Konzeptpapier
Computer Vision Pipeline

André Kestler
a.kestler@oth-aw.de

Marcus Haberl
m.haberl@oth-aw.de

Tobias Lettner
t.lettner@oth-aw.de

Antonio Vidos
a.vidos@oth-aw.de

Tobias Dobmeier
t.dobmeier@oth-aw.de

Tobias Weiß
t.weiss@oth-aw.de

***Zusammenfassung*—Ziel des Projekts ist die Implementierung einer Computer Vision Pipeline als Cloud-Anwendung. In der Benutzeroberfläche kann der Benutzer ein zu verarbeitendes Bild hochladen und eine Verarbeitungskette aus einzelnen Modulen zusammenstellen. Das Backend verarbeitet die Daten in den angegebenen Schritten und stellt dem Benutzer die Ergebnisse als verarbeitetes Bild auf der Webseite zur Verfügung.**

I. Einleitung

Computer Vision ist ein wichtiges Gebiet der digitalen Bildverarbeitung und hat in den letzten Jahren erhebliche Fortschritte gemacht. Es ermöglicht die automatisierte Analyse und Interpretation von Bildern und Videos und hat zahlreiche Anwendungsbereiche in der Industrie, Medizin und Sicherheitstechnik. Eine wichtige Komponente bei der Entwicklung von Computer Vision Anwendungen ist die Erstellung einer robusten und effizienten Pipeline, die die Datenverarbeitung und Analyse durch mehrere Schritte automatisiert und optimiert.

In diesem Projekt wird eine Computer Vision Pipeline als Cloud-Anwendung implementiert, die es dem Benutzer ermöglicht, ein Bild hochzuladen und durch eine Verarbeitungskette das gewünschte Ergebnis zu erhalten. Die Benutzeroberfläche soll eine intuitive Möglichkeit bieten, um den Prozess der Bildanalyse und -verarbeitung zu steuern. Zudem bietet die Cloud-Anwendung eine skalierbare und flexible Infrastruktur, die die Möglichkeit bietet, die Kapazität je nach Bedarf anzupassen.

II. Verwandte Arbeiten

Die Webseite „Scratch" [1] hat ein ähnliches Konzept, das teilweise als Inspiration für die Entwicklung des hier beschriebenen Systems genutzt wurde. Mit „Scratch" können Nutzer ohne oder mit wenig Programmiererfahrung die Denkweise eines Softwareentwicklers erlernen und ihre Erfahrungslücken verringern. Die Programmierung erfolgt grafisch, indem der Benutzer Aktionen und Ereignisse als Bausteine auf ein leeres Feld zieht und miteinander verknüpft. Wenn die Bausteine ausgeführt werden, führen sie die entsprechenden Aktionen aus, z.B. die Animation eines Laufenden Mannes.

III. Anforderungen

A. Bild upload

Als Benutzer möchte ich ein Upload-Feld haben, mit dem ich mein Bild an die Webseite hochladen kann. Akzeptanzkriterien:

- Upload-Feld für das Bild
- Bild wird hochgeladen

B. Bild download

Als Benutzer möchte ich ein Download-Feld haben, um das fertige Bild lokal herunterzuladen. Akzeptanzkriterien:

- Download-Feld für das Bild
- Bild wird heruntergeladen

C. Metadaten

Als Entwickler möchte ich dem Benutzer die Metadaten zu seinem Bild anzeigen lassen. Akzeptanzkriterien:

- Dimensionen des Bildes
- Pixelanzahl
- Graustufenbild oder Farbbild
- Optional Histogramm aller Kanäle

D. Visualisierung Bild

Als Benutzer möchte ich mein originales Bild auf der Webseite sehen. Akzeptanzkriterien:

- Bild in ausreichender Qualität darstellen

E. Visualisierung Ergebnisse

Als Benutzer möchte ich das Endergebnis, nach der Pipeline und auch alle Zwischenergebnisse betrachten können. Vorallem ist es wichtig, dass zugeordent wird zu welchem Schritt das angezeigte Bild gehört. Akzeptanzkriterien:

- Bild in ausreichender Qualität darstellen
- Bild des Zwischenschrittes anzeigen, wenn von Benutzer gefordert

F. Pipeline erstellen

Als Benutzer möchte ich die Pipeline intuitiv erstellen und ändern können. Akzeptanzkriterien:

