechnen, lassen sich zwei grundlegende Methoden ausmachen: *Sparse Optical Flow* und *Dense Optical Flow*. Der Sparse Optical Flow berechnet die Pixelbewegung nur für wenige, vorher festgelegte Pixel. Das können z.B. solche sein, die in einem Vorverarbeitungsschritt als markante Pixel ausgewählt wurden und verfolgt werden sollen. Ein Beispiel für ein Sparse Optical Flow-Verfahren ist der Lucas-Kanade Algorithmus [14], welcher Informationen aus der Nachbarschaft des betrachteten Pixels mit einbezieht. Der Dense Optical Flow ermittelt die Pixelverschiebung für jedes Pixel innerhalb des gewählten Bildbereichs und kann sich somit auch über das gesamte Bild erstrecken. Das macht seine Berechnung deutlich aufwändiger als die der Sparse-Variante, kann dafür aber genauere Ergebnisse liefern. Eine bekannte Technik zur Berechnung des Dense Optical Flows wurde von Gunnar Farnebäck [15] eingeführt.

3 Stand der Technik

Es existieren bereits mehrere Versuche, das Verdeckungsproblem in AR zu lösen. Ebenfalls wurde schon eine Vielzahl verschiedener Techniken entwickelt, Image Matting auf natürlichen Bildern durchzuführen. Ein kleiner Anteil davon hat sich auch mit dem Berechnen einer Alpha Matte in Echtzeit beschäftigt. Auf diese bereits bestehenden Verfahren wird im Folgenden weiter eingegangen und deren Vor- und Nachteile diskutiert.

3.1 Verdeckungsproblem in Augmented Reality

Die korrekte Behandlung der Verdeckung virtueller Objekte wurde noch nicht so gründlich studiert wie andere Teilgebiete des AR-Forschungsbereichs, beispielsweise das Tracking. Die Ansätze, wie das Verdeckungsproblem gelöst werden soll, unterscheiden sich daher stark voneinander. Tian et al. [1] wählen einen Objekttracking-Ansatz und adaptieren diesen so, dass Verdeckungen damit behandelt werden können. Dazu werden die relativen Positionen virtueller sowie reeller Objekte verfolgt. Hierfür muss jedoch ein verdeckendes, reelles Objekt vom Benutzer definiert werden, welches das virtuelle Objekt dauerhaft verdeckt. Dadurch entsteht eine fehlerhafte Darstellung, wenn sich die Tiefenbeziehung zwischen virtuellem und realem Objekt verändert.

Ebenfalls von Tian et al. [16] stammt ein Lösungsvorschlag auf Basis einer 3D-Rekonstruktion der realen Szene. Zuerst wird aus Farb- und Tiefendaten eine 3D-Punktewolke erstellt und darauf aufbauend die Beziehung zwischen den Punkten und dem virtuellen Objekt ermittelt. Hierbei ist die Genauigkeit der Verdeckung stark abhängig von der Genauigkeit der Punktewolke. Zudem bleibt auch mit diesem modellbasierten Ansatz die dynamische Behandlung von Verdeckungen ein Problem.

Allgemein ist ein naheliegender Ansatz die Verwendung von Tiefendaten, um zu ermitteln, an welchen Stellen im Bild eine Verdeckung des virtuellen Objekts stattfindet. Dies hängt jedoch stark von der Genauigkeit der Tiefendaten ab. Während Tiefensensoren in vergangenen Jahren einen starken Fortschritt im Bezug auf die Qualität der Tiefenmap gemacht haben, ist diese Qualität weiterhin nicht ausreichend, um sich ausschließlich auf Tiefendaten verlassen zu können. Ein auf Tiefendaten aufbauendes Konzept wird von Du et al. [2] vorgeschlagen. Sie versuchen, die Kanten aus Farb- und Tiefenbild übereinanderzulegen und somit eine genaue Verdeckung zu berechnen. Das erhöht die Qualität der Tiefendaten, sodass damit eine zuverlässige Berechnung der Verdeckung möglich wird. Das Verfahren liefert gute Ergebnisse, wenn Vorder- und Hintergrundobjekt einfach voneinander getrennt werden können. Es scheitert jedoch bei zu starker Ähnlichkeit von Vorder- und Hintergrund und eine detaillierte Behandlung von Haarregionen ist mit diesem Verfahren ebenfalls nicht

möglich, da dafür die Tiefendaten zu ungenau sind.

Der erste Vorschlag zur Lösung des Verdeckungsproblems, der auf Image Matting basiert, stammt von Hebborn et al. [3]. Image Matting wird normalerweise dazu verwendet, einen bestimmten Bereich aus einem Bild möglichst genau zu extrahieren. Diese Idee wird auf das Verdeckungsproblem angepasst. Anhand von Farb- und Tiefendaten wird der Vordergrund aus dem Bild extrahiert. Anschließend wird das virtuelle Objekte zwischen Vorder- und Hintergrund eingefügt. Die Verwendung des Image Mattings sorgt im Vergleich zu anderen Verfahren dafür, dass detaillierte Bereiche, wie z.B. Haarregionen, genauer behandelt werden können. Da dieses Verfahren Grundlage für die vorliegende Arbeit ist, wird dessen Funktionsweise in Abschnitt 4 im Detail beschrieben und auf Problemfälle des Verfahrens in Abschnitt 5.1 eingegangen.

3.2 Natural Image Matting

Im Bereich Natural Image Matting existieren sehr viele unterschiedliche Ansätze, die die Erstellung einer Alpha Matte beschreiben. Die gängigsten Matting-Verfahren wurden von Rhemann et al. [10] auf deren Fehleranfälligkeit getestet und in einer Online-Benchmark zusammengefasst. Fast alle Ansätze basieren auf einer Voraberstellung einer Trimap, welche Vorder-, Hintergrund und unbekannte Bereiche des Bilds kennzeichnet. Dies ist notwendig, um das unterbestimmte Problem des Natural Image Mattings zu behandeln (siehe Gleichung 1). Die meisten Ansätze können zwischen propagations-basiertem und sample-basiertem Matting unterschieden werden. Beim propagations-basierten Matting (siehe u.A. Sun et al. [17], Levin et al. [7]) werden die Farbinformationen aus den bekannten Bereichen in den unbekannten Bereich interpoliert und darauf aufbauend ein Alpha-Wert geschätzt. Dies geschieht unter der Annahme, dass sich die Farbwerte in Vorder- und Hintergrund in kleinen Regionen nicht verändern. Probleme können hier entstehen, wenn der Vorder- oder Hintergrund wenige glatte Strukturen aufweist.

Das sampling-basierte Matting (verwendet unter anderem in Johnson et al. [18], Chuang et al. [19], Rhemann et al. [8]) sucht daher explizite Vordergrund- und Hintergrundpixel aus dem Bild heraus, um auf dessen Basis Alpha-Werte im unbekannten Bereich zu bestimmen. Hier kann das Problem entstehen, dass gute Samples verpasst werden, da nur begrenzte Bereiche im Bild untersucht werden. So kann es vorkommen, dass ein Sample-Paar ausgewählt wird, das den wahren Vorder- und Hintergrund nicht geeignet repräsentiert und somit keine echten Samples darstellt. Wang und Cohen [20] verwenden eine Kombination aus propagations- und sample-basierten Ansätzen, um die Nachteile beider Methoden zu mitigieren. He et al. [21] schlagen vor, das Sampling nicht nur auf umliegende Pixel zu beschränken, sondern ein globales Sampling zu

verwenden, damit keine echten Samples verpasst werden. Die genannten Verfahren sind allerdings aufgrund hoher Berechnungskomplexität nicht für eine Anwendung in Echtzeit geeignet.

Der erste Ansatz, Natural Image Matting in Echtzeit durchzuführen, wurde von Gastal und Oliveira [9] vorgeschlagen. Sie erreichen dies, indem sie die Berechnung der Alpha-Werte für nahe beieinander liegende Pixel im unbekannten Bereich zusammenfassen, sowie durch die Verlagerung der Berechnung auf die GPU.

Ein weiteres Problem ist, dass viele der Matting-Verfahren davon ausgehen, dass Trimaps vorab vom Benutzer definiert werden. Dies ist aber für eine Echtzeitanwendung nicht umsetzbar. Hierfür ist eine automatische Generierung von Trimaps nötig. Pham et al. [22] schlagen eine Video Matting Methode vor, die unter Echtzeitbedingungen läuft. Dazu segmentieren sie das Bild anhand der Farbinformationen in zwei Ebenen und schätzen die Alpha-Werte auf Basis der Ebenen. Diese Idee wird von Wang et al. [23] aufgefasst und mit Tiefendaten kombiniert. Auf Basis der Segmentierung wird dann eine Erosion auf Vordergrund und Hintergrund durchgeführt, um die unbekannten Regionen der Trimap zu bestimmen. Dieses Verfahren berücksichtigt jedoch nicht die unterschiedliche Behandlung von Haarbereichen und Regionen mit klaren Kanten in der Trimap. Zudem wird für die spätere Alpha-Schätzung keine temporale Information verwendet. Sindeev et al. [24] präsentieren einen Ansatz, die Alpha Matte eines Videos auf Basis bestimmter Keyframes anhand des Optical Flows einer Bildsequenz zu bestimmen. Auf diese Weise wird die temporale Information zu einem entscheidenden Faktor zur Berechnung der Alpha-Werte. Allerdings arbeitet ihr Verfahren auf offline-Daten, was es ihnen ermöglicht, Informationen aus zukünftigen Frames mit zu berücksichtigen. Diese Informationen sind in AR-Anwendungen, welche online arbeiten, nicht abrufbar. Hier können nur Daten aus vergangenen Frames berücksichtigt werden.

4 Occlusion Matting

Ein großer Teil der vorliegenden Arbeit basiert auf dem Occlusion Matting-Verfahren von Hebborn et al. [3], welches in diesem Abschnitt im Detail beschrieben wird. Von ihnen stammt die Idee, das Verdeckungsproblem in Augmented Reality durch ein echtzeitfähiges Natural Image Matting zu lösen. Bisherige Ansätze zur Behandlung von Verdeckung in AR ([1], [16] [2]) können nur für bestimmte, vorher ausgewählte reale Objekte die Verdeckung behandeln oder liefern einen ungenauen Übergang im Bereich der Kanten. Die Verwendung von Alpha Matting hat den Vorteil, dass auch für detailreiche Objekte (z.B. haarige Objekte) eine akkurate Verdeckung ermittelt werden kann. Durch die Zuhilfenahme von Tiefendaten ist es möglich, einen einfachen Tiefentest zwischen virtuellen und realen Objekten durchzuführen und die Übergänge von Vorder- und Hintergrund genauer zu bestimmen.

Während des Occlusion Mattings werden die folgenden Schritte durchgeführt:

1. Rendern der virtuellen Objekte (Farbe und Tiefenwerte) in Texturen
2. Adaptive Generierung der Trimap anhand Farb- und Tiefenkanten
3. Propagation der Farbwerte aus Vorder- und Hintergrund in den unbekannten Bereich der Trimap
4. Schätzen des Alpha-Werts
5. Zusammenfügen von virtueller und realer Szene anhand der Alpha Matte

Diese Schritte werden in den nächsten Abschnitten im Detail besprochen. Eine Ausnahme bildet Schritt 1, welcher trivial ist. Hier werden lediglich die Farb- und Tiefenwerte der virtuellen Szene ermittelt und in getrennten Texturen abgespeichert. Die einzelnen Schritte des Occlusion Mattings sind außerdem in Abbildung 6 visualisiert.

4.1 Adaptive Trimap-Erstellung

Die Erstellung einer Trimap ist notwendig, um die Alpha-Werte an den Übergängen von Vorder- und Hintergrund genau bestimmen zu können. Dazu muss zusätzlich zum definitiven Vorder- und Hintergrund ein unbekannter Bereich gebildet werden, für den die Zuordnung zur jeweiligen Ebene nicht eindeutig ist. Hierbei wird darauf geachtet, dass dieser unbekannte Bereich in detailreichen Regionen (z.B. bei Haaren) groß sein muss, bei klaren Farbkanten jedoch kleiner sein kann. Der unbekannte Bereich wird durch eine Dilatation gebildet, welche berücksichtigt, dass

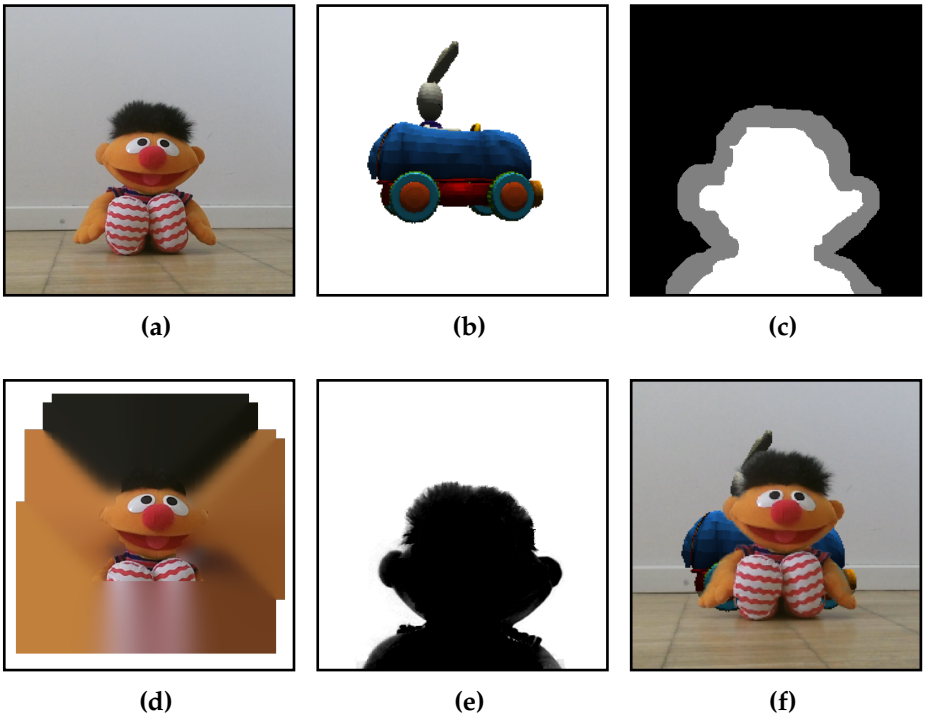
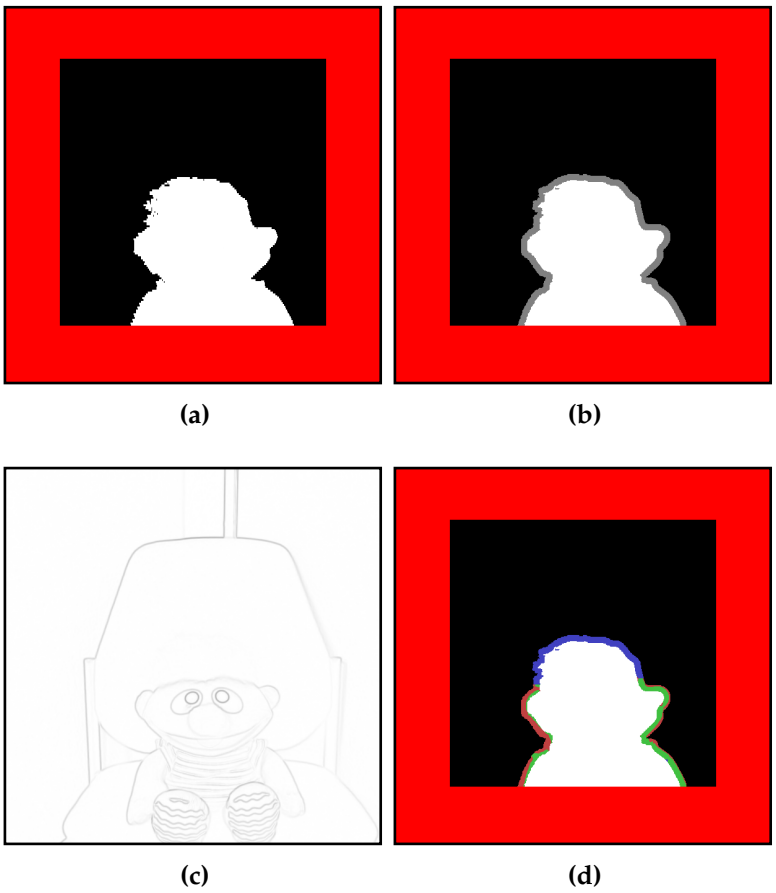


Abbildung 6: Ablauf des Occlusion Matting. Als Eingabe wird das Farbbild (a) sowie die zugehörigen Tiefendaten benötigt. Zudem wird eine virtuelle Szene (b) gerendert. Anschließend wird eine Trimap erstellt (c), um den unbekannten Bereich festzulegen. Unter (d) werden die Farbwerte aus dem Vordergrund in den unbekannten Bereich propagiert (Hintergrund analog). Aus diesen Farbwerten werden Samples ausgewählt und daraus die Alpha Matte berechnet (e). Anhand dieser wird das virtuelle Objekt in die Szene eingefügt (f) und somit vom Vordergrund verdeckt.

sowohl Farb- als auch Tiefenkanten in diesem Bereich liegen. Das soll bezwecken, dass nicht fälschlicherweise Hintergrundbereiche als Vordergrund angesehen werden oder umgekehrt. Dabei soll der unbekannte Bereich allerdings so klein wie möglich sein, da sonst kleine Vordergrundbereiche verloren gehen (z.B. Finger einer Hand oder Schwanz eines Plüschtiers).

Der erste Schritt besteht in der Erstellung einer initialen Trimap, welche aus dem Tiefenbild gewonnen wird. Diese Trimap wird, ebenso wie die darauffolgenden Trimaps und die entstehende Alpha Matte, nur für den Bereich ermittelt, in dem sich das virtuelle Objekt befindet. Alle Pixel außerhalb des virtuellen Objekts werden daher als ungültig markiert (rot). Der Vordergrund ist weiß, der Hintergrund schwarz und der unbekannte Bereich grau gefärbt. Vordergrund und Hintergrund werden nun ermittelt, indem ein Tiefentest zwischen den Pixeln aus dem Tiefenbild der

realen Szene und der Tiefenmap des virtuellen Objekts durchgeführt wird. So entsteht eine *Bimap* (siehe Abbildung 7a), welche noch keine unbekannten Bereiche beinhaltet. Um den unbekannten Bereich zu erhalten, wird die Bimap zuerst geglättet, um das Rauschen der Tiefendaten zu reduzieren. Anschließend wird sie mit einem Sobel-Filter behandelt, um die Kanten im Tiefenbild hervorzuheben. Die Idee dabei ist, dass sich der unbekannte Bereich der Trimap nur an den Kanten zwischen Vorder- und Hintergrund befinden kann. Daher werden Pixel, in deren Umkreis sich starke Kanten befinden, nun als unbekannter Bereich definiert und die Berechnung der initialen Trimap (Abbildung 7b) abgeschlossen.



(a) **(b)**

(c) **(d)**

Abbildung 7: Erste Schritte zur Generierung der Trimap. (a): Bimap, erstellt aus den Tiefendaten, bestehend aus Vordergrund (weiß), Hintergrund (schwarz) und ungültigen Bereichen (rot), welche außerhalb des virtuellen Objekts liegen. (b): Initiale Trimap mit unbekanntem Bereich (grau) auf Basis der Bimap. (c): Farbkanten des Eingabebilds (grau). (d): Einteilung der unbekannten Bereiche in *vorderer Halbraum* (rot), *hinterer Halbraum* (grün) oder *keine Kante* (blau), je nach Lage zur Farbkante.

Da diese Trimap nur auf Basis der Tiefendaten entstanden ist, ist nun das Ziel, den unbekannten Bereich der initialen Trimap in Richtung der Farbkanten zu erweitern. Dazu müssen die Kanten im Farbbild gefunden werden, was ebenfalls durch Filtern mittels einem Sobel-Kern realisiert wird. Ein solches Sobel-gefiltertes Farbbild ist in Abbildung 7c zu sehen. Nun werden die unbekannten Pixel markiert, je nachdem, wo sie sich im Bezug auf die Farbkante befinden. Dafür gibt es 3 Kategorien:

- *Vorderer Halbraum*, falls sich das unbekannte Pixel im vorderen Halbraum einer Farbkante befindet;
- *Hinterer Halbraum*, falls dieses Pixel im hinteren Halbraum einer Farbkante liegt;
- *keine Kante*, falls sich keine signifikante Kante in der Nähe befindet.

In welchem Halbraum sich das unbekannte Pixel befindet, lässt sich durch die Berechnung des Skalarprodukts zwischen diesem Pixel und nahegelegenen Punkten der Farbkante bestimmen. Diese Markierungen (visualisiert in Abbildung 7d) entscheiden nun, wie stark die Dilatation im jeweiligen Bereich ist.

Die Dilatation wird nun durchgeführt, indem die Nachbarschaft jedes Vordergrund- und Hintergrundpixels $i \in (F, B)$ in einem Fenster durchsucht wird. Die Größe des Fensters kann durch den Nutzer festgelegt werden. Findet sich in diesem Fenster ein Pixel u aus dem unbekannten Bereich U, welches als *vorderer* oder *hinterer Halbraum* markiert wurde, wird i ebenfalls als unbekannt markiert. Finden sich in dem Fenster jedoch zu viele Pixel, die als *keine Kante* markiert wurden, wird die Größe der Dilatation erhöht (auf einen zweiten vom Nutzer festgelegten Wert). Meist befinden sich viele dieser Pixel ohne Kante in detaillierten Bereichen (wie z.B. Haaren). Die Erhöhung der Dilatationsmenge bewirkt daher, dass auch in diesen Bereichen alle nötigen Pixel im unbekannten Bereich landen und keine echten Vordergrundpixel verpasst werden. Der Unterschied zwischen Verwendung einer simplen und einer adaptiven Trimap kann in Abbildung 8 betrachtet werden. Es lässt sich erkennen, dass bei der adaptiven Trimap eine stärkere Dilatation im Bereich der Haare entsteht. Das resultiert daraus, dass sich hier viele unbekannte Pixel befinden, die mit *keine Kante* markiert wurden.

4.2 Propagation der Farbwerte

Für die Erstellung der Alpha Matte ist es von großer Bedeutung, dass gute Samples ausgewählt werden, um den Alpha-Wert jedes unbekannten Pixels zu schätzen. Das Occlusion Matting verwendet hierbei eine Kombination aus propagations- und sample-basiertem Matting. Dafür

(a) (b) (c)

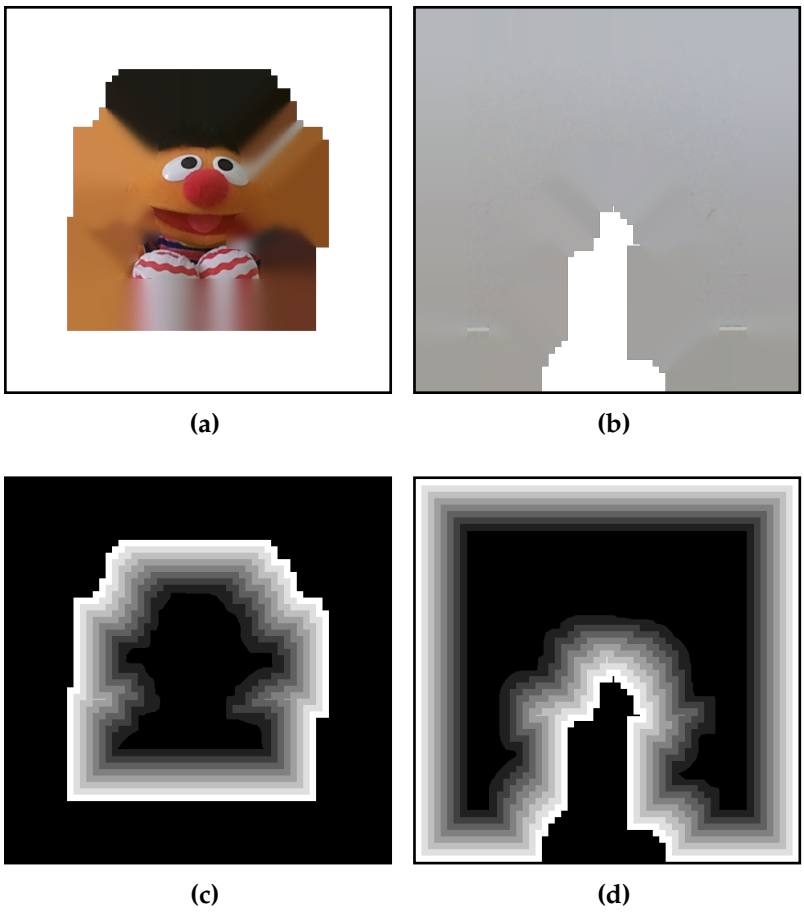
Abbildung 8: Automatisch generierte Trimaps. (a): Das Eingabebild. (b): Trimap ohne adaptive Dilatation. (c): Mit adaptiver Dilatation.

werden zuerst die Farbwerte aus den bekannten Bereichen in die unbekannten Bereiche diffundiert. Der folgend beschriebene Algorithmus wird für Vorder- und Hintergrund separiert durchgeführt.

Die Propagation der Farbwerte geschieht durch ein wiederholtes Weichzeichnen des Eingabebilds und findet in mehreren Iterationen statt. Hierzu wird für jede Iteration eine Bildpyramide aufgebaut, welche das Eingabebild in verschiedenen Auflösungen enthält. Die feinste Ebene der Pyramide ist dabei das Eingabebild selbst. Durch Heruntersetzen der Auflösung wird nun die nächste Ebene der Pyramide erstellt. Dies geschieht unter Verwendung eines Gauß-Filters. Dieser Prozess wird so lange wiederholt, bis die gröbste Ebene der Pyramide erreicht ist. Die Größe der Pyramide ist von der Nutzereingabe abhängig. Dieses Prinzip kann durch die Verwendung von Mip Maps realisiert werden, was eine schnelle Umsetzung auf der GPU ermöglicht. Jedes Mip Map-Level entspricht dabei einer Ebene der Pyramide.

Wurden alle Pyramidenlevel erstellt, wird die Pyramide top-down durchlaufen, um die Farben aus dem Eingabebild weichzuzeichnen. Die Farbwerte aus dem gröberen Level werden durch eine quadratische B-Spline Interpolation (genauer beschrieben unter [25]) in das nächst feinere Level übertragen. Anschließend werden die geglätteten Farben zurück in das Eingabebild geschrieben. Dies geschieht mittels linearer Interpolation, was für einen glatten Farbübergang sorgt. In der folgenden Iteration wird nun das neue, geglättete Bild als Eingabebild gewählt und die oben genannten Schritte wiederholt, bis die gewünschte Anzahl Iterationen erreicht ist. Um sich zu merken, während welchem Iterationsschritt die Farbe im Ergebnisbild entstanden ist, wird die Anzahl der durchgeführten Iterationen für jedes Pixel im Alpha-Kanal des RGBA-Bilds gespeichert. Dieser Wert dient später dazu, bei der Auswahl der Samples eine Aussage über die Genauigkeit des verwendeten Pixels treffen zu können. Farbwerte, die beispielsweise weiter vom tatsächlichen Vordergrund entfernt sind, be-

sitzen eine höhere Fehlerwahrscheinlichkeit als solche, die nah am echten Vordergrundpixel liegen. Solche weit entfernten Werte sollten daher bei der Alpha-Berechnung weniger stark berücksichtigt werden. Das Ergebnis der propagierten Farbwerte sowie eine Visualisierung des Propagationsfehlers kann in Abbildung 9 betrachtet werden.



(a) **(b)**

(c) **(d)**

Abbildung 9: Propagation der Farbwerte in den unbekannten Bereich. (a): Der Vordergrund mit diffundierten Werten. (b): Der zugehörige Hintergrund, ebenfalls mit propagierten Werten. (c)-(d): Der Propagationsfehler des Vorder- bzw. Hintergrunds. Die Farbe gibt die Iteration an, in welcher der Farbwert entstanden ist. Je heller der Grauwert, desto weiter wurde die Farbe propagiert und desto höher ist die Fehlerwahrscheinlichkeit. Für dieses Beispiel wurden 4 Mip Maps gebildet und 8 Iterationen durchgeführt.

4.3 Alpha-Schätzung

Sobald die Propagation der Farbwerte abgeschlossen wurde, werden aus diesen Werten Samples ausgesucht, welche in geeigneter Weise den Vorder- und Hintergrund repräsentieren. Um dies zu erreichen, wird für jedes unbekannte Pixel je ein Suchfenster für Vorder- und Hintergrund angelegt. In diesem Suchfenster soll nun das am besten geeignete Vorder- und Hintergrundsample gefunden werden. Dazu wird jedes Vordergrundpixel im Fenster in Kombination mit jedem Hintergrundpixel getestet und anhand einer Kostenfunktion bestimmt, ob das gewählte Sample-Paar zur Berechnung des Alpha-Werts geeignet ist. Diese Kostenfunktion C_{total} lautet wie folgt:

$$C_{\text{total}} = w_{\text{cost}} \cdot C_{\text{photo}} + (1 - w_{\text{cost}}) \cdot C_{\text{propa}} \tag{3}$$

Die Gesamtkosten werden also aus zwei weiteren Kostenfunktionen berechnet: Den photometrischen Kosten C_{photo} sowie den Propagationskosten C_{propa}. Diese beiden Kostenfunktionen gehen mit unterschiedlicher Gewichtung, welche durch w_{cost} beschrieben wird, in die Gesamtkosten ein.

Die photometrischen Kosten geben an, wie genau die Farbe am unbekannten Pixel durch die Farben der Sample-Pixel dargestellt werden können. Optimal ist es, wenn das unbekannte Pixel durch eine Linearkombination der Farben aus Vorder- und Hintergrund dargestellt werden kann. Das bedeutet, dass sich die beiden Farbvektoren von Vorder- und Hintergrundsample mithilfe eines Skalars so miteinander addieren lassen, dass als Ergebnis genau der Farbvektor des echten Pixels entsteht. Als Skalarwert wird in diesem Fall α verwendet. Ist dies nicht der Fall, sollte die Differenz zwischen tatsächlichem Farbwert und dem des Sample-Paares wenigstens so niedrig wie möglich sein. Als Gleichung lässt sich das folgendermaßen ausdrücken:

$$C_{\text{photo}}(F_i, B_j, I_p) = ||I_p - (\alpha F_i + (1 - \alpha)B_j)|| \tag{4}$$

Hierbei steht I_p für die tatsächliche Pixelfarbe und (F_i, B_j) bezeichnet das Sample-Paar bestehend aus Vorder- und Hintergrundfarbe. Diese photometrischen Kosten wurden von [8] als geeignete Metrik eingeführt, um ein gut passendes Sample-Paar zu finden. Ist die Ähnlichkeit der kombinierten Sample-Farbe und der tatsächlichen Farbe sehr hoch, ist davon auszugehen, dass das Paar zur Schätzung des Alpha-Werts gut geeignet ist.

