MoonDream AI

Table of Contents

What is MoonDream AI....2
Setup....3
    Python Setup....3
    MoonDream AI Models....3
MoonDream Image Captions....4
MoonDream Query....6
MoonDream Pointer HTML Display....7
Object Detection with Bounding Box....9
Auto Caption Gallery with Bottle and SQLite....11
Intruder Detection....14

What is MoonDream AI

MoonDream AI is an Open Source light weight Vision Model that allows you to easily add computer vision to your Python projects.

The value of MoonDream over OpenCV is that you can simply query about an object instead of needing to specifically train to detect it.

A significant value of MoonDream is that ability to give X/Y Coordinates and Sizes of the objects it discovers like we've demonstrated in prior OpenCV classes. This allows for actions to occur not just based on the detection of objects, but also based on the location of the objects.

There is a 2B model and a smaller .5B model. The .5B model "works" but is very limited. For testing I recommend using the 2B int 8 model. There are 8 int and 4 int versions of the models. This relates to the compression of the model. 8 int is the best.

MoonDream released the ability to output Structured Text in JSON, XML and CSV format. During testing I found the results to be "quirky" at best.

They also released "Gaze Detection" functionality that determines where people are looking. I have not tested this.

The local models run on CPU not GPU. I have run the models on a 2012 MacBook Pro with 4GB of RAM and they are slow but the results are consistent. I created these classes and did testing using a 2023 M2 MacBook Pro with 32GB of RAM and the performance is pretty slow. MoonDream may be useful for your projects using low resource systems, but realize the speed may be slower than you expect.

I might consider using MoonDream for Prototyping and Testing, but then train a dedicated model for my specific use case.

Setup

Python Setup

```
python3 -m pip install moondream
```

```
python3 -m pip install pillow
```

```
python3 -m pip install bottle
```

```
python3 -m pip install opencv-python
```

Note:

If you are going to use their API you may need to use a venv. For me I had a SSL security failure when trying to run the script from the standard enviornment.

MoonDream AI Models

I recommend using the 2B int 8 model for initial testing on a “modern” computer, and then use the smaller models once you know what to expect.

Download the model to the working directory of your project.

GitHub Page:

https://github.com/vikhyat/moondream?tab=readme-ov-file#-moondream

.5B int 8

https://huggingface.co/vikhyatk/moondream2/resolve/9dddae84d54db4ac56fe37817aeaeb502ed083e2/moondream-0_5b-int8.mf.gz?download=true

2B int 8

https://huggingface.co/vikhyatk/moondream2/resolve/9dddae84d54db4ac56fe37817aeaeb502ed083e2/moondream-2b-int8.mf.gz?download=true

MoonDream Image Captions

In the center of the image, a woman is engaged in a shooting activity. She is wearing a white tank top and light blue shorts, and is wearing red earmuffs. She is holding a black handgun in her right hand, aiming it towards the left side of the frame. The gun is mounted on a black stand. The woman is standing on a grassy area, with a small, light-colored table in the foreground. The background features a dirt or gravel area with some rocks and trees. The woman is also wearing a dark-colored wristband on her right wrist.

In this lab we use the caption() function to auto create a caption for an image, and then create an HTML file with the image embedded and the caption written below

moondream-caption.py

```
import moondream as md
from PIL import Image

model = md.vl(model='./moondream-2b-int4.mf.gz')
# model = md.vl(model='./moondream-0_5b-int8.mf.gz')

picture = '../image/img2.jpeg'
image = Image.open(picture)
encoded_image = model.encode_image(image)

# Caption any image (length options: "short" or "normal" (default))
caption = model.caption(encoded_image, "short")
print("Caption:", caption['caption'])

with open('moondream-caption.html', 'w') as file:
    file.write(f'''
               <div style="position: relative;
               height: 400px;
               width: auto;
               display: inline-block;">

               <img src="{picture}"
               style="height: 100%;
               display: block;">

                <div>{caption['caption']}</div>

                </div>''')
```

MoonDream Query

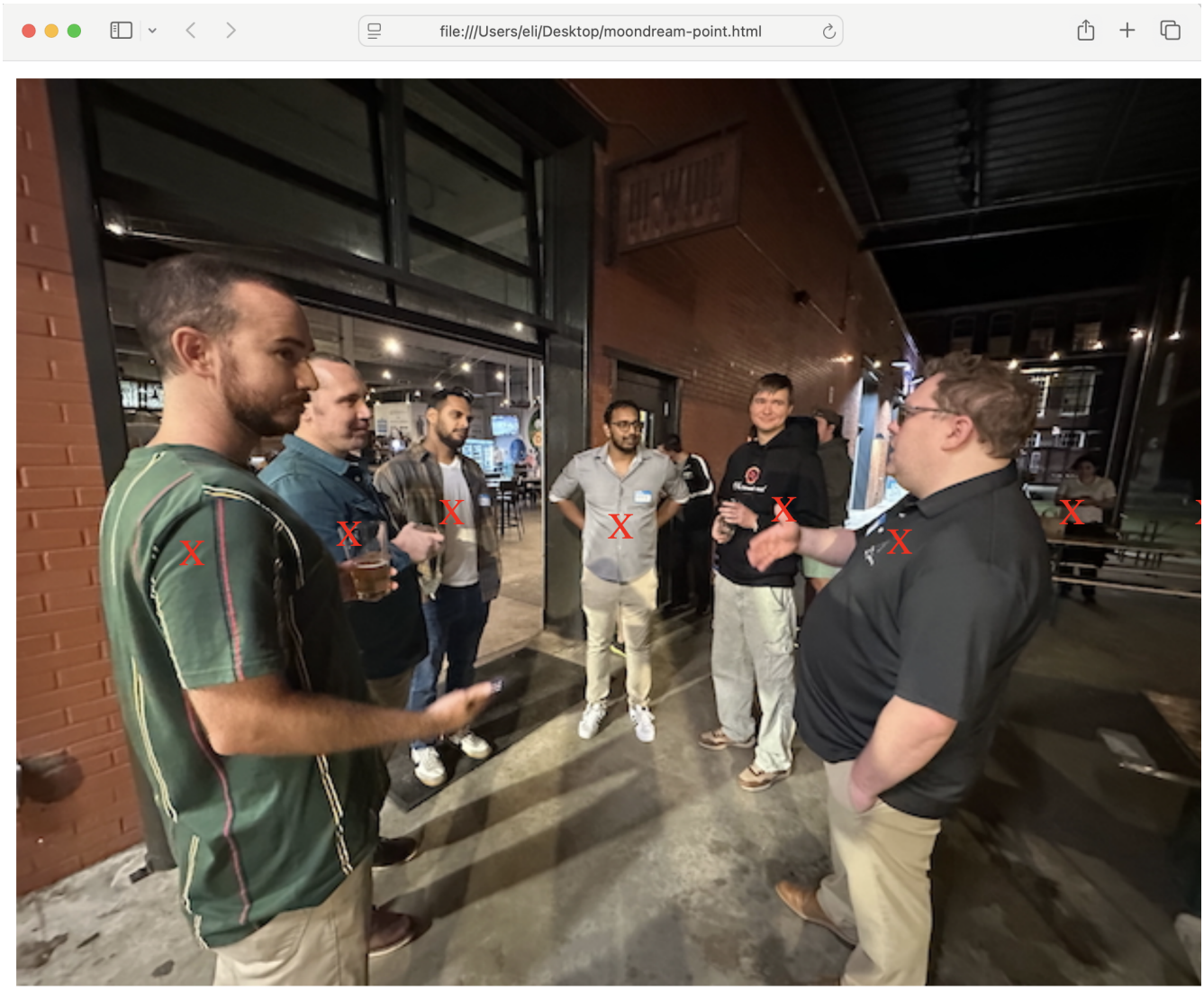
```
is this at the beach - answer only YES or NO

Answer:  YES
```

In this lab we use the query() function to ask MoonDream about an image. It can either describe an image, or return a value if your question is true.

moondream-query.py

```
import moondream as md
from PIL import Image

model = md.vl(model='./moondream-2b-int4.mf.gz')
# model = md.vl(model='./moondream-0_5b-int8.mf.gz')

picture = './image/img.jpeg'
image = Image.open(picture)
encoded_image = model.encode_image(image)

query = 'is this at the beach - answer only YES or NO'
answer = model.query(encoded_image, query)["answer"]
print(query)
print("\nAnswer:", answer)
```

MoonDream Pointer HTML Display



This lab allows you to query MoonDream to detect objects within an image, and then to create an HTML page with the image and an X at the X/Y coordinates of each object detected.

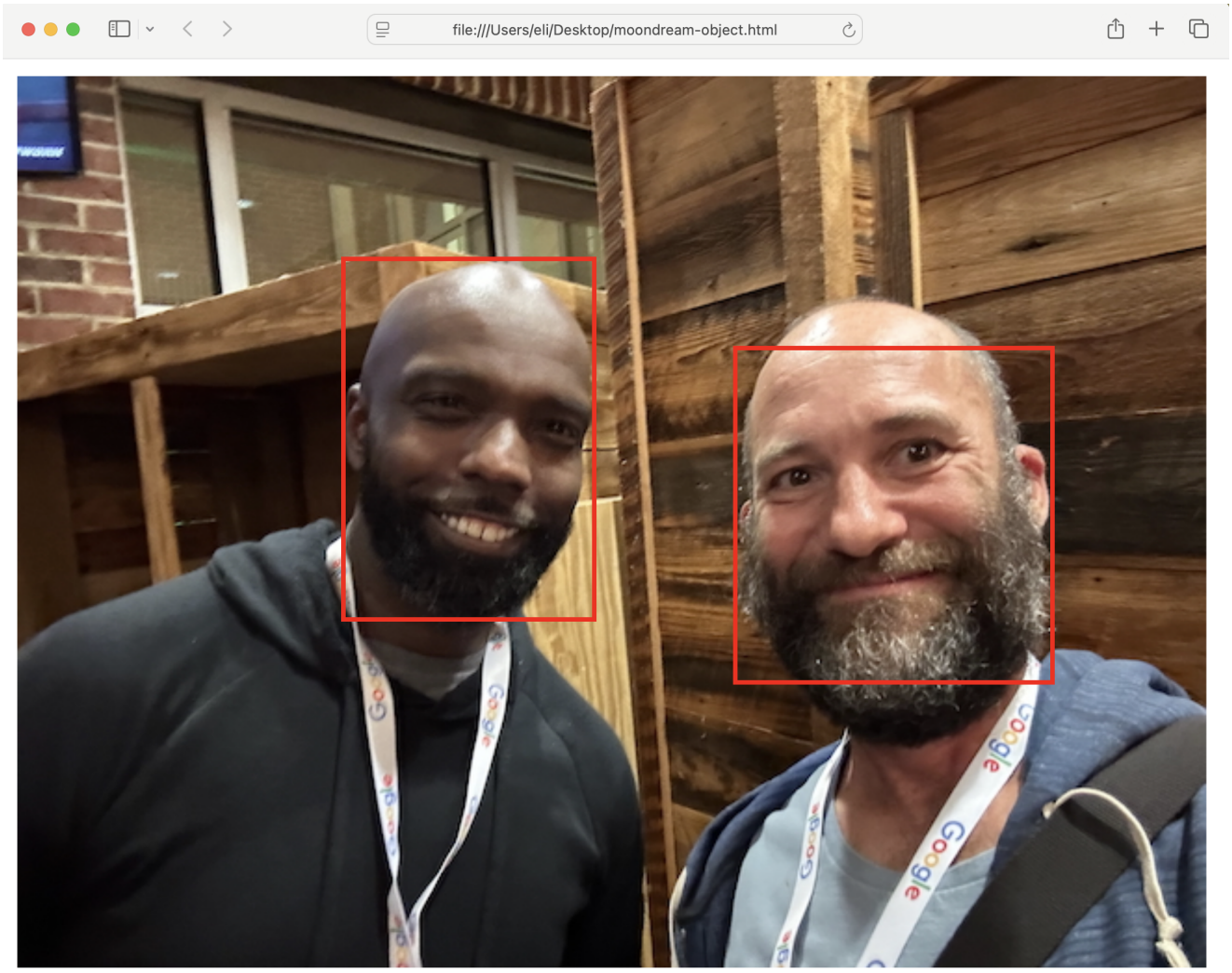
Point_result[“points”] returns a list of the coordinates of the objects detected. These numbers are from top and left and represent from 0-1. We multiply the results by 100 to get a percent value and can then use the value for the top and left css attributes.

You need to use the 2B model for this project.

moondream-pointer.py

```
import moondream as md
from PIL import Image

model = md.vl(model='./moondream-2b-int4.mf.gz')

picture = './image/img3.jpeg'
image = Image.open(picture)
encoded_image = model.encode_image(image)

query = 'people'
point_result = model.point(encoded_image, query)
print("Points:", point_result["points"])

with open('moondream-point.html', 'w') as file:
    file.write('''
               <div style="position: relative;
               height: 400px;
               width: auto;
               display: inline-block;">
               ''')
    file.write(f'''
               <img src="{picture}"
               style="height: 100%;
               display: block;">
               ''')
    for point in point_result["points"]:
        x = point['x'] * 100
        y = point['y'] * 100
        print(f"{x} -- {y}")
        file.write(f'''
                   <div style="color:red;
                   position: absolute;
                   top:{y}%;
                   left:{x}%;"
                   >X</div>
                   ''')
    file.write('</div>')
```

Object Detection with Bounding Box



This lab uses the detect() function to find the X/Y coordinates of objects with their height and width. This allows you to create bounding boxes when you out put the image information.

You can use 2B or .5B model for this function, but the .5B model is pretty poor.

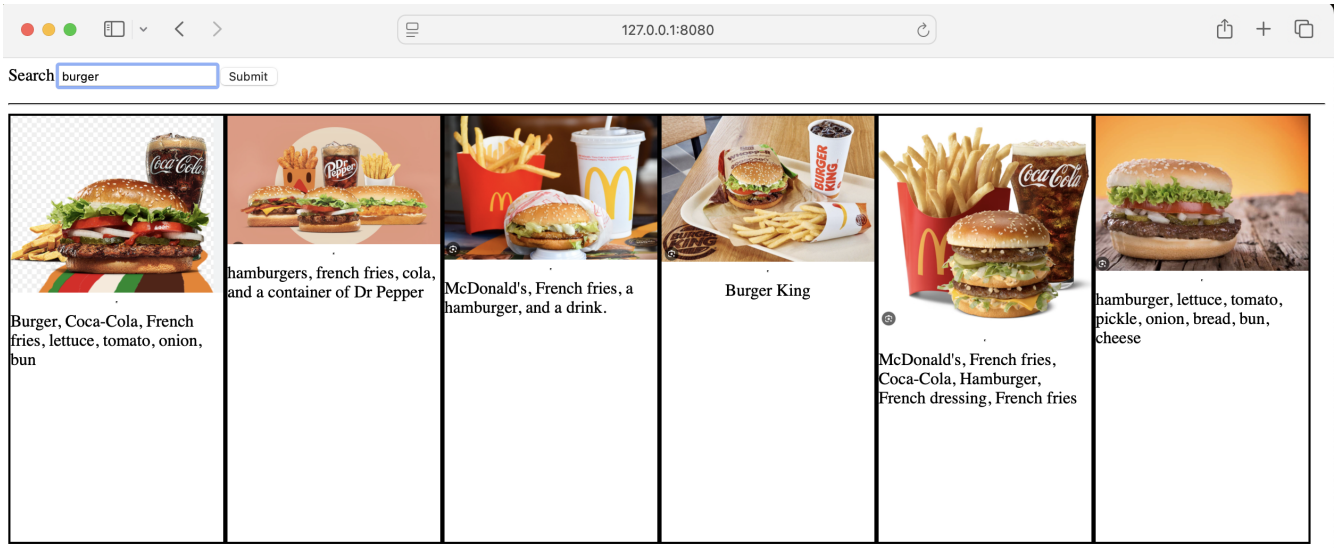
The X/Y values are between 0-1 and are returned as x_min and y_min. To get the position you multiply by 100 and use this as a percentage of the div box.

For Height and Width you subtract x_min from x_max and multiply by 100 for the width percentage. Repeat for y values.

moondream-object.py

```
import moondream as md
from PIL import Image

model = md.vl(model='./moondream-2b-int4.mf.gz')  # Initialize model
# model = md.vl(model='./moondream-0_5b-int8.mf.gz')  # Initialize model

picture = './image/img2.jpeg'
image = Image.open(picture)
encoded_image = model.encode_image(image)

query = "face"
detect_result = model.detect(encoded_image, query)
print("\nDetected:", detect_result["objects"])

with open('moondream-object.html', 'w') as file:
    file.write('''
               <div style="position: relative;
               height: 400px;
               width: auto;
               display: inline-block;">
               ''')
    file.write(f'''
               <img src="{picture}"
               style="height: 100%;
               display: block;">
               ''')
    for object in detect_result['objects']:
        x = object['x_min'] * 100
        y = object['y_min'] * 100
        width = (object['x_max'] - object['x_min']) * 100
        height = (object['y_max'] - object['y_min']) * 100

        print(f"{x} -- {width} / {y} -- {height}")
        file.write(f'''
                   <div style="color:red;
                   position: absolute;
                   border: 2px solid red;
                   top:{y}%;
                   left:{x}%;
                   width:{width}%;
                   height:{height}%;"
                   ></div>
                   ''')
    file.write('</div>')
```

Auto Caption Gallery with Bottle and SQLite



This project uses MoonDream to create a description of an image based on your custom query and then save the image name and description to a database.

With this we can then create a web app with Bottle to allow us to search for image based off of keywords.

The auto-caption script will scan a directory and put all of the files into a list. We then iterate through the list asking MoonDream a query. The image name and the query result is then inserted into the SQLite database.

moondream-autocaption.py

```
import moondream as md
from PIL import Image
import os
import sqlite3

pic_directory = './image_food'

model = md.vl(model='./moondream-2b-int8.mf.gz')

image_list = os.listdir(pic_directory)
print(image_list)

query = 'what is in this picture. use as many proper nouns as possible. be very
descriptive. state any brands in the picture'

def insert(picture, answer):
    conn = sqlite3.connect('moondream-autocaption.db')
    cursor = conn.cursor()
    cursor.execute('CREATE TABLE IF NOT EXISTS image(image text,description text)')
    cursor.execute('INSERT INTO image(image,description) VALUES(?,?)',(picture,
answer))
    conn.commit()
    conn.close()

for image in image_list:
    picture = os.path.join(pic_directory, image)
    image = Image.open(picture)
    encoded_image = model.encode_image(image)

    answer = model.query(encoded_image, query)['answer']
    print(picture)
    print('\nDescription:', answer)
    insert(picture, answer)
```

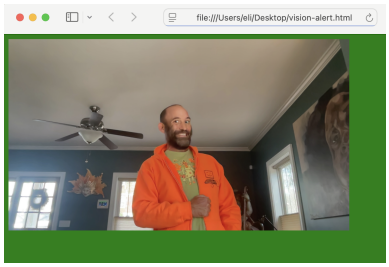
The Gallery script uses Bottle to create a web app. We use an HTML form to take a search query from the user and then select the records that match that query from the SQLite database. Then we use the results to dynamically create an image gallery.

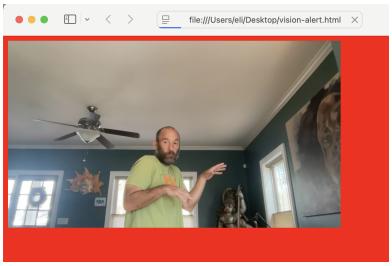
gallery-search.py

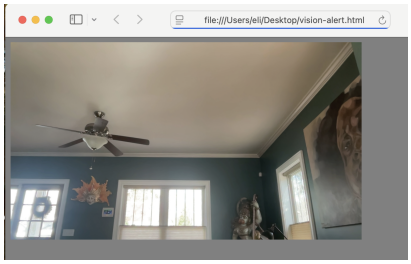
```
from bottle import run, route, static_file, post, request
import sqlite3
import os

pic_directory = './image_food'

def search(term):
    print(term)
    conn = sqlite3.connect('moondream-autocaption.db')
    cursor = conn.cursor()
    query = cursor.execute(f"SELECT * FROM image where description like ?",
                           (f'%{term}%',))
    response = query.fetchall()
    print(query)
```

```
    conn.close()

    return response

@route('/image_food/<filename:path>')
def serve_static(filename):
    return static_file(filename, root='./image_food')

@post('/')
@route('/')
def index():
    query = request.forms.get('query')
    response = search(query)

    form = '''<form action="./" method="post">
            Search <input type="text" name="query">
            <input type="submit">
            </form>
            <hr>
            '''

    page=''


    for image, description in response:
        image = os.path.join(pic_directory, image)
        page += f'''<div
                        style="height:400px;
                        width:200px;
                        display:inline-block;
                        border:2px solid black;
                        display: flex;
                        flex-direction: column;
                        justify-content: flex-start;
                        align-items: center;
                        overflow: hidden;">
                <img src="{image}"
                    style="height:auto;
                    width:200px;">
                <hr>
                {description}
                </div>
                '''

    page = f'''{form}
                <div style="display: flex;
                flex-wrap: wrap;
                align-items: flex-start;">
                {page}
                </div>'''

    return(page)

run(host='127.0.0.1', port='8080')
```

Intruder Detection







This lab shows you how to use MoonDream to create a visual security system that detects whether an area is empty, whether a person is in a room with a colored clothing, or if an intruder without a uniform is in the room. This could be used for security to set off an alarm or send out an alert, or could be used to turn on an IoT light in a warehouse enviornment.

When this script runs it will determine what is in the image and then set a color based on that. It will then dynamically create an HTML page with the image embedded and set the background color to the color value that was set.

NOTES:

- To keep the script simple we simply overwrite the image that is created. This creates a lag between when an image is taken and when the background color changes.
- The cv2.VideoCapture() values will generally be 0, but on M Macs and other systems might be 1.

moondream-see.py

```
import moondream as md
from PIL import Image
import cv2
import time

model = md.vl(model='./moondream-2b-int8.mf.gz')

def camera():
    # Open webcam (0 = default camera) - SD: M Processor Macs Use 1
    cap = cv2.VideoCapture(0)
    time.sleep(1) #Sleep added to make the image brighter

    if not cap.isOpened():
        print("Cannot open webcam")
        exit()

    ret, frame = cap.read()

    if ret:
        cv2.imwrite("captured_image.jpg", frame)
        print("Image saved as captured_image.jpg")
    else:
        print("Failed to grab frame")

    cap.release()
    cv2.destroyAllWindows()

while True:
    color = ''
    answer = ''
    camera()
    picture = './captured_image.jpg'
    image = Image.open(picture)
    encoded_image = model.encode_image(image)

    query = '''if person is wearing orange clothing say "good",
                else say "bad"'''

    query2 = '''if there is no one in say "empty"'''

    answer = model.query(encoded_image, query)["answer"]

    if answer.strip() == 'bad':
        answer = model.query(encoded_image, query2)["answer"]
        if answer.strip() == 'empty':
            answer = 'Room is Empty'
            color = 'grey'
        else:
            answer = 'Bad Person in Room'
            color = 'red'
    else:
        answer = 'Good Person is Room'
        color = 'green'

    print(f'Result: {answer}')
```

```
    with open('vision-alert.html', 'w') as file:
        file.write('<meta http-equiv="refresh" content="2">')
        file.write(f'<body style="background-color:{color};">')
        file.write(f'<img src="./captured_image.jpg"
style="height:300px;width:auto;">')
```