# Computer Vision Challenge

Klaus Diepold
Stefan Röhrl
Lehrstuhl für Datenverarbeitung
Technische Universität München

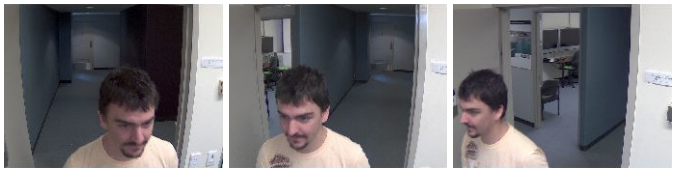
9. Juli 2020

## Zusammenfassung

Zur Zeit finden immer mehr Video-Calls und Konferenzen in den eigenen vier Wänden statt. Dabei möchte man vielleicht nicht unbedingt das unaufgeräumte Arbeitszimmer, die heimische Küche oder die Familienfotos an der Wand hinter dem Sofa an Vorgesetzte und Kollegen streamen. Die Computer Vision Challenge beschäftigt sich daher in diesem Jahr mit der Unterscheidung von Vorder- und Hintergrund, sowie mit der Möglichkeit, ungewollte Szenenbestandteile zu erstzen.

## 1 Datenset

Die Datenbasis bildet das ChokePoint Dataset[1], welches aus Aufnahmen von Überwachungskameras an verschiedenen Portalen ($P1, P2$) besteht. Es gibt jeweils drei Kameras ($C1, C2, C3$), welche simultan die gleichen Szenen ($S1$ - $S5$) filmen. Jedes Portal wurde zudem einmal von innen ($E$ enter) und von außen ($L$ leave) aufgenommen. Die dabei entstandenen Bilder haben eine Größe von $800 \times 600$ Pixel und wurden mit $30$fps aufgenommen. Drei gleichzeitige Frames aus dem Szeneordner *P1E_S4* sind in der folgenden Abbildung dargestellt.



(a) Kamera 1 (b) Kamera 2 (c) Kamera 3

Abbildung 1: Portal 1, Enter, Szene 4

In Abbildung 2 sehen Sie einen Ausschnitt aus der Ordnerstruktur des Datensets. In Sie können das gesamte Datenset hier herunterladen. Beim Ausführen des Programms können Sie davon ausgehen, dass sich der Name des Kameraordners (*P1E_S2_C1*) immer aus dem des Szenenordners (*P1E_S2*) und der Nummer der Kamera (*_C1*) zusammensetzt.

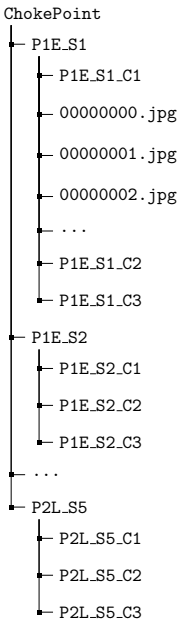



Abbildung 2: Ordnerstruktur ChokePoint Dataset

## 2 Spezifikation

Im Folgenden finden Sie die detaillierte Aufgabenbeschreibung der Computer Vision Challenge 2020. Wir bitten Sie, diese genau zu lesen. Diese Aufgaben sind verpflichtend für alle Gruppen.

### 2.1 Image Reader

Schreiben Sie eine Klasse `ImageReader`, welche in der Lage ist, zwei der drei Videostreams aus einem Szenenordner als Endlosschleife abzuspielen.

**Initialisierung** Die Instanzierung der Klasse

```
ir = ImageReader(src, L, R, start, N)
```

soll einen beliebigen Dateipfad zu einem Szenenordner als Quelle `src` erhalten. Beispiele für `src` sind relative Pfade *"../files/ChokePoint/P1E_S2"* als auch absolute Pfade *"C:\Users\Hannes\CV\ChokePoint\P2L_S4"* in Windows

oder Unix Style. Die Parameter `L` und `R` wählen aus, welche Kamera als linkes oder als rechtes Bild verwendet werden soll. Somit kann `L` die Werte $\{1, 2\}$ und `R` die Werte $\{2, 3\}$ annehmen. Je nach Auswahl sollen die entsprechenden Unterordner *P**_S*_C** als Quellen für die Stereobildpaare genutzt werden. Der Parameter `start` ist optional und gibt an, ab welcher Framenummer die Szene abgespielt werden soll. Ist kein Wert für `start` angegeben, beginnt das Programm bei 0. Der Wert `N` gibt an, wie viele Nachfolgebilder pro Lesevorgang geladen werden sollen. Standardmäßig ist $\texttt{N} = 1$, um den aktuellen Frame und dessen Nachfolger zu laden.

