

STARHack Asimov

Dokumentation

David Schafer, Serge Hauri, Steve Ineichen, Remo Schwarzenruber

2019-03-22

Inhaltsverzeichnis

1 Getting Started	3
1.1 Challenges	3
1.2 AUTONSENSE / VOLVO Challenge	3
1.3 Setup	4
2 Tasks	5
3 Data Processing	6
4 Damage drawer	7
4.1 Funktionsumfang	7
4.2 Funktionsdesign	7
4.2.1 Software Abhängigkeiten	7
4.2.2 Prozess des Funktionsdesigns	7
4.3 Realisation / Umsetzung	8
4.4 Klasse DamageImage	9
4.4.1 Methode: Init	9
4.4.2 Methode: Kulturen Auto	9
4.4.3 Methode: Zeichnen	10
4.4.4 Methode: Zeichnen - Pfeil	10
4.4.5 Methode: Zeichnen - Kreis	10
4.4.6 Methode: Text auf das Bild	10
4.4.7 Methode: Berechnung Beschädigung	11
4.4.8 Methode: Schreiben der Bilddatei	11
4.4.9 Methode: Pfad der geschriebene Datei	11
4.4.10Methode: Löschen von allen gerendert Dateien	12
4.4.11Methode: Anzeige der Datei auf dem Bildschirm	12
4.5 Implementierung im Projekt	12
4.6 Mögliche Darstellung der Datei in einem Protal	13
5 Server / Frontent und Backend	14

1 Getting Started

1.1 Challenges

There were 8 different challenges which you could apply. We were mainly interested in the Challenges from the following partners:

- Autosense (Crash Visualization)
- SBB (Recycle)
- Laica (AR)
- BOSCH IOT-Lab (Sensor Car)

All case descriptions can be viewed here: <http://live.starthack.ch/case-descriptions/>

We applied for the Autosense challenge and got it (limit of 15 Teams per challenge).
The challenge is as follows:

1.2 AUTONSENSE / VOLVO Challenge

Generate Car Crash Image, visualize impact and direction using sensor data

Your challenge if you choose to accept:

Build Microservice(s) to generate Image with 3D object simulating impact forces for given time offset (from crash). Deploy Microservice(s) on Swisscom Application Cloud (cloud foundry). Provide API(s) for submitting Input data (stream) and getting the Result. Generate output for each submitted Crash Record : Direction of the impact (Impact angle and energy), visualize the damage show expected place of impact on car

Winner is the Team who:

Has identified the maximum number of crashes correctly providing - Correct impact direction & Most accurate 3D simulation (compared to real crash picture)

How it will be measured:

For each submitted Crash Record AND time offset, generate Image with Direction of the impact (Impact angle and energy), Visualized damage and Time offset with the maximum force/damage on the object. Crash Record is submitted to the service. The calculated impact direction will be compared with pictures from real crash.

Restrictions:

Service must be deployable on cloud infrastructure (AWS/Cloud Foundry/Kubernetes/Docker). Service should use as few as possible external APIs. Given Data Models and API POST Requests structure must be used.

1.3 Setup

xxxxx

pwd

2 Tasks

- Data Parsing (transform in more structured way -> acceleration, calibration)
 - define useful functions
 - implement functions
 - crash_record.py
- Webserver
 - create webserver (sanic)
 - implement requests
 - return some dummy data for the moment
 - webserver.py (rename main.py)
 - docker container
- Image
 - define interface
 - library to draw arrows
 - library to draw circles
 - image.py
- Visualization & Math
 - jupyter notebook visualization
 - define functions to calculate angles & impact
 - start crash_record_calculator.py

3 Data Processing

tbd

4 Damage drawer

4.1 Funktionsumfang

Mit der Klasse DamageImage (File: `damage_image.py`) werden alle Funktionalitäten im Zusammenhang mit der Bild zusammengebündelt. Die Umfasst als Beispiel das Erkennen der Kulturen von dem Auto, das Zeichnen der Beschädigung wie auch das Beschriften der Milisekunden des Aufpralls, die Crash-ID und weiter Informationen.

