Exercise8

1 CS-E4850 Computer Vision Exercise Round 8

The problems should be solved before the exercise session and solutions returned via MyCourses. For this exercise round you should return a pdf file containing written answers to the questions below.

1.0.1 Exercise 1. Face tracking example using KLT tracker

Run the example as instructed below and answer the questions.

a) Run `Exercise8.ipynb`
b) Run `Exercise8.ipynb` with a different input by changing the input to obama.avi: `frames=faceTracker('obama.avi')`
c) What could be the main reasons why most of the features are not tracked very long in case b) above?
d) How could one try to avoid the problem of gradually losing the features? Suggest one or more improvements.
e) Voluntary task: Capture a video of your own face or of a picture of a face, and check that whether the tracking works for you. That is, replace the input video path in `faceTrackingDemo.py` with the path to your own video.

1.0.2 Exercise 2. Kanade-Lucas-Tomasi (KLT) feature tracking (Pen & paper problem)

Read Sections 2.1 and 2.2 from the paper by Baker and Matthews. Show that the Equation (10) in the paper gives the same solution as the equations on slide 25 of Lecture 7, when the geometric warping W (between the current frame and the template window in the previous frame) is a translation.

[1]:
```
# This cell is used for creating a button that hides/unhides code cells to␣
↪quickly look only the results.
# Works only with Jupyter Notebooks.

import os
from IPython.display import HTML

HTML('''<script>
code_show=true;
function code_toggle() {
```

```
if (code_show){
$('div.input').hide();
} else {
$('div.input').show();
}
code_show = !code_show
}
$( document ).ready(code_toggle);
</script>
<form action="javascript:code_toggle()"><input type="submit" value="Click here␣
↪to toggle on/off the raw code."></form>''')
```

[1]:
```
<IPython.core.display.HTML object>
```

[2]:
```
# Description:
#   Exercise8 python demo.
#
# Copyright (C) 2018 Santiago Cortes, Juha Ylioinas, Tapio Honka
#
# This software is distributed under the GNU General Public
# Licence (version 2 or later); please refer to the file
# Licence.txt, included with the software, for details.

import matplotlib.pyplot as plt
import matplotlib.animation as animation
from IPython.display import HTML
from faceTrackingDemo import faceTracker
```

```
The data directory is /coursedata
Data stored in /coursedata/exercise-08-data
```

[5]:
```
%%capture
fig = plt.figure(figsize=(10,10))

# frames of the processed input video
# change the input to obama.avi in part b)
frames = faceTracker('santi.avi')

# create an animation that can be embedded in the notebook
ani = animation.ArtistAnimation(fig, frames, interval=50, blit=True,␣
↪repeat_delay=2000)
```

[4]:
```
display(HTML(ani.to_html5_video()))
```

```
<IPython.core.display.HTML object>
```

```
[9]: %%capture
     fig = plt.figure(figsize=(10,10))

     # frames of the processed input video

     frames = faceTracker('obama.avi')

     # create an animation that can be embedded in the notebook
     ani = animation.ArtistAnimation(fig, frames, interval=50, blit=True,␣
      ↪repeat_delay=2000)
```

```
[8]: display(HTML(ani.to_html5_video()))
```

```
<IPython.core.display.HTML object>
```

c) What could be the main reasons why most of the features are not tracked very long in case b) above?

Because the tracking algortihm will tracks linears and eliminate outliers from the last frame by using RANSAC. key points will be missed when the image is rotated or the camera moves ata fast speed.

d) How could one try to avoid the problem of gradually losing the features? Suggest one or more improvements.

we try to avoid large movements over a short period of time so the system can learn new features and track them after a period of time. another try is to keep outliers rather than eliminating them for further tracking, but the problem is points always exist even if you move your face out of the camera.