FYS2150
Lab Report: Drag

Nicholas Karlsen

April 26, 2018

Abstract

A study on the flow of an assortment of spheres in a fluid and the use of image processing to determine the terminal velocity.

1 Introduction

This report contains the description and analysis of data collected in the lab 21.03.2018 concerning the flow of several spherical objects in a large range of different sizes and densities. The balls were immersed in fluid, dropped and filmed. Post-lab, the raw footage was then processed using a Python script in order to quantify the motion of the spheres. This

2 Theory

3 Experimental Procedure

3.1 Video capture

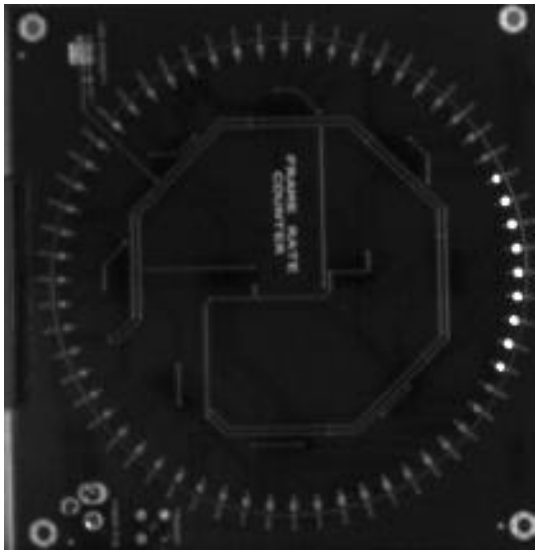

Figure 1: Signal used to determine the FPS of the camera

In order to capture the motion of the balls, a USB video camera was connected to a

computer running uEye cockpit [1], which we used to change the settings of the camera and make recordings.

First, the error of the stated FPS of the camera had to be determined. This was done by connecting a series of circular LEDs (see Fig. 1) to a signal generator. The light emitted would "circle" at a rate which could be changed using the signal generator. By adjusting the signal such that the emitted light would seem stationary when observed through the video feed in uEye cockpit, the FPS of the camera determined to be

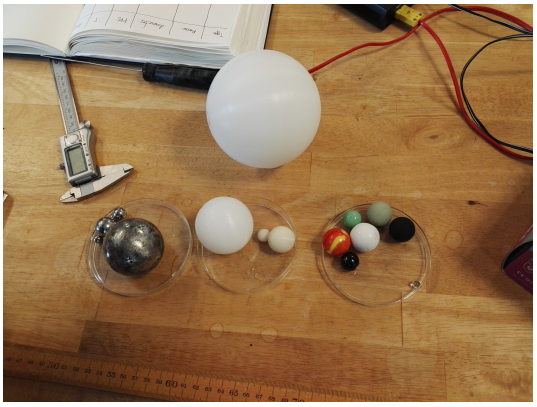

Figure 3: Most of the balls used in the experiment, excluding the ones labeled small 1 and 2.

4 Results

Type	Mass [g]	Diameter [mm]	FPS	T [°C]	Filename
Metal	502.76	48.98	100	22.7	A1.avi
Metal	28.13	19.02	100	22.8	A2.avi
Metal	6.99	11.97	100	22.6	A3.avi
Metal	2.08	7.99	100	22.6	A4.avi
Metal	0.68	5.48	100	22.5	A5.avi
Metal	0.10	2.98	100	22.6	A6.avi
Plastic	488.41	99.4	100	22.5	B1.avi
Plastic	61.56	50.02	100	22.5	B2.avi
Plastic	7.12	23.89	100	22.6	B3.avi
Plastic	0.87	12.06	100	22.6	B4.avi
White	29.74	25.24	100	22.6	C1.avi
BigB lack	31.42	21.08	100	22.5	C4.avi
Small Black	5.67	16.45	100	22.5	C5.avi
Big Green	31.60	21.86	100	22.4	C3.avi
Small Green	5.60	16.38	100	22.3	C6.avi
Big Red	18.44	24.01	100	22.3	C2.avi
Glass	0.27	5.81	100	22.3	C7.avi
Small 2	12.0E-3	1.59	100	23.7	D2.avi
Small 1	4.1E-3	1.0	100	23.7	D1.avi

Table 1: Spheres

5 Discussion

6 Conclusion

References

[1] DS Imaging Development Systems GmbH. ueye cockpit.

