MULTICORE VIDEO PROCSSING

Project for research

Contents

ABSTRACT

To perform video processing using parallel processing methods. A code will be developed which will assign different tasks to each core of the main processor. A comparison between two parallel processing methods will be done to determine which method is better for the following categories: small, medium, high resolution and of small, medium large length

MOTIVATION

- With the growth of IoT devices, the implementation of microcontrollers and peripheral sensors and networking modular devices is increasing.
- One major segment of such projects and products is achieved via processing live streaming videos through cameras present as data receiving modules.
- Such microcontrollers have computing and power constraints; this proves the difficulty of processing images on such constrained devices.
- The proposed methodologies and techniques aim at resolving this problem by implementing parallel processing architecture on a small scale for video processing.

Method 1: SFST

- SFST: Single Frame Single Thread Method
- Frames of the video are received in a loop.
- **Each frame will be processed by one thread**.
    - All calculations for processing the frame will be done by one thread only.
    - Frames will be sent to threads which are idle
    - If no threads are idle then retrieval of frames will be paused till one thread is idle.

Method 2: SFMT

- SFMT: Single Frame Multi Thread
- Frames of the video are received in a loop.
- **Each frame will be divided into equal sections.**
    - Number of sections in which the frame will be divided will be equal to the number of threads in the main processor.
    - Each thread will execute the calculation assigned.
    - If other threads haven't finished calculation then thread will remain idle till all the other threads are done with the frame.

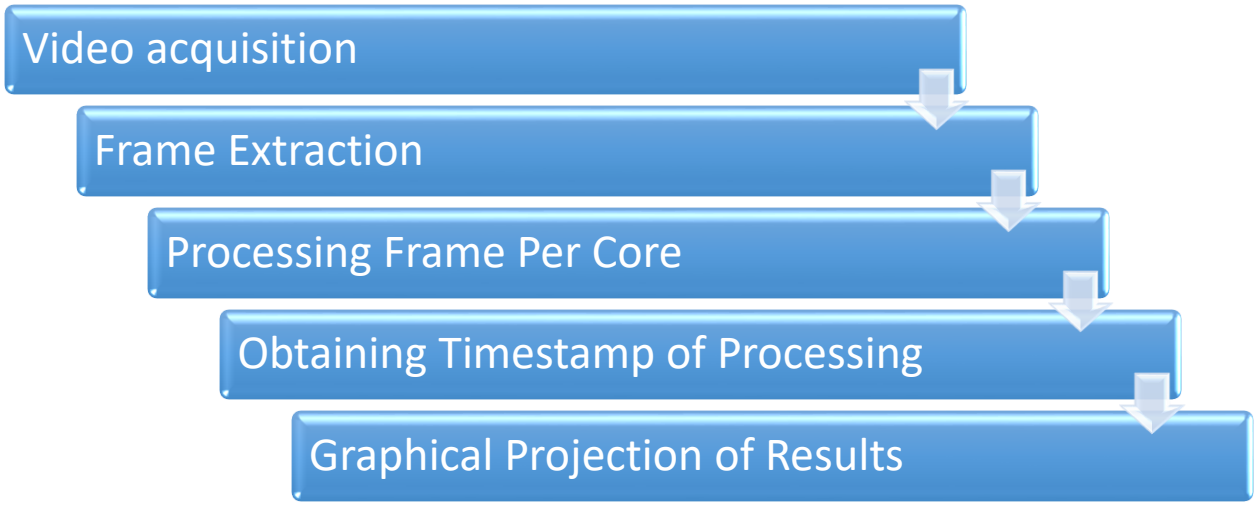
Objective

- **To check the behavior of both the processing methods.**
    - Both methods will process the same set of videos.
    - The videos will be of LOW,MED,HIGH resolution
    - Also of Small, medium and long duration.

Software & Hardware Requirements

- HARDWARE Requirements:
    - Multicore processing computer system.
    - Cameras
- SOFTWARE Requirements
    - Language : **Python**
    - Video Processing : **OpenCV**
    - Parallel Processing : **OpenMP, DISPY**
    - RStudio for plotting graphs

System Architecture: SFST



System Architecture: SFMT

Expected Project Outcome

1. Respective Timestamp data of SFST and SFMT methods.
2. Graphical projection of the acquired data for comparative analysis.

