Projektbericht

Klassifikation von Szenen mittels Maschinellem Lernen im Kontext der Fertigung

Betreuer: Oliver Heiman

Gruppenmitglieder: Arian Khalaf | Qinyuan Fan | Tobias Königer | Zixuan Liu

Inhaltsverzeichnis

[Abbildungsverzeichnis 4](#_Toc32586188)

[1. Problemdefinition 5](#_Toc32586189)

[2. Vorarbeiten 6](#_Toc32586190)

[2.1 Unity als Entwicklungsumgebung 6](#_Toc32586191)

[2.2 Szenenverarbeitung 7](#_Toc32586192)

[2.3 Maschinelles Lernen 7](#_Toc32586193)

[2.4 Neuronale Netze 8](#_Toc32586194)

[2.5 SVM (Support Vector Machine) 8](#_Toc32586195)

[2.6 Nearest-Neighbour-Clustering 9](#_Toc32586196)

[3. Das Hu-Moment 9](#_Toc32586197)

[3.1 Der mathematische Hintergrund des Hu-Moments 9](#_Toc32586198)

[3.2 Bild-Moment 9](#_Toc32586199)

[3.3 Hu-Moment 11](#_Toc32586200)

[3.4 Implementierung in OpenCV 12](#_Toc32586201)

[3.4.1 MatchShapes (OpenCV) 13](#_Toc32586202)

[4. Das Zernike-Moment 14](#_Toc32586203)

[5. PCA - Manuelles Matching 16](#_Toc32586204)

[5.1 PCA - Principal Component Analysis 16](#_Toc32586205)

[5.3 Kovarianzmatrix 17](#_Toc32586206)

[5.3 SVD - Singular value decomposition 17](#_Toc32586207)

[5.2 Berechnungsprinzip der PCA 18](#_Toc32586208)

[6 Vergleichspipeline 18](#_Toc32586209)

[7 Implementierung 19](#_Toc32586210)

[7.2 Der Szenengenerator 19](#_Toc32586211)

[8 Versuchsauswertung 21](#_Toc32586212)

[8.1 Ergebnisse der Untersuchung der Hu-Momente 22](#_Toc32586213)

[8.2 Ergebnisse der Versuche mit Zernike-Momenten 22](#_Toc32586214)

[8.3 Ergebnisse des Szenenvergleichs mit PCA 22](#_Toc32586215)

[9 Anhang 22](#_Toc32586216)

[9.1 Quellen 22](#_Toc32586217)

[9.2 Projektplan 24](#_Toc32586218)

[9.3 Quellcode 25](#_Toc32586219)

# Abbildungsverzeichnis

[Abbildung 1: Versuchsaufbau 5](#_Toc32586220)

[Abbildung 2: Hu-Momente Beispiele11 13](#_Toc32586221)

[Abbildung 3: Berechnung der MatchShapes Funktion (OpenCV) 10 14](#_Toc32586222)

[Abbildung 4: Rekonstruktion eines Bildes mit Zernike-Momenten12 15](https://d.docs.live.net/5502acec562d9821/AT%20Projekt/Projektbericht/Projektbericht.docx#_Toc32586223)

[Abbildung 5: Rekonstruktion mit Zernike Momenten mit 60. Ordnung16 16](#_Toc32586224)

[Abbildung 6: Anwendung der OpenCV PCA17 17](#_Toc32586225)

[Abbildung 7: Translation + Rotation 18](#_Toc32586226)

[Abbildung 8: Vier unterschiedliche Ausprägungen für die linke, obere, untere und rechte Kante 19](#_Toc32586227)

[Abbildung 9: Drei zufällig erzeugte Beispielszenen mit rotierten und translatierten Objekten. 20](#_Toc32586228)

[Abbildung 10: Aliasing-Effekt bei einem der generierten Objekte 21](https://d.docs.live.net/5502acec562d9821/AT%20Projekt/Projektbericht/Projektbericht.docx#_Toc32586229)

[Abbildung 11: Zufällig erzeugte Objekte nach dem Ausfüllen 22](#_Toc32586230)

[Abbildung 12: Gantt-Chart des Projektplan 24](#_Toc32586231)

# Problemdefinition

Im automatisierungstechnischen Projekt *"Klassifikation von Szenen mittels Maschinellem Lernen im Kontext der Fertigung"* wird das Problem der Wiedererkennung von Szenen im Rahmen einer fertigungstechnischen Aufgabe auf einem Roboterstand behandelt. Ein System aus einem Roboterarm, einem Projektor und einem optischen Scanner kann derzeit nicht erkennen, ob ein Stückpaar bereits in einer ähnlichen Konstellation auf dem Arbeitstisch angeordnet und bearbeitet worden ist. Es können nur (gemeinsame) Kanten der Werkstücke optisch detektiert und ein vom Anwender mit dem Finger individuell erstellter Pfad von dem Roboterarm abgefahren werden.

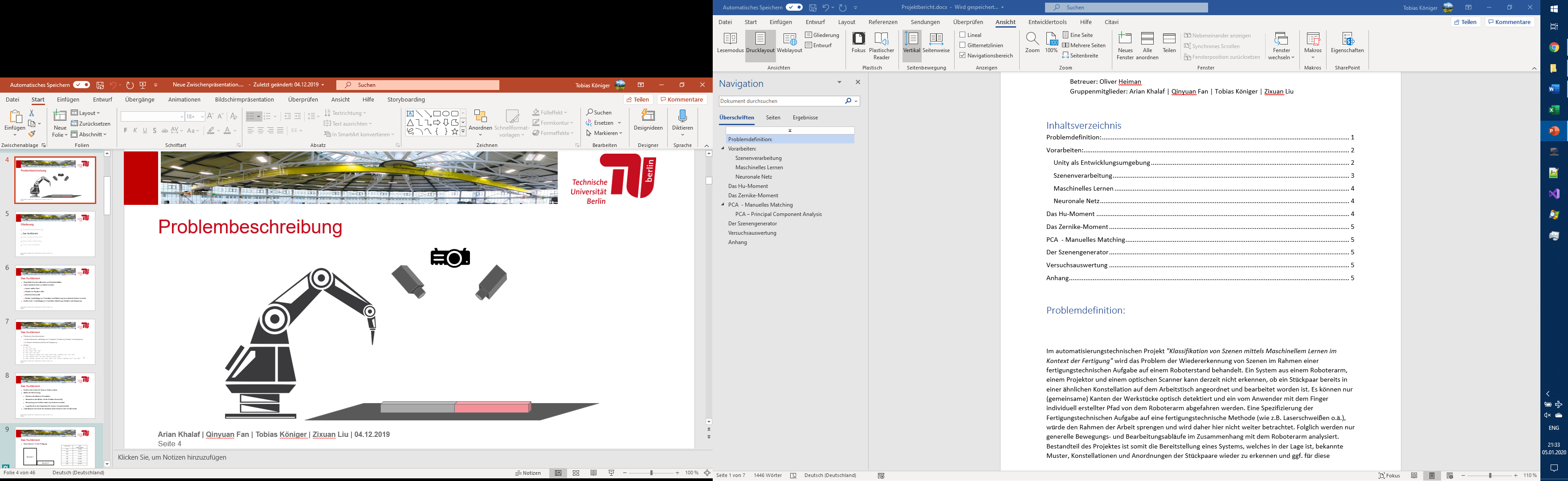


Abbildung 1: Versuchsaufbau

Eine Spezifizierung der Fertigungstechnischen Aufgabe auf eine fertigungstechnische Methode (wie z.B. Laserschweißen o.ä.), würde den Rahmen der Arbeit sprengen und wird daher hier nicht weiter betrachtet. Folglich werden nur generelle Bewegungs- und Bearbeitungsabläufe im Zusammenhang mit dem Roboterarm analysiert. Bestandteil des Projektes ist somit die Bereitstellung eines Systems, welches in der Lage ist, bekannte Muster, Konstellationen und Anordnungen der Stückpaare wieder zu erkennen und ggf. für diese eingeprägte und geeignete generelle Bearbeitungsverläufe vorzuschlagen, die es aus vorherigen Bearbeitungsverläufen erlernt hat. Der Anwender wird im besten Fall dadurch von redundanter Programmierung der Fertigungsabläufe befreit und kann seine gewonnene Arbeitszeit anderweitig gewinnbringend einsetzen.

Da im Zuge der Digitalisierung (Stichwort: Industrie 4.0) eine immer stärkere Vernetzung und Effizienzsteigerung von Produktionsabläufen stattfindet, ist es kurz- bis mittelfristig unabdingbar, dass Systeme die Fähigkeit erhalten bestimmte redundante Routinen eigenständig zu erkennen und auszuführen. In diesem Kontext kann mit dem vorliegenden Projekt der Grundstein für eine smarte und effiziente Produktion gelegt werden, welche auch die zunehmende Relevanz von „Machine Learning“ bzgl. dem Aufbau von Produktionsstätten miteinbezieht. Im Zuge des Projektes wird somit geprüft, ob sich das bestehende System mit einer Wiedererkennungsfähigkeit erweitern und verbessern lässt. Ferner wird geprüft, ob diese wiedererkannten Szenen dem Anwender in Form von Bearbeitungsvorschlägen übergeben werden können.

# Vorarbeiten

Alle Beteiligten besitzen in Bezug auf das Projektvorhaben Programmiererfahrungen, welche sie sich im Kontext ihres Studiums oder anderweitig angeeignet haben. Weitere Kenntnisse zu optischer Bildverarbeitung, der Verwendung von Unity als Entwicklungsumgebung, Erstellung eines Feature-Vektors und „Machine Learning“-Verfahren müssen im Laufe des Projektes von allen Beteiligten noch erlernt werden. Im Folgenden wird grob der Stand der Technik aufgezeigt, welche für das Projekt eine entscheidende Rolle spielt:

## **2.1 Unity als Entwicklungsumgebung**

Die Entwicklungsumgebung „Unity“ wird normalerweise für die Entwicklung von Computerspielen und anderen interaktiven 3D-Anwendungen genutzt. Sie ist an üblichen 3D-Animationsprogrammen angelehnt. Über ein Hauptfenster kann eine 3D-Szene gezeigt und über unterschiedlichen Menüs mit den Eingabegeräten skaliert, verschoben als auch gedreht werden. Jede Szene besteht aus „GameObjects“, die wiederum Eigenschaften (wie Skripte, Materialien usw.) besitzen.1

Unitys Programmbibliothek umfasst dabei zahlreiche primitive Objekte (wie z.B. Würfel, Kugeln o.ä.), die eine separate Erzeugung von diesen Objekten obsolet machen und dem Entwickler viel Zeit schenken. Sofern komplexere Strukturen, wie Animationen, Texturen o.ä., in den Programmcode eingebunden werden müssen, so können sie in der Form von „Assets“ in das Programm implementiert werden. Unitys Laufzeitumgebung detektiert dabei fortlaufend jegliche Änderungen in der Programmierung der Strukturen und aktualisiert sie dann automatisch. In weiteren Ansichten kann das jeweilige Programm dann simuliert und über einen Export in eine ausführbare Anwendung übersetzt werden.

Unitys Grafik-Engine benutzt ein Deferred-Shading-Verfahren2 auf Basis von OpenGL oder Direct3D. Folgende Beleuchtungsmodelle können dabei genutzt werden: Environment Mapping, Bumpmapping, Parallax Mapping, Umgebungsverdeckung, variable Schatteneffekte (auf Basis von Shadow Maps), Render-To-Texture und Vollbild-Postprocessing-Effekte (z. B. Spiegelungen und Glühen, basierend auf Framebufferobjekten). Zudem ist es möglich selbst entwickelte Shader für die implementierten Beleuchtungseffekte zu verwenden.3

Unity besitzt eine eigene Physik-Engine mit dem Namen „PhysX“. Die Bewegung der „GameObjects“ kann darüber mit definierten Pfaden, Skripte und physikalischen Eigenschaften erfolgen. Figuren können zudem über eine Skin-And-Bones-Technik (ähnlich einem Polygonnetz, über ein unsichtbares Skelett) animiert werden. Des Weiteren können gasförmige Eigenschaften (wie Feuer, Rauch usw.) über ein spezielles Partikelsystem dargestellt werden.

Für das Projekt insbesondere relevant ist die Möglichkeit mit eigens erstellten Skripten in der Skriptsprache C# die in „Unity“ bestehenden Funktionen zu erweitern. Aus mehreren Skripten bzw. Assets können dabei sogenannte „Prefabs“ gebildet werden, welche es erlauben selbst entwickelte Strukturen ähnlich wie die mitgelieferten Komponenten zu verwenden.1 Dank den Skripten kann auch die Entwicklungsumgebung selbst mit neuen Funktionalitäten erweitert werden. Diese sogenannten Plug-Ins ermöglichen ein breites Spektrum an Werkzeugen, die fast jeden Anwendungsfall abdecken und „Unity“ zu einer allseits beliebten Entwicklungsumgebung für viele technische Projekte machen.