**Laden von Bildern** Um eine entsprechende Menge an Bildern aus den Kameraordnern zu laden, soll die Klasse die Methode

```
left, right, loop = ir.next()
```

bereitstellen. Beginnend bei dem in `start` angegebenen Wert liefert die Methode das nächste Stereobildpaar und `N` nachfolgende Bildpaare. Die Frames der linken Kamera werden als Tensor `left`, sowie die Frames der rechten Kamera als Tensor `right` zurück gegeben. Beide Tensoren haben die Dimensionen $600 \times 800 \times (N + 1) \cdot 3$, da es jeweils `N`+1 Bilder mit drei Farbkanälen sind. Der Wert `loop` ist dabei standardmäßig auf 0. Sollten im Ordner nicht mehr genügend Bilder zur Verfügung stehen, werden unabhängig von `N` nur noch die vorhandenen Nachfolger zurückgegeben und `loop` auf 1 gesetzt. Beim nächsten Aufruf von `next()` soll unabhängig vom Wert `start` wieder am Anfang der Szene (Bild *00000000.jpg*) angefangen werden. Beispiel für `N`=3: Beim letzten Aufruf von `next()` wurde das Stereobildpaar zur Framenummer 42 geladen. Wird nun die Funktion `next()` erneut aufgerufen befinden sich in `left` die Frames aus den Bildern 00000043.jpg, 00000044.jpg, 00000045.jpg und 00000046.jpg der linken Kamera. Das gleiche gilt analog für den rechten Kamerastream. Die Framenummer bezieht sich hier auf die Position des Bildes im Ordner, da Sprünge in der Benennung der Bilder (z.B. 00000098.jpg, 00000099.jpg, 000000325.jpg, 000000326.jpg...) auftreten können.
Wenn Sie eine kleine Anzahl an Frames geladen haben, spielt es keine große Rolle, welchen Frame Sie bearbeiten. Angenommen Sie benötigen immer zwei Vorgänger und zwei Nachfolger Bilder zum aktuellen Frame, dann können Sie N=4 setzen. Sie erhalten dann fünf aufeinander folgende Bildpaare (1,2,3,4,5). Sie können dann Bild 3 als aktuellen Frame ansehen und dessen Maske berechnen. Bitte achten Sie dabei auf eine konsistente Umsetzung im ganzen Programm, da dann immer der dritte Frame aus einem Tensor der aktuelle Frame ist.

## 2.2 Segmentation

Nutzen Sie in der Funktion

```
mask = segmentation(left, right)
```

die beiden zuvor erstellten Tensoren `left` und `right` mit aufeinanderfolgenden Bildpaaren, um Vordergrund und Hintergrund für das erste Bildpaar zu schätzen. Sie können die Tiefeninformationen des Stereobildpaares nutzen, müssen dies aber nicht tun. Geben Sie für das aktuelle Bild der linken Kamera eine binäre Segmentierungsmaske `mask` zurück, welche die gleiche Größe hat wie das Bild selbst. Pixel, die dem Hintergrund zugeordnet werden, erhalten den Wert 0. Pixel, die zum Vordergrund gehören, bekommen den Wert 1.

## 2.3 Rendering

Schreiben Sie eine Funktion

```
result = render(frame, mask, bg,
render_mode),
```

welche das aktuelle Bild der linken Kamera `frame` mit Hilfe der dazugehörigen Segmentierungsmaske `mask` verarbeiten kann. Der Parameter `render_mode` wählt dabei zwischen folgenden Modi aus:

- **foreground**: Der Hintergrund wird auf Schwarz gesetzt. Der Vordergrund im Bild soll so wenig wie möglich verändert werden.
- **background**: Der Vordergrund wird auf Schwarz gesetzt und nur noch der Hintergrund ist zu sehen.
- **overlay**: Vordergrund und Hintergrund werden mit gut unterscheidbaren Farben transparent eingefärbt.
- **substitute**: Ersetzt den Hintergrund durch einen virtuellen Hintergrund welcher in `bg` übergeben wird.

Der virtuelle Hintergrund `bg` ist dabei ein RGB-Bild mit beliebiger Größe $N \times M \times 3$.

## 2.4 Konfiguration

Um Ihre rechnerspezifischen Einstellungen von der Berechnung zu entkoppeln, vervollständigen Sie die Datei *config.m*. Diese wird immer vor der *challenge.m* aufgerufen und füllt den Workspace mit allen nötigen Variablen, Einstellungen und Pfaden.