Für die Berechnung der photometrischen Kosten ist auch die Schätzung des α-Werts für das Sample-Paar nötig. Dieser wird wie folgt ermittelt:

$$\alpha = \frac{(I - B) \cdot (F - B)}{||(F - B)||^2} \tag{5}$$

Der zweite Teil der Gesamtkosten wird durch die Propagationskosten bestimmt. Sie geben an, wie weit der Farbwert aus dem definitiven Vordergrundbereich propagiert wurde. Je weiter das gewählte Sample von einem echten Vorder- bzw. Hintergrundpixel entfernt ist, desto höher ist die Wahrscheinlichkeit, dass der propagierte Wert fehlerhaft und deshalb weniger geeignet zur Alpha-Berechnung ist. Die Formel zur Bestimmung der Propagationskosten lautet wie folgt:

$$C_{\text{propa}}(F_i, B_j, I_p) = \frac{d(F_i) + d(B_j)}{D_n} \tag{6}$$

Die Funktion $d(X)$ gibt an, in welcher Iteration des Propagationsalgorithmus der Farbwert X entstanden ist (visualisiert in Abbildung 9). D_n bezeichnet die Anzahl Iterationen, die insgesamt durchgeführt wurden. Diese beiden Kostenfunktionen werden nun mit dem nutzerdefinierten Gewicht w_{cost} in die Gesamtkostenfunktion (Gleichung 3) eingebracht. Für $w_{\text{cost}} = 0.8$ lassen sich gute Ergebnisse erzielen. Mindestens sollte dieser Wert aber über 0.5 liegen, da die photometrischen Kosten wichtiger eingestuft werden als die Propagationskosten.

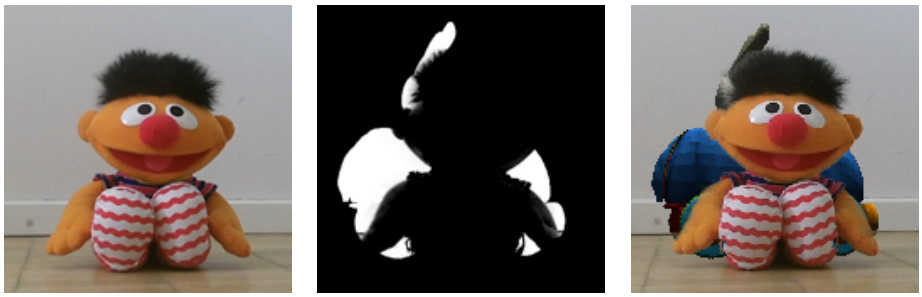
Wurden die Gesamtkosten für jedes Sample-Paar bestimmt, wird ermittelt, für welches der Sample-Paare die Gesamtkosten C_{total} minimal werden. Das Paar mit den niedrigsten Kosten wird ausgewählt und auf dessen Basis der Alpha-Wert geschätzt. Dies geschieht wie in Gleichung 5 beschrieben. Dieser Prozess wird für alle Pixel im unbekannten Bereich wiederholt, bis eine vollständige Alpha Matte entstanden ist.

4.4 Zusammensetzung der Szenen

Wurden alle Alpha-Werte bestimmt, können die virtuelle und reale Szene miteinander kombiniert werden. Hierzu wird Gleichung 1 verwendet. Dabei wird das virtuelle Objekt als Hintergrund angesehen, da dieses von den realen Objekten im Vordergrund verdeckt werden soll. Die tatsächliche Pixelfarbe bestimmt sich also aus Kombination von Vordergrundobjekt und virtuellem Objekt, abhängig von α. Die Zusammensetzung des finalen Bilds zeigt Abbildung 10.

Mit der genannten Methodik lässt sich also eine Verdeckung von virtuellen Objekten in AR-Anwendungen berechnen. Das Occlusion Matting liefert dabei gewisse Vorteile im Vergleich zu anderen bestehenden Verfahren wie [1] oder [2]. Zum einen sorgt die Verwendung der Tiefendaten dafür, dass das Bild automatisch anhand dieser Daten grob in Vorder- und Hintergrund segmentiert werden kann. Dadurch ist kein Tracking oder Markieren des Vordergrundobjekts notwendig. Es kann also ein beliebiges Vordergrundobjekt zur Verdeckung beitragen und dieses auch zur Laufzeit durch ein anderes Objekt ausgetauscht werden. Ebenso gut funktioniert der Algorithmus, wenn sich die Tiefenbeziehung zwischen Vordergrund

und virtuellem Objekt ändert. Dynamische Verdeckungen lassen sich durch Occlusion Matting also generell genau und zur Laufzeit behandeln. Des Weiteren können durch die Anwendung von Natural Image Matting auch Haarregionen oder andere Regionen mit viel Detail und weichen Kanten genau behandelt werden. Diese Vorteile tragen zu einer realistisch wirkenden Darstellung der Verdeckung bei.



(a) **(b)** **(c)**

Abbildung 10: Aus Eingabebild (a) und Alpha Matte (b) wird das finale Bild (c) zusammengesetzt.

5 Verbesserung des Occlusion Matting

Im Rahmen dieser Arbeit wird auf Verbesserungsmöglichkeiten für das in Kapitel 4 beschriebene Verfahren zum Alpha Matting in Echtzeit eingegangen. Im Folgenden werden somit Problemfälle aufgezeigt (Abschnitt 5.1), in denen beim Occlusion Matting Fehler oder Artefakte auftreten. Anschließend werden Konzepte vorgestellt, welche die Qualität der Alpha Matte in diesen Fällen anheben (Abschnitt 5.2). Für Szenen, in denen das bestehende Verfahren schon sehr gute Ergebnisse liefert, soll die Matte die bereits erreichte Qualität mindestens beibehalten. Abschnitt 5.3 fokussiert sich auf die Details der Implementierung.

5.1 Probleme des Occlusion Matting

Das in Kapitel 4 beschriebene Occlusion Matting Verfahren verwendet Informationen aus Farbbild und Tiefenbild, um die Bildregionen in Vorder- und Hintergrund einzuteilen. Die Tiefendaten dienen hier zur Unterstützung, um den unbekannten Bereich in der Trimap zu definieren, sodass sowohl die Farb- als auch die Tiefenkanten in diesem Bereich liegen. Allerdings werden bei der Propagation der bekannten Werte in den unbekannten Bereich sowie bei der anschließenden Auswahl des Sample-Paars ausschließlich die Farbinformationen der Aufnahme verwendet. Das macht es schwierig für den Algorithmus, Bereiche, in denen der Farbunterschied zwischen Vorder- und Hintergrund sehr niedrig ist, korrekt zu behandeln. In diesen Fällen kommt es oft vor, dass Teile des Vordergunds fälschlicherweise als Hintergrund deklariert werden und umgekehrt. Des Weiteren wird an diesen Stellen meist ein halbtransparenter Alpha-Wert geschätzt, welcher weder eine klare Zugehörigkeit zu Vorder- noch zu Hintergrund darstellt und für eine ungenaue Verdeckung des virtuellen Objekts sorgt.

Abbildung 11 zeigt ein Matting-Ergebnis, in welchem Artefakte durch die falsche Behandlung der Vorder- und Hintergrundfarben auftreten. Hier ist eine Szene abgebildet, in welcher Vorder- und Hintergrund sehr ähnlich zueinander sind. In diese Szene wird nun ein einfarbiges Rechteck als virtuelles Objekt eingefügt. In der Alpha Matte in Abbildung 11b lässt sich erkennen, an welchen Stellen das Occlusion Matting besondere Probleme hat, die beiden Ebenen voneinander zu unterscheiden. Die Artefakte äußern sich hier dadurch, dass einzelne Pixel und teilweise auch kleinere Bereiche falsch in die Ebenen eingeteilt werden. Durch die falsche Einteilung von einzelnen Pixeln kann teilweise der Eindruck eines Rauschen entstehen (siehe oberer, rot markierter Bereich). Im unteren, markierten Bereich werden größere Regionen im Vordergrund als Hintergrund eingeordnet und umgekehrt. Dadurch werden Löcher im realen sowie im virtuellen Objekt sichtbar. Diese Fehler lassen sich auch im Ergebnisbild (Abbildung 11c) betrachten.

(a) (b) (c)

Abbildung 11: Problemdarstellung einer Szene mit ähnlichen Vorder- und Hintergrundfarben. (a): Das Eingabebild. (b): Die Alpha Matte. Bereiche, in denen Artefakte besonders sichtbar werden, sind in rot markiert. (c): Das Ergebnisbild mit virtuellem Objekt.

Ein weiteres Problem kann entstehen, wenn die Bilddaten verrauscht sind. Das äußert sich vor allem im Tiefenbild. Bei der Aufnahme einer Szene können im Tiefenbild Löcher entstehen und die Aufnahme kann sich von Frame zu Frame stark verändern, was sich durch Ungenauigkeiten bei der Messung der Tiefendaten erklären lässt. Dadurch, dass die Trimap sowie der Alpha-Wert für jeden Frame neu berechnet werden, kann so ein leichtes Rauschen in der Alpha Matte entstehen. Dieses kann sich auch dann ereignen, wenn keine Bewegung in Vorder- oder Hintergrund stattfindet und stellt daher einen zu reduzierenden Fehler dar.

Schwerwiegender als das Rauschen durch Tiefendaten ist ein "Flimmern", was in der Alpha Matte in Haarbereichen entstehen kann. Dies hängt mit der adaptiven Dilatation zusammen, anhand welcher die Trimap erstellt wird. Mithilfe der adaptiven Dilatation wird der unbekannte Bereich der Trimap in solchen Gegenden erweitert, bei denen keine deutlichen Kanten erkennbar sind. Dies kann besonders in solchen Bereichen auftreten, an denen sich z.B. Haare des Vordergrundobjekts befinden oder viele Ecken anzutreffen sind. Das wird anhand der Anzahl an markierten Punkten in der Nachbarschaft jedes unbekannten Pixels festgelegt. Wurde für die Mehrzahl dieser Punkte ermittelt, dass sich dort keine Kante befindet, wird die Dilatation in diesem Bereich verstärkt. Das Problem hierbei ist jedoch, dass die adaptive Dilatation von Frame zu Frame stark schwanken kann. Das wird in Abbildung 12 deutlich. Die Beispielbilder zeigen ein Vordergrundobjekt, welches zu einem großen Teil aus Haaren besteht. Dadurch wird in fast allen Bereichen die Trimap mittels adaptiver Dilatation erweitert. Das Ergebnis dieser Dilatation schwankt jedoch von Frame zu Frame und Details an den Haaren können somit in manchen Frames verloren gehen. So kann ein "Flimmern"des Vordergrundobjekts entstehen, welches den visuellen Eindruck des Occlusion Matting vermindert.

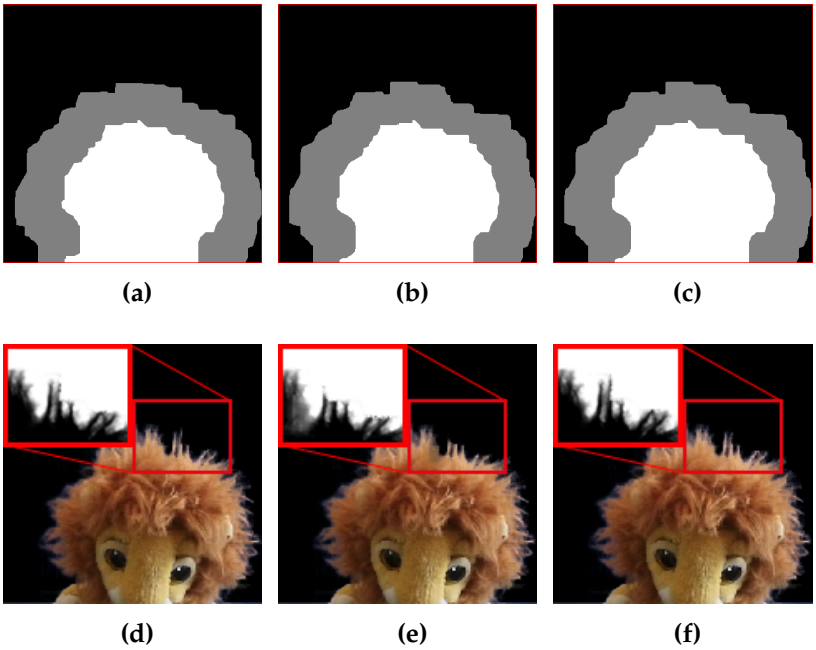


Abbildung 12: (a)-(c): Trimaps drei aufeinanderfolgender Frames, erstellt durch adaptive Dilatation. (d)-(f): Der auf Basis dieser Trimaps extrahierte Vordergrund. Für die rot markierte Region, in der Fehler auftreten, ist oben links der Bereich aus der Alpha Matte vergrößert dargestellt.

Bei dem bestehenden Sampling-Verfahren ist die Qualität der Matte zudem stark davon abhängig, wie groß das Suchfenster ist und welches Sample-Paar endgültig zum Schätzen des Alpha-Werts verwendet wird. Durch ein zu klein gewähltes Fenster kann es vorkommen, dass gute Samples verpasst werden. Zudem hängt der letztendlich ermittelte Alpha-Wert lediglich von einem Sample-Paar ab. Es kann durchaus vorkommen, dass dieses Paar die Farben aus Vorder- oder Hintergrund nicht geeignet repräsentiert, aber trotzdem anderen, möglicherweise besser geeigneten Samples vorgezogen wird.

5.2 Konzept zur Verbesserung

Um die im vorherigen Abschnitt genannten Probleme des Occlusion Matting zu adressieren, müssen die Fälle, in denen das bisherige Verfahren versagt, genau betrachtet werden. Aus den Ursachen, welche zu den unerwünschten Artefakten führen, lassen sich dann Verbesserungsmöglichkeiten an dem bestehenden Matting-Vorgang ableiten. Daher werden im Folgenden drei Konzepte beschrieben, die in den Ablauf des bestehenden Matting-Verfahrens integriert werden und eine erhöhte Qualität der

Alpha Matte herbeiführen sollen. Dabei sind zwei Randbedingungen gegeben, die eingehalten werden müssen:

- Die einzuführenden Veränderungen am Ablauf des Matting-Algorithmus sollen bei Szenen, in welchen das bereits bestehende Verfahren sehr gute Ergebnisse liefert, in einer mindestens gleich hochwertigen Alpha Matte resultieren (und somit das bestehende Verfahren nicht verschlechtern);
- Das Matting soll weiterhin in Echtzeit ausgeführt werden, da eine Verwendung für AR-Anwendungen nur unter diesen Umständen möglich ist.

Um die zweite Randbedingung einzuhalten, muss auf zu komplexe Algorithmen verzichtet werden, damit die Berechnungszeit der Matte nicht zu stark erhöht wird. Auf dieser Basis werden drei Verbesserungsmöglichkeiten vorgeschlagen, welche keine der Randbedingungen verletzen sollen:

- Um ein Rauschen/Flimmern der Alpha Matte in stillen Szenen oder Szenen mit wenig Bewegung zu reduzieren, wird die Alpha Matte nur einmal in einem bestimmten Zeitrahmen wie gewohnt berechnet und in darauf folgenden Frames mittels Optical Flow aktualisiert;
- Das bestehende Verfahren zur Auswahl der Vorder- und Hintergrundsamples wird durch ein *Iteratives gewichtetes Window Sampling* erweitert, welches anstatt nur einem Sample-Paar mehrere lokal optimale Samples für Vorder- und Hintergrund findet und deren Einfluss auf die Alpha Matte anhand der Kostenfunktion gewichtet;
- Es wird ein Test der Tiefendaten durchgeführt, um die Alpha-Werte an den Stellen anzupassen, bei denen der Farbunterschied der propagierten Vordergrund- und Hintergrundwerte zu ähnlich ist.

Diese Verbesserungsvorschläge werden in den folgenden Unterabschnitten 5.2.1 bis 5.2.3 im Detail erläutert. Unter Kapitel 6 wird ermittelt, welchen Mehrwert die Verwendung der oben aufgelisteten Erweiterungen für die Qualität der Alpha Matte liefert.