4.2 Funktionsdesign

4.2.1 Software Abhängigkeiten

Für die Erkennung der Kulturen wurde die Bildverarbeitungs Library OpenCV (Open Source Computer Vision Library) verwendet. Die Kulturen werden verwendet um den Eintrittspunkt der Beschädigung zu berechnen.

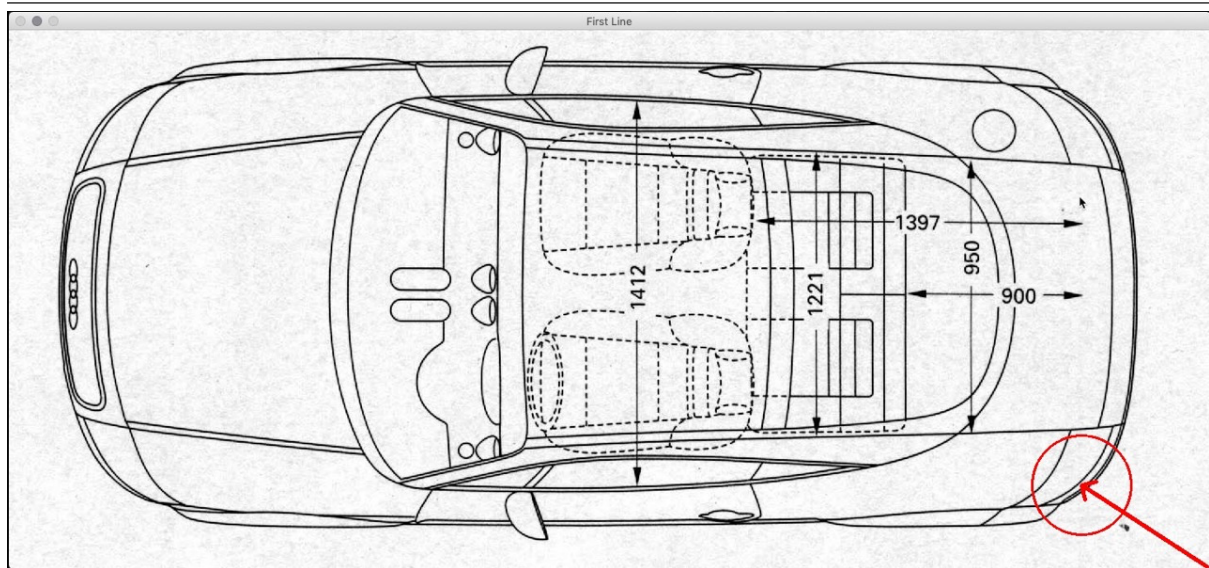
Für die Berechnung innderhalb der Klasse DamageImage wurde auf die bekannte Python Library Numpy (<http://www.numpy.org>) zurückgegriffen.

Die Python Standardbibliothek `os / math / shutil` werden für kleinere Funktionen benötigt.