CODE

1) Processing Webcam Feed

```
import cv2
import argparse
import imutils


#function to detect shapes and return text
def detect(c):
	shape = "unidentified"
		peri = cv2.arcLength(c, True)
		approx = cv2.approxPolyDP(c, 0.04 * peri, True)
		if len(approx) == 3:shape = "triangle"
		elif len(approx) == 4:
			(x, y, w, h) = cv2.boundingRect(approx)
			ar = w / float(h)
			shape = "square" if ar >= 0.95 and ar <= 1.05 else "rectangle"
		elif len(approx) == 5:shape = "pentagon"
		else:shape = "circle"
		return shape


#Single Core Processing code
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,help="path to the input image")
args = vars(ap.parse_args())
image = cv2.imread(args["image"])
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
thresh = cv2.threshold(gray, 60, 255, cv2.THRESH_BINARY)[1]
```

```
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if imutils.is_cv2() else cnts[1]
for c in cnts:
        M = cv2.moments(c)
        cX = int((M["m10"] / M["m00"]))
        cY = int((M["m01"] / M["m00"]))
        shape = detect(c)
        c = c.astype("float")
        c = c.astype("int")
        cv2.drawContours(image, [c], -1, (0, 255, 0), 2)
        cv2.putText(image, shape, (cX, cY), cv2.FONT_HERSHEY_SIMPLEX,0.5, (255, 255, 255), 2)
        cv2.imshow("Image", image)
        cv2.waitKey(0)
```

2) Single Core Processing and Writing to File

```
import cv2
import imutils
import timeit
f=open('single_thread.txt','w')
l=[]
ctr=0
def detect(c):
    shape="unidentified"
    peri=cv2.arcLength(c,True)
    approx=cv2.approxPolyDP(c,0.04 * peri,True)
    if len(approx)==3:shape="triangle"
    elif len(approx)==4:
        (x,y,w,h)=cv2.boundingRect(approx)
        ar=w/float(h)
        shape="square" if ar>=0.95 and ar<=1.05 else "rectangle"
    elif len(approx)==5:shape="pentagon"
    else:shape="circle"
    return shape
cam=cv2.VideoCapture(0)
def process(image):
    gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    thresh=cv2.threshold(gray,60,255,cv2.THRESH_BINARY)[1]
    cnts=cv2.findContours(thresh.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    cnts=cnts[0] if imutils.is_cv2() else cnts[1]
    for c in cnts:
        M=cv2.moments(c)
```

```
            cX,cY=0,0
            try:
                cX=int((M["m10"]/M["m00"]))
                cY=int((M["m01"]/M["m00"]))
            except:pass
            shape=detect(c)
            c=c.astype("float")
            c=c.astype("int")
            cv2.drawContours(image,[c],-1,(0,255,0),2)
            cv2.putText(image,shape,(cX,cY),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),2)
            cv2.imshow("Image",image)
while timeit.default_timer()<=10:
    ret_val,img=cam.read()
    start=timeit.default_timer()
    process(img)
    stop=timeit.default_timer()
    if ctr==3:l,ctr=[],0
    l.append(stop-start)
    f.writelines(str(sum(l)/4)+'\n')
    ctr=ctr+1
    if cv2.waitKey(1)==27: break
cv2.destroyAllWindows()
f.flush()
f.close()
```

3) SFMT

```
import cv2
from imutils import is_cv2
import multiprocessing
from timeit import default_timer
f=open('SFMT.txt','w')
l=[]
ctr=0
def detect(c):
    shape="unidentified"
    peri=cv2.arcLength(c,True)
    approx=cv2.approxPolyDP(c,0.04 * peri,True)
    if len(approx)==3:shape="triangle"
    elif len(approx)==4:
        (x,y,w,h)=cv2.boundingRect(approx)
        ar=w/float(h)
        shape="square" if ar>=0.95 and ar<=1.05 else "rectangle"
    elif len(approx)==5:shape="pentagon"
    else:shape="circle"
    return shape
cam=cv2.VideoCapture(0)
def process(image,stri):
    gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    thresh=cv2.threshold(gray,60,255,cv2.THRESH_BINARY)[1]
    cnts=cv2.findContours(thresh.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    cnts=cnts[0] if is_cv2() else cnts[1]
    for c in cnts:
```

```
            M=cv2.moments(c)
            cX,cY=0,0
            try:
                cX=int((M["m10"]/M["m00"]))
                cY=int((M["m01"]/M["m00"]))
            except:pass
            shape=detect(c)
            c=c.astype("float")
            c=c.astype("int")
            cv2.drawContours(image,[c],-1,(0,255,0),2)
            cv2.putText(image,shape,(cX,cY),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),2)
            cv2.imshow(stri,image)
if __name__=='__main__':
    while default_timer()<=10:
        ret_val,img=cam.read()
        start = default_timer()
        h,w,d=img.shape
        p1=multiprocessing.Process(target=process,args=(img[:int(h/2),:int(w/2)],"top-left",))
        p2=multiprocessing.Process(target=process,args=(img[int(h/2):,:int(w/2)],"bottom-left",))
        p3=multiprocessing.Process(target=process,args=(img[:int(h/2),int(w/2):],"top-right",))
        p4=multiprocessing.Process(target=process,args=(img[int(h/2):,int(w/2):],"bottom-right",))
        p1.run()
        p2.run()
        p3.run()
        p4.run()
        stop = default_timer()
        if ctr==3:l,ctr=[],0
        l.append(stop-start)
```

```
    f.writelines(str(sum(l)/4)+'\n')
    ctr=ctr+1
    if cv2.waitKey(1)==27:break
cv2.destroyAllWindows()
f.flush()
f.close()
```