5.2.1 Aktualisierung der Alpha Matte mittels Optical Flow

Ein Problem des in Abschnitt 4 und in [3] beschriebenen Occlusion Matting Verfahrens zeigt sich dadurch, dass in stillen Szenen, bzw. Szenen mit sehr wenig Bewegung, ein Rauschen bzw. Flimmern in der Alpha Matte bemerkbar wird. Dieses entsteht, da in jedem Frame die Alpha Matte inklusive Trimap von Grund auf neu berechnet wird. In die Berechnung der Trimap fließen auch Informationen aus Tiefendaten mit ein, welche jedoch

verrauscht sind. Zudem kann, wie bereits unter 5.1 beschrieben, aufgrund der adaptiven Dilatation ein störendes Flimmern in Bereichen ohne klare Kanten entstehen.

Um dies zu verhindern, wäre eine simple Idee, die Trimap lediglich einmal in einem bestimmten Zeitraum zu berechnen und in den dazwischen liegenden Frames nur die Alpha Matte zu aktualisieren. Dieser Ansatz würde allerdings schon bei kleinen Bewegungen zu Fehlern führen, da z.B. das Vordergrundobjekt den unbekannten Bereich der Trimap verlassen könnte und somit als definitiver Hintergrund gewertet werden würde.

Ein besserer Ansatz ist es, die vollständige Berechnung der Alpha Matte nur zu bestimmten Frames durchzuführen. Zwischen diesen Frames kann dann lediglich eine Aktualisierung der Matte durch die Berechnung des Optical Flow zwischen aktuellem und vorherigem Frame durchgeführt werden. Dabei wird für jedes Pixel $\vec{p_t} = (x, y)$ im Bereich des virtuellen Objekts der Alpha-Wert aus der Matte herausgelesen und anhand des berechneten Optical Flow $\vec{f_p} = (u, v)$ in die entsprechende Richtung verschoben, sodass gilt:

$$\alpha(\vec{p}_{t+1}) = \alpha(\vec{p_t} + \vec{f_p}) \tag{7}$$

Hierbei ist $\alpha(\vec{p_t})$ eine Funktion, die den Alpha-Wert aus der Matte an Pixel $\vec{p}$ zum Zeitpunkt t zurückgibt. Der Alpha-Wert an Pixel $\vec{p}_{t+1}$ entspricht also dem Alpha-Wert an Pixel $\vec{p_t}$, welcher anhand des Verschiebungsvektors $\vec{f_p}$ verschoben wird. Die auf diese Weise aktualisierte Alpha Matte wird im Folgenden als *Flow Matte* bezeichnet. Eine Berechnung der Flow Matte erspart die Erstellung einer Trimap sowie eine Propagation der Farbwerte und die Identifizierung eines passenden Sample-Paares. Stattdessen werden lediglich die Alpha Matte des vorherigen Frames sowie die Farbbilder des aktuellen und vorherigen Frames als Eingabe benötigt.

Die Flow Matte kann auf folgende Weise in den bestehenden Algorithmus integriert werden: In jedem i-ten Frame wird wie gewohnt eine Alpha Matte berechnet, inklusive Berechnung der Trimap und Propagation der Farbwerte. Im darauffolgenden Frame kann dann anhand der Alpha Matte des letzten Frames eine Flow Matte berechnet werden. Anschließend wird die neue Flow Matte nur noch anhand der Flow Matte des vorherigen Frames bestimmt, bis wieder der i-te Frame erreicht und eine Neuberechnung der Alpha Matte durchgeführt wird. Die regelmäßige Neuberechnung verhindert, dass Fehler beim Bilden der Flow Matte zu stark in die weiteren Frames propagieren. Ist z.B. bereits die Alpha Matte fehlerhaft, wird dieser Fehler durch die Flow Matte nur weiter übernommen und gegebenenfalls verstärkt. Das Neuberechnungsintervall der Alpha Matte sollte daher nicht zu hoch gewählt werden.

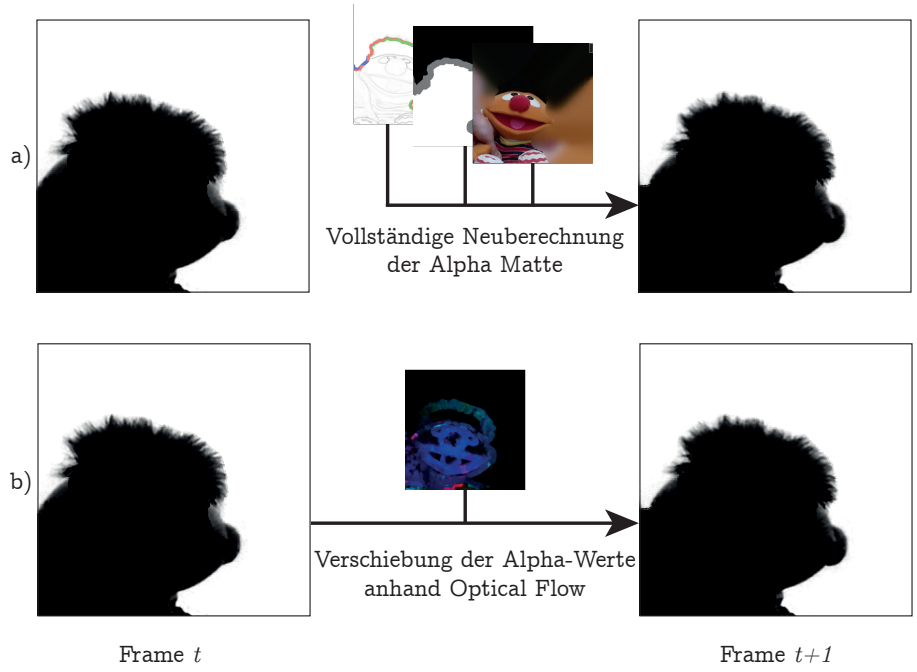


Abbildung 13: Konzept der Flow Matte. Bildreihe (a): Das in Abschnitt 4 beschriebene Occlusion Matting. Bildreihe (b): Update der Alpha Matte aus den temporalen Informationen des Optical Flows.

Das Konzept zur Verwendung einer Flow Matte ist in Abbildung 13 grafisch dargestellt. In Bildreihe (a) wird zuerst für Frame t eine Alpha Matte berechnet. Die Matte des nächsten Frames $t + 1$ bezieht keine Daten aus vorherigen Frames in die Berechnung mit ein, sondern verwendet nur die Bilddaten aus diesem Frame. Hierfür wird erneut aus Farb- und Tiefendaten eine Trimap erstellt, die Farbwerte propagiert und anschließend für die unbekannten Bereiche eine Alpha Matte geschätzt. Bildreihe (b) zeigt den Ablauf bei Verwendung einer Flow Matte. Zur Berechnung der Matte an Frame $t + 1$ wird lediglich die Alpha Matte aus Frame t und der Optical Flow, der auf Basis beider Frames berechnet wurde, benötigt. Nun kann in der Optical Flow-Textur nachgesehen werden, wie weit sich der Alpha-Wert jedes Pixels aus dem letzten Frame verschoben hat. Dieser Wert wird anhand des Verschiebungsvektors bewegt und in die neue Matte aus Frame $t + 1$ eingetragen werden. So entsteht durch Einbeziehung der temporalen Informationen eine Flow Matte.

Bei zu starken Bewegungen des Vorder- oder Hintergrundobjekts kann es zu Problemen kommen, da die Berechnung des Optical Flows für solche Szenarien zu ungenau ist. Das liegt daran, dass der Optical Flow voraussetzt, dass sich die Helligkeitswerte in zwei aufeinanderfolgenden Frames innerhalb einer gewissen Nachbarschaft befinden (siehe Abschnitt

2.4). Daher wird zusätzlich zur Neuberechnung der Alpha Matte alle i Frames überprüft, ob starke Verschiebungen der Pixel bei der Berechnung des Optical Flows bemerkbar sind. Das kann anhand der Stärke des Verschiebungsvektors $\vec{f_p}$ für jedes Pixel überprüft werden. Finden in diesem Frame zu starke Verschiebungen der Pixel statt, wird die Alpha Matte neu berechnet. Auch dies beugt dem Fall vor, dass Fehler aus einer ungenau berechneten Alpha Matte in den folgenden Frames noch weiter verstärkt werden.

5.2.2 Iteratives gewichtetes Window Sampling

In Abschnitt 4 wurde erläutert, wie die Auswahl eines Sample-Paares zur anschließenden Bestimmung des Alpha-Wertes in [3] durchgeführt wird. Um das aktuell betrachtete unbekannte Pixel wird ein Suchfenster für Vordergrund- und Hintergrundsamples erstellt und alle daraus entstehenden Sample-Kombinationen werden anhand einer Kostenfunktion verglichen. Das Sample-Paar, für welches die Kostenfunktion den minimalen Wert liefert, wird dann als finales Paar gewählt und anhand diesem der Alpha-Wert geschätzt. Allerdings können auf diese Weise gute Samples, die außerhalb des Suchfensters liegen, verpasst werden. Zudem ist die Qualität der Alpha-Matte von einem einzigen Sample-Paar abhängig, welches jedoch nicht immer das Paar sein muss, was die Farben aus Vorder- oder Hintergrund am besten repräsentiert. Wenn stattdessen auch andere, gut geeignete Sample-Paare in die Berechnung des Alpha-Werts einbezogen werden, kann mit größerer Sicherheit angenommen werden, dass der geschätzte Alpha-Wert zutreffend ist.

Die angesprochene Problematik beim bestehenden Sampling-Verfahren ist Grundlage für das hier vorgestellte iterative gewichtete Window Sampling. Der Grundgedanke ist hier, dass in der Nähe bereits gefundener, geeigneter Sample-Paare möglicherweise weitere gute Sample-Paare liegen, die dann ebenfalls in die Berechnung der Alpha Matte eingehen können. Dies soll zum einen sicherstellen, dass keine besser geeigneten Samples verpasst werden, und andererseits dafür sorgen, dass sich nicht nur auf ein einziges Sample-Paar verlassen werden muss. Die Idee, den berechneten Alpha-Wert nicht nur von je einem Vordergrund- und Hintergrundsample abhängig zu machen, wurde bereits von Wang und Cohen [20] aufgefasst.

Das iterative gewichtete Window Sampling baut auf dem bereits bestehenden Sampling Verfahren auf. Es wird zuerst weiterhin ein Fenster der Größe $n \times n$ für jeweils Vorder- und Hintergrundpixel erstellt. In diesem werden die propagierten Farbwerte nach geeigneten Samples durchsucht. Anstatt nun aber das Paar mit der günstigsten Kostenfunktion zu wählen und das Sampling zu beenden, wird nun ein weiteres Suchfenster der selben Größe jeweils um das bereits gefundene Vorder- und Hintergrundsample gelegt. Anschließend wird in den neuen Fenstern erneut nach dem besten Sample-Paar in diesem Bereich gesucht. Dieses

Schema wird für eine festgelegte Anzahl Iterationen wiederholt. Das während jeder dieser Iterationen gefundene, lokal optimale Sample-Paar wird zur späteren Verwendung abgespeichert. Der Algorithmus kann frühzeitig abgebrochen werden, sobald während einer Iteration kein besseres Sample-Paar als das bereits bestehende gefunden wurde.

Wurden alle Iterationen durchgeführt, werden die gefundenen Sample-Paare gemeinsam zur Schätzung des Alpha-Werts benutzt. Dabei werden die Sample-Paare anhand des Ergebnisses ihrer Kostenfunktion gewichtet, sodass Samples mit niedrigen Kosten stärker berücksichtigt werden. Daraus resultierend wird der finale Alpha-Wert des Pixels wie folgt berechnet:

$$\alpha_{total} = \sum_{i=1}^{N} \alpha(I, F_i, B_i) \cdot w_i \tag{8}$$

Hierbei steht N für die Anzahl der durchgeführten Iterationen, $\alpha(I, F_i, B_i)$ für den Alpha-Wert des i-ten Sample-Paars F_i, B_i und w_i für dessen Gewichtung. Diese Gewichtung wird anhand des Ergebnisses der Kostenfunktionen aller verwendeten Sample-Paare wie folgt bestimmt:

$$w_i = \frac{(1 - C_i)}{\sum_{k=1}^{N}(1 - C_k)} \tag{9}$$

Dies resultiert in einer höheren Gewichtung für Sample-Paare, die ein niedriges Ergebnis der Kostenfunktion aufweisen. Die Gewichtung wird anhand der Summe der Kosten aller gewählten Samples normiert, sodass die Summe aller Gewichte 1 ergibt. Die Kostenfunktion wird dabei weiterhin wie unter Abschnitt 4 und in [3] beschrieben ermittelt.

5.2.3 Einbindung der Tiefendaten für Alpha Matting

Szenen, in denen Vordergrund und Hintergrund ähnliche Farben besitzen, stellen viele bekannte Segmentierungs- und Matting-Verfahren vor Herausforderungen. Da die meisten dieser Ansätze lediglich auf dem Farbbild arbeiten, lässt sich hier eine Unterscheidung zwischen Vorder- und Hintergrund nur schwer eindeutig bestimmen. Das unter Abschnitt 4 und in [3] beschriebene Verfahren verwendet daher Tiefendaten in Kombination mit den Farbbildern, um darauf aufbauend eine Trimap zu erstellen, die Tiefenkanten und Farbkanten in ihrem unbekannten Bereich umfasst. Allerdings werden für das spätere Schätzen des Alpha-Werts lediglich das Farbbild bzw. propagierte Werte aus dem Farbbild in die Berechnung inkludiert. Die Tiefendaten werden hier jedoch vollständig ignoriert. Ein Ansatz, die Tiefendaten in die Berechnung der Alpha-Werte einzubringen, wurde bereits von Cho et al. [26] vorgeschlagen. Im Fall, dass eine starke Ähnlichkeit der Farbwerte zu erkennen ist, verwenden sie anstatt eines Farbsamples den jeweiligen Tiefenwert zur Berechnung der finalen Alpha Matte. Während

zwar aufgrund des Rauschens der Tiefendaten nicht zu viel Wert auf deren Genauigkeit gelegt werden kann, lassen sich dennoch wertvolle Informationen aus diesen gewinnen und in die finale Berechnung des Alpha-Werts mit einbeziehen.

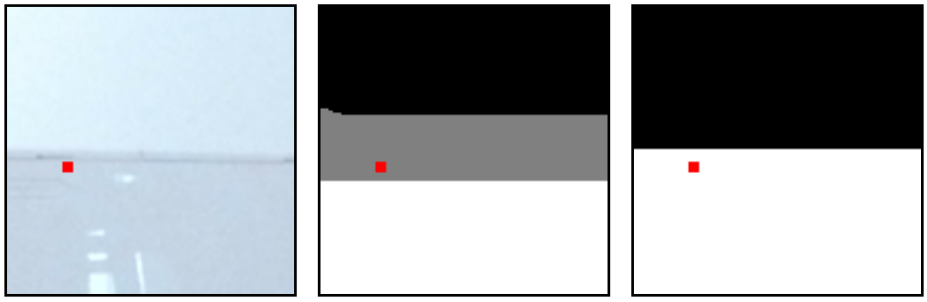
Hierzu wird der folgende grundlegende Ansatz in das bestehende Sampling-Verfahren integriert: Es wird davon ausgegangen, dass Vorder- und Hintergrundsamples, welche eine sehr hohe Ähnlichkeit in ihrem Farbwert aufweisen, keine geeigneten Sample-Paare sein können. Wird ein solches Sample-Paar gewählt und für dieses der Alpha-Wert geschätzt, ist die Wahrscheinlichkeit zu hoch, dass der resultierende Wert fehlerhaft ist. Daher wird ein solcher Fall gesondert behandelt. Lassen sich Vorder- und Hintergrundfarbe jedoch deutlich genug voneinander unterscheiden, wird die Berechnung des Alpha-Werts wie gewohnt fortgeführt.

Um zu messen, wie stark sich Vorder- und Hintergrundfarbe voneinander unterscheiden, wird ein Ähnlichkeitsmaß benötigt, an welchem festgelegt werden kann, ab wann die Pixel als Sample nicht mehr geeignet sind. Als solches wird die euklidische Distanz der beiden Farben im RGB-Farbraum gewählt. Diese ist für zwei Pixel $P_1 = (R_1, G_1, B_1)$ und $P_2 = (R_2, G_2, B_2)$ wie folgt definiert:

$$D(P_1, P_2) = \sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2} \tag{10}$$

Ob ein Sample ungeeignet ist, wird nun anhand eines Schwellwerts entschieden, der auf die RGB-Farbdistanz angewendet wird. Für den Fall, dass das Sample-Paar tatsächlich ungeeignet ist, werden nun Informationen aus den Tiefendaten entzogen. Hierfür kann die bereits zum Berechnen der Trimap erstellte Bimap verwendet werden, welche auf Basis des Tiefenbilds erstellt wurde. Das wird in Abbildung 14 visualisiert. Sobald die Farbähnlichkeit zu groß ist, wird der Wert in der Bimap überprüft, der sich am aktuell zu schätzenden unbekannten Pixel befindet. Hierbei können aus der Bimap nur zwei mögliche Werte ausgelesen werden: Vordergrund ($\alpha = 1$) oder Hintergrund ($\alpha = 0$). Diesem Alpha-Wert wird nun ein hoher Wert für die Kostenfunktion zugewiesen und mit anderen Samples verglichen. Das soll den Einfluss des Bimap-Werts reduzieren, da dieser aufgrund der Tiefendaten verrauscht und ungenau sein kann. Findet sich unter den durchsuchten Sample-Kandidaten kein Paar mit günstigerer Kostenfunktion (oder keine Paare, bei denen die Farbdistanz groß genug ist), geht der aus der Bimap entnommene Wert in die weitere Berechnung der Alpha-Matte ein.

Da sich Abschnitt 5.2.2 bereits mit einer verbesserten Version des Alpha-Sampling beschäftigt und das hier vorgeschlagene Konzept ebenfalls zur Verbesserung der Sampling-Strategie dient, werden beide Ansätze miteinander kombiniert und unter Abschnitt 6 gemeinsam evaluiert.



(a) (b) (c)

Abbildung 14: Konzept zur Einbindung der Tiefendaten. Im Eingabebild (a) weisen Vorder- und Hintergrund ähnliche Farben auf. Für das rot markierte Pixel in der Trimap (b) besitzt das bisherige Sampling-Verfahren Schwierigkeiten, den Alpha-Wert korrekt zu bestimmen. Daher wird an diesem Pixel in der Bimap (c) nachgesehen und anhand dem dort befindlichen Wert Alpha ermittelt.

5.3 Realisierung

Während Abschnitt 5.2 ein Konzept zur Verbesserung des bestehenden Matting-Verfahrens beinhaltet, wird in diesem Kapitel die genaue Umsetzung der Vorschläge erläutert. Für diese Arbeit wurde das AR-Framework *CVARK* verwendet, welches innerhalb der Universität Koblenz-Landau entwickelt wurde. Das Framework basiert auf Funktionalitäten aus *OpenGL* und *OpenCV*. Der OpenGL-Anteil ist unter anderem für das Rendern der virtuellen Objekte zuständig. Außerdem übernimmt er zusätzlich einen Teil der Bildverarbeitungsaufgaben. Dazu zählen z.B. die Suche nach Pixeln in einer Nachbarschaft, die Berechnung der Dilatation im unbekannten Bereich oder die Anwendung verschiedener Filter (insbesondere der Gauß-Filter zum Weichzeichnen sowie der Sobel-Filter zur Kantenerkennung). Da diese Aufgaben mittels OpenGL an Shader übergeben und somit auf der GPU berechnet werden, besitzt das Verfahren einen hohen Grad an Parallelität. Dies macht es überhaupt erst möglich, dieses Matting-Verfahren unter Echtzeitbedingungen laufen zu lassen. Der OpenCV-Anteil ist für komplexere Bildverarbeitungsaufgaben zuständig und wird größtenteils auf der CPU berechnet. Hier wäre eine Umsetzung der Algorithmen auf der GPU aber ebenfalls möglich. Dies kann z.B. durch die Kombination von OpenCV mit *CUDA* realisiert werden. Während dadurch garantiert eine Beschleunigung des Verfahrens erreicht werden kann, ist eine Verbesserung der Qualität jedoch nicht zu erwarten. Zudem ist das Verfahren bereits mit den CPU-Varianten der OpenCV-Algorithmen in Echtzeit ausführbar und eine Beschleunigung daher nicht unbedingt notwendig.

Im Folgenden wird im Detail die Implementierung der Verbesserungsmöglichkeiten genannt, die in den Abschnitten 5.2.1 bis 5.2.3 eingeführt wurden.

5.3.1 Einbindung der Flow Matte

Wie bereits in Abschnitt 5.2.1 beschrieben, wird die Alpha Matte nicht mehr jeden Frame von Grund auf neu berechnet. Stattdessen wird zwischendurch mithilfe des Optical Flows eine Flow Matte erstellt, welche anstatt der bestehenden Alpha Matte verwendet werden kann. Abbildung 15 zeigt den neuen Ablauf der Alpha-Updates mithilfe der Flow Matte.

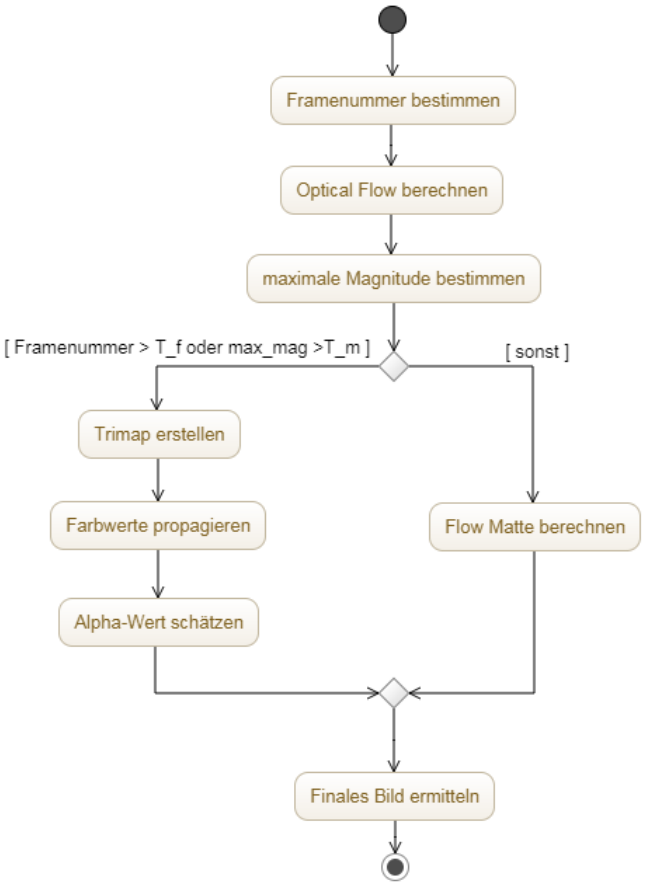


Abbildung 15: Aktivitätsdiagramm, welches die Einbindung der Flow Matte schildert. Diese Aktionen werden in jedem Frame ausgeführt.

In jedem Frame wird ermittelt, wie lange es her ist, dass zum letzten Mal eine vollständige Berechnung der Alpha Matte stattgefunden hat. Dies wird anhand einer Variablen angegeben, die jeden Frame inkrementiert

und zurückgesetzt wird, sobald die Alpha Matte neu berechnet wurde. Sie wird in der Abbildung als Framenummer bezeichnet.

Anschließend wird eine Berechnung des Optical Flows durchgeführt. Hierfür wird eine in OpenCV enthaltene Implementation des Dense Optical Flow verwendet, welche den Optical Flow für jedes Pixel des Eingabebereichs ermittelt. Dieses Verfahren basiert auf dem Verfahren von Gunnar Farnebäck [15]. Vor der Berechnung muss das Farbbild des aktuellen und vorherigen Frames dazu erst in ein Grauwertbild umgewandelt werden. Zudem wird der Bildbereich, für den der Flow berechnet wird, auf einen Bereich innerhalb einer Bounding Box eingeschränkt (Abbildung 16). Diese Bounding Box wird um das/die virtuelle/n Objekt/e gelegt. Eine Berechnung des Flows für das gesamte Bild hat in Testläufen für einen zu starken Einbruch der Framerate gesorgt.

Sobald der Flow für jedes Pixel innerhalb der Bounding Box ermittelt wurde, wird geprüft, wie hoch die insgesamt stärkste Verschiebung im Flow-Bild ist. Diese wird als maximale Magnitude abgespeichert. Nun wird entschieden, ob für den neuen Frame eine Alpha Matte oder eine Flow Matte verwendet werden soll. Die Alpha Matte wird in zwei Fällen verwendet: Im ersten überschreitet die Anzahl der Frames, in welchen bereits aufeinanderfolgend die Flow Matte verwendet wurde, einen gewissen Schwellwert (in Abbildung 15 als T_f bezeichnet). Eine Neuberechnung der Alpha Matte soll dann verhindern, dass Fehler aus den vorherigen Frames zu stark propagiert werden. Der zweite Fall tritt ein, wenn die Stärke des Verschiebungsvektors einen vorab festgelegten Schwellwert T_m überschreitet. In beiden Fällen wird eine Neuberechnung der Alpha Matte auf dem aus Kapitel 4 bekannten Weg durchgeführt. Treten beide Bedingungen nicht ein, wird eine Flow Matte für den aktuellen Frame berechnet.

(a) **(b)**

Abbildung 16: (a) Das Farbbild aus dem aktuellen Frame. (b) Der aus vorherigem und aktuellem Bild bestimmte Optical Flow, berechnet innerhalb einer gewünschten Bounding Box. Die Farbe gibt die Richtung, die Helligkeit die Stärke des Verschiebungsvektors an.

Zur genauen Berechnung der Flow Matte muss ermittelt werden, wie stark der Alpha Wert jedes Pixels verschoben werden muss. Dazu wird ein Shader-Programm kreiert, welches die im vorherigen Frame verwendete Matte als Input erhält. Dies kann ebenfalls entweder eine Alpha Matte oder eine Flow Matte sein, je nachdem, welche Entscheidung im vorherigen Frame getroffen wurde. Zusätzlich wird der Optical Flow als Textur übergeben. Pixel, welche sich nicht im Bereich der Bounding Box befinden und somit nicht relevant sind, werden im Vorhinein von der Berechnung exkludiert. Anschließend werden die benötigten Werte aus den jeweiligen Texturen entnommen. Diese sind der Optical Flow am aktuellen Pixel sowie der Alpha-Wert am ursprünglichen Pixel, an welchem sich der Farbwert im vorigen Frame befand.

Hierbei muss beachtet werden, dass sich in der Flow-Textur der Verschiebungsvektor am aktuellen Pixel befindet anstatt an der Position, wo sich das Pixel im letzten Frame befand. Um den Alpha-Wert aus dem vorherigen Frame auszulesen, muss in der entgegengesetzten Richtung des Verschiebungsvektors nachgesehen werden. Die Berechnung erfolgt daher in umgekehrtem Wege als in Gleichung 7 beschrieben. Anstatt den Verschiebungsvektor auf das Pixel des letzten Frames zu addieren, wird das aktuelle Pixel ausgelesen und der Verschiebungsvektor von dessen Position subtrahiert, um an die Position des Alpha-Werts im vorherige Frame zu gelangen. Dieser Wert wird nun am aktuellen Pixel eingetragen. Dieser Vorgang wird in Algorithmus 1 in Form von Pseudocode beschrieben.

forall *Pixel i := (x,y) im Eingabebild I* **do**

Lese Optical Flow $f := (u, v)$ an Pixel i aus;

```
/* Springe an die Position des Pixels im
   vorherigen Frame                          */
```

$old\_i := i - f$;

Lese Alpha-Wert α an Stelle old_i aus ;

Schreibe α in die Flow Matte an Stelle i ;

end

Algorithmus 1: Ablauf der Erstellung einer Flow Matte.

Wird die Flow Matte auf diese Weise berechnet, können Artefakte an den Rändern des virtuellen Objekts entstehen. Dies kommt dadurch, dass Alpha-Werte von einem Bereich außerhalb des virtuellen Objekts in das virtuelle Objekt „gezogen“werden können. Hierdurch kann es passieren, dass Pixel am Rand des virtuellen Objekts von Hinter- in Vordergrund wechseln und das eigentlich unveränderte Objekt manipuliert wird. Um dies zu vermeiden, werden nur die Pixel verschoben, die im letzten Frame auch Teil des Vordergrunds waren. Dabei wird angenommen, dass sich die

Hintergrundobjekte nicht bewegen. Gehörte das aktuell betrachtete Pixel zum definitiven Hintergrund, wird die Verschiebung nicht durchgeführt und stattdessen weiterhin der Hintergrundwert übernommen. So wird sichergestellt, dass nur Alpha-Werte, bei denen tatsächlich eine Bewegung stattfindet, verschoben werden.

5.3.2 Anpassung des Sampling-Verfahrens

Um für eine bessere Auswahl der Sample-Paare zu sorgen, werden mehrere Änderungen am bestehenden Sample-Verfahren vorgenommen. Anstatt ein einzelnes Vordergrund- und Hintergrundsample auszuwählen, werden mehrere Samples in die Berechnung mit einbezogen. Diese werden in je einem Fenster für Vorder- und Hintergrund ausgewählt, welches sich abhängig von den ausgewählten Pixeln verschiebt. Falls sich nur ungünstige Sample-Paare in einem Fenster finden, werden Informationen aus den Tiefendaten zu Hilfe gezogen.

Ein Überblick über den Ablauf des iterativen gewichteten Window Samplings lässt sich in Abbildung 17 finden. Da der Algorithmus auf der GPU implementiert ist, können die Berechnungen für jedes Pixel in hoher Parallelität ausgeführt werden. Zuerst wird je ein Fenster um das aktuelle unbekannte Pixel gelegt. Eines der Fenster wählt die Vordergrundsamples, das Andere die Hintergrundsamples aus. Die Größe des Fensters wird vom Nutzer festgelegt und kann zur Laufzeit nach Bedarf verändert werden. Ebenfalls per Nutzereingabe wird die Anzahl der durchgeführten Iterationen bestimmt. In jeder Iteration wird ein Sample-Paar ausgewählt und später in die Berechnung des Alpha-Werts einbezogen. Eine höhere Anzahl Iterationen sorgt also für eine höhere Anzahl ausgewählter Paare, aber ebenso erhöht sich die Suchdistanz weiter vom betrachteten unbekannten Pixel. Das kann dafür sorgen, dass Farben in die Berechnung eingehen, die nicht den tatsächlichen Vorder- oder Hintergrundfarben entsprechen. Diese Faktoren müssen bei der Wahl der Parameter bedacht werden.

Nachdem die Suchfenster erstellt wurden, wird aus beiden Fenstern je ein Sample selektiert. Dabei werden nur solche Samples ausgewählt, welche tatsächlich zu Vorder- bzw. Hintergrund gehören oder in den unbekannten Bereich propagiert wurden. Auf Basis der ausgesuchten Sample-Paare wird anschließend ein initialer Alpha-Wert geschätzt. Dies geschieht folgendermaßen:

$$\alpha = \frac{(I - B) \cdot (F - B)}{||(F - B)||^2} \tag{11}$$

Hierbei bezeichnet I den tatsächlichen Farbwert des unbekannten Pixels, F den Farbwert des Vordergrundsamples und B den des Hintergrundsamples. Aus dem ermittelten Alpha-Wert werden anschließend die

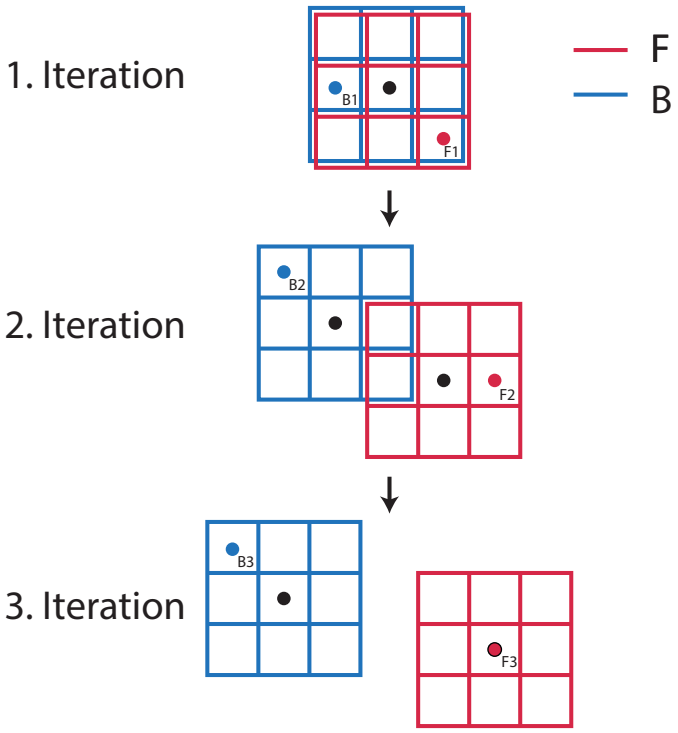


Abbildung 17: Ablauf des iterativen gewichteten Window Samplings. Das 3×3 Fenster für den Vordergrund ist rot, das für den Hintergrund blau markiert. F1-F3 sind die Vordergrundsamples, die in die Berechnung der Alpha Matte eingehen. B1-B3 sind die korrespondierenden Hintergrundsamples.

photometrischen Kosten C_{photo} bestimmt, welche Teil der gesamten Kostenfunktion sind. Diese werden analog zum ursprünglichen Verfahren mit der folgenden Gleichung bestimmt:

$$C_{\text{photo}}(F_i, B_j, I_p) = ||I_p - (\alpha F_i + (1 - \alpha)B_j)|| \tag{12}$$

Der andere Teil der Kostenfunktion besteht aus den Propagationskosten C_{propa}, welche durch

$$C_{\text{propa}}(F_i, B_j, I_p) = \frac{d(F_i) + d(B_j)}{D_n} \tag{13}$$

bestimmt werden. Hierbei steht D_n für die Anzahl der durchgeführten Diffusionen. $d(F_i)$ bzw. $d(B_j)$ gibt an, während der wievielten Iteration des Propagationsalgorithmus die gesamplete Vorder- bzw. Hintergrundfarbe entstanden ist. Die genaue Funktionsweise der Propagation kann unter Abschnitt 4 nachgelesen werden.

Diese beiden Kostenfunktionen werden anhand einer bestimmten Gewichtung w_{cost} in die finale Kostenberechnung mit einbezogen. Dies geschieht wie folgt:

$$C_{\text{total}} = w_{\text{cost}} \cdot C_{\text{photo}} + (1 - w_{\text{cost}}) \cdot C_{\text{propa}} \tag{14}$$

Sobald die Kostenfunktion ermittelt wurde, wird das nächste Sample-Paar aus den Fenstern ausgewählt und der beschriebene Prozess zur Kostenermittlung wiederholt. Insgesamt findet die Berechnung der Kostenfunktion für alle möglichen Sample-Paare also n^4-Mal statt, wobei n die Größe der quadratischen Sample-Fenster ist.

Die Kostenfunktion jedes Sample-Paars wird nun untereinander verglichen. Unter ihnen wird dasjenige Vordergrund- und Hintergrundpaar (F_k, B_k) ausgewählt, für das die Kostenfunktion minimal ist, also sodass gilt:

$$(F_k, B_k) = \underset{F_i, B_j}{\arg\min}\, C_{\text{total}}(F_i, B_j) \tag{15}$$

Hierbei steht k für die aktuell durchlaufene Iteration. Wurde ein solches Paar ermittelt, werden für dieses die folgenden Informationen zur weiteren Verwendung abgespeichert:

- Das Ergebnis der Kostenfunktion C_{total} für (F_k, B_k),
- der zugehörige geschätzte Alpha-Wert α_k und
- die Pixelposition von F_k und B_k.

F_k und B_k, die ausgewählten Samples aus Iteration k, sind in Abbildung 17 als roter bzw. blauer Punkt gekennzeichnet.

Im Anschluss werden die Vorder- und Hintergrundfenster so verschoben, dass ihr Zentrum das Pixel beinhaltet, welches als bestes Sample im vorherigen Fenster ausgewählt wurde (in Abbildung 17 schwarz markiert). Dies wird anhand der abgespeicherten Pixelposition bestimmt. Daraufhin wird die nächste Iteration der bisherigen Prozedur begonnen, ein neues Sample-Paar in den verschobenen Suchfenstern ausgewählt und dessen Informationen gespeichert. Nach Durchlauf aller Iterationen werden also insgesamt K Sample-Paare ermittelt. Da K auch die Anzahl der Iterationen definiert und innerhalb jeder Iteration n^4 Sample-Paare getestet werden, werden also insgesamt $n^4 \cdot K$ Tests durchgeführt. Hierbei ist zu beachten, dass es durchaus vorkommen kann, dass die gleichen Vorder- und Hintergrundpixel mehrmals verwendet werden, da das verschobene Fenster zum Teil weiterhin Pixel aus dem Originalfenster beinhaltet. Es ist durchaus vorstellbar, solche Pixel als bereits getestet zu markieren und die berechneten Werte zu α und C_{total} in einer Datenstruktur abzuspeichern. Sie können

dann beim nächsten Test direkt abgerufen werden. Während dies durchaus eine Beschleunigung des Algorithmus darstellt, ist auch ohne diese Optimierung weiterhin eine Echtzeitfähigkeit gewährleistet, solange die Fenstergröße n nicht zu hoch gewählt wird.

Wurden alle Sample-Paare bestimmt, wird für jedes der Paare eine Gewichtung festgelegt, welche beschließt, wie stark dessen geschätzter Alpha-Wert in das finale Alpha eingeht. Um den Wertebereich $[0, 1]$ von Alpha nicht zu verlassen, wird auf folgende Weise das Gewicht w_k berechnet:

$$w_k = \frac{(1 - C_{\text{total}}(F_k, B_k))}{\sum_{i=1}^{K}(1 - C_{\text{total}}(F_i, B_i))} \tag{16}$$

Der Divisor dient hierbei zur Normalisierung des Gewichts auf den gewünschten Wertebereich und sorgt dafür, dass

$$\sum_{k=1}^{K} w_k = 1 \tag{17}$$

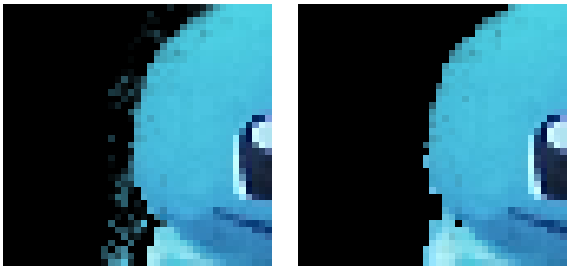
gilt. Darauf folgend müssen nur noch die berechneten Gewichtungen mit den korrespondierenden α_k verrechnet werden. Dies geschieht mithilfe folgender Formel:

$$\alpha_{\text{total}} = \sum_{k=1}^{K} \alpha_k w_k \tag{18}$$

Dieser Alpha-Wert wird nun letztendlich als finales Ergebnis in die Alpha Matte eingetragen.

Bei der Wahl eines Sample-Paares kann der Fall auftreten, dass Vorder- und Hintergrundsample einen sehr ähnlichen Farbwert besitzen. Da solche Fälle bisher zu Fehlern in der resultierenden Alpha Matte geführt haben, müssen sie gesondert behandelt werden. Dies wird auf folgende Weise getan:

Bei der Auswertung jedes Sample-Paars wird zuerst die Farbähnlichkeit der Pixel überprüft. Dies wird anhand der euklidischen Distanz im RGB-Farbraum ermittelt (Gleichung 10). Findet sich ein Paar, für das diese Distanz zu niedrig ist, werden die Tiefeninformationen am aktuell betrachteten unbekannten Pixel mit einbezogen. Dazu wird die bereits vorab erstellte Bimap zu Hilfe gezogen. In dieser ist für alle Pixel im relevanten Bereich festgelegt, ob diese anhand der reinen Tiefendaten zu Vorder- oder Hintergrund gehören. Dieser Wert wird am unbekannten Pixel ausgelesen und anhand diesem ein festgelegter, eindeutiger Alpha-Wert bestimmt ($\alpha = 1$ für Vordergrund, $\alpha = 0$ für Hintergrund). Der gewählte Alpha-Wert wird nun behandelt, als wäre er als Resultat der Alpha-Schätzung entstanden. Damit die rohen Tiefendaten allerdings nicht zu stark in die



(a) **(b)**

Abbildung 18: Ausschnitt aus dem zusammengesetzten Bild (a) ohne Elimination der Ausreißer, (b) Mit Elimination.

weitere Berechnung eingehen, da sie dafür zu ungenau sind, wird ihnen für die Kostenfunktion ein hoher Wert zugewiesen. Anschließend wird die Berechnung wie bereits beschrieben durchgeführt. Da dem Bimap-basierten Alpha-Wert ein hoher Wert in der Kostenfunktion zugewiesen wurde, wird das zugehörige Sample nur dann als lokal optimales Sample-Paar (F_k, B_k) ausgewählt, wenn folgende Fälle eintreten:

- Alle anderen Sample-Paare, die in den Suchfenstern gefunden wurden, besitzen höhere Kosten als die für das Bimap-Pixel festgelegten Kosten;
- Es befinden sich ausschließlich Samples innerhalb der Fenster, deren Farben sich nicht signifikant voneinander unterscheiden.

In manchen Szenen kann es passieren, dass Pixel in kleineren Regionen halbtransparente Alpha-Werte erhalten, während der umliegende Bereich klar einer Ebene zugeordnet wird. Dies geschieht ebenfalls besonders in Bereichen mit ähnlichen Vorder- und Hintergrundfarben. Dadurch entstehen unerwünschte Artefakte ähnlich einem Rauschen. Um solche Artefakte zu vermeiden, wird eine kleine Nachbarschaft um jedes solcher Pixel untersucht. Dabei wird für jedes Nachbarschaftspixel die absolute Differenz zwischen dessen Alpha-Wert und dem des zentralen Pixels gemessen. Ist die Summe dieser Differenzen zu hoch, wird der Alpha-Wert nach oben bzw. unten angepasst und somit eliminiert. Liegt der Alpha-Wert näher am echten Hintergrund als am echten Vordergrund (somit gilt $\alpha < 0.5$), wird das Pixel noch stärker dem Hintergrund zugeordnet und Alpha somit verringert. Der umgekehrte Fall gilt für Pixel, die näher am echten Vordergrund liegen. Abbildung 18 zeigt ein Beispiel, in welchem solche Ausreißerpixel eliminiert werden.

6 Evaluation

Mehrere Testdaten wurden erhoben, um zu ermitteln, ob das in Abschnitt 5 beschriebene Konzept zur Verbesserung auch tatsächlich die gewünschten Ergebnisse zeigt. Zusätzlich zur Qualität der Alpha Matte muss dazu auch überprüft werden, inwiefern die Implementation der Verbesserungen weiter echtzeitfähig bleibt. Dazu wird u.a. evaluiert, welche Parameter sich in welchem Maße auf die Performance auswirken.

Alle in dieser Arbeit verwendeten Testdaten wurden auf einem System, bestehend aus einem Intel i7-6700 CPU mit einer NVIDIA GeForce GTX TITAN X Grafikkarte und 32 GB RAM gemessen. Die Auflösung des Eingabebilds, das mit einer Microsoft Kinect aufgenommen wurde, beträgt 1920x1080 Pixel. Bis auf die Berechnung des Optical Flows werden alle intensiven Berechnungen auf der GPU durchgeführt. In die gemessenen Zeiten wurden die Dauer des Texturuploads, welche jedoch konstant bleibt, mit einbezogen.

Ein Faktor, der eine große Rolle bei der Performance des Algorithmus spielt, ist die Größe der Bounding Box, die um das virtuelle Objekt bzw. die virtuellen Objekte gelegt wird. Nur innerhalb dieser Bounding Box wird der Optical Flow berechnet, der zu Erstellung der Flow Matte dient. Der Dense Optical Flow-Algorithmus aus [15], der in dieser Arbeit verwendet wird, ist eine aufwändige Berechnung, die für jedes Pixel innerhalb der Bounding Box durchgeführt werden muss. Daher steigt der Rechenaufwand mit der Größe dieser Bounding Box stark an. Dies lässt sich in Tabelle 1 nachvollziehen. Nehmen die virtuellen Objekte einen zu großen Teil des Bilds ein, kann ein Einbruch in der Performance erkennbar werden. Daher sollte in solchen Fällen anstatt der Flow Matte auf eine Neuberechnung der Alpha Matte zurückgegriffen werden, da diese nicht von der Größe der virtuellen Objekte abhängt. Für kleine und mittelgroße Objekte ist die Berechnung des Optical Flows allerdings echtzeitfähig. Hierbei sollte angemerkt werden, dass der Optical Flow auf der CPU berechnet wird. Es ist zu erwarten, dass eine Umsetzung des Algorithmus auf der GPU den Rechenaufwand drastisch reduziert.

Größe der Bounding Box	**Zeit(ms)**
300x300	24
480x480	60
600x600	94
720x720	138

Tabelle 1: Gemessene Zeiten zur Berechnung des Optical Flows, in Abhängigkeit der Größe der Bounding Box. Die Zeiten ergeben sich aus einem Durchschnitt aus 500 Messungen.

Anzahl Iterationen	Größe des Suchfensters	Zeit(ms)
3	5x5	18
5	5x5	22
7	5x5	28
3	7x7	32
3	9x9	66

Tabelle 2: Einfluss der Parameter des iterativen gewichteten Window Sampling auf die Performance.

Die Parametereinstellungen können für die meisten Szenen konstant gewählt werden und müssen nur in bestimmten Fällen angepasst werden. Zur Erstellung der Trimap ist die Größe der fixen Dilatation auf 9, die Größe der adaptiven Dilatation auf 17 festgelegt. Für Szenen mit sehr großen Haarbereichen, wie die Simba-Testszene in Abbildung 19, sollte die adaptive Dilatation noch höher gewählt werden, um den gesamten Haarbereich abzudecken. Zur Propagation der Farbwerte werden standardmäßig 8 Iterationsschritte mit 5 Ebenen in der Bildpyramide durchgeführt. Das iterative gewichtete Window Sampling liefert bei 3 Iterationen und einer Fenstergröße von 5×5 Pixeln den besten Kompromiss zwischen Performance und Qualität der ausgewählten Sample-Paare. Die Verwendung von 3 Iterationen bedeutet in diesem Fall auch, dass je bis zu 3 Sample-Paare in die Berechnung der Alpha-Werte mit eingehen. In Tabelle 2 kann eingesehen werden, welche Auswirkungen die Anzahl der Iterationen sowie die Größe des Suchfensters auf die Performance haben. Dabei ist festzustellen, dass die Anzahl der Iterationen einen geringeren Einfluss auf den Rechenaufwand hat als die Fenstergröße. Die Größe des Suchfensters sollte daher nicht zu hoch gesetzt werden. Neben einer stark erhöhten Berechnungszeit führt ein zu großes Suchfenster auch dazu, dass sich die Distanz zum ursprünglichen Pixel immer weiter erhöht und zudem viele Sample-Paare unnötigerweise mehrmals überprüft werden. Mit den beschriebenen Standardeinstellungen beläuft sich die vollständige Berechnung der Alpha Matte pro Frame auf ca. 22ms, was für eine Anwendung im Echtzeitbereich ausreichend ist.