## 2.2 Szenenverarbeitung

Im Rahmen dieser Arbeit werden aus Szenenaufnahmen Daten extrahiert, die sich für eine Szenenbeschreibung und insbesondere zur Unterscheidung zwischen zwei Szenen eignen. Mittels der Bildverarbeitung können hierbei Merkmale, die jede Szene einzigartig machen, bestimmt werden.

Jede Szene besteht lediglich aus einer niedrigen Anzahl von Objekten, die sich auf einer Platte befinden. Dabei wird die Szene von mehreren Kameras von oben aufgenommen. Grundsätzlich wäre denkbar eine Szenenbeschreibung auf detektierten Umrissen der Objekte vorzunehmen. Hierzu muss mithilfe eines Bildverarbeitungs-Algorithmus eine Kantendetektion durchgeführt werden.

Zur Aufnahme der Szene werden Stereo-Kameras verwendet. Aus den Bildern werden – parallel zum menschlichen Sehen – mit Stereo Imaging-Algorithmen eine 3D-Abbildung der Szene generiert. 3D-Objekte werden durch eine Vielzahl verbundener Dreiecke, sog. Meshes, beschrieben. Zur Kantenerkennung muss lediglich ermittelt werden, welche Kanten der Dreiecke sich in einer bestimmten Ebene am weitesten außen befinden. Diese Kanten können dann zu einem Linienzug verbunden werden, der den äußeren Umriss des Objektes vollständig beschreibt.

Eine weitere denkbare Szenenbeschreibung basiert auf der Auswertung von besonderen Merkmalen von Bildern, sog. Features. Unter Anwendung eines Bildverarbeitungs-Algorithmus werden dabei Features einer Szenen-Aufnahme erkannt, die diese Szene einzigartig machen. Mit OpenCV steht eine Open Source Lösung zur Verfügung, die eine automatische Feature-Erkennung ermöglicht.

OpenCV bietet desweiteren Algorithmen zum Feature-Matching an (z.B. die FLANN-Algorithmus), wie auch Funktionen zur Ermittlung der Transformation zwischen den Features.

Egal ob eine Szenenbeschreibung entlang der Objektumrisse oder anhand von erkannten Features durchgeführt wird (oder aus einer Kombination), so muss die Transformation von Features/ Umrissen zwischen zwei Szenen ermittelt werden. Hierzu gehört eine translatorische wie auch eine rotatorische Transformation. Da zwei Szenen nie vollständig übereinstimmen werden, muss mit einer Abweichung zwischen den Objektpositionen auch nach der Transformation gerechnet werden.

Die Entscheidung, ob zwei Szenen identisch sind, könnte dabei regelbasiert getroffen werden: Die mittlere Abweichung zwischen den Koordinaten muss geringer als delta\_x/ delta\_y sein. Alternativ dazu ist ebenfalls eine Entscheidung mithilfe von Machine Learning Algorithmen denkbar. Hierfür müsste allerding zunächst eine geeignete Datengrundlage generiert werden, auf der die Machine Learning Algorithmen trainiert werden.

## 2.3 Maschinelles Lernen

Maschinelles Lernen (ML) ist die wissenschaftliche Untersuchung von Algorithmen und statistischen Modellen, mit denen Computersysteme eine bestimmte Aufgabe ausführen, ohne explizite Anweisungen zu verwenden. Stattdessen stützen sie sich auf Muster und Schlussfolgerungen. Es wird als Teilmenge der künstlichen Intelligenz angesehen. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.20 Es gibt zwei Haupttypen des mechanischen Lernens: überwachtes Lernen und unüberwachtes Lernen. Beim überwachten Lernen besitzen die Eingabedaten gemeinsame Merkmale (d.h. es existieren Standartantworten). Beim unüberwachtem Lernen hingegen besitzen die Eingabedaten keine gemeinsamen Merkmale (d.h. es existieren keine Standartantworten). Gemäß unseren Projektanforderungen ist es bekannt, dass, mittels Kameras, ein System kontinuierlich einen Werktisch beobachtet und in der Lage ist, die dort platzierten Teile und ihre Position zu erfassen. Wenn jetzt die gleiche Szene (sprich die gleichen Werkstücke mit vergleichbarer relativer Position zu einander) erneut auf dem Tisch platziert wird, soll das System in der Lage sein, dies zu erkennen und dem Werker anbieten, das bereits bekannte Programm erneut anzuwenden. Für das Projekt kommen drei “Machine Learning”-Verfahren in Frage, welche im Folgenden näher beschrieben werden:

## 2.4 Neuronale Netze

Ein neurales Netz zeichnet sich erstens durch eine selbstlernende Funktion aus. Wenn beispielsweise eine Bilderkennung implementiert ist, werden nur eine Vielzahl von unterschiedlichen Bildvorlagen und entsprechenden Erkennungsergebnissen in das künstliche neuronale Netzwerk eingegeben, und das Netzwerk lernt langsam, ähnliche Bilder durch die Selbstlernfunktion zu erkennen. Die Selbstlernfunktion ist besonders wichtig für die Vorhersage der Zukunft. Künstliche neurale Netzwerkcomputer können dabei u.a. wirtschaftliche Vorhersagen, Marktvorhersagen und Nutzenvorhersagen für den Menschen liefern.

Zweitens verfügt ein neuronales Netz über die Speicherfunktion von Lenovo. Diese Eigenschaft kann unter Verwendung eines Rückkopplungsnetzwerks künstlicher neuronaler Netze erreicht werden. Drittens hat ein neuronales Netz die Fähigkeit, optimierte Lösungen mit hoher Geschwindigkeit zu finden. Das Finden einer optimalen Lösung für ein komplexes Problem erfordert häufig einen großen Rechenaufwand unter Verwendung eines künstlichen neuronalen Rückkopplungsnetzwerks, das für ein Problem ausgelegt ist. Ein neuronales Netz kann dabei die Hochgeschwindigkeits-Rechenleistung eines modernen Computers nutzen und so eine schnelle optimierte Lösung finden.

## 2.5 SVM (Support Vector Machine)

Eine Support Vector Machine ist ein mathematisches Modell, welches seinen Ursprung im maschinellen Lernen besitzt. Das Ziel einer Support-Vector-Machine ist es Objektdaten so zu klassifizieren, dass um den Klassengrenzen herum ein möglichst breiter Bereich frei von Objekten bleibt. Dabei wird zwischen den Objekten in Raum eine Hyperebene aufgespannt, welche alle Objekte in zwei Klassen teilt. Die Lage der Hyperebene wird nur von der an ihr nächsten anliegenden Objekten bestimmt. Um eine Hyperebene zu erstellten, werden alle Objekte in Form von Vektoren dargestellt. Die nächstliegenden Vektoren werden dabei als „Stützvektoren“ bezeichnet und geben dem Modell seinen Namen. Zudem muss beachtet werden, dass eine Hyperebene nicht verbogen werden kann. Sofern die Objekte nicht linear trennbar sind, so lässt sich eine Hyperebene nicht zwischen den Objektdaten aufspannen und es muss ein sogenannter Kernel-Trick angewendet, um eine nichtlineare Klassengrenze einzuziehen.

## 2.6 Nearest-Neighbour-Clustering

Die “Nearest-Neighbour-Clustering-Methode” klassifiziert Objekte anhand der Klassen ihrer knächsten Nachbarn. Durch eine Mehrheitsentscheidung wird mit Hilfe der bereits definierten Klassen der Nachbarobjekte die jeweilige Klasse des Objektes bestimmt. Dabei ist die Anwendung von vielen Abstandmaßen wie (Manhatten-, Euklidischer Abstand usw.) möglich. Die Klasse, die die größte Anzahl der k-Nachbarn besitzen, wird dann dem Objekt zugewiesen. Ein Unentschieden kann durch eine ungerades k vermieden werden. Bei zu kleinem k kann Rauschen und bei zu großen k kann Ungenauigkeit, aufgrund der Einbeziehung zu weit entfernt liegender Objekte, die Qualität der Klassifikationsergebnisse deutlich senken.

# Das Hu-Moment

## 3.1 Der mathematische Hintergrund des Hu-Moments

In der Mathematik ist ein Moment ein spezifisches quantitatives Maß für die Form einer Funktion. Es wird sowohl in der Mechanik als auch in der Statistik verwendet. Wenn die Funktion physikalische Dichte darstellt, dann ist das nullte Moment die Gesamtmasse, wobei das erste Moment durch die Gesamtmasse geteilt den Massenmittelpunkt ergibt, und das zweite Moment die Rotationsträgheit. Wenn die Funktion eine Wahrscheinlichkeitsverteilung ist, dann ist das nullte Moment die Gesamtwahrscheinlichkeit, das erste Moment der Erwartungswert, das zweite Zentralmoment die Varianz, das dritte genormte Moment die Schiefe und das vierte standardisierte Moment der Kurtosis. Das mathematische Konzept ist eng mit dem Konzept des Moments in der Physik und der Bildverarbeitung verwandt.

Seien und Diskrete Zufallsvariable, c ist eine Konstante und k ist eine positive ganze Zahl. Wenn existiert, wird das Moment -ter Ordnung von in Bezug auf Punkt genannt. Wenn ist, wird es das Ursprungsmoment -ter Ordnung genannt; wenn ist, wird es das Zentralmoment -ter Ordnung genannt.

Wenn existiert, heißt es , in Bezug auf den Punkt -ter Ordnung. Wenn ist, wird es das gemischte Ursprungsmoment -ter Ordnung genannt. Wenn , ist, wird es das gemischte Zentralmoment -ter Ordnung genannt.

Wenn , stetige Variablen sind, wird als das Ursprungsmoment -ter Ordnung von in Bezug auf Punkt bezeichnet, als das Zentralmoment -ter Ordnung von in Bezug auf Punkt bezeichnet.

Die Essenz des Moments ist die mathematische Erwartung, und die erwartete Berechnungsformel sollte sein, wobei die Wahrscheinlichkeitsdichte von ist, was bedeutet, die obige Formel geht tatsächlich davon aus, dass alle Zufallsvariablen die gleiche Wahrscheinlichkeit haben.

## 3.2 Bild-Moment

Durch die Einführung der obigen Mathematik-Momente können wir eine Reihe von Momenten verwenden, die aus digitalen Grafiken berechnet wurden, um die globalen Merkmale des Bildes zu beschreiben und eine Vielzahl von Informationen über die verschiedenen Arten von geometrischen Merkmalen des Bildes wie Größe, Position und Richtung und geometrisch Form usw. bereitzustellen, d. h Bildmoment

- Moment nullter Ordnung:

Das Bild hier ist ein Einkanalbild, und repräsentiert den Grauwert des Bildes am Punkt .

Wir können feststellen, dass, wenn das Bild ein Binärbild ist, die Summe der weißen Bereiche auf dem Bild ist, so dass verwendet werden kann, um den Bereich des Binärbilds (Kontur, verbundene Domäne) zu finden.

- Moment erster Ordnung:

Wenn das Bild binär ist, hat nur zwei Werte von 0 (schwarz) und 255 (weiß). ist die Akkumulation der -Koordinatenwerte aller weißen Bereiche im Bild. Daher kann das Moment erster Ordnung verwendet werden, um den Schwerpunkt eines binären Bildes zu finden, nämlich um die Graustufenmitte des Bildes zu bestimmen.

- Moment zweiter Ordnung:

Es gibt drei Momente zweiter Ordnung, , , , die ebenfalls zu Trägheitsmomenten werden. Sie können mehrere Merkmale eines Objekts bestimmen:

1. Das das Zentralmoment zweiter Ordnung wird verwendet, um die Hauptachse des Zielbilds zu bestimmen. Die lange und die kurze Achse entsprechen den maximalen und minimalen zentralen Moment zweiter Ordnung. Der Winkel der Hauptachsenrichtung kann berechnet werden.
2. Bildellipse: Aus dem Moment erster und zweiter Ordnung kann eine Bildellipse bestimmt werden, die der ursprünglichen Bildträgheit entspricht. Die sogenannte Bildellipse ist eine gleichmäßige Ellipse, die dem Moment zweiter Ordnung des Originalbildes und der Summe der Graustufen des Originalbildes entspricht. Diese Bildellipse übereinstimmt die Hauptachse der Bildellipse mit der Richtung der Hauptachse des Zielbilds, um die Art des Bildes zu analysieren.

