Final Project

李龙昊 23210980049 GitHub

推荐使用html格式进行观看！！！！！，pdf无法展示视频效果。

视频介绍

- **视频时长**：视频总时长为12小时00分34.91秒。
- 视频流信息：
  - **视频流**：包含一个视频流，编码为 `H.264`，分辨率为 `1280x720 (720p)` ，帧率为 `29.97 FPS` 。
  - **音频流**：包含一个音频流，编码为 `AAC`，采样率为 `44100 Hz`，双声道。

由于视频内容过于庞大，因此需要处理视频。目前的处理思路是**使用 `ffmpeg` 截取了前10份钟的视频作为输入**

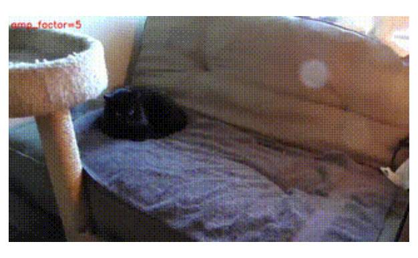
Eulerian中的Linear方法

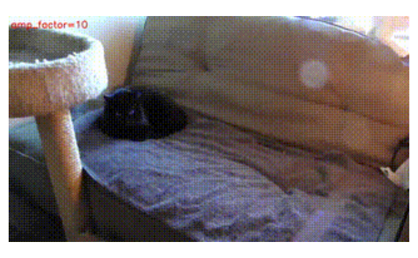
```
def magnify_motion(images, magnification_factor):
    """
    放大图像序列中的运动。该函数通过计算连续两帧之间的差异，并将差异乘以一个放大系数，
    然后将增强的差异加回原始图像中，从而实现运动的放大效果。
    """
    # 创建一个空列表，用于存储处理后的图像
    output_images = []
    # 初始化前一帧图像为序列的第一帧
    prev_image = images[0]
    for i in range(1, len(images)):
        # 获取当前处理的帧
        current_image = images[i]
        # 计算当前帧与前一帧之间的差异
        frame_diff = cv2.absdiff(current_image, prev_image)
        # 将差异乘以放大系数
        magnified_diff = cv2.multiply(frame_diff, np.array([magnification_factor], dtype=np.uint8))
        # 将放大的差异添加回当前帧
        enhanced_image = cv2.add(current_image, magnified_diff)
        # 将处理后的图像添加到输出列表中
        output_images.append(enhanced_image)
        # 更新前一帧图像为当前帧，为下一次循环做准备
        prev_image = current_image

    return output_images
```

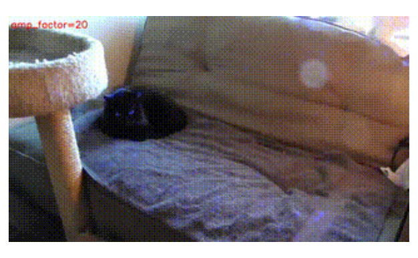
将生成的图片转化为视频输出

```
def create_video_from_images(image_folder, output_video_file, amp=5, fps=30):
    images = [img for img in sorted(os.listdir(image_folder))]
    frame = cv2.imread(os.path.join(image_folder, images[0]))
    height, width, layers = frame.shape

    # 定义视频编码器和创建 VideoWriter 对象
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    video = cv2.VideoWriter(output_video_file, fourcc, fps, (width, height))

    for image in images:
        img = cv2.imread(os.path.join(image_folder, image))
        cv2.putText(img, 'amp_factor={}'.format(amp), (7, 37),
                    fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, color=(0, 0, 255), thickness=2)
        video.write(img)

    video.release()  # 释放资源
```

放大五倍、十倍、二十倍、四十倍的视频如下：







Eulerian中的Phase-based方法

```
def build_gaussian_pyramid(img, levels):
    """使用 OpenCV 创建高斯金字塔。"""
    pyramid = [img]
    for _ in range(levels - 1):
        img = cv2.pyrDown(img)
        pyramid.append(img)
    return pyramid

def laplacian_from_gaussian(gaussian_pyr):
    """从高斯金字塔生成拉普拉斯金字塔。"""
    laplacian_pyr = []
    for i in range(len(gaussian_pyr) - 1):
        size = (gaussian_pyr[i].shape[1], gaussian_pyr[i].shape[0])
        L = cv2.subtract(gaussian_pyr[i], cv2.pyrUp(gaussian_pyr[i + 1], dstsize=size))
        laplacian_pyr.append(L)
    laplacian_pyr.append(gaussian_pyr[-1])
    return laplacian_pyr

def reconstruct_from_laplacian_pyramid(lpyr):
    """从拉普拉斯金字塔重建图像，确保数据类型和范围。"""
    img = lpyr[-1].astype(np.float32)  # 确保顶层是float类型
    for layer in reversed(lpyr[:-1]):
        img = cv2.pyrUp(img, dstsize=(layer.shape[1], layer.shape[0])).astype(np.float32)
        layer_float = layer.astype(np.float32)  # 确保layer也是float类型
        img = cv2.add(img, layer_float)  # 使用相同类型的数组进行加法
    img = np.clip(img, 0, 255)  # 确保图像值在0-255范围内
    return img.astype(np.uint8)  # 转换为uint8类型以便显示和保存


def phase_magnify(channel, magnification_factor, levels):
    """相位放大单个颜色通道。"""
    g_pyr = build_gaussian_pyramid(channel, levels)
    l_pyr = laplacian_from_gaussian(g_pyr)

    # 处理每一层的相位
    for i in range(len(l_pyr)):
        complex_layer = np.fft.fft2(l_pyr[i].astype(np.float32))
        magnitude = np.abs(complex_layer)
        phase = np.angle(complex_layer)
        # 直接计算放大的相位
        magnified_phase = phase * (1 + magnification_factor)  # 放大相位
        new_complex_layer = magnitude * np.exp(1j * magnified_phase)
        l_pyr[i] = np.fft.ifft2(new_complex_layer).real

    return reconstruct_from_laplacian_pyramid(l_pyr)

def phase_based_motion_magnification(images, magnification_factor, levels=3):
    output_images = []
    # 为每幅图像的每个颜色通道应用相位放大并重建
    for img in tqdm(images, desc="处理图像"):
        channels = cv2.split(img)
        magnified_channels = []
        for channel in channels:
            magnified_channel = phase_magnify(channel, magnification_factor, levels)
            magnified_channels.append(np.clip(magnified_channel, 0, 255).astype(np.uint8))
        magnified_image = cv2.merge(magnified_channels)
        output_images.append(magnified_image)
    return output_images
```

将生成的图片转化为视频输出

```
def create_video_from_images(image_folder, output_video_file, amp=5, fps=30):
    images = [img for img in sorted(os.listdir(image_folder))]
    frame = cv2.imread(os.path.join(image_folder, images[0]))
    height, width, layers = frame.shape

    # 定义视频编码器和创建 VideoWriter 对象
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    video = cv2.VideoWriter(output_video_file, fourcc, fps, (width, height))

    for image in images:
        img = cv2.imread(os.path.join(image_folder, image))
        cv2.putText(img, 'amp_factor={}'.format(amp), (7, 37),
                    fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, color=(0, 0, 255), thickness=2)
        video.write(img)

    video.release()  # 释放资源
```

放大五倍、十倍、二十倍、四十倍的视频如下：