Freeze Me! - Dokumentation

A Media Processing Project

Stand: 28.02.2023

Gruppe: Jan Rekemeyer, Iskander Yusupov und Hendrik Finke

1 Motivation

Da das Ziel des Projekts darin bestand, ein Softwareprodukt zu erstellen, entschied sich die Gruppe, eine Motivation im Marketingformat zu erstellen.
Glückliche Momente lassen sich in einem Foto festhalten, das dann verschenkt werden kann. Aber was tun mit den glücklichen Momenten, die auf Video festgehalten wurden? Mit diesem Ansatz suchte die Gruppe nach Möglichkeiten, ein Video so in Form eines Bildes festzuhalten, dass dieses später beispielsweise als gerahmtes Geschenk präsentiert werden könnte. Eine weitere Inspiration für unsere Gruppe waren Videos mit viel Dynamik. Sport, Tanz, aber auch Tiktok-Challenges enthalten viel Bewegung. Beim Fotografieren von sich bewegenden Objekten erzeugt die Einstellung einer Langzeitbeleuchtung den Effekt einer „eingefrorenen"Bewegung. Diese Fotos haben ihre eigene Ästhetik und unsere Gruppe wollte versuchen, diesen Effekt mit den im Modul erlernten Fähigkeiten zu reproduzieren.