- Mehrfachauswahl möglich
- Drag-Drop der Filter

G. Fehlermeldungen Pipeline

Als Entwickler möchte ich dem Benutzer Fehlermeldungen für die Pipeline anzeigen, dass dieser weiß, was er bei Bedarf ändern muss. Akzeptanzkriterien:

- Bildformat passend
- Vorverarbeitungsschritt vor Filter nötig
- Pipeline ändern
- Filterparameter passend

H. Pipeline Filter

Als Benutzer möchte ich Filter auf mein Bild anwenden. Akzeptanzkriterien:

- Bilateral
- Average
- Faltung

I. Pipeline Morphologische Transformationen

Als Benutzer möchte ich morphologische Transformationen auf meinem Bild durchführen. Akzeptanzkriterien:

- Erosion
- Dilation
- Opening
- Closing

J. Pipeline Transformationen

Als Benutzer möchte ich Transformationen zur Kontrast- und Bildänderung mit meinem Bild ausführen. Akzeptanzkriterien:

- Log-Transformationen
- Gamma-Transformationen
- Geometrische Transformation

K. Pipeline Segmentierung

Als Benutzer möchte ich Segmentierungsalgorithmen auf mein Bild anwenden. Akzeptanzkriterien:

- Watershed
- Otsu Thresholding

L. Pipeline Kanten-/Konturenerkennung

Als Benutzer möchte ich die Kanten auf meinem Bild erkennen können Akzeptanzkriterien:

- Canny-Edge-Detector
- Sobel
- Laplacian

M. Parameter anpassen

Als Benutzer möchte ich Einfluss auf die einzelnen Parameter der Fitler haben, um den jeweiligen Filter anzupassen. Akzeptanzkriterien:

- Pop-Up Fenster
- Default Werte vorhanden
- Werte ändern

N. Filter suchen

Als Benutzer möchte ich eine Suchfunktion auf der Webseite haben, um bestimmte Filter zu finden. Akzeptanzkriterien:

- Suchfenster
- Tastatureingabe
- Ergebnisse anzeigen

O. Container

Als Entwickler will ich das Projekt mit Containern modularisieren um die Anwendung leicht zu verteilen und die Wartbarkeit dadurch zu erhöhen. Akzeptanzkriterien:

- Frontend Container
- Backend Container
- Docker-Compose verwalten

P. Testabdeckung

Als Entwickler will ich eine Testabdeckung für meine Anwendung, um Fehler frühzeitig erkennen zu können.

- Backend-Testabdeckung zu mindestens 50%. Mithilfe von pytest [8].
- Frontend-Testabdeckung zu mindestens 50%. Mithilfe von Jest [7].

Q. Cloudanwendung

Als Entwickler will ich meine Anwendung über die Cloud verteilen, um möglichst einfach Benutzer anzusprechen.

- Docker-Container der einzelnen Module

IV. Methoden

A. Datenfluss

1) Bild wird über die Benutzeroberfläche in einen S3-Bucket geladen.
2) Frontend überträgt die Verarbeitungskette ans Backend.
3) Backend lädt Bild aus S3-Bucket.
4) Backend speichert die Ergebnisse der Verarbeitungskette im S3-Bucket.
5) Frontend lädt Ergebnisse aus S3-Bucket und stellt diese dar.

B. Frontend

Für die Interaktion mit dem Benutzer wird React [6] mit Javascript verwendet. Über eine REST-Schnittstelle kann mit dem Backend kommumniziert werden.

C. Backend

- Für die serverseitige Verarbeitung wird Python [2] verwendet.
- Für die Bildverarbeitung wird die OpenCV- [3] und SciKit-Image-Bibliothek [4] verwendet.
- Für die Bereitstellung der REST-Schnittstelle wird die Flask-Bibliothek [5] verwendet.

REFERENCES

[1] Scratch Programmierung: https://scratch.mit.edu/projects/editor/?tutorial=getStarted
[2] Python: https://www.python.org
[3] OpenCV: https://opencv.org
[4] scikit-image: https://scikit-image.org
[5] Flask: https://flask.palletsprojects.com/en/2.3.x/
[6] React: https://react.dev
[7] Jest: https://jestjs.io
[8] PyTest: https://docs.pytest.org/en/7.3.x/