In Abbildung 19 ist ein Vergleich zwischen Alpha und Flow Matte abgebildet. Die erste Bildreihe zeigt die Alpha Mattes von drei aufeinanderfolgenden Frames. In diesen Testdaten besteht ein großer Bereich des Vordergrundobjekts aus Haaren, wodurch die adaptive Dilatation hier einen sehr großen unbekannten Bereich in der Trimap erstellt, um den ganzen Haarbereich abzudecken. Dies resultiert aber in einem Flimmern der Alpha-Werte, welches besonders im rot markierten Bereich erkennbar ist. In der Flow Matte (zweite Bildreihe) ist derselbe Bereich grün markiert. Dieser Bereich aus beiden Mattes ist in der dritten Bildreihe vergrößert

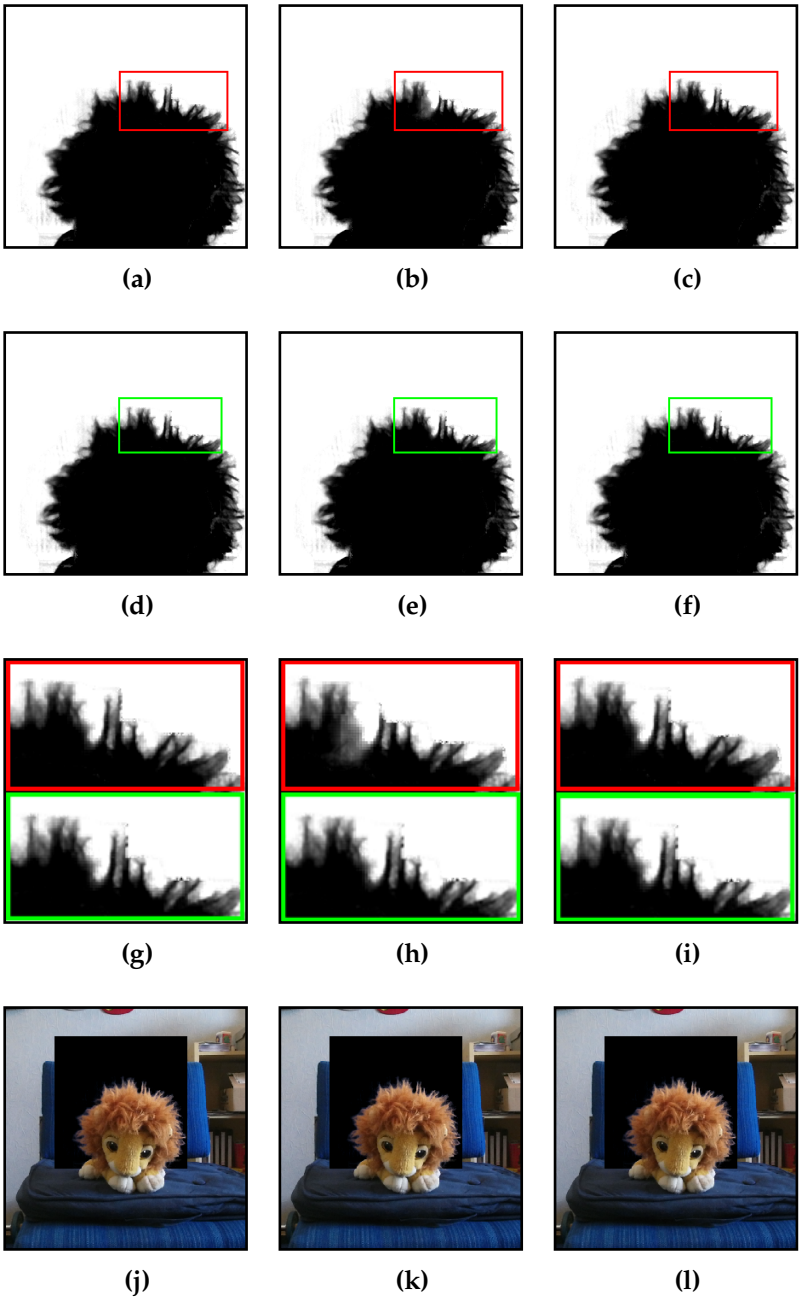


Abbildung 19: (a)-(c): Ergebnisse der Alpha Matte aufeinanderfolgender Frames. (d)-(f): Ergebnisse der Flow Matte aufeinanderfolgender Frames. (g)-(i): Die vergrößerten rot bzw. grün umrandeten Bereiche aus den vorigen Bildern im Vergleich. (j)-(l): Das mithilfe der Flow Matte zusammengesetzte Bild (das schwarze Rechteck ist das virtuelle Objekt).

dargestellt. Da sich das Vordergrundobjekt nicht bewegt, ist auch der Verschiebungsvektor des Optical Flow für jedes Pixel niedrig. Das resultiert in beinahe keiner Veränderung der Matte zwischen diesen Frames. Dementsprechend ist auch das zusammengesetzte Bild (letzte Bildreihe) für alle Frames konsistent.

Abbildung 20: Das Eingabebild. Die Hand soll nun ein virtuelles Objekt verdecken.

Ein Vergleich von ursprünglichem Sampling und dem neuen, iterativen gewichteten Window Sampling kann in Abbildung 21 betrachtet werden. Die Hand im Eingabebild (siehe Abbildung 20) verdeckt zum größten Teil Bereiche mit starkem farblichen Kontrast (blauer Hintergrund). Im oberen Teil des Bilds jedoch wird ein weißlicher Bereich verdeckt, welcher eine starke Farbähnlichkeit mit der hautfarbenen Hand aufweist. In Abbildung 21a lässt sich feststellen, dass die Bereiche mit starkem Farbunterschied vom ursprünglichen Algorithmus sehr genau in Vorder- und Hintergrund getrennt werden können. Hier entsteht also eine Alpha Matte von hoher Qualität. Im oberen Teil des virtuellen Objekts ist jedoch zu erkennen, dass die farbähnlichen Bereiche nur unter Bildung von Artefakten voneinander getrennt werden können. Hier werden Pixelregionen in die falsche Ebene unterteilt oder erhalten leichte Transparenz, da der Algorithmus nicht sicher eine Entscheidung zwischen Vorder- oder Hintergrund treffen kann. Unter Abbildung 21c findet sich dieselbe Szene mit der neuen Sampling-Variante. Die zugehörigen Alpha Mattes sind in Abbildung 21b und 21d abgebildet. Es lässt sich beobachten, dass sich viele der Artefakte beheben lassen und für viele Pixel nun eine klare Unterscheidung in Vorder- oder Hintergrund möglich geworden ist. Dies gilt jedoch nicht für alle Bereiche. Gerade im Bereich der Kante zwischen dem blauen und weißlichen Hintergrundbereich besitzt auch das neue Verfahren Probleme, eine korrekte Alpha-Schätzung durchzuführen. Dennoch lässt sich eine Verbesserung in der Genauigkeit der Verdeckung des virtuellen Objekts feststellen. Diese entsteht durch das eingeführte iterative gewichtete Window Sampling und anschließender Elimination der Ausreißerpixel.

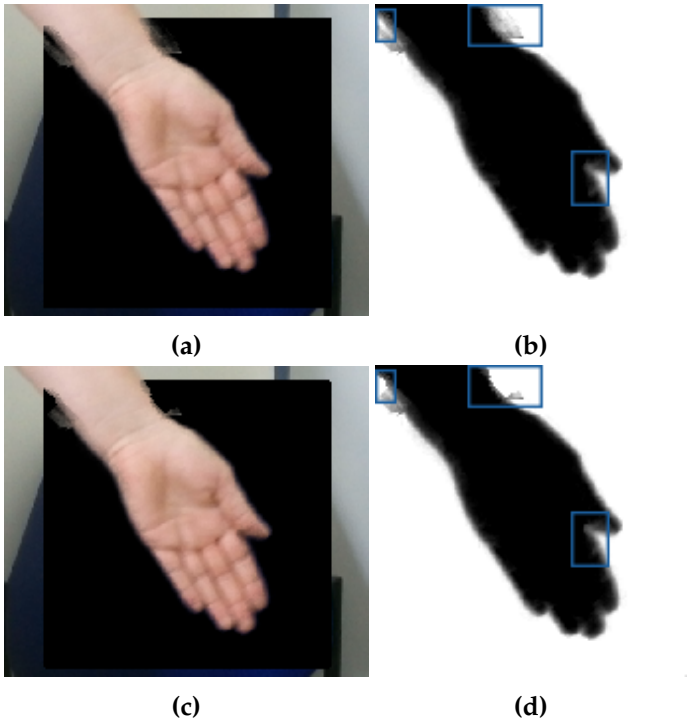


Abbildung 21: (a),(c): Das virtuelle Objekt (schwarzes Quadrat), in das Bild eingefügt mittels Alpha Matting. Ursprüngliche Version oben, neue Version unten. (b),(d): Die zugehörige Alpha Matte. Interessante Bereiche sind blau markiert.

Abbildung 22 verdeutlich ebenfalls, dass eine Verbesserung der Alpha Matte durch die Anpassung der Sample-Auswahl erreicht werden kann. Dort befindet sich eine Szene, in der Vorder- und Hintergrund einen sehr geringen Farbunterschied aufweisen. In diese Szene wird nun eine virtuelle Kreatur eingefügt. Das Matting wird hier sowohl ohne Verbesserungen (obere Bildreihe) als auch mit Verbesserungen (untere Bildreihe) durchgeführt. Besonders im rot markierten Bereich lassen sich hier starke Unterschiede zwischen den beiden Sampling-Verfahren ausmachen. Diese können dadurch erklärt werden, dass gerade solche Fälle mit starker Farbähnlichkeit gesondert behandelt werden. In einem solchen Fall wird zum einen der Tiefenwert des unbekannten Pixels stärker berücksichtigt, welcher die Pixel im rot markierten Bereich eindeutig als Vordergrund definiert. Zum anderen gehen mehr Samples in die Berechnung des Alpha-Werts mit ein, was die Chance vermindert, dass ein schlecht gewähltes Sample die Qualität der Alpha Matte verringert. Zuletzt spielt auch die Elimination einzelner Ausreißerpixel eine tragende Rolle in der verbesserten Qualität. Dennoch sind hier ebenfalls noch einzelne falsch eingeordne-

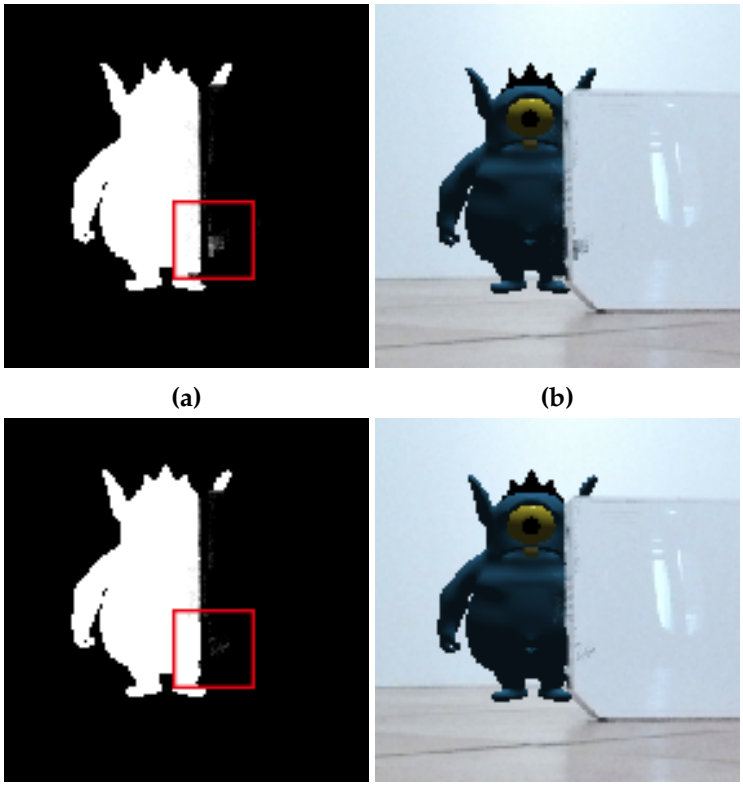


Abbildung 22: (a): Alpha Matte, die nach dem Occlusion Matting Verfahren aus Kapitel 4 entstanden ist. (b): Das daraus resultierende zusammengesetzte Bild. (c): Alpha Matte mit den in Kapitel 5 addierten Verbesserungen. (d): Das neue zusammengesetzte Bild.

te Pixel zu erkennen. An dieser Stelle sollte allerdings angemerkt werden, dass die Verwendung der Tiefendaten nicht immer eine Verbesserung der Alpha Matte hervorruft. Je nach Szene kann es vorkommen, dass durch die Einbindung der Tiefendaten zu viele Pixel aus dem Hintergrund als Vordergrund deklariert werden. Das passiert dadurch, dass Farb- und Tiefenwerte nicht genau aufeinander liegen. So kann es vorkommen, dass ein Pixel laut Tiefenwert im Vordergrund liegt, während er sich laut Farbbild im Hintergrund befinden sollte. So können Artefakte an den Kanten entstehen. Die Anpassung des Sampling-Verfahrens unter Einbindung der Tiefendaten, welches bereits unter [26] verwendet wurde, führt in der hier beschriebenen Implementation nur in wenigen, bestimmten Testszenen zu einer erhöhten Qualität der Matte. In den meisten Szenen wurden jedoch schlechtere Resultate als vorher erzielt.

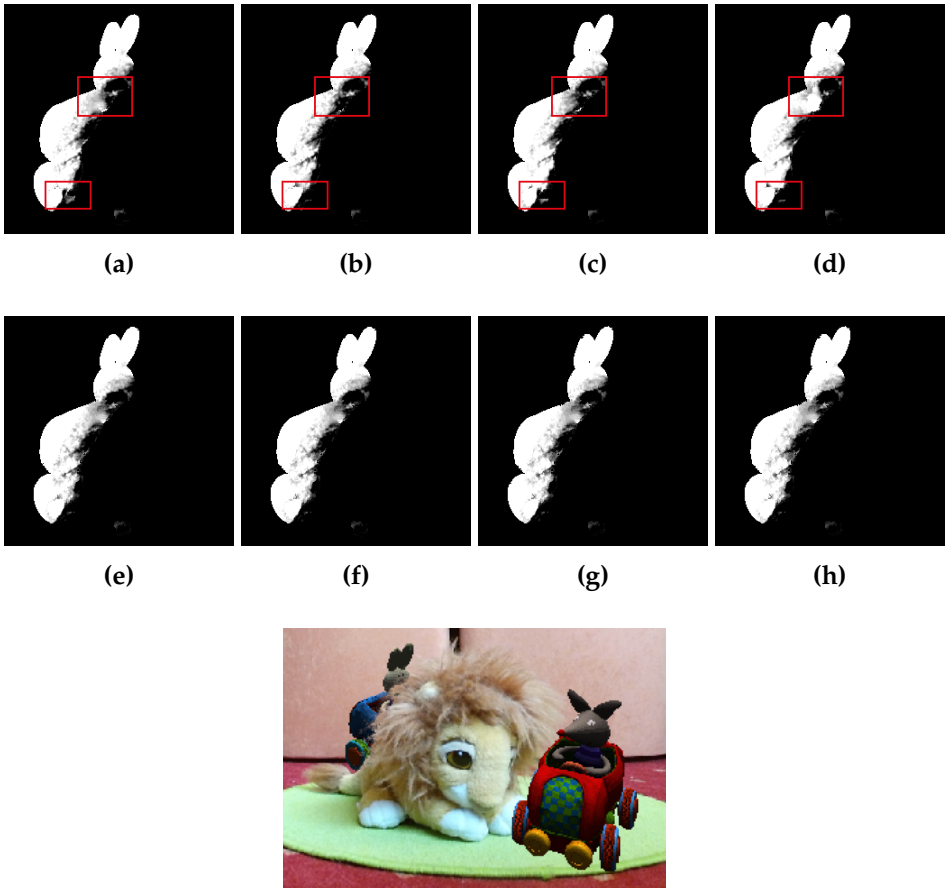


Abbildung 23: (a)-(d): Ergebnisse der Alpha Matte aufeinanderfolgender Frames (ohne Verbesserungen). (e)-(h): Ergebnisse der Alpha bzw. Flow Matte (mit Verbesserungen). (i)-(j): Das zusammengesetzte Bild aus einem der Frames, mit Verbesserungen.