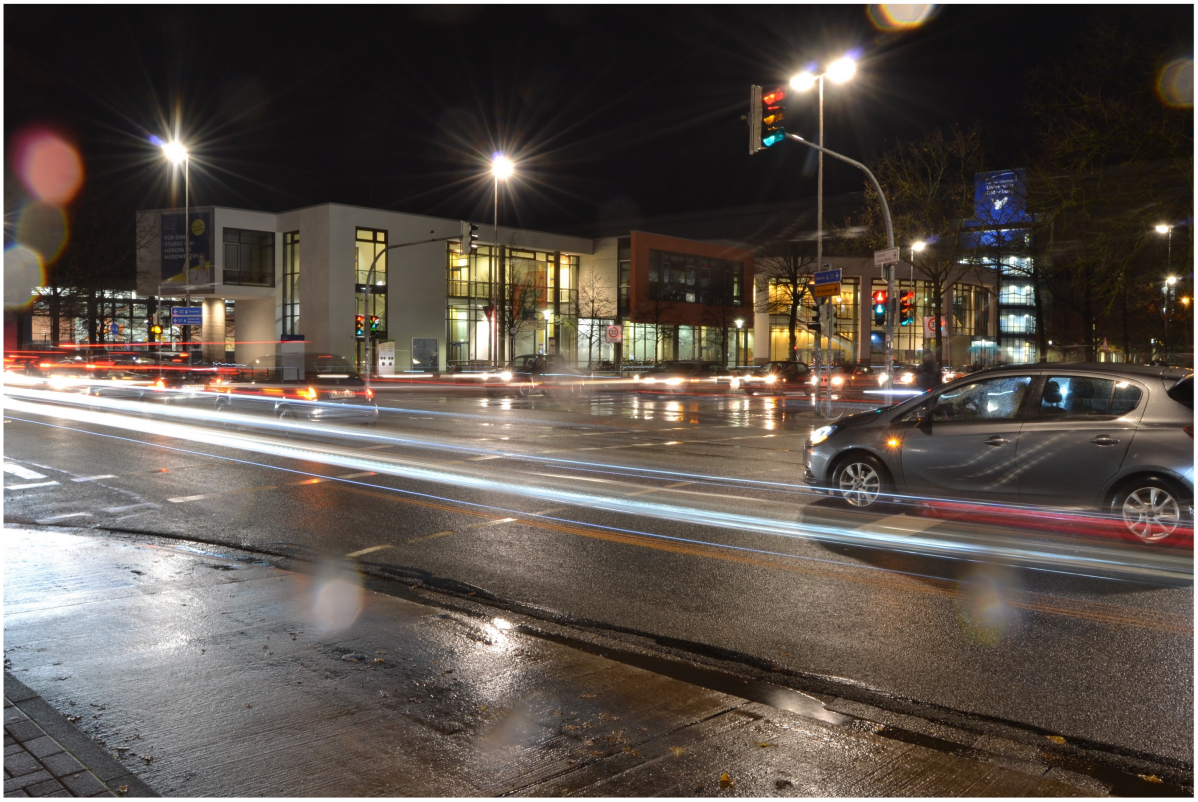
2 Zielsetzung

Unser Projekt haben wir der folgende Zielsetzung gewidmet:
Wir entwickeln ein Programm, welches es ermöglicht aus einem Video ein Bild zu erzeugen, welches die Bewegungsdynamik des Videos ästhetisch einfasst und dabei den Eindruck fortwährender Bewegung erhält.

3 Langzeitbelichtung - Inspirationsquelle

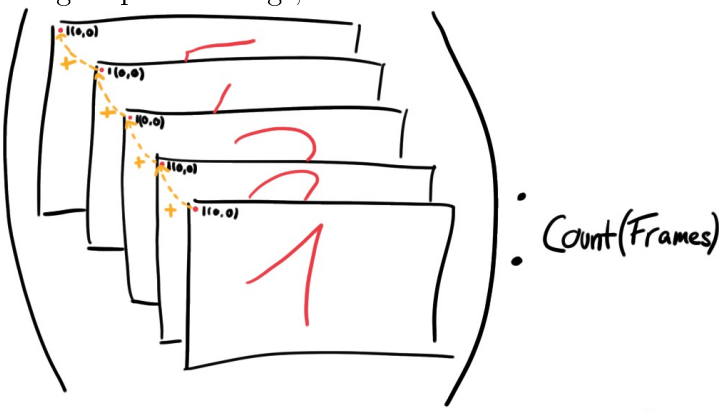
3.1 Motivation

Das in den Eröffnungsfolien gezeigte Bildmaterial erinnerte uns in Teilen doch recht stark an eine klassische Langzeitbelichtung, wie man sie aus der Analogfotografie oder der Fotographie mit DSLRs oder DSLMs kennt. Daher stellten wir uns die Frage, wie man dieses Konzept digital nachträglich unter Zuhilfenahme eines Videos anstelle eines Fotos reproduzieren kann.