Momente zweiter Ordnung können verwendet werden, um die Richtung der Form eines Objekts zu bestimmen.

ist eine Funktion von Opencv, die einen Vektor verwendet und einen Winkel von zurückgibt.

- Moment dritter Ordnung:

Moment dritter Ordnung: Projektionsschiefe, die den Grad der Verzerrung der Bildprojektion beschreibt. Die Schiefe gehört zur klassischen Statistik, mit der der Grad der Abweichung von der symmetrischen Verteilung des Mittelwerts gemessen wird.

- Moment vierter Ordnung:

Moment vierter Ordnung: Projektions-Kurtosis, Kurtosis gehört ebenfalls zur klassischen Statistik. Es werden immer die Kurtosis-Koeffizienten einer Verteilung berechnet. Wenn der Kurtosis-Koeffizient 0 ist, bedeutet dies eine Gauß-Verteilung, wenn der Kurtosis-Koeffizient kleiner als 0 ist, bedeutet dies eine abgeflachte Verteilung mit wenigen Peaks, wenn der Kurtosis-Koeffizient größer als 0 ist, bedeutet dies eine enge Verteilung(mit starken Peaks) und mehreren Spitzen.

## 3.3 Hu-Moment

Das Zielbild in unserem Projekt wird oft von einer räumlichen Transformation (Translation, Skalierung, Rotation) begleitet, so dass es notwendig ist, auf der Grundlage der Bildmomente eine invariante Moment-Gruppe das sog. Hu-Moment zu konstruieren.

Durch die Einführung der obigen Bildmomente können wir das Ursprungsmoment berechnen:

Das Zentralmoment ist:

Wenn sich das Zielbild ändert, zeigt die obige Interpretation der Bildmomente, dass sich das Ursprungsmoment auch immer ändert. Das Zentralmoment ist eine Funktion, die translationsinvariant, aber dennoch sehr empfindlich gegenüber Rotationsänderungen ist, deshalb können das Ursprungsmoment und das Zentralmoment nicht direkt als Merkmalsmoment des Bildes verwendet werden. Im Zuge der Recherche der Gruppe kristallisierte sich heraus, dass das normalisierte zentrale Moment als Merkmal des Bildes verwendet wird, weil das normalisierte zentrale Moment des Bildes nicht nur translationsinvariant, sondern auch skalierungsinvariant ist.

Das normalisierte zentrale Momente lautet:

Das Hu-Moment verwendet normalisierte zentrale Momente zweiter und dritter Ordnung, um 7 invariante Momente zu konstruieren, die unter kontinuierlichen Bildbedingungen invariant für Translation, Skalierung und Rotation sind. 11

s

Das 7. Hu-Moment ändert das Vorzeichen wenn das Objekt gespiegelt wird. 11

Aufgrund der beschriebenen Eigenschaften eignen sich die Hu-Momente prinzipiell zur Szenen-Wiedererkennung.

## 3.4 Implementierung in OpenCV

Zur Berechnung des Hu-Moments ist in OpenCV die Funktion HuMoments() implementiert. Zunächst muss das betreffende Bild in Graustufen eingelesen werden. Anschließend wird das Bild mit der threshold() Funktion binarisiert. Da sich das Hu-Moment aus den zentralen Bildmomenten berechnet, werden die Momente zunächst mit moments() ermittelt. Mit HuMoments() werden aus den zentralen Momenten die Hu-Momente erzeugt.

Zur besseren Vergleichbarkeit müssen die Hu-Momente noch logarithmiert werden. Abbildung 2 zeigt die berechneten Hu-Momente von sechs Beispielbildern.

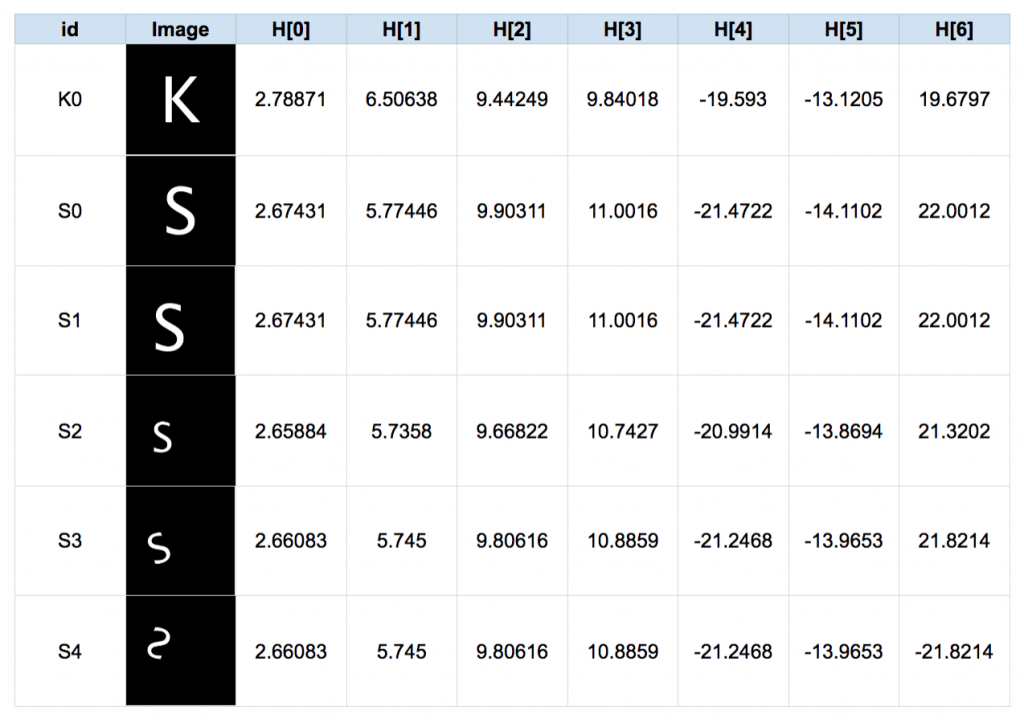


Abbildung 2: Hu-Momente Beispiele11

### 3.4.1 MatchShapes (OpenCV)

Die Funktion matchShapes aus OpenCV vergleicht zwei Konturen mithilfe der Hu-Momente. Die Funktion stellt dazu drei verschiedene Berechnungsmethoden zum Vergleich der Hu-Momente zur Verfügung.

Die erste Methode bildet die Differenz aus den Kehrwerten der Hu-Momente, die zweite Methode subtrahiert direkt die Momente voneinander und die dritte bezieht die Differenz auf das erste Hu-Moment.

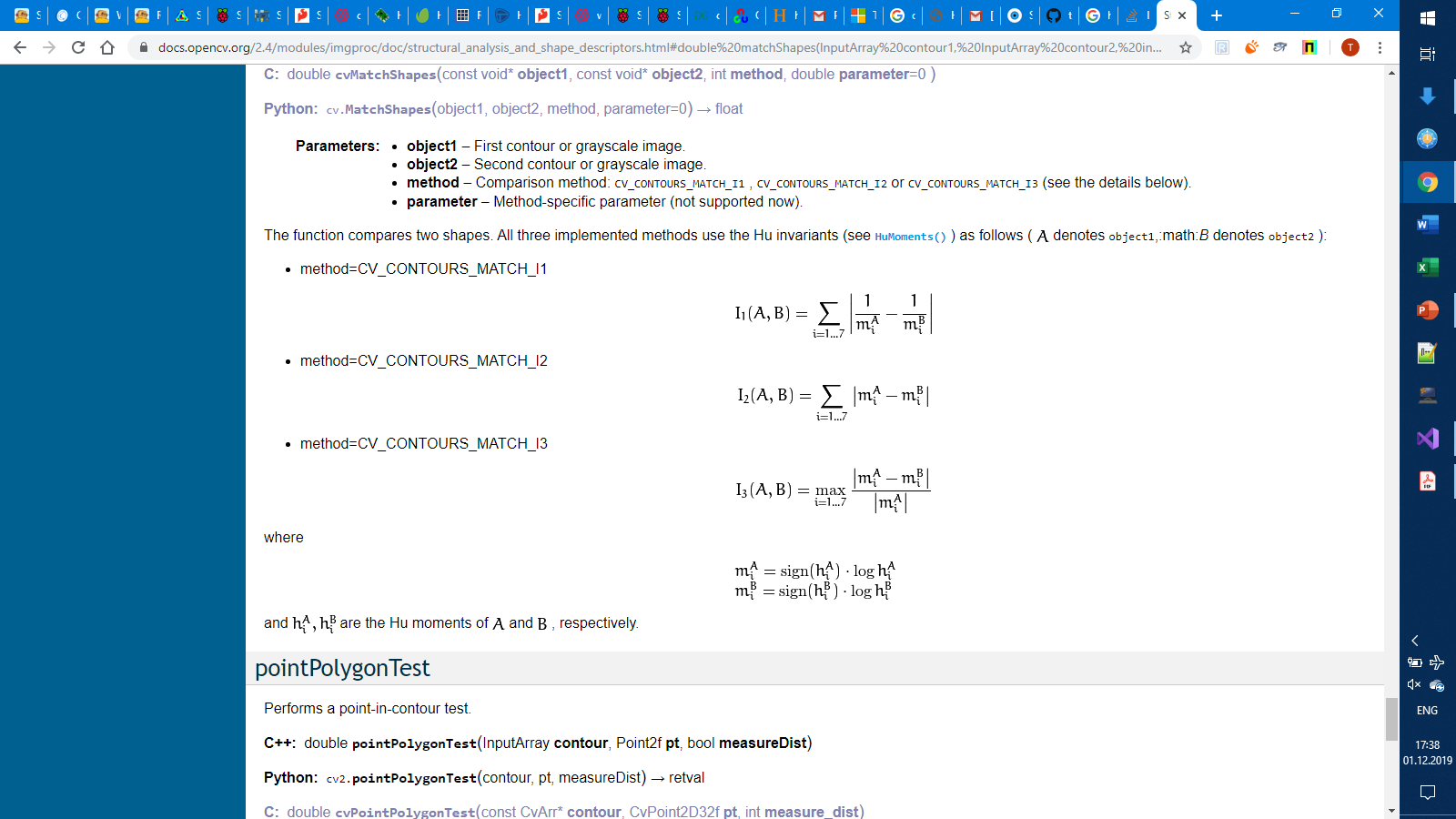


Abbildung 3: Berechnung der MatchShapes Funktion (OpenCV) 10

Problem:

Die weitere Recherche ergab, dass Hu-Momente einwandfrei bei symmetrischen Objekten funktionieren. 4 5 6

Lösung:

* Zernike Momente bieten noch höhere Genauigkeit (besitzen aber viel höheren Rechenaufwand).7 8
* Zernike Momente können mit der Open Source Implementierung der Python Library Mahotas9 berechnet werden.

# Das Zernike-Moment

Zernike-Momente sind Bildmomente, die bei einer Rotation der Szene invariant verhalten. Zum Vergleich zweier beliebiger Szenen müssen die Zernike-Momente deshalb zunächst translatorisch in Deckung gebracht werden. Die Translation kann zum Beispiel mithilfe der Bildschwerpunkte durchgeführt werden.

Gegenüber dem Hu-Moment besitzt das Zernike-Moment den Vorteil, dass die Zernike-Momente orthogonal zueinander sind, wodurch die Werte linear unabhängig sind. Daraus folgt, dass die Zernike-Momente im Gegensatz zu den Hu-Momenten keine redundanten Informationen enthalten.

Ein weiterer Vorteil der Zernike-Momente besteht darin, dass sie bis zu einer beliebigen Ordnung berechenbar sind. Es können daher prinzipiell beliebig viele Informationen aus einem Bild extrahiert werden.

Diesen Vorteilen steht jedoch der quadratisch steigende Rechenaufwand zur Berechnung höherer Zernike-Momente gegenüber. Für ein quadratisches Bild mit der Kantenlänge N und der Ordnung des Zernike-Moments M steigt die Zahl der benötigten Additionen und Multiplikationen mit O(M2N2).

82 \* 2002 = 40.064

Aktuelle Forschungsergebnisse zeigen, dass die Bilderkennung mit Zernike-Momenten deutlich bessere Ergebnisse hinsichtlich der Erkennungsrate liefert als mit Hu-Momenten. Es können Erkennungsraten von bis zu 97 bis 99 % erreicht werden.21 22

Zur Berechnung der Zernike Momente steht eine Implementation aus der python-Bibliothek mahotas unter der Open Source MIT-Lizenz zur Verfügung.9

Die Berechnung der Zernike-Momente stellt sich relativ kompliziert dar: Die folgende Formel zeigt die Berechnung des Zernike-Polynoms der Ordnung n und Stufe m. Die zweite Formel stellt eine Zusammenfassung der ersten dar, wobei lediglich die Formelzeichen variiert wurden.

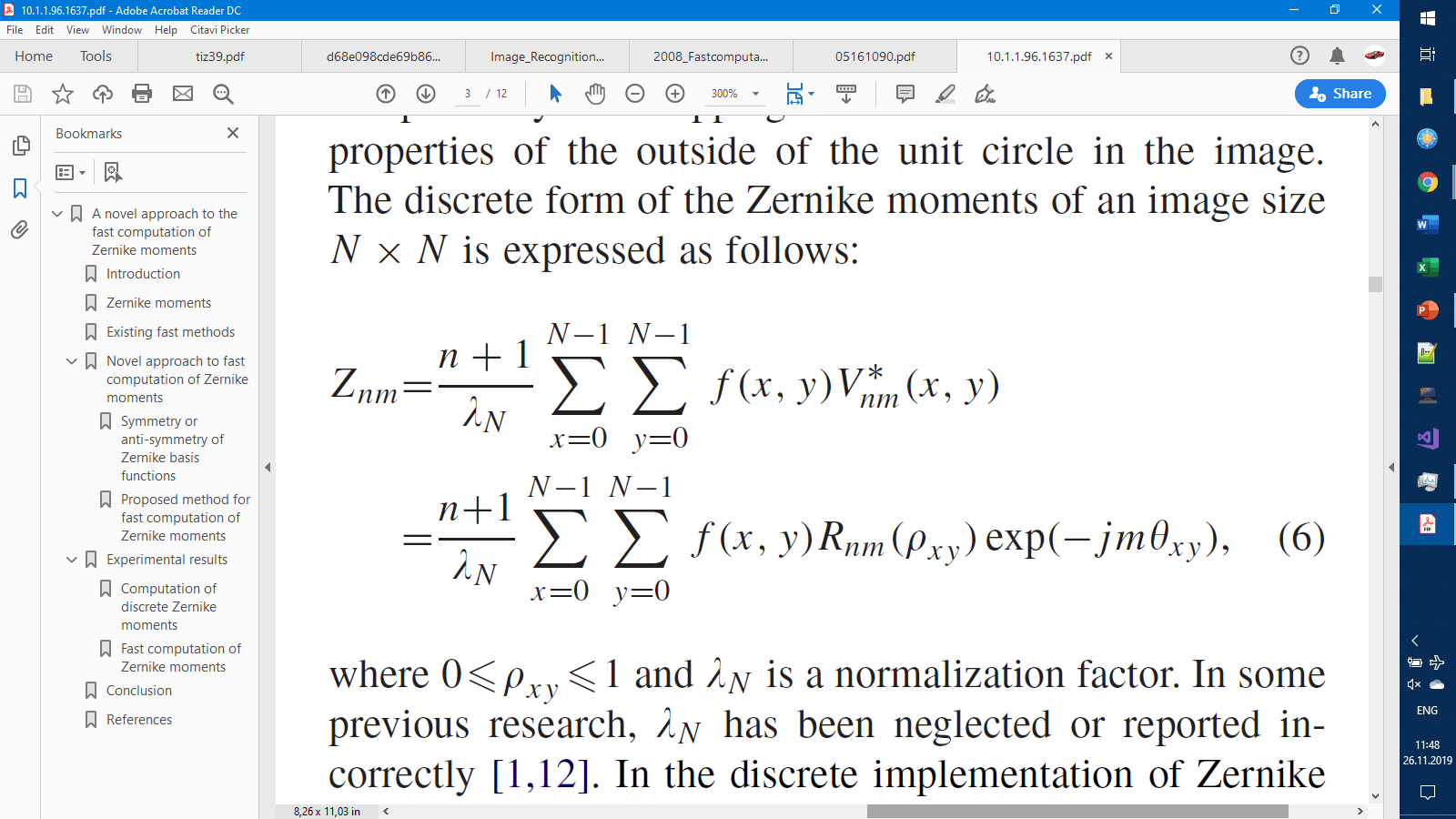
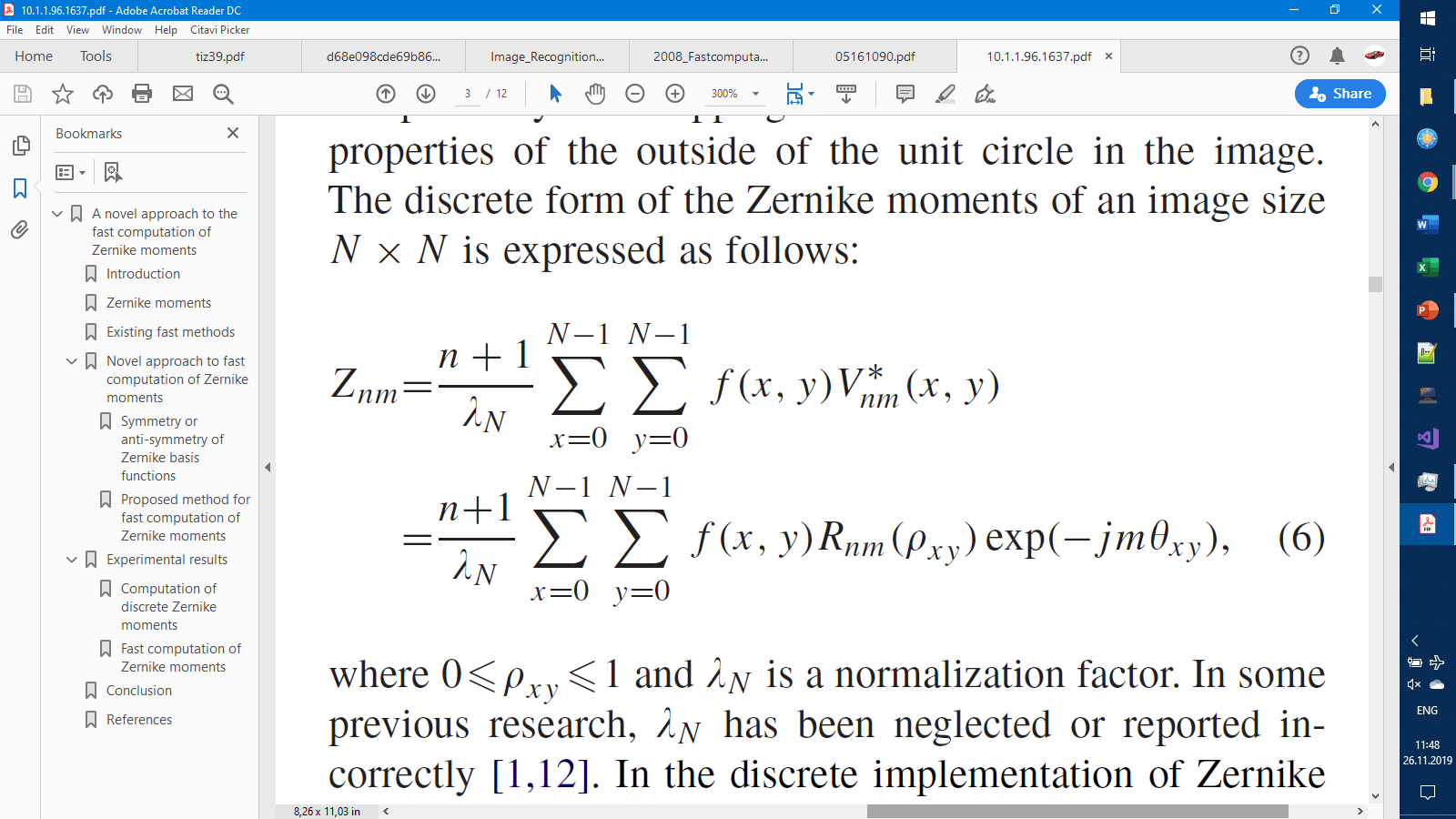
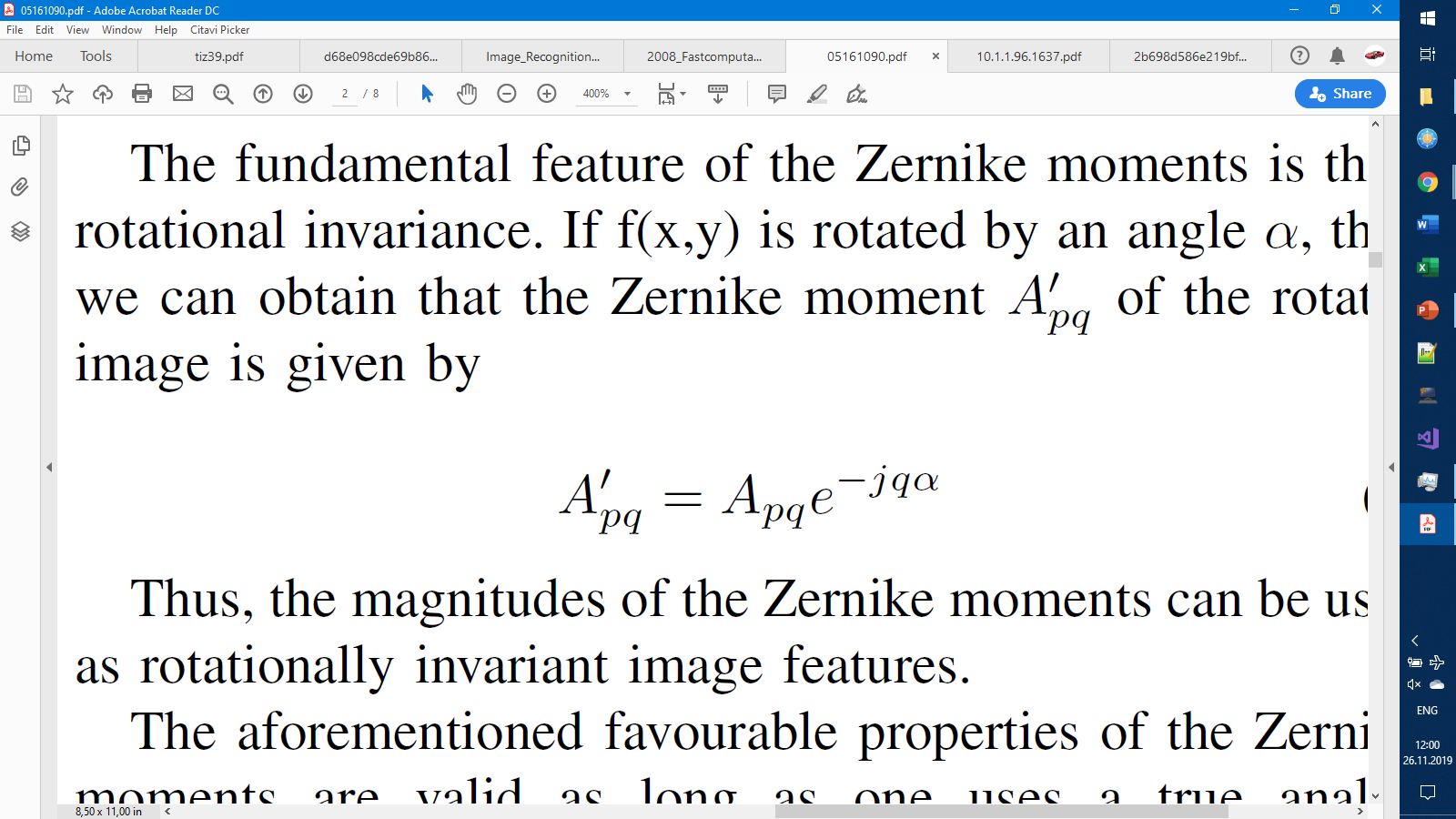
Bemerkenswert sind jeweils die letzten Terme: Hierbei handelt es sich um eine Exponentialfunktion. Bei Zernike-Momenten handelt es sich um komplexe Zahlen, deren Betrag bei einer Rotation um einen beliebigen Winkel gleich bleibt. 13 14 15

Abbildung 4 zeigt die Leistungsfähigkeit der Zernike Momente: Bild (a) stellt das Ausgangsbild dar. Die Bilder (b) bis (f) zeigen Rekonstruktionen des Ausgangsbildes, die aus den berechneten Zernike-Momenten mit ansteigender Ordnung gewonnen wurden.