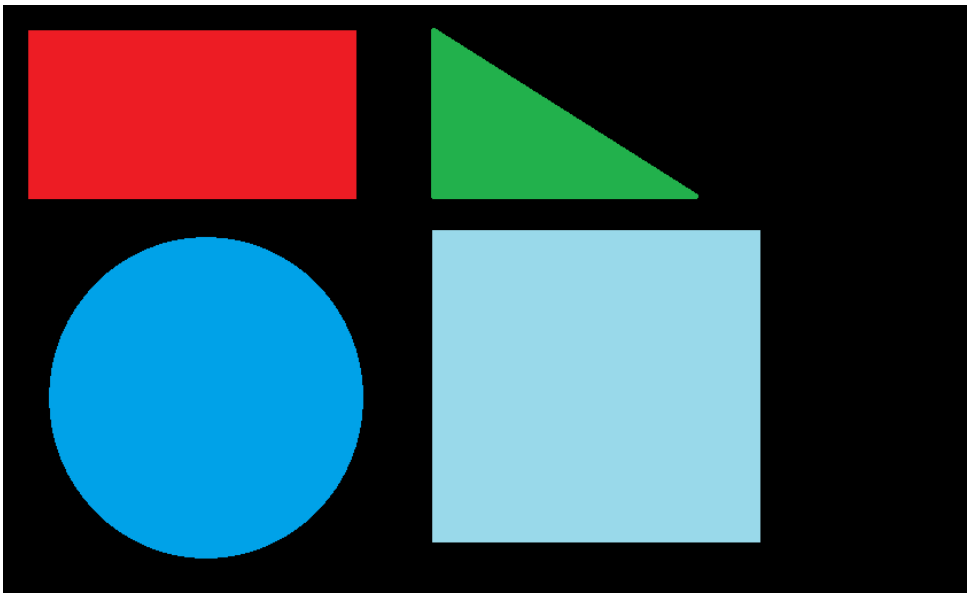
4) SFST

```
import cv2
from imutils import is_cv2
import multiprocessing
from timeit import default_timer
f=open('STSF.txt','w')
l=[]
ctr=0
def detect(c):
    shape="unidentified"
    peri=cv2.arcLength(c,True)
    approx=cv2.approxPolyDP(c,0.04 * peri,True)
    if len(approx)==3:shape="triangle"
    elif len(approx)==4:
        (x,y,w,h)=cv2.boundingRect(approx)
        ar=w/float(h)
        shape="square" if ar>=0.95 and ar<=1.05 else "rectangle"
    elif len(approx)==5:shape="pentagon"
    else:shape="circle"
    return shape
cam=cv2.VideoCapture(0)
def process(image,stri):
    gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    thresh=cv2.threshold(gray,60,255,cv2.THRESH_BINARY)[1]
    cnts=cv2.findContours(thresh.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    cnts=cnts[0] if is_cv2() else cnts[1]
    for c in cnts:
```

