NvDecoderBug

September 3, 2022

[8]:
```
import torch
import torchvision.transforms.functional as TF
import PyNvCodec as nvc
import PytorchNvCodec as pnvc
import numpy as np
import os
```

[48]:
```
def encode(t:torch.Tensor, path:str) -> None:
    n,c,h,w = t.shape
    print(t.shape)
    assert t.dim() == 4

    encoded = np.ndarray(shape=(0), dtype=np.uint8)
    surface = nvc.Surface.Make(nvc.PixelFormat.NV12, w, h, 0)
    plane = surface.PlanePtr()
    written = 0

    with open(path, "wb") as stream:
        nvenc = nvc.PyNvEncoder({
            "preset":"P7",
            "tuning_info":"high_quality",
            "codec":"hevc",
            "rc":"constqp",
            "s": f"{w}x{h}"}, 0)

        for yuv in t:
            yuv = yuv.cuda()
            y,uv = yuv[[0]], TF.resize(yuv[[1,2]], (h//2, w//2))
            uv = uv.view(2,-1).permute(1,0).reshape(1,-1,w)
            nv12 = torch.cat((y, uv), dim=1)
            pnvc.TensorToDptr(nv12, plane.GpuMem(),
                              plane.Width(), plane.Height(),
                              plane.Pitch(), plane.ElemSize())

            ok = nvenc.EncodeSingleSurface(surface, encoded)
            if ok:
                stream.write(bytearray(encoded))
```

```
                written += 1
                print(f"Encoded frame {written}/{n}")

        while written < n:
            ok = nvenc.FlushSinglePacket(encoded)
            if ok:
                stream.write(bytearray(encoded))
                written += 1
                print(f"Encoded frame {written}/{n} (flush)")

    fs = os.path.getsize(path)
    print(f"Filesize: {fs/(1024*1024):0.02f}MB. Compression ratio: {100 * fs /␣
 ↪t.numel()}%")
```

[50]:
```
def decode(path:str) -> torch.Tensor:
    t = []
    read = 0
    nvdec = nvc.PyNvDecoder(path, 0)
    while True:
        surface = nvdec.DecodeSingleSurface()
        if surface.Empty(): break
        read += 1
        print(f"Decoded frame {read}")

        plane = surface.PlanePtr()
        nv12 = pnvc.DptrToTensor(
            plane.GpuMem(), plane.Width(), plane.Height(),
            plane.Pitch(), plane.ElemSize()).cpu()
        assert nv12 is not None
        h,w = nv12.shape
        h = 2 * h // 3
        y,uv = nv12[:h], nv12[h:]
        uv = uv.view(-1, 2).permute(1,0).view(2,h//2,w//2)
        uv = TF.resize(uv, (h,w))
        t.append(torch.cat((y.unsqueeze(0), uv)))

    return torch.stack(t)
```

[56]:
```
# Encode / decode 2 frames

t = (torch.rand(2, 3, 1080, 1920) * 255).byte()
encode(t, "random.hevc")
decompressed = decode("random.hevc")
```

```
torch.Size([2, 3, 1080, 1920])
Encoded frame 1/2 (flush)
Encoded frame 2/2 (flush)
```

```
Filesize: 1.81MB. Compression ratio: 15.272657857510287%
Decoded frame 1
Decoded frame 2
```

[57]:
```
# Encode / decode a SINGLE frame

t = (torch.rand(1, 3, 1080, 1920) * 255).byte()
encode(t, "random2.hevc")
decompressed = decode("random2.hevc")
```

```
torch.Size([1, 3, 1080, 1920])
Encoded frame 1/1 (flush)
Filesize: 1.81MB. Compression ratio: 30.522939171810698%
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
/tmp/ipykernel_30697/1595040808.py in <module>
      3 t = (torch.rand(1, 3, 1080, 1920) * 255).byte()
      4 encode(t, "random2.hevc")
----> 5 decompressed = decode("random2.hevc")

/tmp/ipykernel_30697/3272144749.py in decode(path)
     21         t.append(torch.cat((y.unsqueeze(0), uv)))
     22
---> 23     return torch.stack(t)

RuntimeError: stack expects a non-empty TensorList
```

[ ]: