FYS2150
Lab Report: Drag

Nicholas Karlsen

April 25, 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

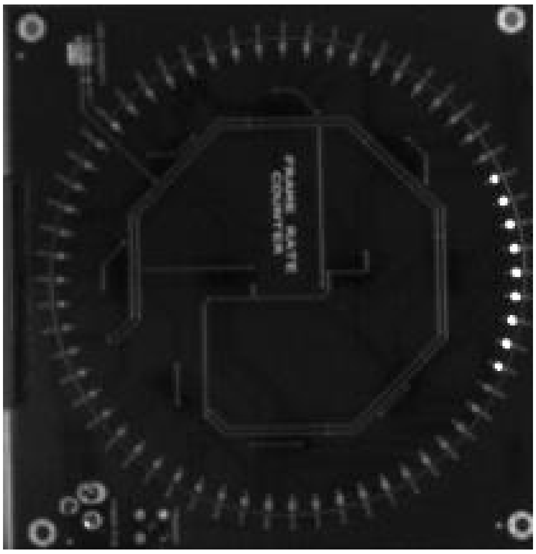

Figure 1: Signal used to determine the FPS of the camera

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

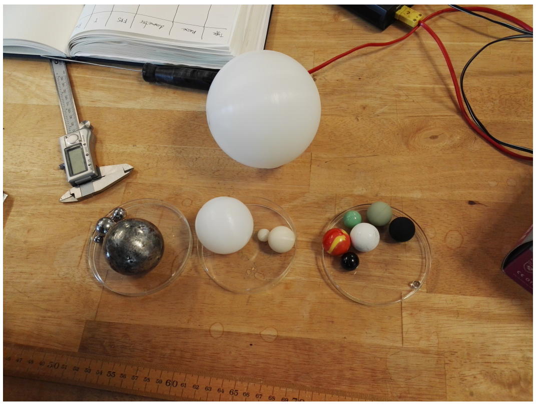

Figure 3: Most of the balls used in the experiment, excluding the ones labeled small 1 and 2.

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

```python
#!/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):
    """
    Generates an array to filter out
    static background based on first frame

    NOT YET IMPLEMENTED
    """
    gsImage = rgb2gray(image)
    bwImage = gray2binary(gsImage)
    bwImage = bwImage / 255.0
    return bwImage.astype(int)

def trackCircle(filename="litenmetallkule.avi", path="current",
                hMin=0, hMax=-1, wMin=0, wMax=-1):
    """
    Takes video file as input, filters out static background based on
    first frame and finds the CM of circle in every frame. Requires
    circle to be only object in frame (after filtering), so requires static
    background. If not, try adjust hMin, hMax, wMin, wMax to crop out moving
    background.
    filename: filename of video
    path: FULL path of file, eg '/home/nick/Videos/fys2150drag'.
          if left as default, it will asume same path as script.
    hMin, hMax, wMin, wMax: used for cropping the image.
    """

    # Fetching current dir path
    folderPath = os.path.dirname(
        os.path.abspath(
            inspect.getfile(
                inspect.currentframe())))

    # If path is specified, use that instead.
    if path != "current":
        folderPath = path
        "if path is specified"

    fullFilename = folderPath + "/" + filename

    print "Reading video..."

    video = skvideo.io.vread(fullFilename)
    totalFrames = len(video)

    print "Number of frames:", len(video)

    frameStart = 0
    frameStop = totalFrames

    "Creates array to store x, y vals of CM"
    cmPos = np.zeros([frameStop - frameStart, 2])

    validFrames = []  # Keeps track of usable frames
```

```python

    def detectCirc(image):
        """
        Inverts color of image and detects center of circle shape.
        Asumes circle is the ONLY object in image, so noise
        needs to be filtered out
        """
        #staticBg = genFilter(video[0])

        invFrame = image
        bwFrame = gray2binary(
            rgb2gray(
                util.invert(invFrame)))[hMin:hMax, wMin:wMax]
        bwFrame = bwFrame  # * staticBg
        # Detects shapes in image
        props = regionprops(label_image=bwFrame.astype(int))

        return props, invFrame, bwFrame

    for frame in xrange(totalFrames):
        """
        Need to invert image for regionprops to work, only finds white obj
        on black background, not black on white.
        """
        # convert to binary grayscale to filter out noise
        props = detectCirc(video[frame])[0]

        # Bad way of checking if the ball is in frame
        if len(props) == 0:
            cmPos[frame] = "nan"
        else:
            cmPos[frame] = props[0].centroid  # Detects centroids
            validFrames.append(frame)  # Keeps track of frames with ball

            # Print info to terminal while processing
            print "frame", frame,\
                  "-", "Center of mass:",\
                  "x=%i, y=%i" % (cmPos[frame][1], cmPos[frame][0])

    def plot_im(frame=int(totalFrames / 2.0)):
        "plot frame + CM, used to check functionality"
        im = video[frame]
        props, invFrame, bwFrame = detectCirc(im)
        cmPos[frame] = props[0].centroid
        plt.subplot(311)
        plt.imshow(invFrame)
        plt.title("Raw image, frame:%i" % frame)
        plt.plot(cmPos[frame, 1], cmPos[frame, 0] + hMin,
                 "ro", label="Center of mass")
        plt.legend()
        plt.subplot(312)
        plt.imshow(bwFrame, cmap=plt.get_cmap('gray'))
        plt.plot(cmPos[frame, 1], cmPos[frame, 0],
                 "ro", label="Center of mass")
        plt.title("Processed image, frame:%i" % frame)
        plt.legend()
        plt.show()
    plot_im()

    return cmPos.astype(int), validFrames


def testFunc():
    """
    Testing that method of finding C.M works properly
    """
    import skimage.color
    #img = imread("bilde5.png")
    img = imread("frame_inv2.png")
    bwImg = skimage.color.rgb2gray(img)
    plt.subplot(211)
    plt.imshow(bwImg, cmap=plt.get_cmap('gray'))

    props = regionprops(label_image=bwImg.astype(int))
    cm = props[0].centroid

    plt.subplot(212)
    plt.imshow(img)
    plt.plot(cm[1], cm[0], "ro",
             label="Center of mass = (%i, %i)" % (cm[1], cm[0]))
    plt.legend()
    plt.show()




if __name__ == "__main__":

    def readlabdat(filename):
        """
        Used to read the file which stores the parameters of the
        sphere
        """
        vids = []; mass = []; radius = []; temp = []

        file = open(filename, "r")
        for line in file:
            cols = line.split()
            mass.append(cols[1])
            radius.append(cols[2])
            temp.append(cols[-2])
            vids.append(cols[-1])
        file.close()

        return mass, radius, temp, vids
    mass, radius, temp, vids = readlabdat("data/labdata.dat")
```

```python

    folderPath = "/home/nick/Videos/fys2150drag"

    rows = range(1, 19)

    outfile = open("data/results.dat", "w")

    for row in rows:
        cm, validFrames = trackCircle(filename=str(vids[row]),
                                      path=folderPath,
                                      hMin=67, hMax=216)

        if len(cm[:, 1]) != len(validFrames):
            print "Tracking interupted in some frames,"
            print "Only returning uninterupted frames."
            x = []
            y = []
            for validFrame in validFrames:
                x.append(cm[validFrame, 1])
                y.append(cm[validFrame, 0])
            x = np.array(x).astype(int)
            y = np.array(y).astype(int)
        else:
            x = cm[validFrames[0]:validFrames[-1], 1]
            y = cm[validFrames[0]:validFrames[-1], 0]

        x = np.array(x)
        y = np.array(y)
        validFrames = np.array(validFrames)

        print "Find start/stop of terminal velocity (straight, steep line) to perform linfit:"

        plt.plot(validFrames, x, "o")
        plt.xlabel("Frame")
        plt.ylabel("x-position of center of mass [px]")
        plt.title("Use to determine start/stop frame of linfit")
        plt.show()

        start = int(input("Start index:"))
        stop = int(input("Stop index:"))

        m, c, dm, dc = fys.linfit(validFrames[start:stop], x[start:stop])

        plt.subplot(211)
        plt.plot(validFrames, x, ".", label="Position of CM")
        plt.plot(validFrames[start:stop],
                 validFrames[start:stop] * m + c,
                 label="linear fit, y=mx+c")
        plt.text(0, 1000, "m = %i [px/frame]\n dm = %i [px/frame]" % (m, dm))
        plt.xlabel("Frame")
        plt.ylabel("x-pos [px]")
        plt.legend()
        plt.subplot(212)
        plt.plot(validFrames, y, ".", label="Position of CM")
        plt.xlabel("Frame")
        plt.ylabel("y-pos  [px]")
        plt.show()

        outfile.write(vids[row] + " & " + "%i"%(m) + " & " + "%i"%dm + "\\\\n")

    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
```