Supplementary Materials

Calibration Python code

```
import cv2
import numpy as np

file = open("factor71.txt","w")

cam = cv2.VideoCapture(0)

kernel = np.ones((5,5),np.uint8)
font = cv2.FONT_HERSHEY_SIMPLEX
v_x = np.array([])
v_y = np.array([])
delta_mm = 20

while(True):

_,frame = cam.read()

frameHSV = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)



lower_blue = np.array([100,100,20],dtype = np.uint8)

upper_blue = np.array([130,255,255],dtype = np.uint8)

blue_pol = cv2.inRange(frameHSV, lower_blue, upper_blue)

purge_blue_pol = cv2.morphologyEx(blue_pol, cv2.MORPH_OPEN, kernel)

x1,y1,w1,h1 = cv2.boundingRect(purge_blue_pol)

blue_polygon = cv2.rectangle(frame,(x1,y1),(x1 + w1,y1 + h1),(0,255,0),2)

center_blue_pol = (x1 + (w1/2),y1 + (h1/2))

xblue = int(x1 + (w1/2))

yblue = int(y1 + (h1/2))

cv2.circle(frame,(x1 + w1,y1 + h1),7,(0,255,0),−1)

cv2.putText(frame,'{},{}'.format(x1 + w1,y1 + h1),(x1 + w1 + 10,y1 +
h1),font,0.75,(0,255,0),1,cv2.LINE_AA)

print(w1,h1)

v_x = np.append(v_x,w1)

v_y = np.append(v_y,h1)



cv2.imshow("video",frame)



if cv2.waitKey(1) & 0xFF == ord('s'):
```

```
break

cam.release()
cv2.destroyAllWindows()

prom_x = np.average(v_x)
prom_y = np.average(v_y)

prom = (prom_x + prom_y)/2

mm_por_px = float(delta_mm/prom)

print(mm_por_px)

file.write(str(mm_per_px))
file.close()
```

Recording Python code

```
import cv2
import numpy as np
import os
import time

cam = cv2.VideoCapture(0)

kernel = np.ones((5,5),np.uint8)
font = cv2.FONT_HERSHEY_SIMPLEX
a = np.array([])
b = np.array([])
c = np.array([])
d = np.array([])
e = np.array([])
f = np.array([])
g = np.array([])

t1 = np.array([])
t2 = np.array([])
tn = np.array([])
tstart = time.time()

M = np.array([])

while(True):

_,frame = cam.read()

frameHSV = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)



lower_blue = np.array([100,100,20],dtype = np.uint8)

upper_blue = np.array([130,255,255],dtype = np.uint8)

blue_pol = cv2.inRange(frameHSV, lower_blue, upper_blue)

purge_blue_pol = cv2.morphologyEx(blue_pol, cv2.MORPH_OPEN, kernel)
```

```
blue,_ = cv2.findContours(purge_blue_pol,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

for i in blue:

area1 = cv2.contourArea(i)

if area1 > 3000:

M1 = cv2.moments(i)

if (M1[“m00”] = =0):M1[“m00”] = 1

x1 = int(M1[“m10”]/M1[“m00”])

y1 = int(M1[“m01”]/M1[“m00”])

Hull1 = cv2.convexHull(i)

cv2.circle(frame,(x1,y1),7,(0,255,0),−1)

cv2.putText(frame, ‘{},{}’.format(x1,y1),(x1 + 10,y1),font,0.75,(0,255,0),1,cv2.LINE_AA)

cv2.drawContours(frame,[Hull1],0,(0,255,0),1)

t1 = time.time()—tstart

print(t1)

a = np.append(a,x1)

b = np.append(b,y1)

e = np.append(e,t1)



lower_red1 = np.array([0,100,20],dtype = np.uint8)

upper_red1 = np.array([35,255,255],dtype = np.uint8)

lower_red2 = np.array([145,100,20],dtype = np.uint8)

upper_red2 = np.array([179,255,255],dtype = np.uint8)

red_pol1 = cv2.inRange(frameHSV, lower_red1, upper_red1)

red_pol2 = cv2.inRange(frameHSV, lower_red2, upper_red2)

red_pol = cv2.add(red_pol1,red_pol2)

purge_red_pol = cv2.morphologyEx(red_pol, cv2.MORPH_OPEN, kernel)

red,_ = cv2.findContours(purge_red_pol,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

for i in red:
```

```
area2 = cv2.contourArea(i)

if area2 > 3000:

M2 = cv2.moments(i)

if (M2[“m00”] = =0):M2[“m00”] = 1

x2 = int(M2[“m10”]/M2[“m00”])

y2 = int(M2[“m01”]/M2[“m00”])

Hull2 = cv2.convexHull(i)

cv2.circle(frame,(x2,y2),7,(0,255,0),−1)

cv2.putText(frame, ‘{},{}’.format(x2,y2),(x2 + 10,y2),font,0.75,(0,255,0),1,cv2.LINE_AA)

cv2.drawContours(frame,[Hull2],0,(0,255,0),1)

t2 = time.time()—tstart

print(t2)

c = np.append(c,x2)

d = np.append(d,y2)

f = np.append(f,t2)

tn = abs(t2-t1)

g = np.append(g,tn)

print(tn)

cv2.imshow(“video”,frame)



if cv2.waitKey(1) & 0xFF == ord(‘s’):

break

cam.release()
cv2.destroyAllWindows()

print(np.size(a),np.size(b),np.size(c),np.size(d),np.size(e),np.size(f),np.size(g))

sa = np.size(a)
sb = np.size(b)
sc = np.size(c)
sd = np.size(d)
se = np.size(e)
sf = np.size(f)
sg = np.size(g)

if sa==sb==sc==sd==se==sf==sg:

M = np.zeros((np.size(a),7))
```

```
for i in range(0,sa):

M[i][0] = a[i]

M[i][1] = b[i]

M[i][2] = c[i]

M[i][3] = d[i]

M[i][4] = e[i]

M[i][5] = f[i]

M[i][6] = g[i]

np.savetxt('Matrix71.txt',M)
```