- Achten Sie darauf, dass alle geforderten Variablen (`group_number, members, mail`) nach dem Aufruf gesetzt sind.
- Stellen Sie den `ImageReader` für Ihren Rechner ein und erstellen Sie ein Objekt dieser Klasse mit dem Namen `ir`.
- Laden Sie einen virtuellen Hintergrund in die Variable `bg`.

Diese Datei ermöglicht es Ihnen den Code anderer Personen einfach bei sich laufen zu lassen und dabei Ihre eigenen Einstellungen und Pfade behalten zu können.

## 2.5 Challenge

Das Skript *challenge.m* führt die bisher implementierten Funktionen in einer Schleife aus, bis der gewählte Szenenordner erschöpft ist. Messen Sie die Zeit, welche Ihr Programm zur Ausführung der Aufgabe benötigt, speichern sie den Wert in der Variable `elapsed_time` und geben Sie diese in der Commandline aus. Die exakte Ausführung der Schleife ist Ihnen überlassen. Folgende Schritte könnten hilfreich sein:

- Laden Sie in einer Schleife alle verbleibenden Bilder aus dem Szeneordner (bis $\texttt{loop} = 1$).
- Berechnen Sie die Segmentierungsmaske für jedes Bild.
- Fügen Sie Ihren virtuellen Hintergrund ein.

- Speichern Sie das Resultat in der `movie` Variable.

Falls die Variable `store` auf *true* steht, speichern Sie alle berechneten Bilder als AVI Video unter dem Namen *output.avi* ab.

# 3 Grafische Benutzeroberfläche

Entwickeln Sie eine grafische Benutzeroberfläche (GUI), welche über folgende Funktionen verfügt:

- Einen beliebigen Szenenordnerpfad auswählen
- Einen Dateipfad für einen virtuellen Hintergrund auswählen
- Auswahl des Startpunkts (siehe Variable `start`)
- Auswahl des Renderingmodus (siehe Variable `mode`)
- Anzeige der Stereoinputs sowie des gewählten Outputstreams
- Abspielsteuerung: Start, Stop, Loop (endloses Abspielen)
- Speicherung des gerenderten Films unter einem beliebigen Dateipfad

Die GUI soll durch den Befehl `start_gui` geöffnet werden können und darf unabhängig von der *challenge.m* oder *config.m* funktionieren.

# 4 Report

Jede Abgabe wird komplettiert durch die Dokumentation der Software, sowie die Darstellung der geforderten Plots in einem PDF. Die erlaubten Sprachen hierfür sind Deutsch oder Englisch.

- **Readme:** Fügen Sie ihrem Code eine *Readme.txt* Datei bei, in der Sie beschreiben, wie ihre Software auszuführen ist und weisen Sie explizit auf Zusatzfunktionen hin.
- **Dokumentation:** Erstellen Sie ein PDF-Dokument *doku-GXX.pdf*, indem Sie die Funktionsweise ihres Programms kurz erklären und legen Sie auch die Quellen dar, die sie für Ihren Ansatz konsultiert haben. Nutzen Sie dazu gerne auch mathematische Beschreibungen, Skizzen und Blockdiagramme. Fügen Sie in dieses Dokument jeweils ein gerendertes Bild mit subtrahiertem Hintergrund (`mode` = foreground) aus den Szenen P1E_S1, P1L_S3, P2L_S5, P2E_S2, P2E_S4 und P2E_S5 ein. Je nach Szene sollen auf diesen ausgewählten Bildern mindestens eine Person zu sehen sein. Setzen Sie auch die Namen aller am Projekt beteiligten Personen an eine gut sichtbare Stelle im Dokument.
- **Kommentare:** Kommentieren Sie Ihre Schritte ausführlich. Die Sprache des Programms und der Kommentare kann Deutsch oder Englisch sein.

# 5 Bewertungskriterien

Dieser Abschnitt geht auf die vorläufigen Bewertungskriterien ein.

## 5.1 Mindestanforderungen

Diese Anforderungen müssen gegeben sein, um die Challenge zu bestehen.

- Ausführbarkeit des Programms auf Windows und Unix Systemen
- Vollständigkeit des Programms
- Einhaltung der Ordnerstruktur
- Einhaltung der maximalen Laufzeit von 30 Minuten
- Einhaltung der Abgabedeadline
- Vollständigkeit der Dokumentation
- Codequalität und Kommentare

## 5.2 Qualitätsmerkmale

Mit diesen Merkmalen verbessern Sie die Bewertung der Computer Vision Challenge.

- Qualität der Vorder- und Hintergrunddetektion (Visueller Eindruck)
- Komplexität der Szene mit denen Ihr Programm gut umgehen kann (Es gibt einfache und komplexere Szenen im Datenset)
- Randbehandlung beim Übergang zwischen Vorder- und Hintergrund
- Laufzeit des Programms (Je schneller, desto besser)
- Benutzerfreundlichkeit der GUI

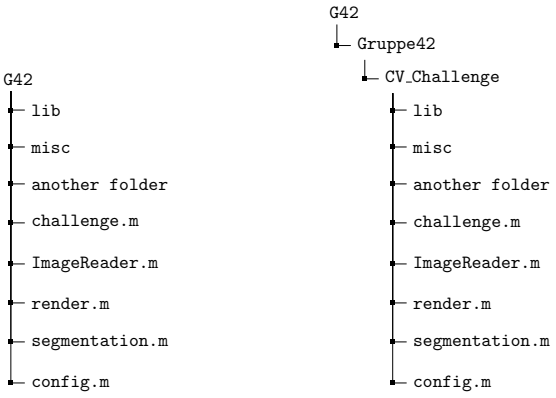
## 5.3 Bonus

Sollten Sie einen der oberen Punkte nicht komplett erfüllt haben, können Sie das mit diesem Zusatzfeature ausgleichen.

- Zu einem virtuellen Hintergrundbild können Sie alternativ die Möglichkeit eines virtuellen Hintergrundvideos anbieten. Bitte machen Sie dies in Ihrer Dokumentation/Readme und GUI deutlich.

# 6 Auswertung

Ihr Code wird nach der Abgabe auf seine Komplettheit und Ausführbarkeit auf den bereits bekannten als auch auf unbekannten Szenen überprüft. Weitere Bewertungsaspekte sind die Erfüllung aller im Abschnitt 2 definierten Funktionen, sowie die Vollständigkeit der Ausgabe und die geforderten Variablen. Des Weiteren hat die Qualität und Vollständigkeit der abgegebenen Dokumente, wie sie in Abschnitt 4 beschrieben sind, Einfluss auf diese Prüfungsleistung. Fehler in den Pflichtanteilen können Sie durch Erfüllen der Zusatzfunktion ausgleichen. Weitere Details zur Bewertung und zum Peer-Review-Verfahren erhalten Sie in kürze.




(a) Richtig (b) Falsch

Abbildung 3: Ordnerstuktur der Abgabe nach dem Entpacken der **.zip*-Datei.

# 7 Abgabe

Geben Sie nur funktionierenden Code ab. Testen Sie ihr Programm vorher auf einem unbekannten Rechner (Windows und Unix) z.B. im Eikon, ob dieses dort auch ausführbar ist und zum gewünschten Ergebnis führt. Zum Testen wird die Matlab Version 2020a verwendet. Wenn Sie mit einer stark abweichenden Matlab Version arbeiten, vergewissern Sie sich, dass keine grundlegenden Funktionen im Vergleich zu dieser Version geändert wurden. Komprimieren Sie Ihre Abgabe in einem **.zip*-Archiv und geben Sie diese Datei auf Moodle für Ihre Gruppe ab. Abbildung 3a zeigt, wie der Inhalt Ihrer Abgabe beispielsweise aussehen könnte.

Abgabedetails:

- **Abgabefrist:** Die Abgabefrist ist der **12.07.2019** um **23:55 Uhr**. Nachträgliche Abgaben können nicht berücksichtigt werden!
- **Abgabeformat:** Sie können nur **.zip*-Dateien abgeben. Achten Sie auf eine eindeutige Benennung nach dem Schema *GXX.zip* wobei *XX* für Ihre Gruppennummer steht. Sind Sie also in Gruppe $42$ geben Sie eine Datei ab, die *G42.zip* heißt. Die Abgabe von Gruppe 2 heißt entsprechend *G02.zip*.
- **Ordnerstruktur:** Stellen Sie sicher, dass beim Entpacken der **.zip*-Datei der Inhalt in genau einem Unterordner landet. Sie können dies z.B. beim Entpacken auf Linux mit dem Befehl `unzip` prüfen. Eine exemplarisch entpackte Abgabe sehen Sie in Abbildung 3.
- **Bilder und Videos:** Ihre Abgabe sollte keine übermäßig großen Dateien oder gar Szenenordner enthalten, diese haben wir bereits selbst vorliegen. Die Uploadgröße ist auf 10MB beschränkt.

# Literatur

[1] Y. Wong, S. Chen, S. Mau, C. Sanderson, and B. C. Lovell, "Patch-based probabilistic image quality assessment for face selection and improved video-based face recognition," in *IEEE Biometrics Workshop, Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 81–88, IEEE, June 2011.