```python
In [1]: import cv2
        import os
        import pathlib
```

```python
In [2]: file_dir = '/home/jupyter-doggo/scratch/NOAA/02-05-2023/sample_img/'
```

```python
In [3]: # def image_resizer(file_dir, perc_reduction):
```

```python
In [4]: def vid_generator(file_dir, start_frame_name, end_frame_name):
            # Get all the possible files in the directory
            filename = os.listdir(file_dir)

            if len(filename) == 0:
                return -3

            # Sorting the filename list is crucial on linux
            filename.sort()

            # Get the start and end frame index
            try:
                start_idx = filename.index(start_frame_name + '.png')
            except:
                # The start file is not found.
                return -2

            try:
                end_idx = filename.index(end_frame_name + '.png')
            except:
                return -1

            # Roll the image list array since the start and end are already found
            images = filename[start_idx:end_idx + 1]

            # Define frame based on the first image, we assume that the size
            # of the image stays the same during the export process
            frame = cv2.imread(os.path.join(file_dir, images[0]))

            # Acquire the dimension and number of layers in the image
            height, width, layers = frame.shape

            # Construct name of the video
            video_name = start_frame_name + '_to_' + end_frame_name + '.avi'

            # Start video writer
            video = cv2.VideoWriter(video_name, 0, 0.25, (width, height))
        #     video = cv2.VideoWriter(video_name, 0, 1, (width, height))

            for image in images:
                video.write(cv2.imread(os.path.join(file_dir, image)))

            cv2.destroyAllWindows()
            video.release()
            return 0
```

```
In [5]:  # Testing purpose

         return_code = vid_generator(file_dir, '2012.10.06.1300', '2016.10.05.1800')
         return_code
```

Out[5]: 0

```
In [6]:  ## TO-DO:
         start_idx_arr = [] # This needs to come from a csv file
         end_idx_arr = [] # This also needs to come from a csv file

         # Only execute when the numbers of timestamps match perfectly
         print('The timestamps align properly: ', len(start_idx_arr) == len(end_idx_a
```

The timestamps align properly: True

In [ ]:

In [ ]:

In [ ]: