Mustererkennung WiSe 12/13
Übung 11

Lutz Freitag, Sebastian Kürten

1 Aufgabe 1: Raucherkennung

Der Rauch in den Bilder erscheint als gleichgerichtete Bewegung von Bildregionen. Ein guter Ansatz wäre also der optische Fluss im Bild, bei dem der Klassifikator zusammengehörige Regionen erkennt und eine Region als Rauchkandidaten aussucht, deren Bewegungsrichtung nicht der Hauptbewegungsrichtung im Bild entspricht.

Um die Regionen in den Bildern zu finden, werden die Helligkeitsunterschiede untersucht. Durch Anwendung des morphologischen ÖpenÖperators auf dem Differenzbild zwischen zwei aufeinanderfolgenden Bildern können zusammenhängende Regionen sichtbar gemacht werden, in denen sich etwas im Bild bewegt. Die Differenz der so entstehenden Bilder schränkt nocheinmal die Menge der Kandidaten ein. Die Methode funktioniert gut in Bildern, in denen sich wenig bewegt (kein Wind, keine Wolken), schlecht in Bildern, in denen die Rauchsäule verhältnismäßig klein ist oder generell viel Bewegung herscht.

Wenn man noch die Information hätte, wohin sich die Region bewegt (optischer Fluss, Tracking), ließe sich ein Klassifikator nach einer einfachen Heuristik implementieren.

2 Code

2.1 a1.m

```
1
2  function img = normalizeImg(in)
3      minimum = min(in(:));
4      img = in .- minimum;
5      maximum = max(img(:));
6      img = img ./ maximum;
7  endfunction
8
9  img10 = double(imread("wildfire/raw/10.png"));
10 img11 = double(imread("wildfire/raw/11.png"));
11 img12 = double(imread("wildfire/raw/12.png"));
12
13 img20 = double(imread("wildfire/raw/20.png"));
14 img21 = double(imread("wildfire/raw/21.png"));
15 img22 = double(imread("wildfire/raw/22.png"));
16
17 img30 = double(imread("wildfire/raw/30.png"));
18 img31 = double(imread("wildfire/raw/31.png"));
19 img32 = double(imread("wildfire/raw/32.png"));
20
21 img10 = normalizeImg(img10);
22 img11 = normalizeImg(img11);
23 img12 = normalizeImg(img12);
24
25 img20 = normalizeImg(img20);
26 img21 = normalizeImg(img21);
27 img22 = normalizeImg(img22);
28
29 img30 = normalizeImg(img30);
30 img31 = normalizeImg(img31);
31 img32 = normalizeImg(img32);
32
33 kernel = ones(5,5)
34
```

```
subplot(2, 3, 1)
bla1 = dilate(erode(((img10 - img11)), kernel),kernel);
bla2 = dilate(erode(((img11 - img12)), kernel),kernel);
imshow(img11 .* ((bla1 - bla2) < 0))

subplot(2, 3, 2)
bla1 = dilate(erode(((img20 - img21)), kernel),kernel);
bla2 = dilate(erode(((img21 - img22)), kernel),kernel);
imshow(img21 .* ((bla1 - bla2) < 0))

subplot(2, 3, 3)
bla1 = dilate(erode(((img30 - img31)), kernel),kernel);
bla2 = dilate(erode(((img31 - img32)), kernel),kernel);
imshow(img31 .* ((bla1 - bla2) < 0))

subplot(2, 3, 4)
imshow(img11)

subplot(2, 3, 5)
imshow(img21)

subplot(2, 3, 6)
imshow(img31)


print("features.png");

pause
```

3 Plots

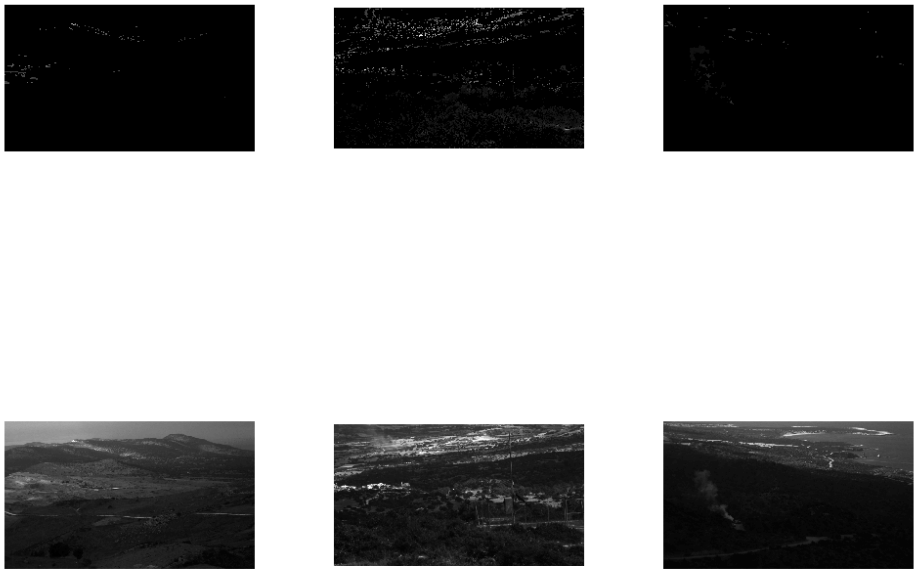

Abbildung 1: Ausgabe des Filters und der zugrundeliegenden Bilder