FYS2150
Lab Report: Drag

Nicholas Karlsen

April 24, 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 Determining framerate of the camera

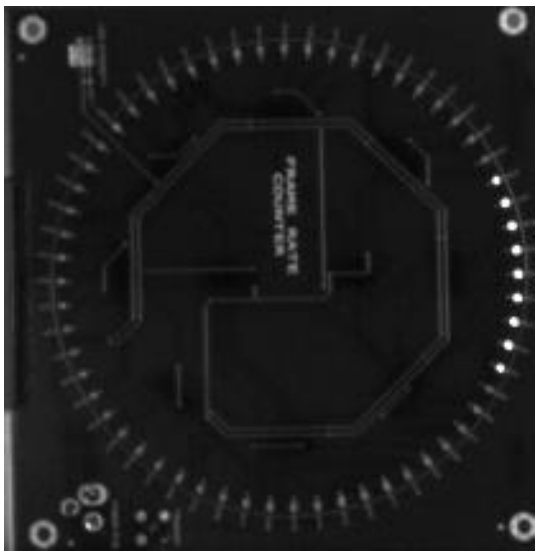

Figure 1: Signal used to determine the FPS of the camera

(a) Large tube (b) Small tube

Figure 2: Cropped images used to find pixel to meter ratio for both of the tubes. (Scaled down in this document)

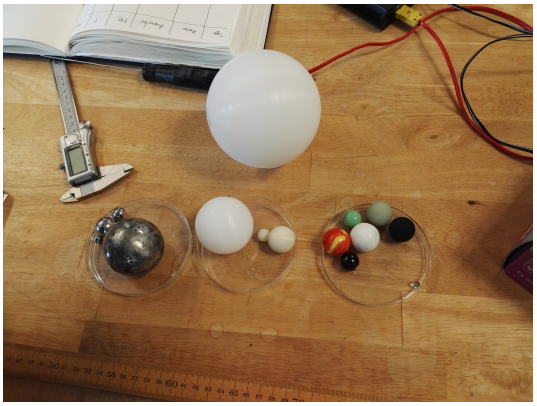

Figure 3: Most of the balls used in the experiment, excluding the ones labeled small 1 and 2.

4 Discussion

5 Results

Type	Mass[g]	Diameter[mm]	FPS	T[C]	Filena
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

Table 1

6 Conclusion

*

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
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

def genFilter(image):
    """
```

```
47      Generates an array to filter out
48      static background based on first frame
49
50      NOT YET IMPLEMENTED
51      """
52      gsImage = rgb2gray(image)
53      bwImage = gray2binary(gsImage)
54      bwImage = bwImage / 255.0
55      return bwImage.astype(int)
56
57  def trackCircle(filename="litenmetallkule.avi", path="current",
58                  hMin=0, hMax=-1, wMin=0, wMax=-1):
59      """
60      Takes video file as input, filters out static background based on
61      first frame and finds the CM of circle in every frame. Requires
62      circle to be only object in frame (after filtering), so requires static
63      background. If not, try adjust hMin, hMax, wMin, wMax to crop out
        moving
64      background.
65      filename: filename of video
66      path: FULL path of file, eg '/home/nick/Videos/fys2150drag'.
67            if left as default, it will asume same path as script.
68      hMin, hMax, wMin, wMax: used for cropping the image.
69      """
70
71      # Fetching current dir path
72      folderPath = os.path.dirname(
73          os.path.abspath(
74              inspect.getfile(
75                  inspect.currentframe())))
76
77      # If path is specified, use that instead.
78      if path != "current":
79          folderPath = path
80          "if path is specified"
81
82      fullFilename = folderPath + "/" + filename
83
84      print "Reading video..."
85
86      video = skvideo.io.vread(fullFilename)
87      totalFrames = len(video)
88
89      print "Number of frames:", len(video)
90
91      frameStart = 0
92      frameStop = totalFrames
93
94      "Creates array to store x, y vals of CM"
95      cmPos = np.zeros([frameStop - frameStart, 2])
96
97      validFrames = []  # Keeps track of usable frames
98
```

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

```
152             plt.title("Processed image, frame:%i" % frame)
153             plt.legend()
154             plt.show()
155         plot_im()
156 
157         return cmPos.astype(int), validFrames
158 
159 
160 def testFunc():
161         """
162         Testing that method of finding C.M works properly
163         """
164         import skimage.color
165         #img = imread("bilde5.png")
166         img = imread("frame_inv2.png")
167         bwImg = skimage.color.rgb2gray(img)
168         plt.subplot(211)
169         plt.imshow(bwImg, cmap=plt.get_cmap('gray'))
170 
171         props = regionprops(label_image=bwImg.astype(int))
172         cm = props[0].centroid
173 
174         plt.subplot(212)
175         plt.imshow(img)
176         plt.plot(cm[1], cm[0], "ro",
177                  label="Center of mass = (%i, %i)" % (cm[1], cm[0]))
178         plt.legend()
179         plt.show()
180 
181 
182 if __name__ == "__main__":
183 
184         folderPath = "/home/nick/Videos/fys2150drag"
185 
186         vidLabel = "A1"
187 
188         cm, validFrames = trackCircle(filename=vidLabel + ".avi",
189                                       path=folderPath,
190                                       hMin=67, hMax=216)
191 
192         if len(cm[:, 1]) != len(validFrames):
193             print "Tracking interupted in some frames,"
194             print "Only returning uninterupted frames."
195             x = []
196             y = []
197             for validFrame in validFrames:
198                 x.append(cm[validFrame, 1])
199                 y.append(cm[validFrame, 0])
200             x = np.array(x).astype(int)
201             y = np.array(y).astype(int)
202         else:
203             x = cm[validFrames[0]:validFrames[-1], 1]
204             y = cm[validFrames[0]:validFrames[-1], 0]
```

```
205
206     x = np.array(x)
207     y = np.array(y)
208     validFrames = np.array(validFrames)
209
210     print "Find start/stop of terminal velocity (straight,\
211            steep line) to perform linfit:"
212
213     plt.plot(validFrames, x, "o")
214     plt.xlabel("Frame")
215     plt.ylabel("x-position of center of mass [px]")
216     plt.title("Use to determine start/stop frame of linfit")
217     plt.show()
218
219     start = int(input("Start index:"))
220     stop = int(input("Stop index:"))
221
222     m, c, dm, dc = fys.linfit(validFrames[start:stop], x[start:stop])
223
224     plt.subplot(211)
225     plt.plot(validFrames, x, ".", label="Position of CM")
226     plt.plot(validFrames[start:stop], validFrames[start:stop] * m + c,
        label="linear fit, y=mx+c")
227     plt.text(0, 1000, "m = %i [px/frame]\n dm = %i [px/frame]" % (m, dm))
228     plt.xlabel("Frame")
229     plt.ylabel("x-pos [px]")
230     plt.legend()
231     plt.subplot(212)
232     plt.plot(validFrames, y, ".", label="Position of CM")
233     plt.xlabel("Frame")
234     plt.ylabel("y-pos  [px]")
235     plt.show()
```

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

19	SmallGreen	5.60	16.38	100	22.3	C6.avi
20	BigRed	18.44	24.01	100	22.3	C2.avi
21	Glass	0.27	5.81	100	22.3	C7.avi
22						
23	Small 1	4.1e−3	1.0	100	23.7	D1.avi
24	Small 2	12.0e−3	1.59	100	23.7	D2.avi