4.2.2 Prozess des Funktionsdesigns

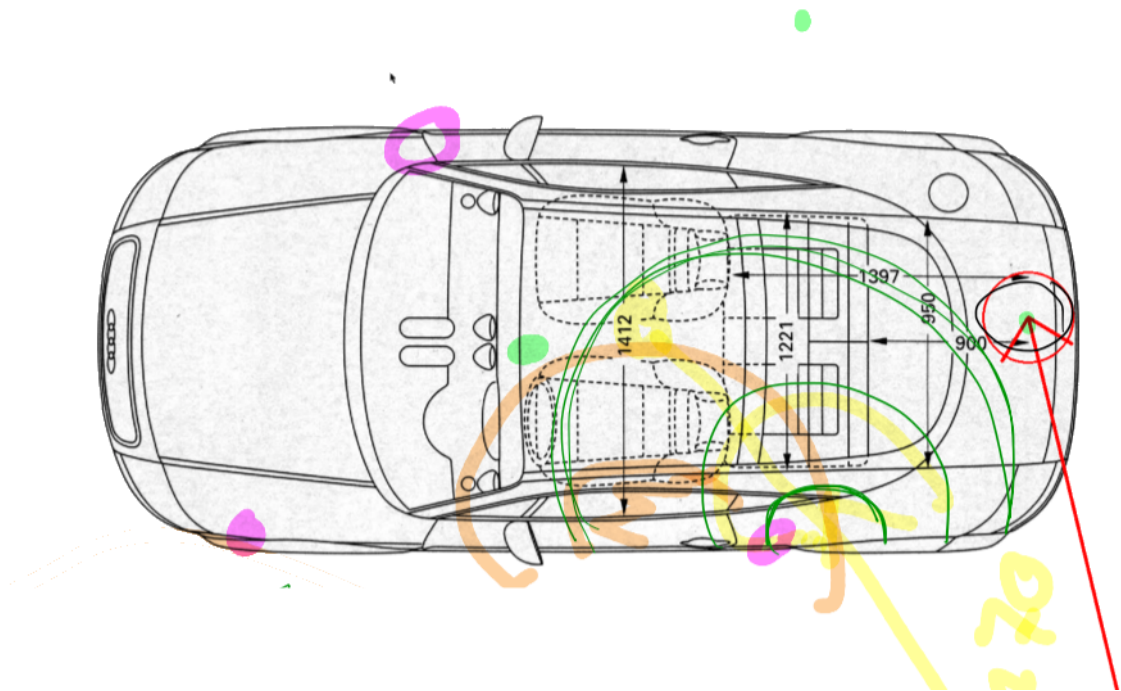
Zu Beginn wurde mittels OpenCV die Datei eingelesen und hardcoded ein Kreis und ein Pfeil gezeichnet. Mit dieser Version haben wir dann im Team das Zieldesign der Bilddatei mittels iPad und Pen gezeichnet.

Erste Version Damage Image



Skizze Damage drawer

Rendered crash image after 6110[ms]

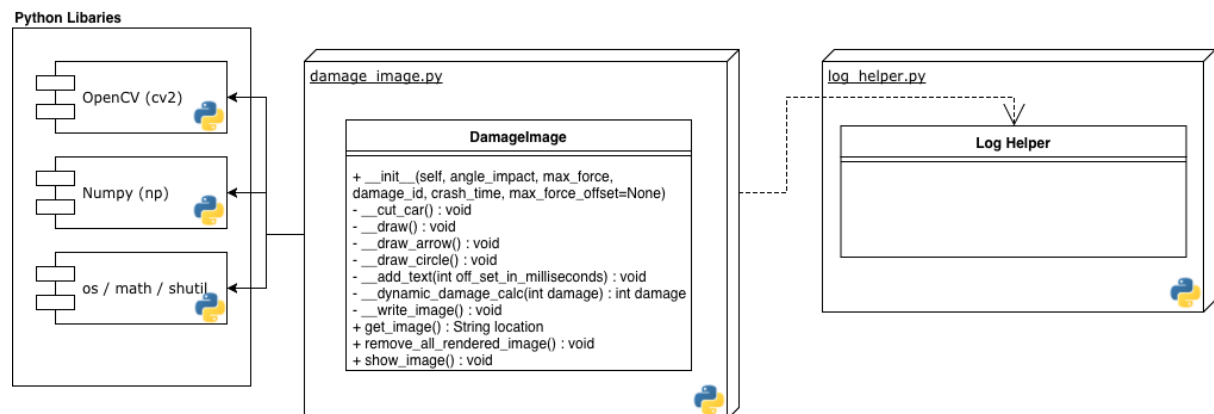


4.3 Realisation / Umsetzung

Klassendiagramm und Beschreibung der Funktionen der einzelnen Methoden.

4.4 Klasse DamageImage

Teilfunktion: Damage Image



4.4.1 Methode: Init

```
def __init__(self, angle_impact, max_force, damage_id, crash_time, max_force_offset=None):
```

Die Init Methode wird aufgerufen bei der Instanzierung von einem Objekt. Als Parameter werden folgende Informationen benötigt:

Parameter	Beschreibung
self	Instanz-Referenz
angle_impact	Winkel in Grad. Mit diesem Winkel wird der Aufprall der Beschädigung gezeichnet.
max_force	Numerischer Wert mit dem die Grösse der Beschädigung am Auto berechnet wird.
damage_id	String. Eindeutiger Crash-Report ID
crash_time	String. Uhrzeit für die Beschriftung auf dem Bild
max_force_offset	Numerischer Wert. Zeitpunkt, nach wie vielen Millisekunden die Beschädigung berechnet wurde (Default=None)

Mit den Methoden Parameter werden die jeweiligen lokalen / privaten Methodenvariablen initialisiert und den Wert zugewiesen. Weiter wird der Pfad wo die Bilddatei gespeichert wird aufgrund der damage_id und der allfälligen max_force_offset. Die Dateien werden in einem speziell definierten Ort (images_rendered/) gespeichert. Dies hat den Grund, dass wenn die Bilddatei von den gleichen Crash-Report und dem gleichen max_force_offset nicht nochmals neu erstellen und schreiben muss, sondern direkt zurückgeben kann.

