Digitale Bildverarbeitung

DHBW Stuttgart, Vorlesung „Computergraphik und Bildverarbeitung“

Praktische Übung

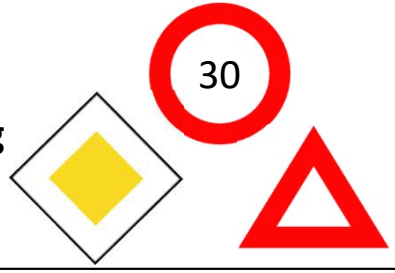
Projekte in dieser Vorlesung

Automotive

Spurerkennung

Farbräume	Histogramme
Bildanalyse (Morphologische Verfahren, Merkmalsextraktion, Kanten- und Flächenbestimmung)	Segmentierung

Verkehrszeichenerkennung



Farbräume	Histogramme
Kontrastverstärkung	Klassifizierung, Objekterkennung

Objekterkennung

Operationen im Ortsbereich (lokale Operatoren, Faltungsfilter)	Optischer Fluss
Segmentierung	Klassifizierung, Objekterkennung

Motivation (einfachtaeglich.de)

Projektziele

- Spaß!
- Neue Erfahrungen im Kontext Bildverarbeitung
- Wecken des langfristigen Interesses

Projektübersicht: Spurerkennung

Spurerkennung

Farbräume	Histogramme
Bildanalyse (Morphologische Verfahren, Merkmalsextraktion, Kanten- und Flächenbestimmung)	Segmentierung

- Auteilung der Note
  - 70 % Programm inkl. Quellcodes und Kommentare
    - Kommentare müssen sinnvoll eingesetzt werden
    - bedeutungsvolle Namensgebung für Variablen
    - gern Python-Bibliothek Sphinx verwenden (https://www.sphinx-doc.org/)
  - 30 % Dokumentation (mögliche Formate Word, Powerpoint, Markdown)
    - Gewählte Vorgehensweise → Warum haben Sie sich für die gewählte Vorgehensweise entschieden? → Welche Alternativen gab es?
    - Dokumentation und Diskussion der Ergebnisse → Welche Probleme traten auf? Welche Lösungswege haben Sie verfolgt?
    - Lessons Learned → Was nehmen Sie aus dem Projekt für sich mit?
    - Ausblick → Welche Probleme konnten Sie im Rahmen des Projektes nicht behandeln?


Curv.: 3277.40, 1998.36, Dev.: −0.01

Projektübersicht: Spurerkennung

Spurerkennung

Farbräume	Histogramme
Bildanalyse (Morphologische Verfahren, Merkmalsextraktion, Kanten- und Flächenbestimmung)	Segmentierung

Mindestanforderungen (entspricht der Note 2,0)

- **Segmentierung des Bildes:** schränken Sie das Bild auf den Bereich ein, in dem sich die Spurmarkierungen befinden
- **Vorverarbeitung:** Führen Sie eine Kamerakalibrierung (für Udacity-Bildquellen) und die Perspektivtransformation durch
- **Farbräume, Histogramme:** erkennen Sie die Spurmarkierungen in den Farben der angegebenen Quellen
  Falls weitere Spurmarkierungen auf dem Bild gefunden werden, müssen die der eigenen Fahrspur priorisiert werden
- **Allgemeines:** Die Verarbeitung von Bildern muss in Echtzeit stattfinden --> Ziel: > 20 FPS für reine Verarbeitung ohne Anzeige) → Prozessor, Grafikkarte, Arbeitsspeicher
- **Allgemeines:** Beschleunigen Sie die Verarbeitung durch weitere Maßnahmen (bspw. Erkennung der Spurmarkierung in den ersten Frames, Tracking der Spurmarkierung in weiteren Frames solange, bis sich Spurmarkierungspositionen zu stark ändern) → mind. eine Maßnahme im Projekt verwenden
- **Curve / Polynom Fitting:** Erkennen Sie die Krümmung der Fahrspur und geben Sie diese im Ausgabebild aus
- **Validierung des Verfahrens:** Umrechnung auf Straßenkrümmung, sodass Simulation erfolgreich bestanden wird
- **Allgemeines:** relevante Spurmarkierungen werden in den Udacity-Bildern und im Video „project_video“ durchgehend erkannt