*

A Code

Following

scripts/lesVideo_conv.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
'''
Reads video file and converts to binary image
resulting in easy data analysis.
author: Nicholas Karlsen

Note: skvideo is not included in anaconda python,
install by 'pip install sk-video' in terminal.
'''

import numpy as np
import skvideo.io
import inspect
import os
import matplotlib.pyplot as plt
import scipy.constants
from skimage.measure import regionprops
from matplotlib.image import imread
from skimage import util
import FYS2150lib as fys # Used for linfit
# import skimage.color
# from PIL import Image
# import skimage.morphology as morph
# from skimage import filters


def rgb2gray(rgb):
    '''
    Converts shape=(N,M,rgb) array to (N, M) grayscale array see wiki page
    '''
    return np.dot(rgb[..., :3], [0.299, 0.587, 0.114]).astype(int)


def gray2binary(gray, limBW=128):
    """Converts grayscale image to binary grayscale of 0 OR 255
    image must be array of shape=(N, M)
    gray: (N, M) array
    limBW:
    """
    bw = np.asarray(gray).copy()
    bw[bw < limBW] = 0       # Black
    bw[bw >= limBW] = 255    # White
    return bw

```

```
46 def genFilter(image):
47     """
48     Generates an array to filter out
49     static background based on first frame
50 
51     NOT YET IMPLEMENTED
52     """
53     gsImage = rgb2gray(image)
54     bwImage = gray2binary(gsImage)
55     bwImage = bwImage / 255.0
56     return bwImage.astype(int)
57 
58 def trackCircle(filename="litenmetallkule.avi", path="current",
59                 hMin=0, hMax=-1, wMin=0, wMax=-1):
60     """
61     Takes video file as input, filters out static background based on
62     first frame and finds the CM of circle in every frame. Requires
63     circle to be only object in frame (after filtering), so requires static
64     background. If not, try adjust hMin, hMax, wMin, wMax to crop out
       moving
65     background.
66     filename: filename of video
67     path: FULL path of file, eg '/home/nick/Videos/fys2150drag'.
68           if left as default, it will asume same path as script.
69     hMin, hMax, wMin, wMax: used for cropping the image.
70     """
71 
72     # Fetching current dir path
73     folderPath = os.path.dirname(
74         os.path.abspath(
75             inspect.getfile(
76                 inspect.currentframe())))
77 
78     # If path is specified, use that instead.
79     if path != "current":
80         folderPath = path
81         "if path is specified"
82 
83     fullFilename = folderPath + "/" + filename
84 
85     print "Reading video..."
86 
87     video = skvideo.io.vread(fullFilename)
88     totalFrames = len(video)
89 
90     print "Number of frames:", len(video)
91 
92     frameStart = 0
93     frameStop = totalFrames
94 
95     "Creates array to store x, y vals of CM"
96     cmPos = np.zeros([frameStop - frameStart, 2])
97 
```

```
98      validFrames = []  # Keeps track of usable frames
99
100     def detectCirc(image):
101         """
102         Inverts color of image and detects center of circle shape.
103         Asumes circle is the ONLY object in image, so noise
104         needs to be filtered out
105         """
106         #staticBg = genFilter(video[0])
107
108         invFrame = image
109         bwFrame = gray2binary(
110             rgb2gray(
111                 util.invert(invFrame)))[hMin:hMax, wMin:wMax]
112         bwFrame = bwFrame  # * staticBg
113         # Detects shapes in image
114         props = regionprops(label_image=bwFrame.astype(int))
115
116         return props, invFrame, bwFrame
117
118     for frame in xrange(totalFrames):
119         """
120         Need to invert image for regionprops to work, only finds white obj
121         on black background, not black on white.
122         """
123         # convert to binary grayscale to filter out noise
124         props = detectCirc(video[frame])[0]
125
126         # Bad way of checking if the ball is in frame
127         if len(props) == 0:
128             cmPos[frame] = "nan"
129         else:
130             cmPos[frame] = props[0].centroid  # Detects centroids
131             validFrames.append(frame)  # Keeps track of frames with ball
132
133             # Print info to terminal while processing
134             print "frame", frame,\
135                   "-", "Center of mass:",\
136                   "x=%i, y=%i" % (cmPos[frame][1], cmPos[frame][0])
137
138     def plot_im(frame=int(totalFrames / 2.0)):
139         "plot frame + CM, used to check functionality"
140         im = video[frame]
141         props, invFrame, bwFrame = detectCirc(im)
142         cmPos[frame] = props[0].centroid
143         plt.subplot(311)
144         plt.imshow(invFrame)
145         plt.title("Raw image, frame:%i" % frame)
146         plt.plot(cmPos[frame, 1], cmPos[frame, 0] + hMin,
147                  "ro", label="Center of mass")
148         plt.legend()
149         plt.subplot(312)
150         plt.imshow(bwFrame, cmap=plt.get_cmap('gray'))
```