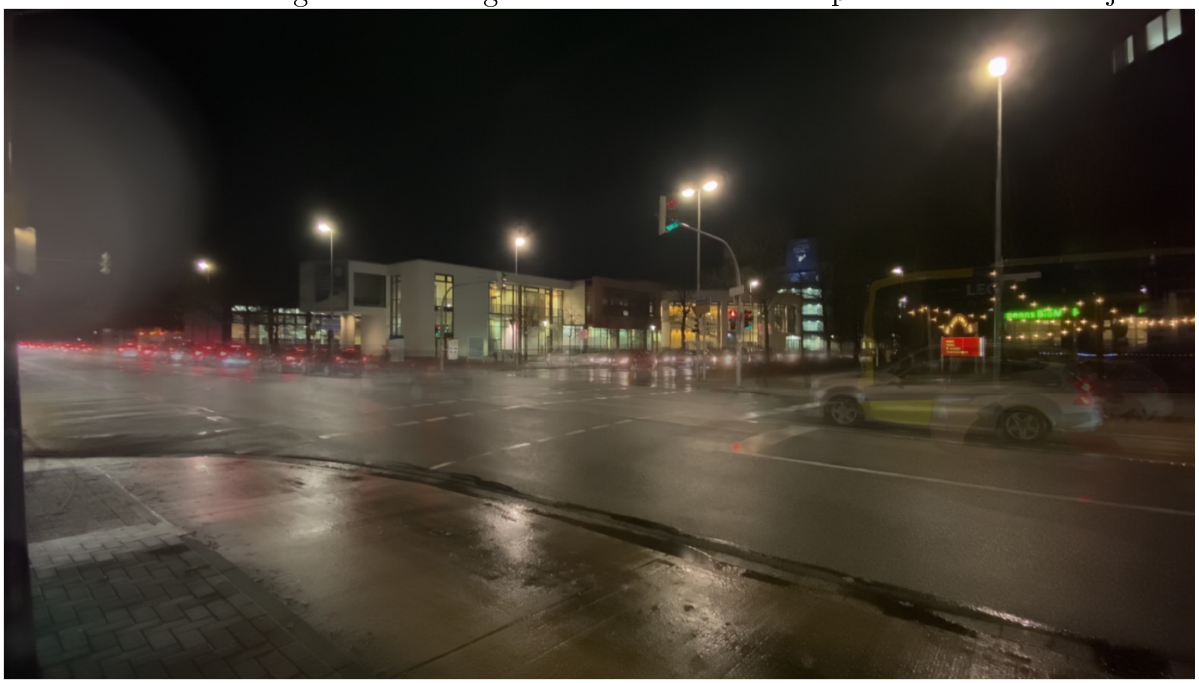
3.2 Funktionsweise

Der Ansatz, den wir hierbei Verfolgt haben ist relativ trivial. Ähnlich wie klassiche Filter den Inhaltswert eines Pixels anhand einer ggf. faktorisierten Durchschnittsberechnung mit den umliegenden Pixeln berechnet tun wir ebendieses - jedoch mit den Pixeln gleicher Koordinaten auf anderen Videoframes. Zu diesem Zwecke werden Videos Frame für Frame betrachtet und dann Pixel für Pixel in ihre jeweiligen 3 Farbkomponenten zerlegt. Im folgenden wird der Durchschnittswert der Farbwerte errechnet und somit eine Art Langzeitpixel erzeugt, deren Summe das künstlich Langzeitbelichtete Bild ergibt.



3.3 Resultate

Die Resultate waren abhängig vom eingegebenen Video durchaus vielversprechend. Ein von uns anfänglich eigens erstelltes Video von der Kreuzung der Ammerländer Heerstraße mit dem Uhlhornsweg resultierte in einem mehr oder minder matschigen Bild. Das Ergebnis des Contemporary Dancers hingegen war schon erstaunlich dicht an den Hintergrunderscheinungen der Fotos in den Veranstaltungsfolien. Auch das TikTok Video mit dem Sprung resultierte in einem relativ ansehnlichen Ergebnis. Da dieser Ansatz somit der erste nennenswerte Schritt in die Richtung des Gesamtzieles war wurde er beibehalten und die Langzeitbelichtung wurde somit Grundkomponente unseres Projektes.



4 Grabcut mit Thresholding

4.1 Motivation

Mit der Langzeitbelichtung haben wir einen ästhetischen Schleiereffekt, der im Video stattfindenden Bewegung, geschaffen. Dieser Schleier muss aber noch in ein Bild eingearbeitet werden, sodass der Effekt entsteht, dass die Bewegungen innerhalb eines Videos in Abhängigkeit zu einem Hauptbild, einem Frame des Videos, sind. Um dieses zu bewerkstelligen, muss die Langzeitbelichtung mit seinem Schleiereffekt sich um die bewegende Objekte des gewählten Frames legen. Die sich bewegenden Objekte trennen wir vom Rest des Bilds durch den Foreground Detection Algorithmus GrabCut und verarbeiten diese rausgeschnittenen Bilder zu einer Maske mittels Thresholding.

4.2 Theorie

Der GrabCut Algorithmus schätzt innerhalb eines definierten Bereichs die Farbverteilung des Vorder- und Hintergrund mithilfe eines Gauß'schen Mischungsmodells. Die daraus resultierenden Pixeletiketten werden zu einem Graphen zusammengesteckt, in Abhängigkeit von der Anzahl verbundener Pixelregionen, die das gleiche Pixeletikett besitzen, sodass eine auf Graphenschnitt basierende Optimierung auf diesem Graphen durchführt werden kann. Das genannte Verfahren kann mit dem resultierenden Ergebnis wiederholt werden bis zur Konvergenz.