Projektübersicht: Spurerkennung

Spurerkennung

Farbräume	Histogramme
Bildanalyse (Morphologische Verfahren, Merkmalsextraktion, Kanten- und Flächenbestimmung)	Segmentierung

Zusatzaufgaben (jeweils – 0,33 → Mindestanforderungen + 3x Zusatzaufgaben = 1,0)

- relevante Spurmarkierungen werden im Video "challenge_video" oder "harder_challenge_video" (nahezu) durchgehend erkannt (sowohl „challenge_video“ als auch „harder_challenge_video“ werden als Zusatzaufgabe gewertet)
- relevante Spurmarkierungen werden auf den Datensatz KITTI angewendet. Welche Anpassungen müssen vorgenommen werden, damit Ihr Algorithmus übertragen werden kann?
- erkennen Sie Objekte im Bild und visualisieren Sie diese (z.B. weitere Fahrzeuge, Motorräder, etc.)
  Die Objekterkennung bitte so implementieren, dass sie deaktivierbar ist und nicht in FPS-Berechnung einzahlt.
- nutzen Sie alternative Möglichkeiten der Spurerkennung (z.B. mit Neuronalen Netzen)
- ergänzen Sie Ihren Algorithmus um eine Kennzeichenerkennung inkl. Texterkennung
- Gerne können Sie eigene Zusatzaufgaben zur Verbesserung Ihres Algorithmus einführen. (Aufwand sollte vergleichbar sein zu o.g. Punkten).
- Alle durchgeführten Aufgaben müssen dokumentiert, kommentiert und abgegeben werden.

Projektübersicht: Spurerkennung

Spurerkennung

Farbräume	Histogramme
Bildanalyse (Morphologische Verfahren, Merkmalsextraktion, Kanten- und Flächenbestimmung)	Segmentierung

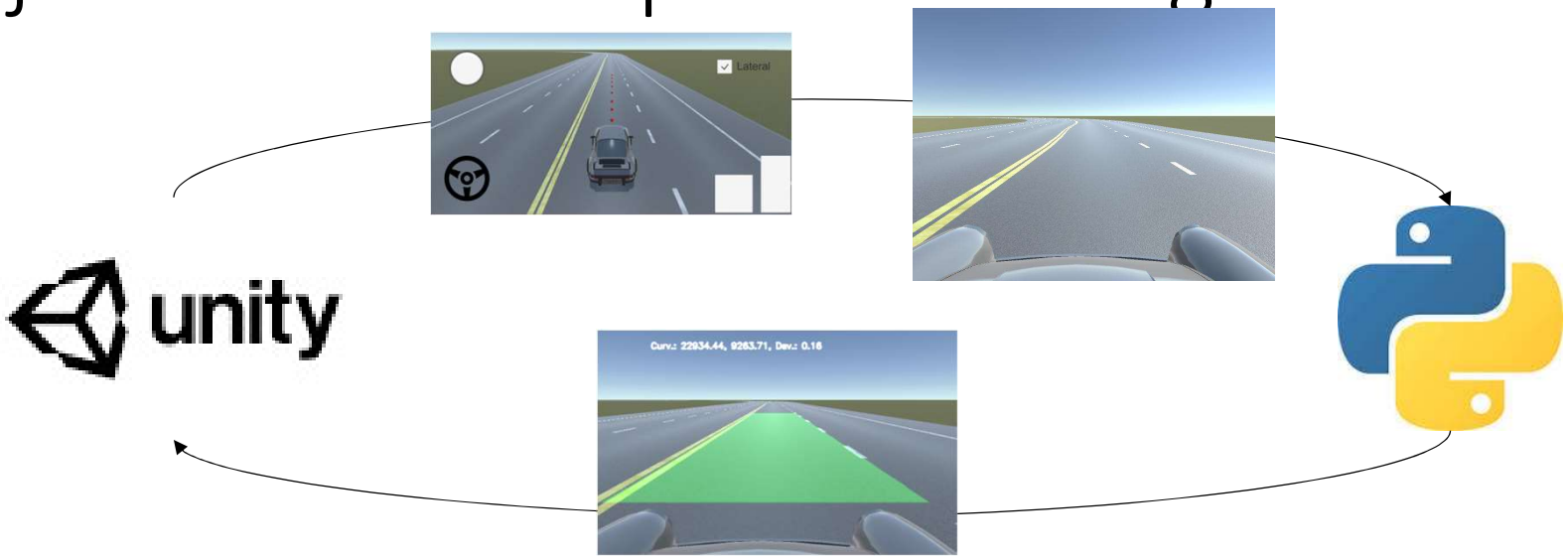
Zusatzaufgabe Android (- 0,7 → Mindestanforderungen + 1x Zusatzaufgaben + Android-Portierung = 1,0)

