This program is designed to capture movements using the user's webcam. It opens several files for video-capturing in greyscale, negative, and colored scale (with contours designed to track on-screen movement). The end result returns a List of frame's; 0 indicating no objects on the screen, and 1 indicating an object. The last portion of code is used to create a Bokeh time series plot to demonstrate movement using the DateTime module!

[5]
```
import cv2, time, pandas
from datetime import datetime


first_frame = None
status_list=[None, None]
times=[]
df=pandas.DataFrame(columns=["Start", "End"])


video=cv2.VideoCapture(0)



while True:
    check, frame = video.read()
    status=0
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray=cv2.GaussianBlur(gray,(21,21),0)

    if first_frame is None:
        first_frame=gray
        continue

    delta_frame = cv2.absdiff(first_frame, gray)

    thresh_frame = cv2.threshold(delta_frame, 30, 255,
cv2.THRESH_BINARY)[1]

    thresh_frame = cv2.dilate(thresh_frame, None, iterations = 2)

    (cnts, _) = cv2.findContours(thresh_frame.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

```
    for contour in cnts:
        if cv2.contourArea(contour) < 1000:
            continue
        status=1

        (x,y,w,h) = cv2.boundingRect(contour)
        cv2.rectangle(frame, (x,y), (x+w, y+h), (0,0,255), 3)
    status_list.append(status)

    status_list = status_list[-2:]
    #For memory efficiency, we return the last 2 frames of
movement in our program, instead of the entire list.

    if status_list[-1]==1 and status_list[-2]==0:
        times.append(datetime.now())
    if status_list[-1]==0 and status_list[-2]==1:
        times.append(datetime.now())

    cv2.imshow("Gray Frame",gray)
    cv2.imshow("Delta Frame",delta_frame)
    cv2.imshow("Threshold frame", thresh_frame)
    cv2.imshow("Color Frame", frame)

    key=cv2.waitKey(1)

    if key == ord('q'):
        if status==1:
            times.append(datetime.now())
        break

print (status_list)
print (times)

for i in range(0,len(times),2):
    df=df.append({"Start":times[i], "End":times[i+1]},
ignore_index=True)

df.to_csv("Times.csv")

video.release()
cv2.destroyAllWindows
```

```
[1, 1]
[datetime.datetime(2019, 9, 5, 15, 36, 9, 126286),
datetime.datetime(2019, 9, 5, 15, 36, 9, 158668),
datetime.datetime(2019, 9, 5, 15, 36, 27, 972050),
datetime.datetime(2019, 9, 5, 15, 36, 29, 574587),
datetime.datetime(2019, 9, 5, 15, 36, 29, 940422),
datetime.datetime(2019, 9, 5, 15, 36, 30, 5499), datetime.datetime(2019,
9, 5, 15, 36, 32, 453223), datetime.datetime(2019, 9, 5, 15, 36, 32,
517052), datetime.datetime(2019, 9, 5, 15, 36, 32, 548006),
```

```
datetime.datetime(2019, 9, 5, 15, 36, 32, 772676),
datetime.datetime(2019, 9, 5, 15, 36, 32, 820972),
datetime.datetime(2019, 9, 5, 15, 36, 33, 13472),
datetime.datetime(2019, 9, 5, 15, 36, 33, 45387),
datetime.datetime(2019, 9, 5, 15, 36, 33, 654496),
datetime.datetime(2019, 9, 5, 15, 36, 38, 757678),
datetime.datetime(2019, 9, 5, 15, 36, 38, 788938),
datetime.datetime(2019, 9, 5, 15, 36, 41, 236289),
datetime.datetime(2019, 9, 5, 15, 36, 46, 983698),
datetime.datetime(2019, 9, 5, 15, 36, 47, 204596),
datetime.datetime(2019, 9, 5, 15, 36, 47, 686458),
datetime.datetime(2019, 9, 5, 15, 36, 47, 877411),
datetime.datetime(2019, 9, 5, 15, 36, 50, 213120),
datetime.datetime(2019, 9, 5, 15, 36, 50, 693776),
datetime.datetime(2019, 9, 5, 15, 37, 24, 422004),
datetime.datetime(2019, 9, 5, 15, 37, 24, 453783),
datetime.datetime(2019, 9, 5, 15, 37, 24, 505533),
datetime.datetime(2019, 9, 5, 15, 37, 24, 727033),
datetime.datetime(2019, 9, 5, 15, 37, 55, 797098)]

<function destroyAllWindows>
```

Please keep in mind that for some users the program might crash. To properly exit the program, click "q" on any of the webcam files. If that doesn't work, simply exit each tab and click close program. The Bokeh Time Series Graph should open up in a separate tab.

[6]

```
from bokeh.plotting import figure, show, output_file
from bokeh.models import HoverTool, ColumnDataSource

#Convert to datetime so that when user hover's over time series
plot, the output is correctly formatted!
df["Start_string"] = df["Start"].dt.strftime("%Y-%m-%d %H:%M:%S")
df["End_string"] = df["End"].dt.strftime("%Y-%m-%d %H:%M:%S")


cds=ColumnDataSource(df)

fig=figure(x_axis_type='datetime', height=120, width=500,
sizing_mode='scale_width', title="Time Series Graph")
fig.yaxis.minor_tick_line_color=None
fig.ygrid[0].ticker.desired_num_ticks=1
```

```
        hov=HoverTool(tooltips=[("Start: ", "@Start_string"), ("End:
        ","@End_string")])
        fig.add_tools(hov)

        fig2=fig.quad(left="Start", right="End", bottom=0, top=1,
        color="green", source=cds)

        output_file("TimeSeriesPlot.html")
        show(fig)
```