README Masterarbeit Julian Geppert

Dependencies:

- python (2.7)
- pip (9.01)
  - numpy (1.12.1)
  - numpy-quaternion (0.0.0.dev2017.02.28)
  - tensorflow (1.0.1)
  - tflearn (0.3)
  - OpenEXR (1.2.0)
  - matplotlib (1.5.1)
  - pyquaternion (0.9.0)
  - trianglesolver (1.1)
- blender (2.78)
- OpenEXR
- OpenCV 3

Dateien/Ordner:

- `rotation_network/` enthält alle Dateien zum Training/Prediction/Konvertieren des MVCNN-Ansatzes
- `rotation_network/networks/` enthält die Implementierungen der MVCNN-Netze
- `rating_network/` enthält alle Dateien zum Training/Prediction/Konvertieren des Bewertungsnetzes
- `rating_network/networks/` enthält die Implementierungen der Rating-Netze
- `tfhelper/` enthält Helper für das Training und die Auswertung der Quaternionen
- `utility/modelgeneration` enthält Scripts zur Generierung der Blendermodelle
- `utility/datageneration` enthält Scripts zur Generierung der Datensätze (Rendering)
- `utility/imagerating` enthält die Implementierung der Bewertungsfunktion

Modellgenerierung

Wird verwendet um Kombinationen der Lebermodelle und Tumormodelle herzustellen. Dazu wird ein Ordner angegeben, welcher die Lebermodelle enthält und eine Datei, welche alle Tumormodelle beinhaltet. Leber-Meshes der Blender-Dateien müssen mit `liver` und Tumor-Meshes mit `tumor` beginnen.

Benutzung:

```
blender --background --python utility/modelgeneration/blender_gen_livers.py
-- [Leber-Ordner] [AusgabeOrdner] [Tumor-Datei] [Leberanzahl]
```

Rendering / Datengenerierung

Wird für die Generierung des Datensatzes verwendet.

Benutzung (Beispiel): Rendert einen Datensatz mit 3 virtuellen Kameras, einer Wahl der Rotationsachse über ein Icosphere mit 162 Kanten und jeweils 10 Rotationen pro Achse mit einer Kamera-Auflösung ovn 100x100.

```
blender --background --python utility/datageneration/render.py --blender_files
[LeberOrdner] --rendered_dir [DatensatzOrdner] --num_rotations 10 --icosphere 3
--res_x 100 --res_y 100 --num_cams 3
```

Die Datensätze müssen daraufhin in TFRecords konvertiert werden, wobei hierfür für das Rotationsnetz/Bewertungsnetz getrennte Scripte (convert.py) verwendet werden.

Beispiel:

```
python convert.py --data_dir [DatensatzPfad] --shuffle True
```

Rotationsnetz

Training

Training des Netzes nach Generierung des Datensatzes und Konvertierung.

Beispiel:

```
python train_rotation.py [Train-Record] [Test-Record] --num_train
[AnzahlTrainingsbeispiele] --num_test [AnzahlTestbeispiele] --num_epochs 25
--batch_size 50 --learning_rate 0.001 --nntype [Netztyp] --save_path trainedModel
```

Prediction

Prediction mittels eines vortrainierten Netzes.

Beispiel:

```
cd trainedModel
python predict_rotation.py [Test-Record] --num_samples [AnzahlBeispiele]
--nntype [Netztyp]
```

Bewertungsnetz

Training

Beispiel:

```
python train_rating.py [Train-Record] [Test-Record] --num_train [AnzahlTrain]
--num_test [AnzahlTest] --num_epochs 25 --batch_size 50 --learning_rate 0.001
--nntype [Netztyp] --save_path trainedModel
```

Prediction

```
cd trainedModel
python predict_rating.py [Test-Record] --num_samples [AnzahlBeispiele]
--nntype [Netztyp]
```