Ein Vergleich zwischen dem unter Kapitel 4 beschriebenen Alpha Matting Verfahren sowie den Anpassungen durch die in dieser Arbeit besprochenen Konzepte lässt sich in Abbildung 23 finden. Hier ist eine Szene abgebildet, in der zwei der bereits besprochenen Probleme auftreten: Zum einen ist die Vorder- und Hintergrundfarbe in bestimmten Bereichen sehr ähnlich, zum anderen besteht ein großer Teil des Vordergrundobjekts aus Haaren. Die erste Bilderreihe zeigt, wie sich die alte Version der Alpha Matte in vier aufeinanderfolgenden Frames verändert. Hier sind problematische Bereiche der Matte in rot markiert. Das sind solche Bereiche, in denen sich ein Flimmern in aufeinanderfolgenden Frames bemerkbar macht, da die Alpha Matte jedes Mal von Grund auf neu berechnet wird. In der zweiten Bildreihe wird je eine Matte zur sel-

ben Bildsequenz anhand der besprochenen Veränderungen erstellt. Dabei wurde das erste Bild links mithilfe des iterativen gewichteten Window Samplings ermittelt. Die restlichen Bilder der Reihe wurden auf Basis des ersten Bilds anhand des Optical Flows geupdated. Hier lässt sich feststellen, dass die Übergänge von Frame zu Frame deutlich glatter sind. Es findet also kein Flimmern mehr statt, sondern nur sehr kleine Verschiebungen der Alpha-Werte durch den Optical Flow. Diese Konsistenz sorgt für eine angenehmere visuelle Erfahrung für den Betrachter der Videosequenz. Die letzte Bilderreihe zeigt das zusammengesetzte Bild aus einem der Videoframes unter dem angepassten Verfahren.

Augmented Reality-Anwendungen werden nicht nur in stillen Szenen verwendet, sondern es ist oft eine bewegliche Kamera vorausgesetzt. Daher sollte der Matting-Algorithmus auch bei Bewegungen der Kamera oder des virtuellen Objekts eine korrekte Verdeckungssituation ermitteln. Ein solches Beispiel mit beweglicher Kamera lässt sich in Abbildung 24 betrachten. Hier wurde das virtuelle Objekt in eine Videosequenz eingefügt. Während sich die Kamera in aufeinanderfolgenden Frames bewegt, bleibt das virtuelle Objekt immer in der gleichen Position relativ zur Kamera. Eine zusätzliche Schwierigkeit ist in dieser Testszene auch durch den Hintergrund gegeben, welcher aus feinen Texturen anstatt einer einheitlichen Farbe besteht. Obwohl sich die Alpha Matte nun in jedem Frame aufgrund der starken Bewegung immer wieder verändert und kleine Artefakte auftreten können, lässt sich insgesamt beobachten, dass im zusammengesetzten Bild kaum Veränderungen bei der Verdeckungsbehandlung festgestellt werden können. Somit ist eine konsistente Darstellung der Kombination von virtueller und realer Szene auch in solchen bewegten Szenen möglich.

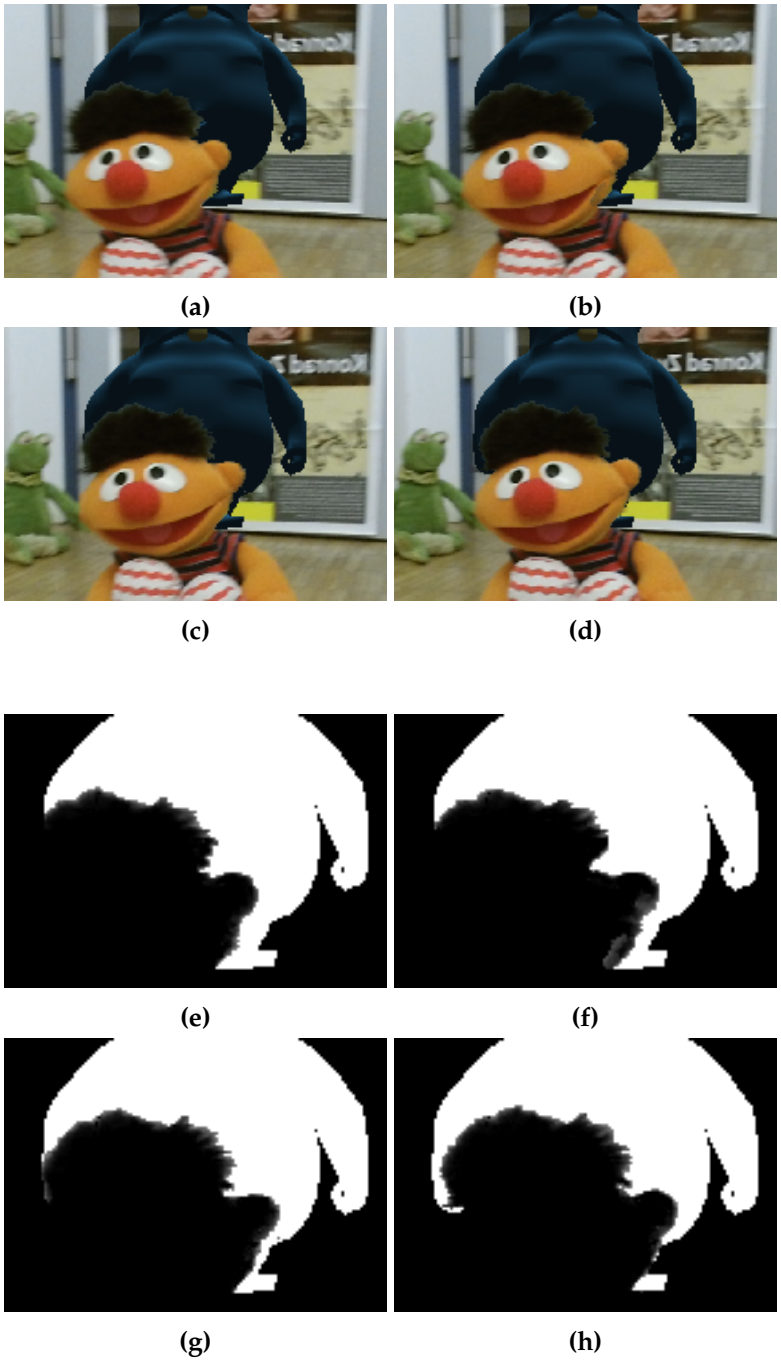


Abbildung 24: Aufnahme einer AR-Szene unter Kamerabewegung. (a)-(d): Das zusammengesetzte Bild in 4 aufeinanderfolgenden Frames. (e)-(h): Die zugehörige Alpha Matte.

7 Fazit und Ausblick

In dieser Arbeit wurde ein Ansatz zur Behandlung von Verdeckungen in Augmented Reality durch Natural Image Matting besprochen. Dieser ermittelt auf Basis von Farb- und Tiefendaten, wann ein virtuelles Objekt durch ein oder mehrere Objekte im Vordergrund verdeckt wird. Dafür wurde das bereits bestehende Verfahren von [3] verwendet und um mehrere Funktionen erweitert. Zum einen wurde eine Aktualisierung der Matte mittels Optical Flows durchgeführt, um temporale Informationen mit einzubeziehen. Dies sorgt dafür, dass ein Flimmern der Alpha Matte vermieden wird, was gerade in detaillierten Bereichen vermehrt auftreten kann. Des Weiteren wurde das bestehende Sampling-Verfahren angepasst, um Bereiche mit hoher Farbähnlichkeit besser behandeln zu können. Dazu wurde der bisherige Sampling-Bereich durch ein sich anpassendes, bewegendes Fenster ersetzt, aus dem mehrere Samples ausgewählt werden. Die gewählten Samples gehen dabei alle anhand einer bestimmten Gewichtung in das Ergebnis der Alpha Matte ein. Zudem werden bei diesem Sampling auch Informationen aus den Tiefendaten berücksichtigt. Es konnte gezeigt werden, dass diese Erweiterungen zu einer Verbesserung in der Qualität der Alpha Matte in schwierigen Szenen beitragen. Trotz der neu dazugewonnenen Funktionalität wird dabei weiterhin eine Berechnung der Verdeckung in Echtzeit erreicht. Dies sorgt dafür, dass eine Verwendung des Mattings für Augmented Reality-Anwendungen weiterhin gewährleistet ist.

Allerdings lässt sich trotz der verbesserten Ergebnisse bei der Verdeckungsberechnung feststellen, dass auch hier nicht in jeder Szene eine einwandfreie Alpha Matte erstellt wird. In Szenen mit schwer unterscheidbarem Vorder- und Hintergrund kann zwar eine Reduktion der entstehenden Artefakte erzielt werden, jedoch finden sich oft weiterhin Bereiche, in denen die Unterteilung der beiden Ebenen falsch durchgeführt wird. Es ist durchaus denkbar, dass weitere Anpassungen bei der Sample-Auswahl solche Fälle beheben können. Da das Natural Image Matting ein intensiv erforschtes Gebiet ist, existiert eine Vielzahl verschiedener Matting-Techniken, wovon einige unter [10] eingesehen werden können. Die meisten dieser Methoden sind für die Verwendung im Video Matting-Bereich nicht geeignet, allerdings ist es denkbar, dass Anpassungen durchgeführt werden können, um eines der robusten Matting-Verfahren in Echtzeit ausführbar zu machen. Ebenfalls wäre es denkbar, dass Veränderungen an der Erstellung der Trimap zu einer genaueren Alpha Matte führen können.

Da der Optical Flow ausschließlich auf den Farbdaten berechnet wird, kann es auch hier weiterhin zu Problemen kommen, wenn die Sample-Farben zu ähnlich sind. Generell lässt sich sagen, dass die alleinige Nutzung der Farbinformation nicht ausreichend für eine Berechnung der

Matte in Echtzeit ist. Hierbei helfen die Tiefendaten erheblich. Allerdings sind diese trotz konstanter Verbesserung der Messgeräte immer noch zu ungenau, um pixelgenaue Berechnungen mit ihnen durchzuführen. Eine weitere Verbesserung der Messgeräte würde auch für eine deutliche Verbesserung des Occlusion Mattings sorgen.

Können die genannten Probleme beseitigt werden, ist es durchaus denkbar, dass Natural Image Matting einen neuen Standard darstellt, um das Verdeckungsproblem in Augmented Reality zu lösen. Hebborn et al. [3] haben bereits gezeigt, dass das Occlusion Matting gegenüber anderen Verfahren in vielen Testfällen für eine realistischere Verdeckung sorgt. Daher sollte auch zukünftige Forschung auf Basis des Image Matting durchgeführt werden.

Literatur

[1] Y. Tian, T. Guan, and C. Wang, "Real-Time Occlusion Handling in Augmented Reality Based on an Object Tracking Approach," pp. 2885–2900, 2010.

[2] C. Du, Y.-L. Chen, M. Ye, and L. Ren, "Edge Snapping-Based Depth Enhancement for Dynamic Occlusion Handling in Augmented Reality," 2016.

[3] A. K. Hebborn, N. Höhner, and M. Stefan, "Occlusion Matting: Realistic Occlusion Handling for Augmented Reality Applications," *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) (accepted)*, 2017.

[4] "Microsoft hololens." https://www.microsoft.com/de-de/hololens/why-hololens. Accessed: 2017-09-01.

[5] "Ar und soziale netzwerke." http://www.informatik.uni-oldenburg.de/ iug10/sn/html/content/augmented_reality.html. Accessed: 2017-09-01.

[6] K. Agusanto, Li Li, Zhu Chuangui, and Ng Wan Sing, "Photorealistic rendering for augmented reality using environment illumination," in *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pp. 208–216, IEEE Comput. Soc, 2003.

[7] A. Levin, D. Lischinski, and Y. Weiss, "A Closed-Form Solution to Natural Image Matting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 228–242, 2008.

[8] C. Rhemann, C. Rother, and M. Gelautz, "Improving Color Modeling for Alpha Matting," in *BMVC*, jan 2008.

[9] E. S. L. Gastal and M. M. Oliveira, "Shared Sampling for Real-Time Alpha Matting," *Comput. Graph. Forum*, vol. 29, no. 2, 2010.

[10] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott, "A Perceptually Motivated Online Benchmark for Image Matting," in *CVPR*, pp. 1826–1833, 2009.

[11] "Microsoft kinect v2." https://www.heise.de/ct/ausgabe/2014-25-Mit-der-Kinect-V2-MIDI-Geraete-steuern-2450350.html. Accessed: 2017-09-01.

[12] B. K. Horn and B. G. Schunck, "Determining Optical Flow," tech. rep., 1980.

[13] D. Fleet and Y. Weiss, *Optical Flow Estimation*, pp. 237–257. Boston, MA: Springer US, 2006.

[14] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, pp. 674–679, Morgan Kaufmann Publishers Inc., 1981.

[15] G. Farnebäck, "Two-Frame Motion Estimation Based on Polynomial Expansion," in *mage Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29 – July 2, 2003 Proceedings*, pp. 363–370, 2003.

[16] Y. Tian, Y. Long, D. Xia, H. Yao, and J. Zhang, "Handling occlusions in augmented reality based on 3D reconstruction method," *Neurocomputing*, vol. 156, pp. 96–104, may 2015.

[17] J. Sun, J. Jia, C.-K. Tang, H.-Y. Shum, J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum, "Poisson matting," in *ACM SIGGRAPH 2004 Papers on - SIGGRAPH '04*, vol. 23, (New York, New York, USA), p. 315, ACM Press, 2004.

[18] J. Johnson, E. S. Varnousfaderani, H. Cholakkal, and D. Rajan, "Sparse Coding for Alpha Matting," *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3032–3043, 2016.

[19] Yung-Yu Chuang, B. Curless, D. Salesin, and R. Szeliski, "A Bayesian approach to digital matting," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 2, pp. II–264–II–271, IEEE Comput. Soc, 2001.

[20] J. Wang and M. F. Cohen, "Optimized Color Sampling for Robust Matting," *IEEE Conference on Computer Vision and Pattern Recognition. CVPR '07.*, 2007.

[21] K. He, C. Rhemann, C. Rother, X. Tang, and J. Sun, "A global sampling method for alpha matting," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2049–2056, 2011.

[22] V.-Q. Pham, K. Takahashi, and T. Naemura, "Real-Time Video Matting Based on Bilayer Segmentation," in *Computer Vision – ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part III*, pp. 489–501, Springer, Berlin, Heidelberg, 2010.

[23] L. Wang, M. Gong, C. Zhang, R. Yang, C. Zhang, and Y.-H. Yang, "Automatic Real-Time Video Matting Using Time-of-Flight Camera and

Multichannel Poisson Equations," *International Journal of Computer Vision*, vol. 97, pp. 104–121, mar 2012.

[24] M. Sindeev, A. Konushin, and C. Rother, "Alpha-Flow for Video Matting," in *Computer Vision – ACCV 2012*, pp. 438–452, Springer, Berlin, Heidelberg, 2013.

[25] E. Catmull and J. Clark, "Recursively generated B-spline surfaces on arbitrary topological meshes," *Seminal graphics*, pp. 183–188, 1998.

[26] J. H. Cho, R. Ziegler, M. Gross, and K. H. Lee, "Improving alpha matte with depth information," *IEICE Electronics Express*, vol. 6, no. 22, pp. 1602–1607, 2009.