Propainter Development

Yanglin ZHANG

2024-10-03

Task 1: Deploy gradio

Development Environment

Clone the repository by running the following command:

```
git clone git@github.com:lucky9-cyou/ProPainter.git
```

Download the propainter checkpoints and SAM checkpoints. For SAM, we use the `sam_vit_h_4b8939.pth` checkpoint.

Install the development environment by running the following commands:

```
# create new anaconda env
conda create -n propainter python=3.8 -y
conda activate propainter

# install pytorch
conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c
nvidia

# intall tensortrt for cuda 11.8
wget https://developer.nvidia.com/downloads/compute/machine-learning/tensorrt/
10.5.0/local_repo/nv-tensorrt-local-repo-ubuntu2204-10.5.0-cuda-11.8_1.0-1_amd
64.deb
dpkg -i nv-tensorrt-local-repo-ubuntu2204-10.5.0-cuda-11.8_1.0-1_amd64.deb
sudo cp /var/nv-tensorrt-local-repo-ubuntu2204-10.5.0-cuda-11.8/nv-tensorrt-
local-EE22FB8A-keyring.gpg /usr/share/keyrings/
sudo apt update
sudo apt install tensorrt
python3 -m pip install --upgrade tensorrt-cu11

# install python dependencies
pip3 install -r requirements.txt

# install web dependences
pip install -r web-demos/hugging_face/requirements.txt
```

Run the Gradio Application

Run the following command to start the Gradio application:

```
cd web-demos/hugging_face/
python3 app.py
```

The Gradio application will be available at 'http://127.0.0.1:6006/' by VSCode port forwarding or 'http://101.126.90.71:50183'.

Task 2: Invoke the Gradio Application

You can use `client.py` to invoke the Gradio application. The following is an example of how to use the client to invoke the Gradio application:

```
python client.py --video inputs/sample/sample.mp4 --pose weights/vitpose.pt
```

The inpainted video will be saved to `outputs/sample.mp4`. If you want to change the output path, you can use the `--output` option.

Task 3: Optimization inference speed

Time Analysis

Current command:

```
/usr/src/tensorrt/bin/trtexec --onnx=raft.onnx --saveEngine=raft-fp8.engine --
fp8 --verbose  --minShapes='gtlf_1:1x3x640x360','gtlf_2:1x3x640x360' --
optShapes='gtlf_1:12x3x640x360','gtlf_2:12x3x640x360' --
maxShapes='gtlf_1:12x3x640x360','gtlf_2:12x3x640x360'  --
dumpOptimizationProfile --builderOptimizationLevel=5 --useSpinWait --
sparsity=enable > raft-fp8.log
```

All the time is based on the `sample.mp4` video. The video resolution is 640x360 (360p), and the video length is 1032 frames.

	VOS tracking	**Raft time**	**Complete flow time**	**Image propagation**	**Feature Propagation**
Time	24090.20447 ms	58275.726223 ms	6067.899583 ms	1963.095136 ms	86457.671271 ms

RAFT Optimization

The RAFT model is composed of three parts: feature block, context block and update block. The following is the optimization strategy for each block:

- Using tensorrt `best` mode to optimization.

Some commands: