HÁSKÓLINN Í REYKJAVÍK

COMPUTER VISION

T-869-COMP

Assignment 1

Sindri Þór Harðarson

sindrih18@ru.is

Instructor

Torfi Þórhallsson

November 27, 2023

https://github.com/SindriTh/ComputerVision

1 Questions

1.1 The processing time for one video frame or image.

The measured processing time varied quite a bit. From 45ms down to 15ms.
This can likely be explained by the system performing other tasks in the background slowing down the rate at which it processes frames.

1.2 How does the processing time change when you add the bright spot detection?

I did not see any difference in performance using the bright spot detection. Frame times varied so much already, that there was no measurable difference in the frame times

1.3 Is the processing time identical when you do not display the image?

```
Frame time: 0.0310
Frame time with image: 0.0320
Frame time: 0.0190
Frame time with image: 0.0210
Frame time: 0.0410
Frame time with image: 0.0420
Frame time: 0.0230
Frame time with image: 0.0240
Frame time: 0.0310
Frame time with image: 0.0320
Frame time: 0.0290
Frame time with image: 0.0300
Frame time: 0.0310
Frame time with image: 0.0320
Frame time: 0.0350
Frame time with image: 0.0400
Frame time: 0.0170
Frame time with image: 0.0210
Frame time: 0.0230
Frame time with image: 0.0240
Frame time: 0.0230
Frame time with image: 0.0240
Frame time: 0.0180
Frame time with image: 0.0190
```

Adding the image does not affect the frame time much, overall it adds around 1ms to the frame time.

1.4 How does your for-loop implementation compare to the built-in function?

The for loop takes WAYYYY longer. The frame time varied between 2.7 to 5.2 seconds. This implementation is also flawed. It has huge issues finding the reddest and brightest pixel. This would NOT be usable. This is likely worsened by the fact that I set the camera resolution to 1280x720. However, this was the case for both tests.
UPDATE: After talking with others, Unnar pointed out that my code was wrong, and mentioned that I need to divide the frame brightness by 3 to start with.
This makes sense, as otherwise, we could get overflowing brightness values. After this change, I also somehow recorded much better frame times or around 1.1sec.

```
Frame time: 1.1698
Frame time with image: 1.1707
Frame time: 1.1803
Frame time with image: 1.1813
Frame time: 1.2811
Frame time with image: 1.2813
```

1.5 Moving your hand in front of the camera, estimate the latency between image capture and display.

I estimate that the latency is around 100ms.
This varies greatly on my machine though, sometimes looking perfect with no perceivable latency, and sometimes struggling so much that the latency goes up to 400ms.

1.6 Is the latency different when capturing from a mobile phone?

Using the e2eSoft iVcam software, I managed to stream the video from my phone using their app over WiFi. The latency was higher with the phone. I estimate around 250ms.

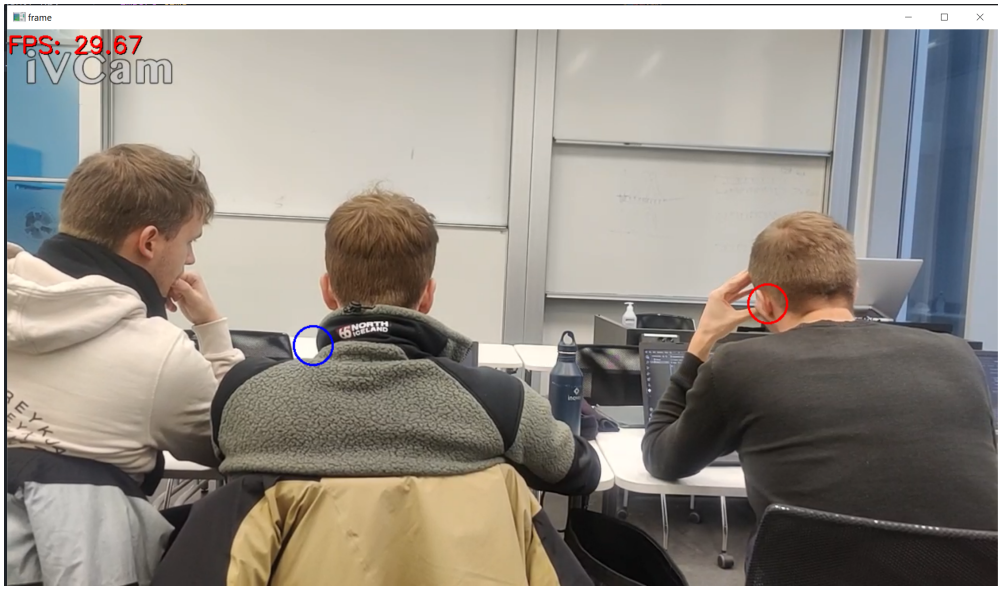

Figure 1: Video from phone

2 Code

2.1 OpenCV

```
# Sindri Þór Harðarson
# OpenCV camera capture
import numpy as np
import cv2
import time

cap = cv2.VideoCapture(1)

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

# FPS counter
start_time = time.time()
frame_counter = 0
fps = 0

while True:
    # Measure Frame time
    start_frame_time = time.time()

    # Capture frame-by-frame
    ret, frame = cap.read()

    # get grayscale image
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # blur image to reduce noise a bit
    blur = cv2.blur(gray, (5,5), 0)

    #Get brightest pixel
    (minVal, maxVal, minLoc, maxLoc) = cv2.minMaxLoc(blur)

    #Draw circle around brightest pixel
    cv2.circle(frame, maxLoc, 25, (255,0,0), 2)



    # Find reddest pixel
    # This is done by subtracting the blue and green channels from the red channel
    # The reddest pixel will have the highest value after subtraction

    # Split into channels
    (B,G,R) = cv2.split(frame)
    # Calculate difference between channels
    RG = cv2.subtract(R,G)
    RB = cv2.subtract(R,B)
```

```
    # Add differences together
    added = cv2.add(RG,RB)
    # Blur image
    blur = cv2.blur(added, (25,25), 0)
    # Get reddest pixel
    (minVal, maxVal, minLoc, maxLoc) = cv2.minMaxLoc(blur)

    # Draw circle around reddest pixel
    cv2.circle(frame, maxLoc, 25, (0,0,255), 2)

    # Calculate FPS
    frame_counter += 1
    if frame_counter >= 10:
        fps = frame_counter / (time.time() - start_time)
        start_time = time.time()
        frame_counter = 0

    cv2.putText(frame, f"FPS: {fps:.2f}", (1, 31), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0)
    cv2.putText(frame, f"FPS: {fps:.2f}", (0, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 25

    # Measure frame time
    end_frame_time = time.time()
    print(f"Frame time: {time.time() - start_frame_time:.4f}")
    # Display the resulting frame
    cv2.imshow('frame', frame)

    #time taken to print
    end_frame_time = time.time()
    print(f"Frame time with image: {end_frame_time - start_frame_time:.4f}")

    # Press q to quit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

2.2 Double For Loop

```
# Sindri Þór Harðarson
# OpenCV camera capture
# Double For Loop
import numpy as np
import cv2
import time

cap = cv2.VideoCapture(0)

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

# FPS counter
start_time = time.time()
frame_counter = 0
fps = 0

while True:
    start_frame_time = time.time()
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Get image dimensions
    height, width, channels = frame.shape
    # Set initial values for brightest pixel
    maxVal = 0
    maxLoc = (0,0)
    # Set inital values for reddest pixel
    maxValRed = 0
    maxLocRed = (0,0)
    # Loop through all pixels
    newFrame = frame/3
    for x in range(width):
        for y in range(height):
            # Get pixel value
            pixel = newFrame[y,x]
            # Calculate brightness
            brightness = pixel[0] + pixel[1] + pixel[2]
            # Check if brightest pixel
            if brightness > maxVal:
                maxVal = brightness
                maxLoc = (x,y)
            # Calculate redness
            redness = (pixel[2] - pixel[0]) + (pixel[2] - pixel[1])
            # Check if reddest pixel
            if redness > maxValRed:
```

```
                maxValRed = redness
                maxLocRed = (x,y)
    # Draw circle around reddest pixel
    cv2.circle(frame, maxLocRed, 25, (0,0,255), 2)

    # Draw circle around brightest pixel
    cv2.circle(frame, maxLoc, 25, (255,0,0), 2)

    # Calculate FPS
    frame_counter += 1
    if frame_counter >= 10:
        fps = frame_counter / (time.time() - start_time)
        start_time = time.time()
        frame_counter = 0

    cv2.putText(frame, f"FPS: {fps:.2f}", (1, 31), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0)
    cv2.putText(frame, f"FPS: {fps:.2f}", (0, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 25

    # Measure frame time
    end_frame_time = time.time()
    print(f"Frame time: {time.time() - start_frame_time:.4f}")
    # Display the resulting frame
    cv2.imshow('frame', frame)

    #time taken to print
    end_frame_time = time.time()
    print(f"Frame time with image: {end_frame_time - start_frame_time:.4f}")

    # Press q to quit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```