4.3 Codebeschreibung

Zu Beginn wird das Video, welches den Schleiereffekt erhalten soll, bis zum Ende gespult, der letzte Frame ausgelesen und als separate JPG-Datei abgespeichert. Danach werden Hilfsvariablen für das Hintergrundmodel, Vordergrundmodel, Maske und auszuschneidende Rechteck initialisiert. Der GrabCut-Algorithmus, der mit den Hilfsvariablen benutzt wird, ist von OpenCV. Der GrabCut-Algorithmus generiert eine Maske, welche mit dem letzten Frame des Videos verrechnet wird, sodass ein Bild von den sich bewegenden Objekten mit sonst schwarzem Hintergrund generiert wird. Die farbigen sich bewegenden Objekte werden dann noch mittels Thresholding wei eingefärbt, sodass die daraus resultierende Maske beim Zusammenführen des Schleiereffekts und des letzten Frames verwendet werden kann. Nach der Zusammenführung bleibt das finale Bild.

4.4 Resultate

Dieser Bildverarbeitungsschritt beschränkt sich derzeit auf das aus unseren Präsentationen bekannte Video vom *contemporary dancer*, da die Hilfsvariable für das auszuschneidende Rechteck fest im Programmcode hinterlegt ist. Die Maske des sich bewegenden Objekts ist in Abbildung 1a, das Bild des letzten Frames in Abbildung 1b, die Langzeitbelichtung in Abbildung 1c und das finale Ergebnisse in Abbildung 1d zu sehen. Bei näherer Betrachtung des finalen Ergebnis ist zu erkennen, dass die Ränder des eingesetzten bewegenden Objekt grafische Artefakte enthält.

5 Edge Detection

5.1 Motivation

Nachdem es der Gruppe gelungen war, Bilder mit einer künstlichen Langzeitblichtung aus dem Video zu erstellen, fiel ein Merkmal auf. Wurde das Video ohne Stativ oder jegliche Art von Unterstützung gedreht, dann führte das leichte Wackeln der Kamera irgendwann dazu, dass sich auch stabile Objekte im Ausgabebild als unscharf herausstellten. Um dieses Merkmal oder Problem zu lösen, wurde entschieden, Canny Edge Detection zu verwenden. Die Idee, der Algorithmus zu verwenden, wurde experimentell geboren. Wir haben versucht, Canny Edge Detection auf ein bereits bearbeitetes Bild

anzuwenden. Die Methode erkannte nur Objekte als Kanten, die auf dem Video stabil waren. Dann entschied die Gruppe, dass es möglich ist, Schärfungsverfahren (z. B. einen Laplace-Filter) auf die Pixelkoordinaten mit stabilen Objekten anzuwenden, die mithilfe des Canny Edge Detection auf dem Ausgabebild erhalten werden.

5.2 Theorie

Der Canny Edge Detection ist ein mehrstufiger Algorithmus zum Erkennen von Kanten in einem Bild. Der Algorithmus besteht aus vier Stufen. In der ersten Stufe ist es notwendig, Rauschen aus dem Originalbild zu entfernen, für diese Aufgabe wird der Gauß Filter angewendet. Der zweite Stufe besteht darin, den Größe entlang der x- und y-Dimension zu erhalten. Dazu wird die Ableitung des Gauß-Filters berechnet, um dann das Gradient der Bildpixel zu berechnen. Der Algorithmus geht alle Punkte der Gradientenintensitätsmatrix durch und findet die Pixel mit dem maximalen Wert in den Kantenrichtungen. Das Betrachten der Gruppe von Nachbarn für jede Kurve in einer Richtung senkrecht zu der gegebenen Kante unterdrückt nicht maximalen Kantenbeitragspixelpunkte. Schließlich wird das Hysterese-Schwellenwertverfahren verwendet, um die Pixel zu bewahren, die höher als die Gradientengröße sind, und diejenigen zu vernachlässigen, die niedriger als der niedrige Schwellenwert sind.

