Bildverarbeitung Praktikum

Übung 4, Abgabe 28.06.2024

Dr. Daniel Wiegreffe

June 11, 2024

1 SIFT

Laden Sie sich die Dateien SIFT_algo.py, SIFT_KeyPoint.py, SIFT_Params.py und SIFT_Visualization.py aus dem Moodle Kurs herunter. Diese Dateien enthalten eine Implementierung des SIFT Algorithmus. Ihre Aufgabe ist es nun, die ersten 3 Schritte des SIFT-Algorithmus zu implementieren. Nutzen Sie nicht den SIFT-Algorithmus von OpenCV oder die bereits implementierten Methoden aus SIFT_algo.py.

Folgende Schritte sollten implementiert werden:

- Erstellen Sie einen Scale Space, die dazugehörigen Delta-Werte und Sigma-Werte
- Berechnen Sie aus dem erstellten Scale Space die DoG-Bilder
- Finden Sie lokale Extrema in den DoG-Bildern

Führen Sie alle Schritte des SIFT-Algorithmus in richtiger Reihenfolge aus. Benutzen Sie hierfür Ihre eigenen Methoden und die Methoden aus SIFT_algo.py.

Das Matching der Keypoints ist nicht Teil dieser Aufgabe. Sie müssen keine Keypoints vergleichen.

SIFT-Implementierung

Bevor Sie mit der Implementierung beginnen, hier noch ein paar Informationen über den Source-Code aus Moodle.

- SIFT_algo.py enthält eine minimalisierte Implementierung aller Schritte von SIFT.
- SIFT_KeyPoint.py enthält eine Klasse, die einen Keypoint / Extremum repräsentiert.

- SIFT_Params.py enthält eine Klasse, die alle Parameter für den SIFT-Algorithmus enthält
- SIFT_Visualization.py enthält zwei Methoden, welche die Scale-Space und die dazugehörigen Keypoints / Extrema visualisieren.

Die Klasse SIFT_Params enthält alle Parameter, die Sie für den SIFT-Algorithmus benötigen. Die Parameter sind bereits mit Standardwerten initialisiert und müssen eigentlich nicht geändert werden.
Hinweis: Der Scale Space sollte in seiner letzten Oktave ein Bild mit mindestens 12 Pixeln Breite und Länge enthalten. Wählen Sie deshalb ein Bild mit genügend Pixeln aus, aber halten Sie das Bild so klein wie möglich. Das beschleunigt die Berechnung deutlich. Empfohlen hier ist ein 128x128 Bild.
Hinweis 2: Der SIFT Algorithmus erwartet Graustufenbilder mit Werten im Intervall [0,1]. Transformieren Sie ihr Bild dementsprechend.

Die Klasse SIFT_KeyPoint enthält alle Informationen, die der SIFT-Algorithmus benötigt. Viele dieser Variablen sind jedoch nicht für Sie wichtig. Sie sollten lediglich die Variablen o=Oktave, s=Scale, m=x-Koordinate und n=y-Koordinate setzen. Die anderen Variablen werden von der Klasse SIFT_Algorithm gesetzt.

Zur Visualisierung können Sie die Methoden visualize_scale_space und visualize_keypoints aus der Datei SIFT_Visualization.py nutzen. Die Methode visualize_scale_space zeigt Ihnen den kompletten Scale-Space in einem Plot an. Die Methode visualize_keypoints legt Extrema/Keypoints auf diesen Scale-Space.
Hinweis: Für eine bessere Darstellung bietet es sich an, die Extrema auf den normalisierten DoG-Bildern anzuzeigen, die KeyPoints jedoch auf der Urspünglichen Skala. Alle SIFT-algo.py Methoden sollten jedoch immer mit nicht-normalisierten DoG oder Skala-Bildern arbeiten.

Ihre Implementierung sollte drei Methoden umfassen, die ebenso in SIFT_algo.py zu finden sind: create_scale_space, create_dogs und find_discrete_extremas. Entscheiden Sie selbst, welche Parameter diese Methoden benötigen. Ihre Extrema sollten vom Typ SIFT_Keypoint sein.

Hinweis: Weitere Methoden des SIFT-Algorithmus SIFT_algo.py benötigen die zusätzlichen Variablen deltas und sigmas. Diese Variablen sollten während create_scale_space erstellt werden.

2 Test des Multi-Layer-Perceptrons beim MNIST Datensatz

a) Laden Sie das Pakete sklearn herunter und binden es mit Hilfe von from sklearn.neural_network import MLPClassifier ein.
b) Laden sie das Paket pyTorch herunter und binden es folgend ein

```
from tensorflow.examples.tutorials.mnist import input_data
```

c) Laden sie den MNIST Datensatz wie folgt. Er ist bereits in Trainings- und Validierungsdaten unterteilt. Die Bilder müssen noch durch ein Reshape in ein 1D array überführt werden!

```
from tensorflow.keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels)
= mnist.load_data()
train = mnist.train.images
validation = mnist.validation.images
```

d) Nutzen sie die Funktion mlp = MLPClassifier() aus der sklearn Bibliothek, um mehrere MLPs mit verschiedenen Konfigurationen zu erstellen. Sehen sie sich dazu die Dokumentation der Funktion an.

- Sie sollen folgende Konfigurationen und deren Kombination ausprobieren (insgesamt 54): - Maximum Iterationen: 5, 10, 15 - Lernrate: 0,1 und 0,01 - Anzahl der Layers: 2,3,4 - Solver bei der Backpropagation: "adam", "sdg" und "lbfgs"
- Für jeden Run sollen sie folgende Performancemetriken für das final trainierte Netzwerk berechnen: - Accuracy - Precision - Recall - F1-Score Dokumentieren Sie ihre Ergebnisse und laden Sie diese im Moodle mit hoch. vor
- Diskutieren sie jeweils welche Konfigurationen sich auf welche Performancemetrik auswirkt. Treffen sie dabei Aussagen wie ?Wenn ich X erhöhe, wird Y erhöht?.