4.4.2 Methode: Kulturen Auto

```
def __cut_car(self):
```

Parameter	Beschreibung
self	Instanz-Referenz

Die Methode `__cut_car` findet mittels OpenCV die Kulturen (Randkulturen) von dem Auto-Bild. Diese Kulturen werden als Pixel-Matrix abgespeichert und später für den Eintrittspunkt der Beschädigung benötigt. Dazu wird das Auto-Bild in Schwarz/Weiss konvertiert und den Threshold für die Linien gesetzt. Mittels der OpenCV Funktion `cv2.findContours` können die äussersten, geschlossene Linien abgefragt werden.

4.4.3 Methode: Zeichnen

```
def __draw(self):
```

Paramenter	Beschreibung
self	Instanz-Referenz

Die generelle Zeichnungsmethode `__draw` ist als Wrapper Methode zu verstehen. Wenn gegebenenfalls noch weitere Beschriftungen auf die Bilddatei geschrieben werden soll, kann dies hier eingefügt werden. Hier passiert auch die Entscheidung ob der Text für den Offset der Millisekunden angezeigt wird oder nicht.

4.4.4 Methode: Zeichnen - Pfeil

```
def __draw_arrow(self):
```

Paramenter	Beschreibung
self	Instanz-Referenz

Hier wird der Pfeil für die Bilddatei gezeichnet. Etwas unschön ist hier, dass auch noch der Nullpunkt des Koordinatensystem in dieser Methode gezeichnet wird. Eine entsprechendes `# TODO:` ist hier vermerkt.

4.4.5 Methode: Zeichnen - Kreis

```
def __draw_circle(self):
```

Paramenter	Beschreibung
self	Instanz-Referenz

Der Kreis für die für die Grösse der Beschädigung wird hier auf die Bilddatei gezeichnet. Die Grösse von dem Kreis (Radius) wird mit Hilfe der Funktion `__dynamic_damage_calc` berechnet.

4.4.6 Methode: Text auf das Bild

```
def __add_text(self, off_set_in_milliseconds):
```

Paramenter	Beschreibung
self	Instanz-Referenz

Parameter	Beschreibung
off_set_in_milliseconds	Numerischer Wert. Zeitpunkt, nach wie vielen Millisekunden die Beschädigung berechnet wurde

Auf der resultierende Bilddatei werden folgende Informationen dargestellt:

- Time-Offset von der maximalen Beschädigung
 - Text: Rendered crash image after " off_set_in_milliseconds + "[ms]"
- Eindeutige Bilddatei Beschriftung mit der entsprechender Uhrzeit aus dem Crashreport
 - Text: crash identifier = " + self.damage_id + " - damage time = " crash_time

4.4.7 Methode: Berechnung Beschädigung

```
def __dynamic_damage_calc(self, damage):
```

Parameter	Beschreibung
self	Instanz-Referenz
damage	Numerischer Wert mit dem die Grösse der Beschädigung am Auto berechnet wird.

Diese Hilfsmethode berechnet aufgrund einem numerischen Wert die grösse der Beschädigung. Die maximale Beschädigung von 15 und die minimale Beschädigung von 2 wurden aus den Daten ermittelt.

4.4.8 Methode: Schreiben der Bilddatei

```
def __write_image(self):
```

Parameter	Beschreibung
self	Instanz-Referenz

Hier wird die PNG-Bilddatei auf den lokalen Storage von dem Server heruntergeschrieben.

4.4.9 Methode: Pfad der geschriebene Datei

```
def get_image(self):
```

Parameter	Beschreibung
self	Instanz-Referenz

Diese Public Methode wird von Server verwendet um die fertige Bilddatei zu erhalten. Innerhalb dieser Methode werden die Zeichnungsmethoden self.__draw() und die self.__write_image() ausgeführt.

Als Rückgabewert erhält der Server den Pfad der Datei, welche auf dem Server geschrieben wurde.

4.4.10 Methode: Löschen von allen gerendert Dateien

```
def remove_all_rendered_image(self):
```

Parameter	Beschreibung
self	Instanz-Referenz

Diese Housekeeping Methode dient dazu, alle gerenderten Bilddateien auf dem Server zu löschen. Die Methode kann vor dem Start des Server wie auch nach dem Testing ausgeführt werden.

4.4.11 Methode: Anzeige der Datei auf dem Bildschirm

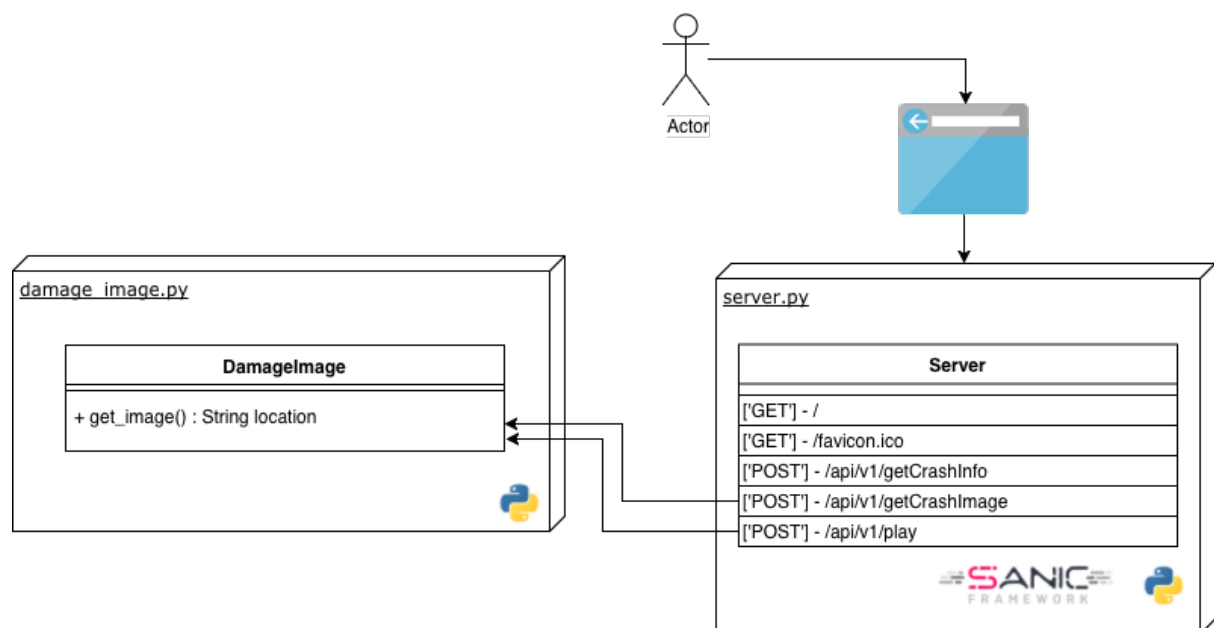
```
def show_image(self):
```

Parameter	Beschreibung
self	Instanz-Referenz

Die show_image dient für eine rasche Entwicklung ohne Server. Sie führt die gleichen Methoden aus, wie die Methode get_image mit dem Unterschied, dass die Bilddatei nicht an den Server zurückgegeben wird sondern mittels OpenCV an dem Display angezeigt wird. So kann Abhängigkeit vom Server neue Funktionen rasche getestet werden.