```
151             plt.plot(cmPos[frame, 1], cmPos[frame, 0],
152                      "ro", label="Center of mass")
153             plt.title("Processed image, frame:%i" % frame)
154             plt.savefig("figs/graphs/%s_1.png"%vids[row][:-4])
155             plt.legend()
156             plt.show()
157         plot_im()
158
159         return cmPos.astype(int), validFrames
160
161
162     def testFunc():
163         """
164         Testing that method of finding C.M works properly
165         """
166         import skimage.color
167         #img = imread("bilde5.png")
168         img = imread("frame_inv2.png")
169         bwImg = skimage.color.rgb2gray(img)
170         plt.subplot(211)
171         plt.imshow(bwImg, cmap=plt.get_cmap('gray'))
172
173         props = regionprops(label_image=bwImg.astype(int))
174         cm = props[0].centroid
175
176         plt.subplot(212)
177         plt.imshow(img)
178         plt.plot(cm[1], cm[0], "ro",
179                  label="Center of mass = (%i, %i)" % (cm[1], cm[0]))
180         plt.legend()
181         plt.show()
182
183
184
185
186     if __name__ == "__main__":
187
188         def readlabdat(filename):
189             """
190             Used to read the file which stores the parameters of the
191             sphere
192             """
193             vids = []; mass = []; radius = []; temp = []
194
195             file = open(filename, "r")
196             for line in file:
197                 cols = line.split()
198                 mass.append(cols[1])
199                 radius.append(cols[2])
200                 temp.append(cols[-2])
201                 vids.append(cols[-1])
202             file.close()
203
```

```
204         return mass, radius, temp, vids
205     mass, radius, temp, vids = readlabdat("data/labdata.dat")
206 
207     folderPath = "/home/nick/Videos/fys2150drag"
208 
209     rows = [7, 8, 9, 10]
210 
211     outfile = open("data/B_results.dat", "w")
212 
213     for row in rows:
214         cm, validFrames = trackCircle(filename=str(vids[row]),
215                                       path=folderPath,
216                                       hMin=67, hMax=216)
217 
218         if len(cm[:, 1]) != len(validFrames):
219             print "Tracking interupted in some frames,"
220             print "Only returning uninterupted frames."
221             x = []
222             y = []
223             for validFrame in validFrames:
224                 x.append(cm[validFrame, 1])
225                 y.append(cm[validFrame, 0])
226             x = np.array(x).astype(int)
227             y = np.array(y).astype(int)
228         else:
229             x = cm[validFrames[0]:validFrames[-1], 1]
230             y = cm[validFrames[0]:validFrames[-1], 0]
231 
232         x = np.array(x)
233         y = np.array(y)
234         validFrames = np.array(validFrames)
235 
236         print "Find start/stop of terminal velocity (straight, steep line)
    to perform linfit:"
237         plt.subplot(211)
238         plt.plot(validFrames, x, "o")
239         plt.xlabel("Frame")
240         plt.ylabel("x-position of center of mass [px]")
241         plt.title("Use to determine start/stop frame of linfit")
242         plt.subplot(212)
243         plt.plot(np.diff(x))
244         plt.show()
245 
246         start = int(input("Start index:"))
247         stop = int(input("Stop index:"))
248 
249         m, c, dm, dc = fys.linfit(validFrames[start:stop], x[start:stop])
250 
251         plt.subplot(211)
252         plt.plot(validFrames, x, ".", label="Position of CM")
253         plt.plot(validFrames[start:stop],
254                  validFrames[start:stop] * m + c,
255                  label="linear fit, y=mx+c")
```

```
256             plt.text(0, 1000, "m = %i [px/frame]\n dm = %i [px/frame]" % (m, dm
        ))
257             plt.xlabel("Frame")
258             plt.ylabel("x-pos [px]")
259             plt.legend()
260             plt.subplot(212)
261             plt.plot(validFrames, y, ".", label="Position of CM")
262             plt.xlabel("Frame")
263             plt.ylabel("y-pos  [px]")
264             plt.savefig("figs/graphs/%s_2.png"%vids[row][:-4])
265             plt.show()
266
267             outfile.write(vids[row] + " & " + "%i"%(m) + " & " + "%i"%dm + "&"
        + "%s"%mass[row] + "&" + "%s"%radius[row] +"\\\ \n")
268
269         # More plots
270
271     outfile.close()
```

scripts/data/labdata.dat

Type	Mass[g]	Diameter[mm]	FPS	T[C]	Filename
Metal	502.76	48.98	100	22.7	A1.avi
Metal	28.13	19.02	100	22.8	A2.avi
Metal	6.99	11.97	100	22.6	A3.avi
Metal	2.08	7.99	100	22.6	A4.avi
Metal	0.68	5.48	100	22.5	A5.avi
Metal	0.10	2.98	100	22.6	A6.avi
Plastic	488.41	99.4	100	22.5	B1.avi
Plastic	61.56	50.02	100	22.5	B2.avi
Plastic	7.12	23.89	100	22.6	B3.avi
Plastic	0.87	12.06	100	22.6	B4.avi
White	29.74	25.24	100	22.6	C1.avi
BigBlack	31.42	21.08	100	22.5	C4.avi
SmallBlack	5.67	16.45	100	22.5	C5.avi
BigGreen	31.60	21.86	100	22.4	C3.avi
SmallGreen	5.60	16.38	100	22.3	C6.avi
BigRed	18.44	24.01	100	22.3	C2.avi
Glass	0.27	5.81	100	22.3	C7.avi
Small 1	4.1e-3	1.0	100	23.7	D1.avi
Small 2	12.0e-3	1.59	100	23.7	D2.avi