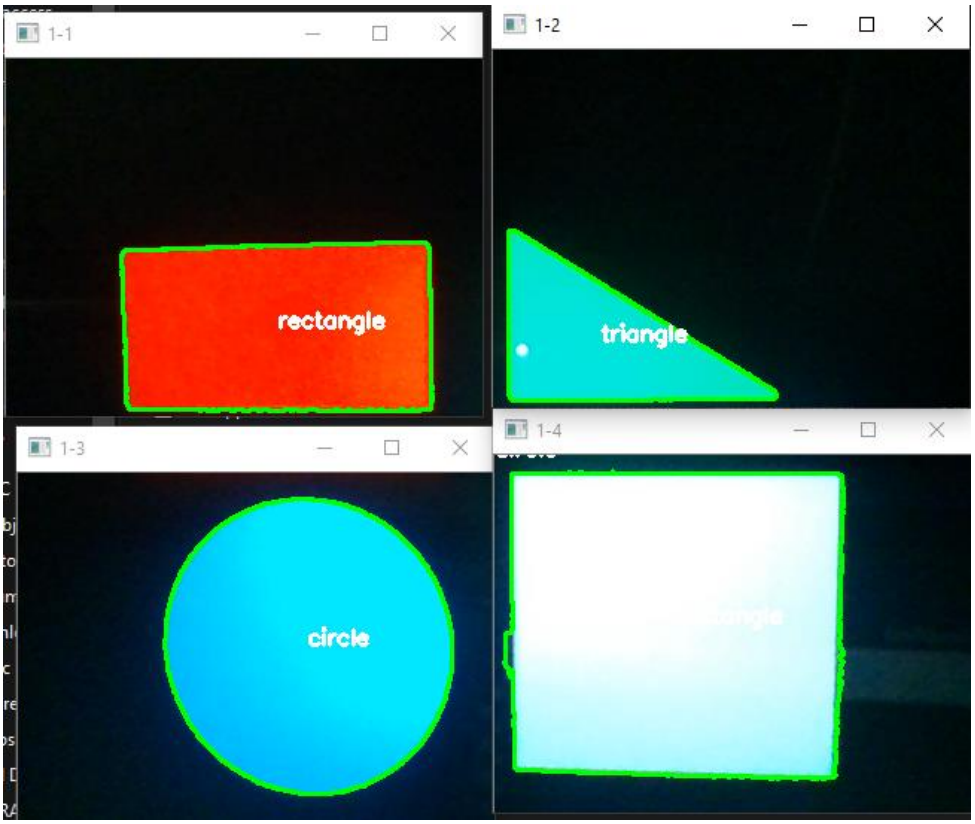
```
            M=cv2.moments(c)
            cX,cY=0,0
            try:
                cX=int((M["m10"]/M["m00"]))
                cY=int((M["m01"]/M["m00"]))
            except:pass
            shape=detect(c)
            c=c.astype("float")
            c=c.astype("int")
            cv2.drawContours(image,[c],-1,(0,255,0),2)
            cv2.putText(image,shape,(cX,cY),cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),2)
            cv2.imshow(stri,image)
if __name__=='__main__':
    ctr=1
    while default_timer()<=10:
        ret_val,img=cam.read()
        start = default_timer()
        h,w,d=img.shape
        if ctr==0:
            p1=multiprocessing.Process(target=process,args=(img,"one",))
            p1.run()
        if ctr==1:
            p2=multiprocessing.Process(target=process,args=(img,"one",))
            p2.run()
        if ctr==2:
            p3=multiprocessing.Process(target=process,args=(img,"one",))
            p3.run()
        if ctr==3:
```

```
            p4=multiprocessing.Process(target=process,args=(img,"one",))
            p4.run()
    stop = default_timer()
    if cv2.waitKey(1)==27:break
    if ctr==3:l,ctr=[],0
    l.append(stop-start)
    f.writelines(str(sum(l)/4)+'\n')
    ctr=ctr+1
cv2.destroyAllWindows()
f.flush()
f.close()
```

