Sperm Detection Code Documentation

Overview

This Python script processes a video to detect and annotate sperm using frames extracted at regular intervals. It uses Azure Custom Vision for sperm detection and applies image enhancement techniques to improve detection accuracy. The annotated frames are saved in the specified output directory.

Modules and Libraries

```
"""Author : Tapash Ranjan Nandi
"""
import cv2
import requests
import os
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from PIL import Image, ImageFilter
import random
```

1. **cv2**: OpenCV library is used for video processing tasks such as reading frames.
2. **requests**: Used to send HTTP POST requests to the Azure Custom Vision API.
3. **os**: Provides functions for interacting with the operating system, such as checking and creating directories.

4. **matplotlib.pyplot**: Used for visualizing and annotating images with bounding boxes.
5. **matplotlib.patches**: Enables adding bounding boxes to images.
6. **PIL.Image**: Provides support for image processing.
7. **PIL.ImageFilter**: Used to enhance images by sharpening them.
8. **random**: Generates random colors for bounding boxes.

Global Variables

1. Azure Custom Vision Prediction API Details

```
# Azure Custom Vision Prediction URL and Key
PREDICTION_URL =
"https://humanspermdetection-prediction.cognitiveservices.azure.com/customvisi
on/v3.0/Prediction/f8487da5-df88-4e03-9652-5ac544d17aed/detect/iterations/Iter
ation1/image"
KEY = "a95cd4c9254140ee93be024fd2e3a471"
```

- **PREDICTION_URL**: URL of the Azure Custom Vision API for prediction.
- **KEY**: Prediction API key to authenticate requests.

These variables are required to communicate with Azure's Custom Vision service.

2. headers:

```
# Headers for the request
headers = {
    "Content-Type": "application/octet-stream",
    "Prediction-Key": KEY
}
```

- Defines HTTP headers for the API request:
  - Content-Type: Specifies that the request body contains binary image data.
  - Prediction-Key: Includes the Azure Custom Vision API key.

Functions

1. random_deep_color()

```
def random_deep_color():
    """Generate a random deep color in RGB format."""
    return (random.uniform(0, 0.5), random.uniform(0, 0.5), random.uniform(0, 0.5))
```

- **Purpose**: Generates a random deep color (RGB format) for bounding box edges.
- **Returns**: A tuple of three random float values between 0 and 0.5 representing RGB values.

- **Usage**: Ensures that bounding boxes have distinct yet subdued colors.

2. sharpen_image(image)

```python
def sharpen_image(image):
    """Sharpen the image to reduce blur."""
    return image.filter(ImageFilter.SHARPEN)
```

- **Purpose**: Sharpens the input image to reduce blurriness.
- **Parameters**:
  - image: A PIL.Image object to be sharpened.
- **Returns**: The sharpened image as a PIL.Image object.
- **Usage**: Enhances image clarity for better detection accuracy.

3. detect_sperm(image_path, output_image_path)

```python
def detect_sperm(image_path, output_image_path):
    """Detect sperm in the image and annotate it."""
    with open(image_path, "rb") as image_file:
        image_data = image_file.read()

    response = requests.post(PREDICTION_URL, headers=headers, data=
image_data)

    if response.status_code == 200:
        predictions = response.json()["predictions"]

        image = Image.open(image_path)
        plt.figure(figsize=(10, 10))
        plt.imshow(image)
        ax = plt.gca()

        for prediction in predictions:
            probability = prediction['probability']

            if probability > 0.96:
                bounding_box = prediction['boundingBox']

                left = bounding_box['left'] * image.width
                top = bounding_box['top'] * image.height
                width = bounding_box['width'] * image.width
                height = bounding_box['height'] * image.height

                color = random_deep_color()
                rect = patches.Rectangle((left, top), width, height,
linewidth=2, edgecolor=color, facecolor='none')
                ax.add_patch(rect)

                plt.text(left, top - 10, f"Prob: {probability:.2%}",
                         color=color, fontsize=12, weight='bold')

        plt.axis("off")
        plt.savefig(output_image_path, bbox_inches='tight', pad_inches=0)
        print(f"Annotated image saved to: {output_image_path}")
        plt.show()

    else:
        print(f"Failed to make prediction: {response.status_code}")
        print(response.json())
```

- **Purpose**: Detects sperm in a given image using Azure Custom Vision and annotates it with bounding boxes and confidence scores.
- **Parameters**:
    - image_path: Path to the input image.
    - output_image_path: Path to save the annotated image.
- **Steps**:

    - Opens the image file in binary mode and sends it to the Azure Custom Vision API.
    - Parses the API response for predictions.
    - Filters predictions with a confidence probability greater than 96%.
    - Annotates the image with bounding boxes and confidence scores.
    - Saves the annotated image to the specified path.
- **Error Handling**:
    - Prints an error message if the API request fails.

4. process_video(video_path, output_folder)

```python
def process_video(video_path, output_folder):
    """Convert video to frames, enhance them, and detect sperm."""
    # Check if the output folder exists; if not, create it
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
    else:
        print(f"The directory '{output_folder}
' already exists. Proceeding with the existing directory.")

    cap = cv2.VideoCapture(video_path)
    frame_rate = cap.get(cv2.CAP_PROP_FPS)
# Get the frame rate of the video
    frame_interval = int(2 * frame_rate)    # Set the interval to 2 seconds
    frame_count = 0
    success = True

    while success:
        # Set the frame position to extract one frame every 2 seconds
        cap.set(cv2.CAP_PROP_POS_FRAMES, frame_count * frame_interval)
        success, frame = cap.read()

        if not success:
            break

        # Convert the frame to a PIL Image for enhancement
        image = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
        enhanced_image = sharpen_image(image)
        # Save the enhanced frame
        frame_file_path = os.path.join(output_folder, f"frame_{frame_count
:04d}_6.jpeg")
        enhanced_image.save(frame_file_path)

        # Detect sperm in the enhanced image and annotate it
        detect_sperm(frame_file_path, frame_file_path.replace(".jpeg",
"_detected.jpeg"))

        frame_count += 1

    cap.release()
    print(f"Processed {frame_count} frames from the video.")
```

- **Purpose**: Extracts frames from a video, enhances them, detects sperm, and annotates the frames.
- **Parameters**:
    - video_path: Path to the input video file.
    - output_folder: Directory to save the processed frames and annotated images.

- **Steps**:
  - Creates the output folder if it does not exist.
  - Opens the video using OpenCV and calculates the frame interval (one frame every 2 seconds).
  - Iteratively:
    - Sets the frame position to extract frames at the desired interval.
    - Converts the frame to a PIL image and sharpens it.
    - Saves the enhanced frame to the output folder.
    - Calls detect_sperm to detect and annotate sperm in the frame.
  - Releases the video file after processing.
- **Output**:
  - Enhanced frames and annotated images saved in the output folder.
  - Prints the total number of processed frames.

Main Execution

```
if __name__ == "__main__":
    # Replace with the path to your video
    video_path = "/home/jyoti/Documents/sperm_detection/sperm_vdo_2.mp4"
    output_folder =
"/home/jyoti/Documents/sperm_detection/Detected_sperm_test_1"

    process_video(video_path, output_folder)
```

if __name__ == "__main__":

- Specifies the video path and output folder.
- Calls process_video to process the video file and save results.