4.5 Implementierung im Projekt

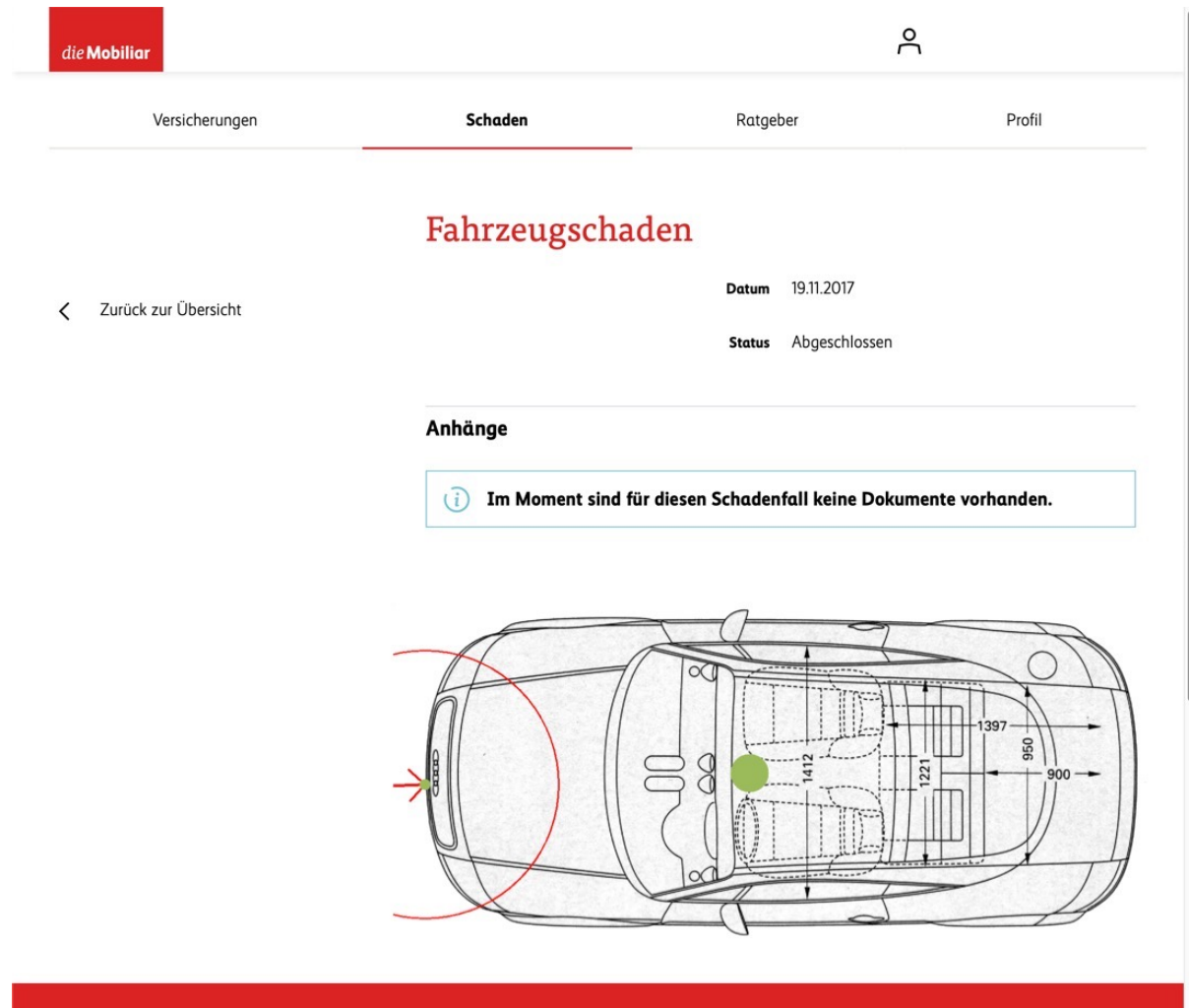
Innerhalb von dem Projekt werden wir die Funktion get_image wie folgt benutzt:



Innerhalb vom server.py wird ein Damage drawer Objekt erstellt und mittels der Funktion get_image die fertig gerenderte Bilddatei zurückgegeben und auf der Webseite dargestellt.

4.6 Mögliche Darstellung der Datei in einem Portal

Ein möglicher Einsatzbereich von unserem Projekt könnte ein Portal von einer Versicherung sein. Hier würde bei Autounfällen der Ort von dem Schaden wie auch das Ausmass der Beschädigung aufgezeigt werden. So kann ein Mehrwert in Form von mehr Informationen an dem Kunde einer Autoversicherung entstehen.



5 Server / Frontent und Backend

tbd