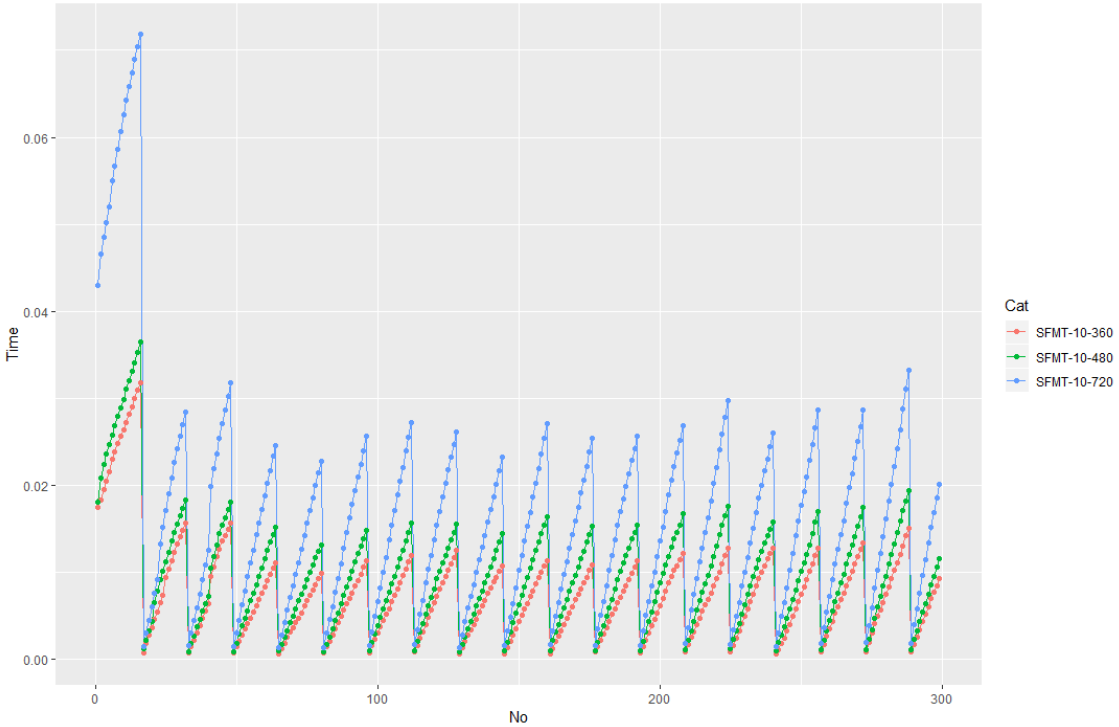
RESULTS

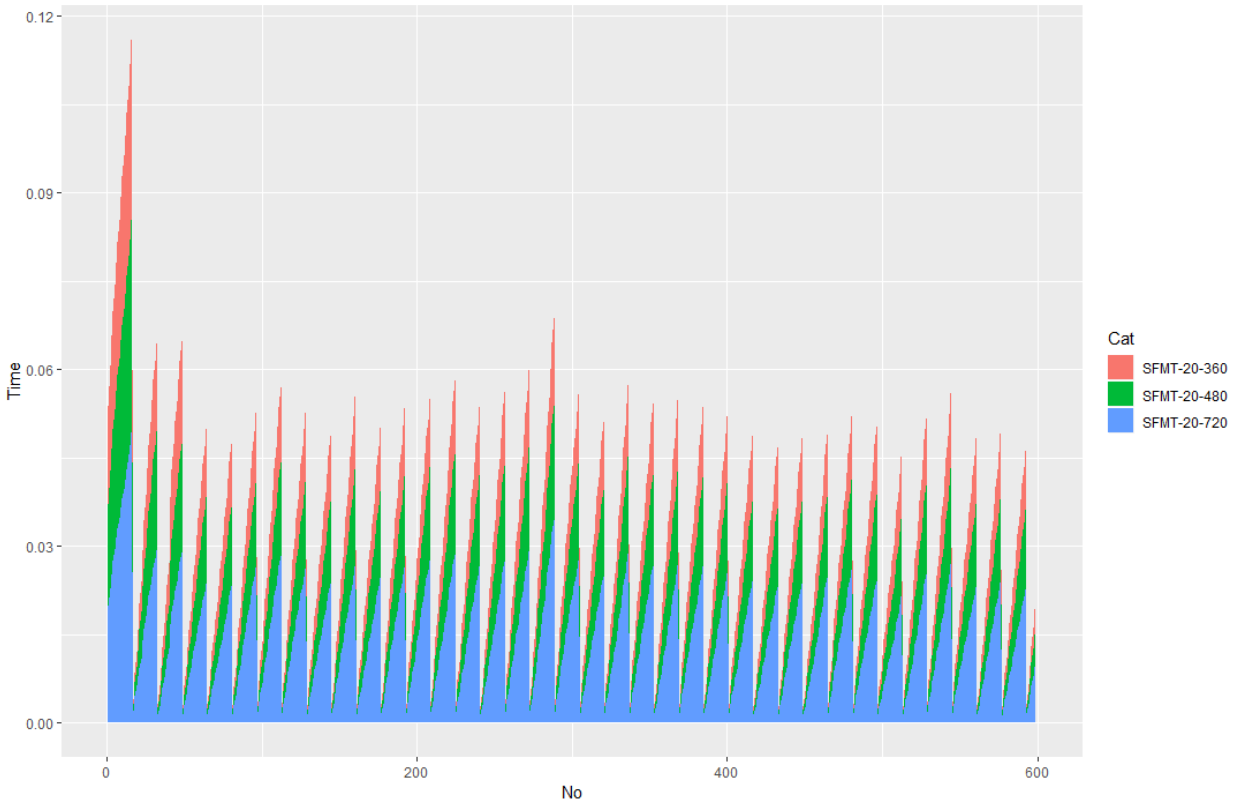


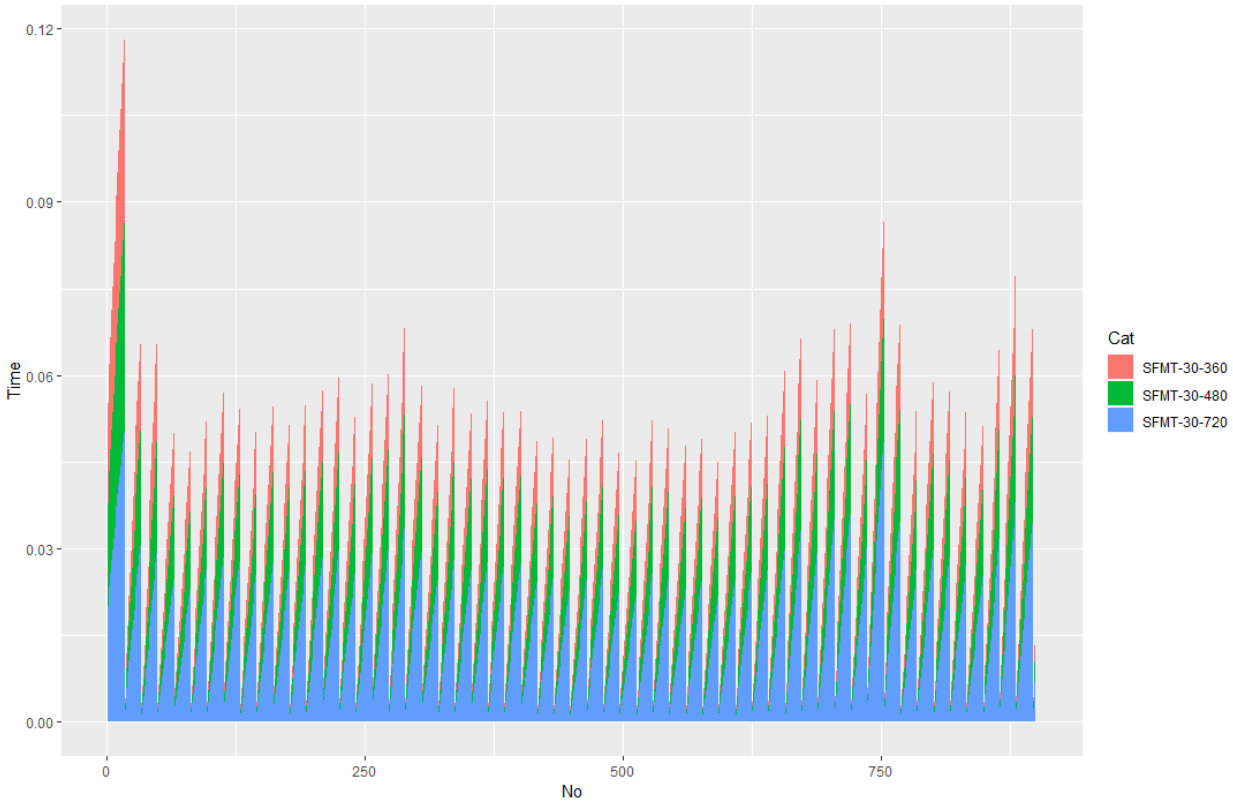
Shape Detection Algorithm performed in each technique as well as division of frame obtained



Graphs obtained using RStudio of Timestamp data (Key: Method-Video Length-Resolution)




0.12
0.09
0.06
0.03
0.00
Time
0
200
400
600
No
Cat
SFMT-20-360
SFMT-20-480
SFMT-20-720


0.12
0.09
0.06
0.03
0.00
Time
0
250
500
750
No
Cat
SFMT-30-360
SFMT-30-480
SFMT-30-720

Conclusion

Achieved:

- Respective Timestamp data of SFST and SFMT methods.
- Graphical projection of the acquired data for comparative analysis

It is observed that SFMT method of image processing is time efficient wherein each frame is divided into sections to utilize thread processing for each frame over SFST wherein each frame is processed by each thread.