(a)

(b)

(c)

(d)

(e)

(f)

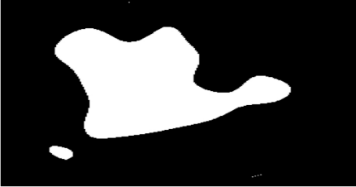
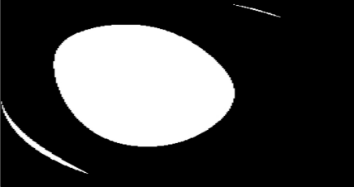


Abbildung 4: Rekonstruktion eines Bildes mit Zernike-Momenten12

Abbildung 5 zeigt die Rekonstruktion eines Bildes aus Zernike Momenten bis zur Ordnung 60 aus einer Forschungsarbeit der NASA: Bei ausreichend genauer numerischer Berechnung können Bilder mit Zernike-Momenten mit nahezu beliebiger Genauigkeit codiert werden.

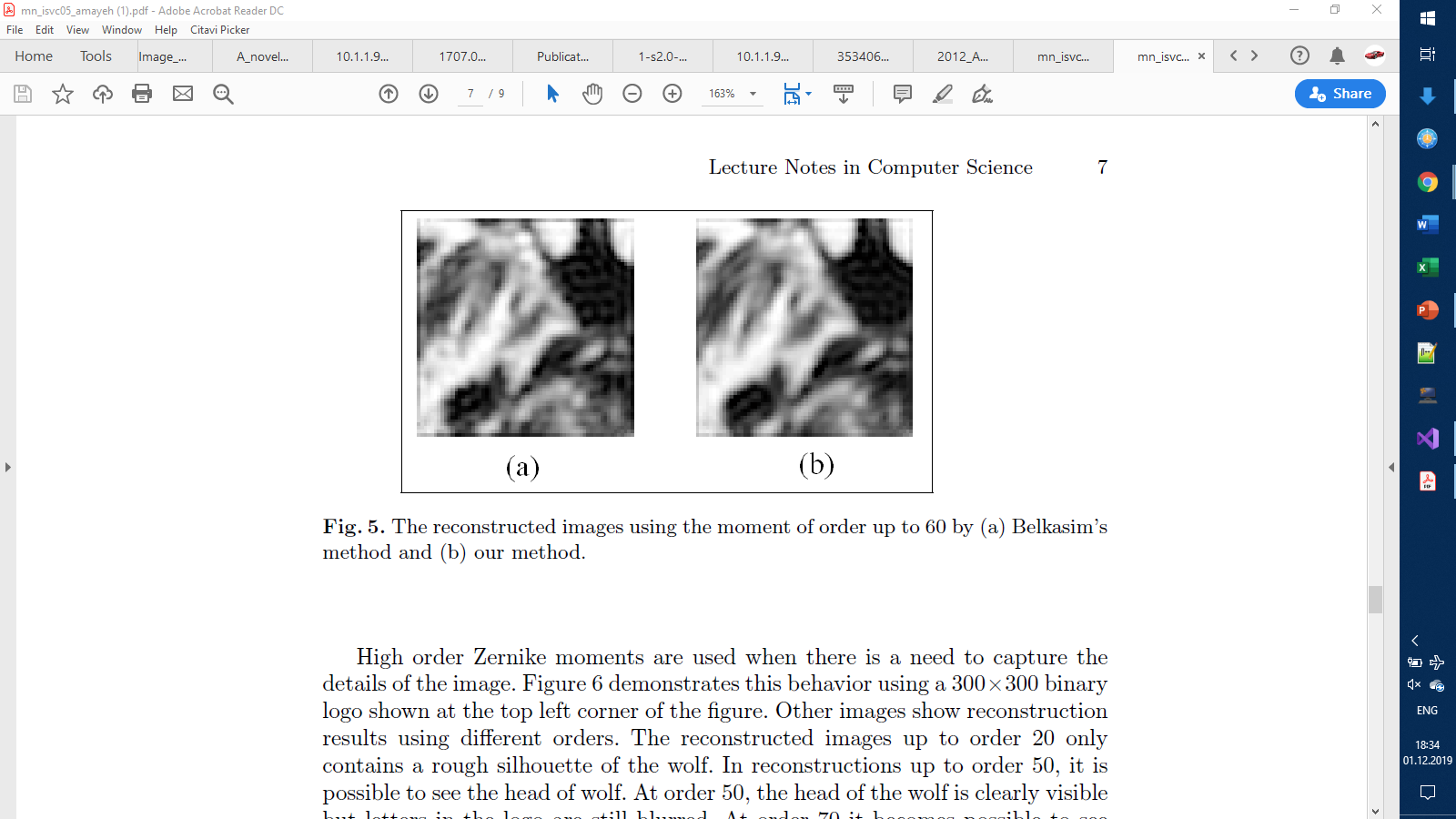


Abbildung 5: Rekonstruktion mit Zernike Momenten mit 60. Ordnung16

# PCA - Manuelles Matching

PCA (Principal Component Analysis) ist ein weit verbreitetes Verfahren zur Extraktion von hochdimensionalen Datenmerkmalen und zur Reduzierung der Dimension. Zunächst wird vorgestellt, wie PCA für unsere Aufgabe verwendet wird.

## 5.1 PCA - Principal Component Analysis

Die Hauptidee von PCA besteht darin, -dimensionale Merkmale auf -Dimensionen abzubilden. Diese k-Dimensionen sind neue orthogonale Merkmale, welche auch als Hauptkomponenten bezeichnet wird. PCA kann k-Dimension-Merkmale basierend auf den ursprünglichen n-Dimension-Merkmalen konstruieren.

Die Aufgabe von PCA besteht darin, nacheinander einen Satz von zueinander orthogonalen Koordinatenachsen aus dem ursprünglichen Raum zu finden. Die Auswahl der neuen Koordinatenachsen hängt eng mit den Daten selbst zusammen. Unter diesen ist die erste neue Koordinatenachse die Richtung mit der größten Varianz in den ursprünglichen Daten, die zweite neue Koordinatenachse die Ebene orthogonal zur ersten Koordinatenachse, um die Varianz zu maximieren, und die dritte Koordinatenachse ist dieselbe wie die erste. In der Ebene mit zwei orthogonalen Achsen ist die Varianz am größten. Analog können -te solcher Koordinatenachsen erhalten werden. Mit den auf diese Weise erhaltenen neuen Koordinatenachsen stellen wir fest, dass die meiste Varianz in den ersten k Achsen enthalten sind und die Varianz in den nachfolgenden Koordinatenachsen nahezu Null sind. Daher können wir die verbleibenden Achsen ignorieren und nur die ersten k Achsen betrachten, die den größten Teil der Varianz enthalten. Tatsächlich ist dies gleichbedeutend damit, nur die Dimensionsmerkmale beizubehalten, die den größten Teil der Varianz enthalten, und die Dimensionsmerkmale zu ignorieren, die nahezu Null Varianz besitzen, um eine Dimensionsreduktion der Datenmerkmale zu erreichen.

Wir berechnen zunächst die Kovarianzmatrix der Datenmatrix, erhalten dann die Eigenvektoren der Kovarianzmatrix und wählen eine Eigenwertmatrix aus, die den k Merkmalen mit den größten Eigenwerten (d.h den größten Varianzen) entspricht. Auf diese Weise können wir die Hauptkomponentenrichtungen erhalten, die den größten Unterschied enthalten, und die Datenmatrix in einen neuen Raum transformieren, um die Dimensionsreduktion der Datenmerkmale zu erreichen.

Der PCA-Algorithmus verfügt über zwei Methoden, um die Eigenwerteigenvektoren der Kovarianzmatrix zu erhalten: Ein PCA-Algorithmus basierend auf der Eigenwert-Dekompositions-Kovarianzmatrix und einPCA-Algorithmus basierend auf der SVD-Dekompositions-Kovarianzmatrix. Wir verwenden hauptsächlich die SVD-Methode, um den Merkmalsvektor zu erhalten, der dem größten Merkmalswert des Zielbildes entspricht, um nach der Dimensionsreduktion mehr gültige Informationen zu erhalten.

## Kovarianzmatrix

## SVD - Singular value decomposition

Ein Bild, das drinnen, Monitor, Bildschirm, sitzend enthält.