5.3 Codebeschreibung

Zur Optimierung der Ergebnisse wird ein Verfahren zur Bestimmung der Schwellwerte (minVal und maxVal) eingesetzt. Die Schwellwerte werden dann in der OpenCV Methode cv.Canny übergegeben.

5.4 Ergebnis

Nach dem Ausführen des Codes ist das erwartete Ergebnis ein Bild mit identifizierten Kanten. Nach Auswertung der Ergebnisse entschied die Gruppe, dass der resultierende Bildschärfungseffekt nicht effektiv genug war, und entschied sich daher, Canny Edge Detection aufzugeben.

6 Background Subtraction

6.1 Motivation

Die Gruppe stand vor dem Problem, dass manchmal ein sich bewegendes Objekt im Video nach Anwendung der künstlichen Langzeitblichtung weniger unterscheidbar wurde. Um Kontrast und Transparenz hinzuzufügen, wurde entschieden, dem endgültigen Bild ein sich bewegende Objekt hinzuzufügen, das aus einem der Videoframes geschnitten

wurde. Zu diesem Zweck entschied sich die Gruppe, zwei Background- und Foregroundsubtraction Methoden von OpenCV anzuwenden:K-Nearest Neighbors (KNN) und Gaussian Mixture-based (MOG2).

6.2 Theorie KNN

Der erste Schritt besteht darin, die Anzahl K der Nachbarn zu wählen. Dann ist es notwendig, den euklidischen Abstand der Anzahl von K Nachbarn zu berechnen. Basierend auf der berechneten euklidischen euklidischen Distanz werden K nächste Nachbarn ausgewählt. Unter diesen k Nachbarn wird die Anzahl der Datenpunkte in jeder Kategorie gezählt. Am Ende werden neue Datenpunkte der Kategorie zugeordnet, für die die Anzahl der Nachbarn maximal ist.

6.3 Theorie MOG2

Die Mix of Gaussian (MOG)-Hintergrundsubtraktion ist abhängig von einer Kombination von Frames anstelle von nur einem Frame. Bei diesem Verfahren wird für jedes Hintergrundpixel eine Mischung aus einer Gaußschen Verteilung k und einem Gewichtungsparameter verwendet, um die Lebensdauer von Pixeln in der Szene zu speichern, wobei k im Bereich von 3 bis 5 variieren kann. Verbleibende Pixel mit einer Zeit größer als a Schwellen in der Szene haben eine höhere Wahrscheinlichkeit, zur Hintergrundszene zu gehören. Wenn das Pixel für einen bestimmten Zeitraum unverändert bleibt, wird es als dominantes Hintergrundpixel betrachtet. Wenn die Differenz von Pixeln in einem Frame größer als ein vordefinierter Schwellenwert ist, werden sie als bewegliche Teile klassifiziert. Diese Methode reagiert sehr empfindlich auf Veränderungen in der Umgebung. In unserem Projekt wird MOG2 verwendet. Der Unterschied besteht darin, dass MOG2 die geeignete Anzahl von Gaußschen Verteilungen für jedes Pixel auswählt, wobei MOG eine K-Gaußsche Verteilung für die Modellierung verwendet. Aus diesem Grund bietet MOG2 eine bessere Anpassungsfähigkeit an wechselnde Szenen aufgrund von Beleuchtungsänderungen.

6.4 Codebeschreibung

Der Code der Funktion "get_foreground_mask"überprüft das Argument foregroundMaskOption",
um zu bestimmen, ob die Methode cv.createBackgroundSubtractorMOG2()öder cv.createBackgroundSu
die Hintergrundsubtraktion verwendet werden soll. Der nächste Schritt besteht darin, eine URL zum Herunterladen des angegebenen Videos mit videoId und cldId als Parameter zu erstellen. Danach lädt es das Video mit urllib.request.urlretrieve() von der URL auf einen lokalen Pfad herunter. Dann wird das VideoCapture-Objekt aus dem heruntergeladenen Video erstellt. Für jeden Frame des Videos wird eine Hintergrundsubtraktion angewendet. Die resultierende Vordergrundmaske wird in einer Datei gespeichert. Der Code der Funktion "get_background_imageïst ähnlich geschrieben, speichert aber auch

Hintergrundbild.