- entwickelter Algorithmus wurde auf Android übertragen
- Dokumentation der erkannten Fahrspuren
- Diskussion über die Herausforderungen bei der Portierung der Python-Umsetzung mit der Java-Umsetzung
- Alle durchgeführten Aufgaben müssen dokumentiert, kommentiert und abgegeben werden.

Projektübersicht: Spurerkennung

- Erwartete Abgabe
  - Quellcodes inkl. Kommentare
    - Jupyter Notebook o.ä. zur prototypischen Implementierung
    - Python / Java / C++ Quellcode zur performanten Implementierung und Validierung
    - Android Studio Quellcode
    - Bilder und Videos inkl. erkannter Linien und Objekte
  - Dokumentation
    - Word, Powerpoint, Markdown

Projektübersicht: Spurerkennung - Validierung



PythonServer_TCP_student

```
import …

while True:
        data = … # receive data
        img = cvt(data) # convert to image
        # YOUR FUNCTION HERE
        result_img, ... = PythonLaneLines
        .find_lines(img_decoded, ...)
        # send polynomials and radius
```

PythonLaneLines_student

```
import …

# YOUR FUNCTION HERE
def find_lines(img, ...):
        result = … # 1. warp and undistort image
        # 2. get lane lines within image
        # 3. get curvature from lane line
        # 4. return image and curvature
        return result, curv_left, curv_right, …
```

Projektübersicht: Spurerkennung

- Vorstellung des Projektes
  - Projektvorstellung (Live-Vorstellung, Methoden, Vorgehen, Probleme)
  - Offene Fragen aus der Gruppe

- Visualisierung **nicht** in Zeitmessung einfließen lassen

Anforderung an eigene Projekte

- Datensätze müssen zum Nachweis der Performance verfügbar sein
- Performance muss auf Basis der Datensätze und weiterer Publikationen vorliegen → Vergleichbarkeit
- Digitale Bildverarbeitung muss zum Einsatz kommen → nicht ausschließlich Bildklassifikation durch NNs

Weitere Projektmöglichkeit: Ampelerkennung

- Ampelerkennung
  - Erkennung der Ampelposition(-en) im Bild
  - Erkennung der jeweiligen Ampelphase
  - Datenbank #1: Bosch Small Traffic Lights Dataset | Kaggle
  - Publikation #1: A deep learning approach to traffic lights: Detection, tracking, and classification | IEEE Conference Publication | IEEE Xplore
  - Publikation #2: (PDF) Traffic Light Recognition using Image Processing Compared to Learning Processes (researchgate.net)
  - Scripts #1: bosch-ros-pkg/bstld: Sample scripts for the Bosch Small Traffic Lights Dataset (github.com)
  - Datenbank #2: LISA Traffic Light Dataset | Kaggle
  - Datenbank #3: DriveU Traffic Light Dataset - Universität Ulm (uni-ulm.de)
  - Scripts #3: julimueller/dtld_parsing: Scripts for loading and visualizing the DriveU Traffic Light Dataset (DTLD) (github.com)

Projekt Ampelerkennung

- Mindestanforderungen (2,0)
  - Ampelerkennung TP-Rate > 80 %, FP-Rate < 10 %
  - Anwendung auf existierende Bilddatenbank
  - Validierung mithilfe Simulationsumgebung
  - Laufzeit > 20 fps
- Zusatzaufgaben (jeweils – 0,33)
  - Verbessern Sie die Performance um 10 % (TP-Rate > 90 % und FP-Rate < 10% oder TP-Rate > 85 % und FP-Rate < 5 %)
  - Erkennen Sie weitere Objekte im Bild (Fahrzeuge, Motorräder, …)
  - Erkennen Sie Kennzeichen im Bild inkl. Texterkennung
  - Erkennen Sie Fahrspuren im Bild
- Zusatzaufgabe Android (-0,7)
  - Portieren Sie Ihren Algorithmus auf Android