Automatisch generierte Beschreibung

Abbildung 6: Anwendung der OpenCV PCA17

Nach Translation & Rotation: **Subtraktion**

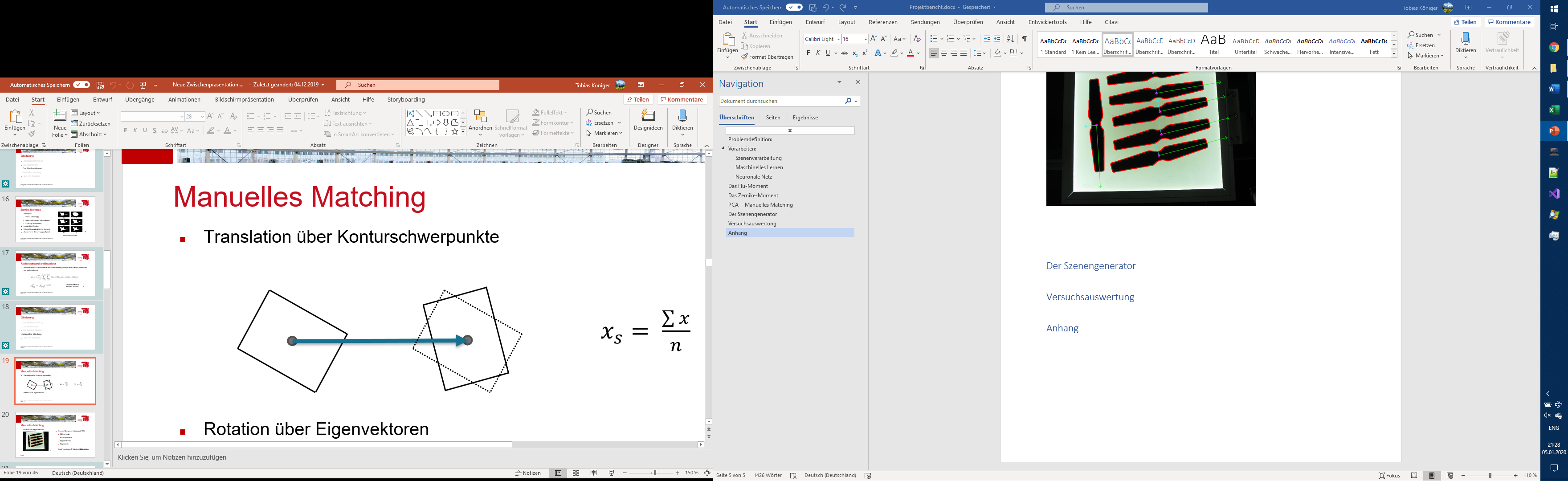
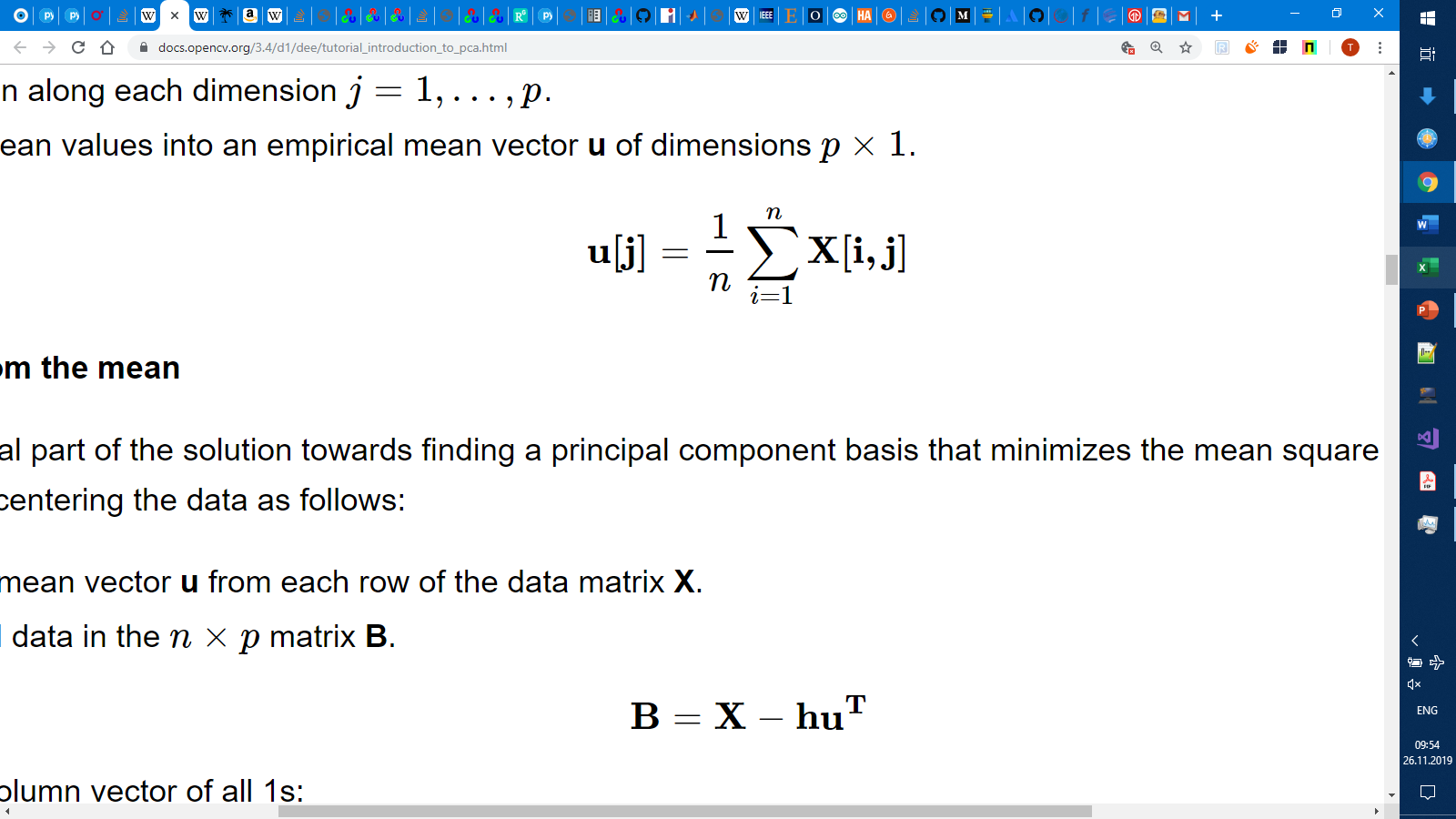
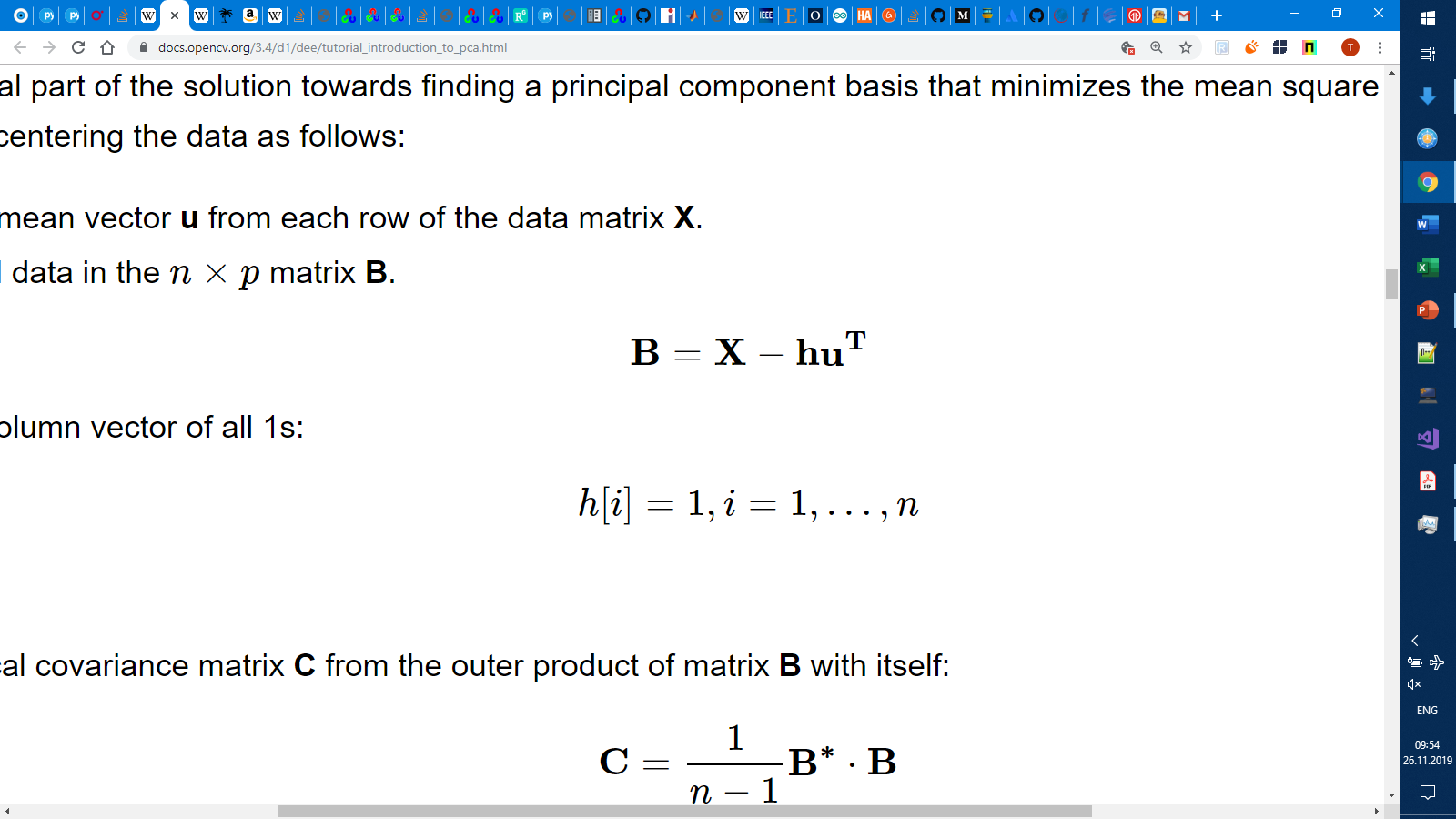
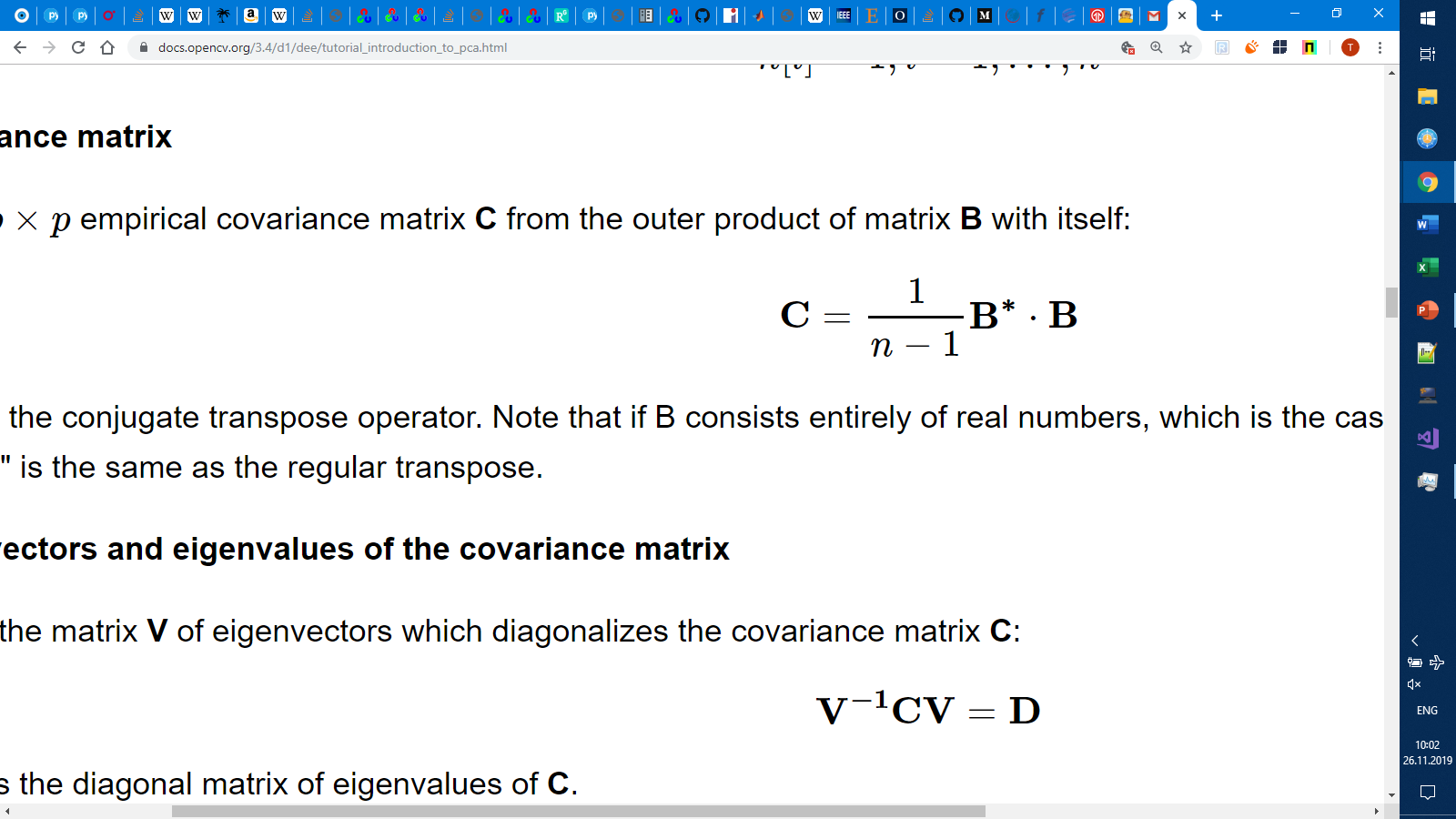


Abbildung 7: Translation + Rotation

## Berechnungsprinzip der PCA

* Bildschwerpunkt
* Abweichungsmatrix
* Kovarianzmatrix
* Eigenvektoren + Eigenwerte



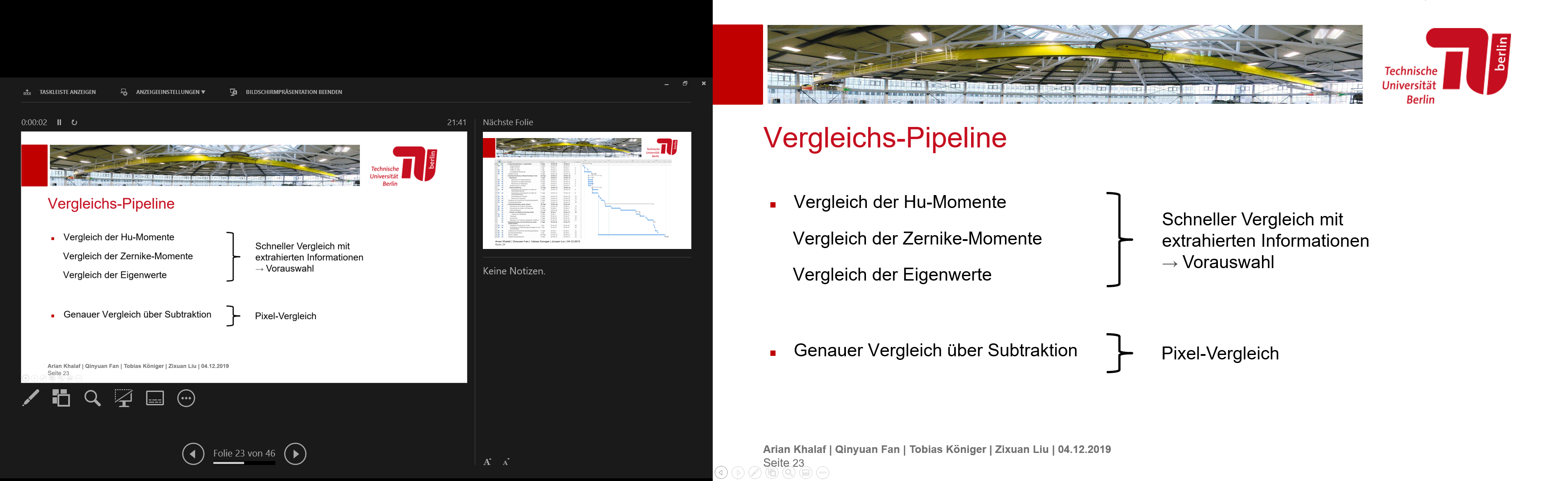
OpenCV Implementierung: PCACompute2()18

Sklearn Implementierung: sklearn.decomposition -> PCA19

# Vergleichspipeline

Zunächst werden mit den vorgestellten Verfahren Informationen aus den generierten Szenen extrahiert. Hierbei werden sowohl Hu-Momente, Zernike-Momente als auch Eigenwerte berechnet. Das Ziel besteht darin, dass auf der Grundlage der extrahierten Informationen eine möglichst genaue Vorauswahl getroffen werden kann. Wenn beispielsweise in der Anwendung am Demonstrator bereits eine Datenbank mit 300 Szenen besteht, sollen mithilfe der Parameter nur wenige Kandidaten-Szenen übrig bleiben.

Ein rechenaufwändigerer, genauerer Vergleich auf Basis der PCA mithilfe der Subtraktion der Szenen sollte die identischen Kandidaten-Szenen finden können.



# Implementierung

## Der Szenengenerator

Im ersten Schritt erzeugt der Generator zwei kompatible Objekte. Dazu werden vier verschiedene Kantenbilder miteinander kombiniert. Jeweils ein Bild für die obere, untere, rechte und linke Kante. Durch eine arithmetische Addition der vier Bilder wird das Objekt erzeugt.

Speziell müssen die beiden Bilder jeweils zunächst invertiert, dann addiert und anschließend wieder invertiert werden. Dadurch wird sichergestellt, dass bei der Addition die schwarzen Pixel (mit Grauwert 0) erhalten bleiben und nicht die weißen Pixel (255).

Für jede Kante existieren vier unterschiedliche Ausprägungen: Eine gerade Kante mit dreieckiger, rechteckiger und elliptischer Ausbuchtung, sowie eine ohne Ausbuchtung. Durch eine zufällige Auswahl der vier Kantenausprägungen können 44 = 256 unterschiedliche Objekte erzeugt werden.

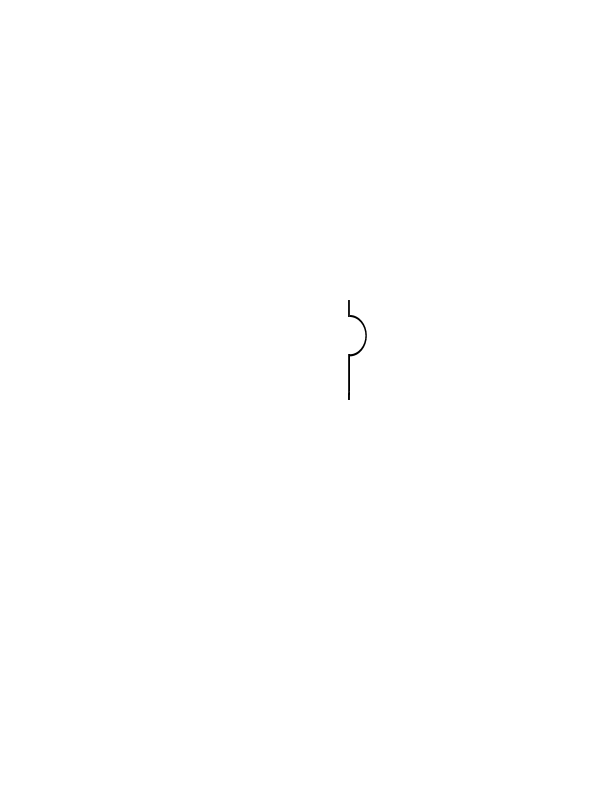
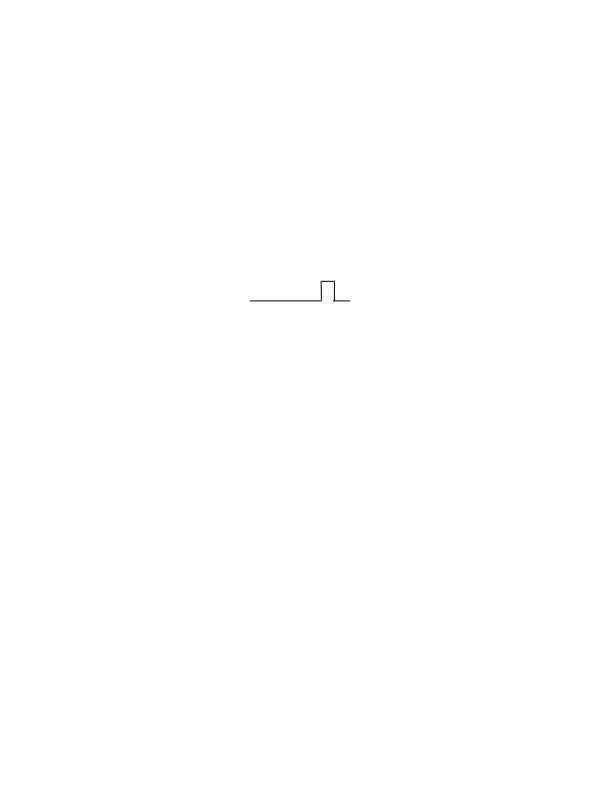
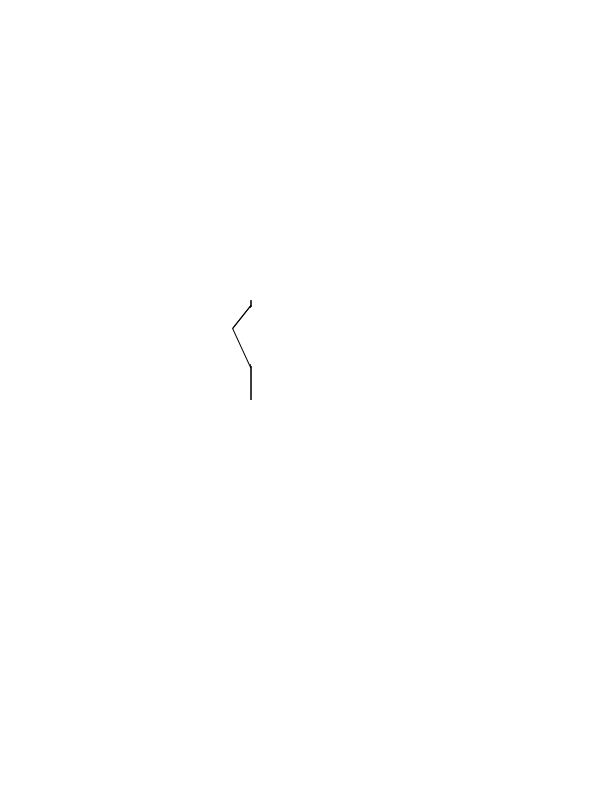


Abbildung 8: Vier unterschiedliche Ausprägungen für die linke, obere, untere und rechte Kante

Die Objekterzeugung wird ein zweites Mal ausgeführt, um das zweite Objekt zu erzeugen. Es muss jedoch darauf geachtet werden, dass eine Kante des zweiten Objekts (die obere) mit einer des ersten (die untere) übereinstimmt, da nur so eine Schweißoperation an der gemeinsamen Kante der beiden Objekte durchgeführt werden kann. Das zweite erzeugte Objekt wird anschließend nach unten verschoben.

Durch die zufällige Wahl der Kanten können 47 = 16384 einzigartige Szene generiert werden.

Zur Erzeugung der Szenen werden nun beide Objekte noch einer Translation und Rotation mit zufällig gewählten Parametern unterworfen.

Da die Szenengenerierung damit abgeschlossen ist, können nun sowohl identische Szenen, die sich nur durch Rotation und Translation unterscheiden, als auch unterschiedliche Szenen miteinander mit den gewählten Verfahren verglichen werden.

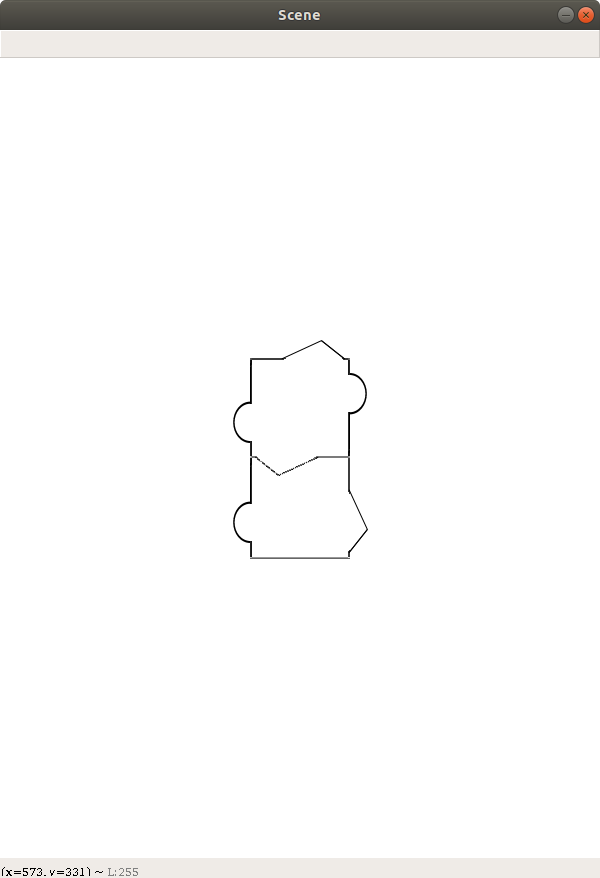
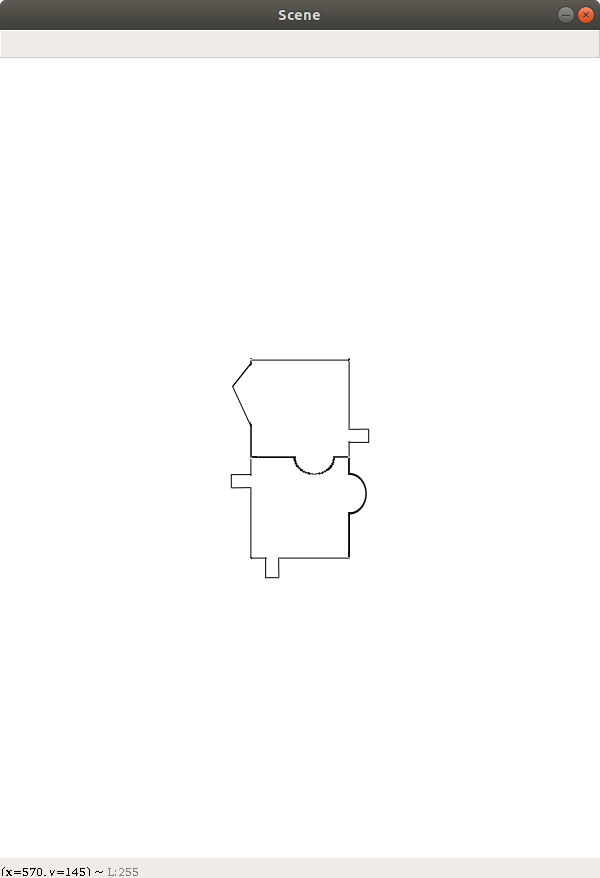
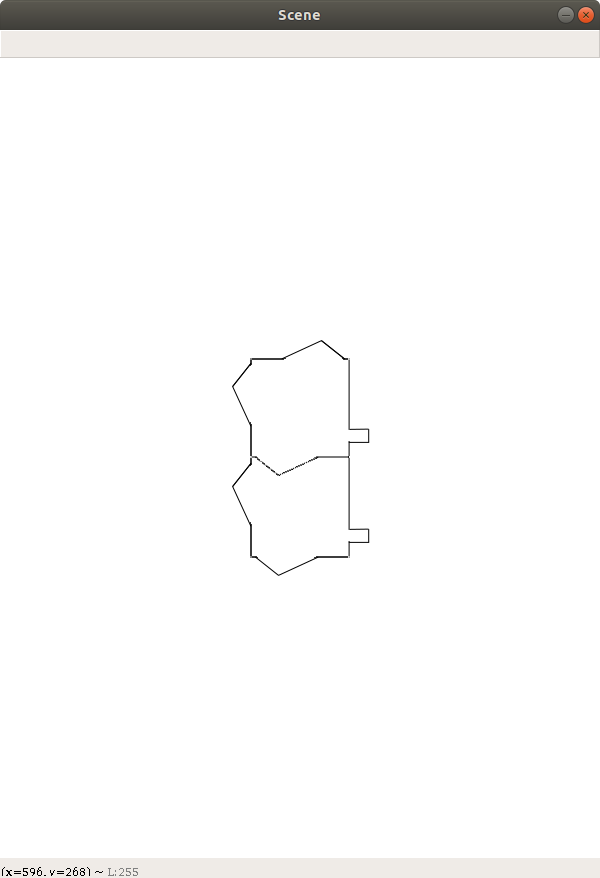


Abbildung 9: Drei zufällig erzeugte Beispielszenen mit rotierten und translatierten Objekten.

# Versuchsauswertung

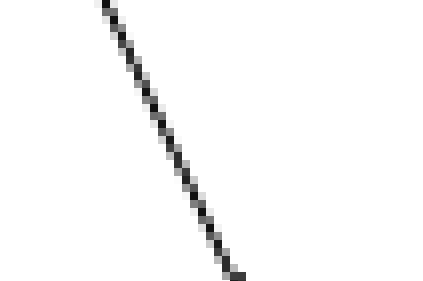
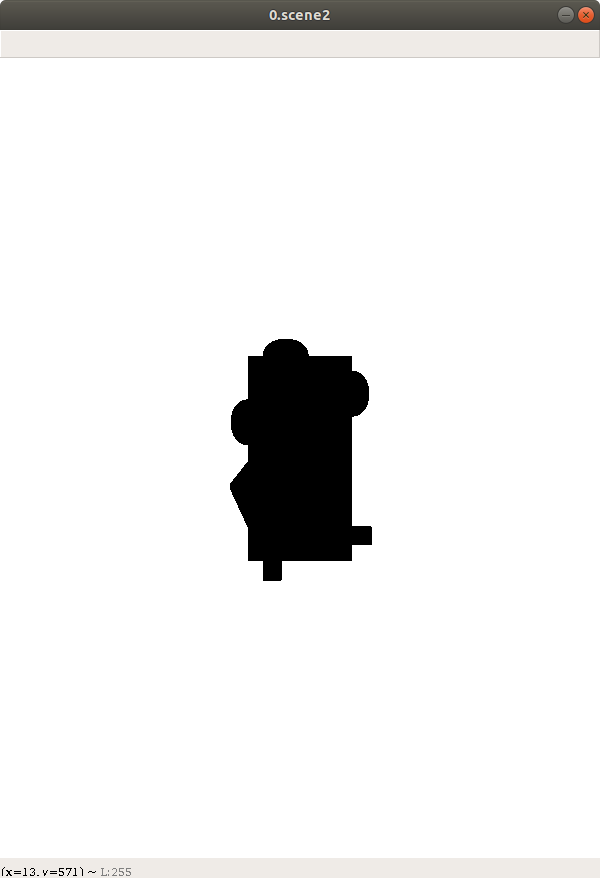
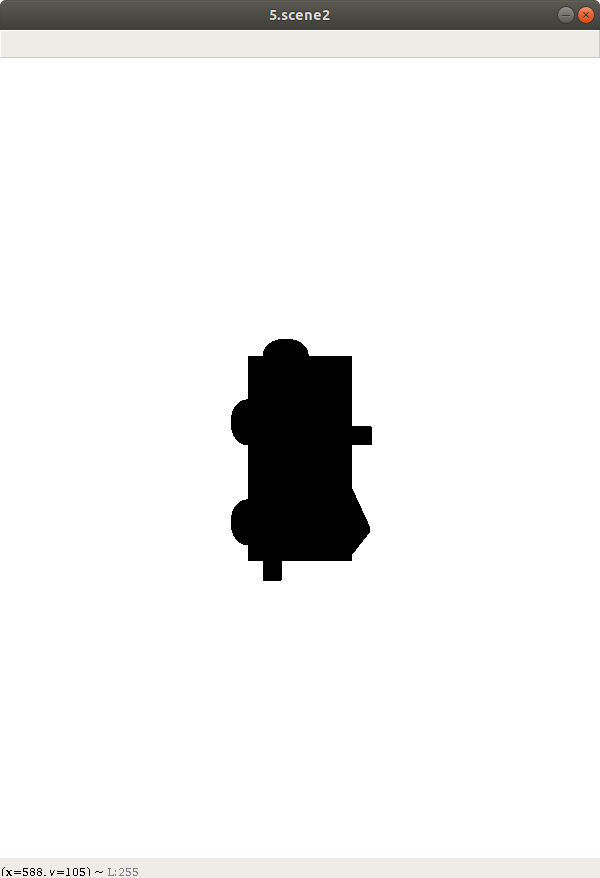
Im Verlauf der Versuchsdurchführung wurde festgestellt, dass der Aliasing-Effekt den Wirkungsgrad der Verfahren negativ beeinflusst. Der Aliasing-Effekt wird durch eine Unterabtastung der Kanten verursacht. Da Bildschirme nur eine begrenzte Auflösung besitzen, können schräge Kanten nicht exakt dargestellt

Abbildung 10: Aliasing-Effekt bei einem der generierten Objekte

werden. Die generierten Szenen werden zur Begrenzung der Rechenzeit nur mit einer Auflösung von 800x600 Pixeln erzeugt.

Aus diesem Grund wurden im Folgenden alle Objekte schwarz ausgefüllt. Das Aliasing-Problem wird dadurch verringert, kann jedoch nicht vollständig ausgeschlossen werden. Abbildung 11 zeigt die ausgefüllten zufällig generierten Objekte.

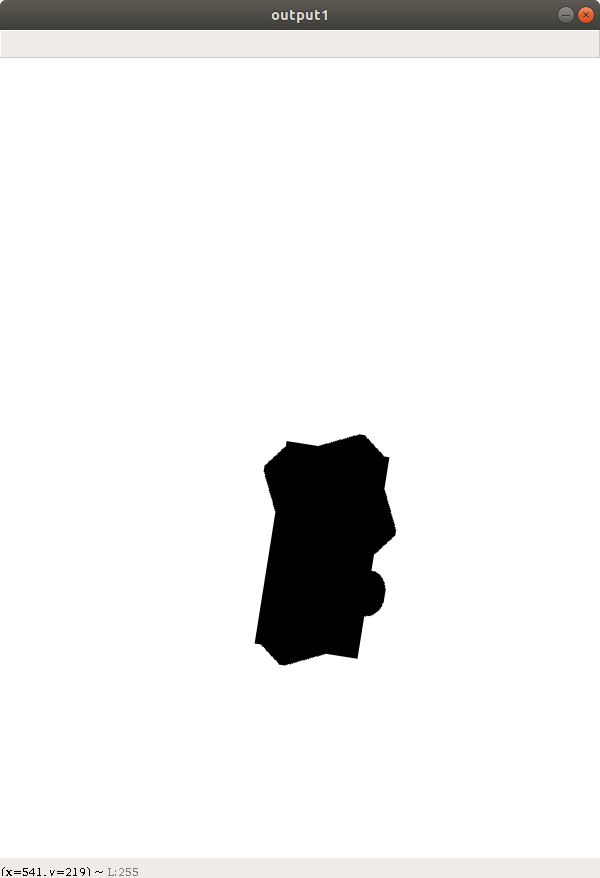
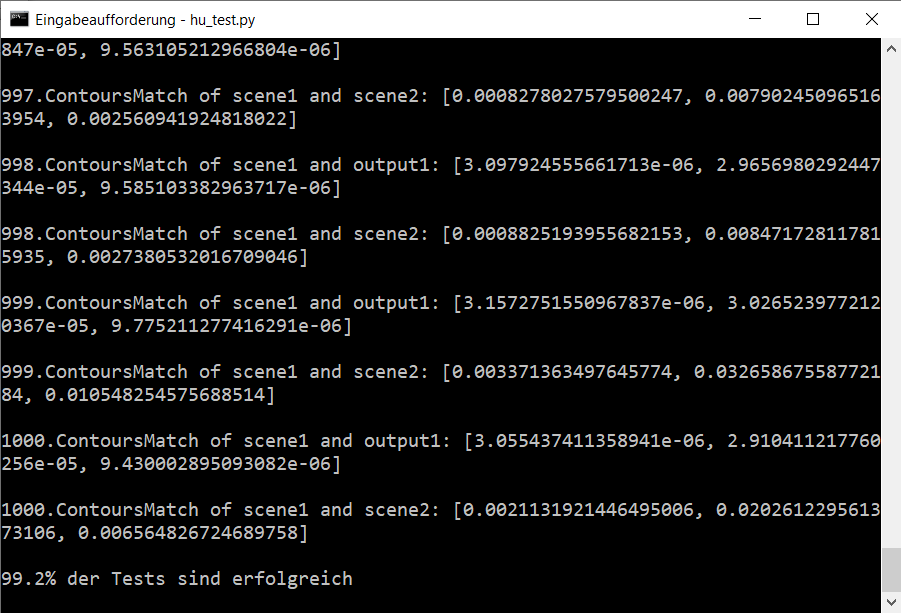


Abbildung 11: Zufällig erzeugte Objekte nach dem Ausfüllen

## Ergebnisse der Untersuchung der Hu-Momente



Im ersten Schritt unserer Versuchsreihe haben wir die Fehlerrate der ContoursMatch-Funktion von OpenCV mit einem Hu-Momenten-Testprogramm geprüft. Das Programm hat hierfür zwei Szenen generiert, wovon die erste Szene zufällig translatorisch und rotatorisch zu der Orginalszene verändert und mit der zweiten Szene verglichen wird. Dabei wird der ContoursMatch-Wert von der Orginalszene1 mit der veränderten Orginalszene und der ContoursMatch-Wert der veränderten Orginalszene mit der zweiten generierten Szene, gemäß den vorher beschriebenen Formeln, berechnet und verglichen. Falls dabei der ContoursMatch-Wert der veränderten Orginalszene mit der zweiten Szene größer ausfällt als der Wert der veränderten Orginalszene mit der unveränderten Orginalszene, so erfasst das Programm einen Fehlerfall und gibt die betroffenen Szenen aus. Der Stichprobenumfang / die Anzahl der Testfälle wird zu Beginn des Programms von dem Anwender abgefragt. Bei einem Stichprobenumfang von 1000 Fällen gab es in der Regel eine Fehlerrate von 1-2% bzw. in 98-99% der Tests waren die .

## 8.2 Ergebnisse der Versuche mit Zernike-Momenten

## 8.3 Ergebnisse des Szenenvergleichs mit PCA

# Anhang

## Quellen

1Vgl. https://www.informatik.hu-berlin.de/de/forschung/gebiete/viscom/thesis/final/Studienarbeit\_Kolbe\_201302.pdf  
2Vgl. https://learnopengl.com/Advanced-Lighting/Deferred-Shading  
3Vgl. <http://www.mttcs.org/Skripte/Pra/Material/vorlesung8.pdf>

4Vgl. <http://library.utia.cas.cz/separaty/historie/flusser-rotation%20moment%20invariants%20for%20recognition%20of%20symmetric%20objects.pdf>

5Vgl. <https://pdfs.semanticscholar.org/f094/f74196288aa6dd0376b4c0a463dd7d0e14a3.pdf>

6Vgl. <http://www.via.cornell.edu/ece547/text/survey.pdf>

7Vgl. <https://pdfs.semanticscholar.org/711b/e25acba94fbf913d2b734d312c27d5fe5226.pdf>

8Vgl. <http://www.decom.ufop.br/menotti/rp112/TrabalhoFinal-papers/SignLanguageRecognition.pdf>

9Vgl. <https://mahotas.readthedocs.io/en/latest/api.html>

10Vgl. <https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#double%20matchShapes(InputArray%20contour1,%20InputArray%20contour2,%20int%20method,%20double%20parameter)>

11Vgl. <https://www.learnopencv.com/wp-content/uploads/2018/12/HuMoments-Shape-Matching.png>

12Vgl. <https://www.math.uci.edu/icamp/summer/research_11/yoo/Zernike.pptx>

13Vgl. <https://pdfs.semanticscholar.org/b4d3/d68e098cde69b8608a8c61d12027ef9caf81.pdf>

14Vgl. <https://www.researchgate.net/publication/289961546_Image_Recognition_Using_Modified_Zernike_Moments>

15Vgl. <https://pdfs.semanticscholar.org/f12b/2b698d586e219bfa07a56615d1cefb8557e1.pdf>

16Vgl. <https://www.cse.unr.edu/~mircea/Publications/mn_isvc05_amayeh.pdf>

17Vgl. <https://docs.opencv.org/3.4/d1/dee/tutorial_introduction_to_pca.html>

18Vgl. <https://docs.opencv.org/3.4/d2/de8/group__core__array.html>

19Vgl. <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>

20Vgl. Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer

21Vgl. <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>

21Vgl.[https://www.researchgate.net/publication/261387261\_Local\_Zernike\_Moments\_A\_new\_representation\_for\_face\_recognitionhttps://www.researchgate.net/publication/261387261\_Local\_Zernike\_Moments\_A\_new\_representation\_for\_face\_recognition](https://www.researchgate.net/publication/261387261_Local_Zernike_Moments_A_new_representation_for_face_recognition)

22Vgl. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.431&rep=rep1&type=pdf>

## Projektplan

Ein Bild, das Karte enthält.

Mit sehr hoher Zuverlässigkeit generierte Beschreibung

Abbildung 12: Gantt-Chart des Projektplan

## Quellcode