6.5 Ergebnis

Das Ergebnis beider Methoden war für die Aufgabenstellung nicht geeignet. Die unter Verwendung dieser beiden Algorithmen erhaltenen Masken waren nicht monolithisch, sondern hatten Lücken darin. Eine Maske mit Lücken war nicht tauglich. Daher entschied sich die Gruppe für eine andere Methode für Hintergrund- und Vordergrundsubtraktion.

7 Fazit

7.1 Limitierungen

Die Erkennung des Vordergrundes beschränkt die Funktionsweise der Software in Teilen. Von den anfänglich 3 angenommenen Testvideos (Tänzer, Kreuzung, TikTok Sprungvideo) funktioniert lediglich der Tänzer wirklich gut. Das Ergebnis des TikTok Sprungvideos ist ebenfalls durchaus annehmbar, wobei hier das Hinzufügen eines maskierten Standbildes weitestgehend wertlos ist. Zu diesem Zwecke müsste ein Framepicker in der Software implementiert werden. Gleichermaßen ist der Kontrast der Hintergrundeffekte relativ stark beschränkt, was ein Betrachten der Resultate in schlechten Lichtverhältnissen oder gar auf Beamern stark einschränkt. Um diesen Effekt zu verstärken hätte eine helligkeitsbedingte Faktorisierung genutzt werden können, die wir anfänglich mal betrachtet haben, deren Notwendigkeit uns im Laufe des Projektes nicht mehr sinnvoll erschien. Retrospektiv wäre diese Funktion ein sinniges Opt-In gewesen.

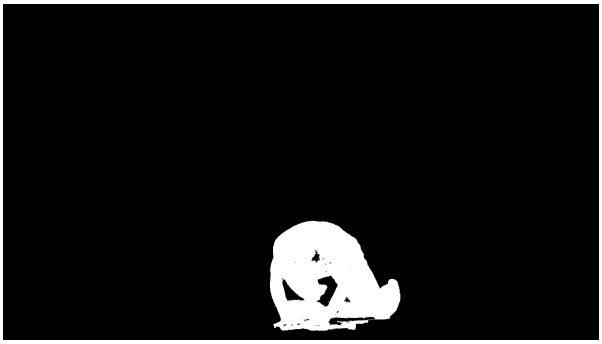
7.2 Ausbaumöglichkeiten

Für eine künftige Ausgestaltung bieten sich die im letzten Block genannten Optionen an: Eine Optimierung des Interfaces um ein Frame zu selektieren, die Option im entsprechenden Frame ein Rechteck zu wählen, das dann an GrabCut übergeben wird um den Vordergrund zu bestimmen. Dazu wäre es eine sicherlich gute Option gewesen die Lichtstärke in die Langzeitbelichtung einfließen zu lassen, um den Kontrast der Schlieren im Hintergrund zu steigern. DarÃ¼ber hinaus blieben sicherlich noch Optimierungen an der GUI und der Effizienz und damit der Performanz des Systems.

7.3 Resumee

Das letztendliche Ergebnis unserer Arbeit entspricht unser Meinung nach dem ursprünglichen Ziel - aufgrund der Subjektivität der Ästhetik aber lässt sich dieses Urteil wohl kaum objektiv fällen. Mit der Anwendung der Langzeitbelichtung fand sich für uns relativ

früh eine Möglichkeit, einen Hintergrund mit den verwaschenen bzw. verschwommenen Hintergrundartefakten zu erzeugen, durch das Ergänzen um ein einzelnes, maskiertes Videoframe ist dabei auch die Charakteristik eines Fotos gewÃ□hrt.



(a) Maske

(b) Letzer Frame

(c) Langzeitbelichtung

(d) Langzeitbelichtung + Letzter Frame

Abbildung 1: GrabCut Zwischenschritte und Ergebnis