Things Done (10.9)

---

1. Read the MONO paper and written a brief summary of it. Decided to start working on the Segmentation section
2. Coded the data loader together with image preprocessing. Looking good.
3. Starting to gain familiarity with PyTorch & implemented some basic NN
    - Built/Train an MLP tested on MNIST
    - Built/Train an easy CNN tested on Dogs vs. Cats

TODO

---

1. Understand more about the Segmentation Network:
    a. Draw a clear picture of the architecture
    b. What is the detection engine used for? why first detect before segmentation?
    c. What is ROI_align?
2. Implement the Segmentation Network
3. (Train the Segmentation Network)

Paper summary: Monocular Real-Time Volumetric Performance Capture

---

textured_capture.pdf

- Reconstructs textured **3D human** from each **frame** of a video without multi-view studio setup or pre-captured template
- **Progressive surface localization algorithm** and **mesh-free direct rendering** (2 orders faster than brute-force Marching Cube alg.)
- Adopt the **Online Hard Example Mining** (OHEM, ex1) technique to suppress failure of challenging examples

Performance Capture Methods

- *Cue-based*: uses silhouettes, multi-view correspondences, reflective information - could achieve high quality - required many cameras and controlled illumination
- *Template-based*: Joint/face detection -> pose estimation -> template fitting - could be extended to monocular image - most lacked personalized details such as clothing and hairstyles; Habermann et al. recovered texture detail through creating a textured 3D template.

- *Deep learning*: FCNN used to infer 3D skeletal joint. Saito et al. combined fully convolutional image features with implicit (surface) functions representation.

Proposed Method

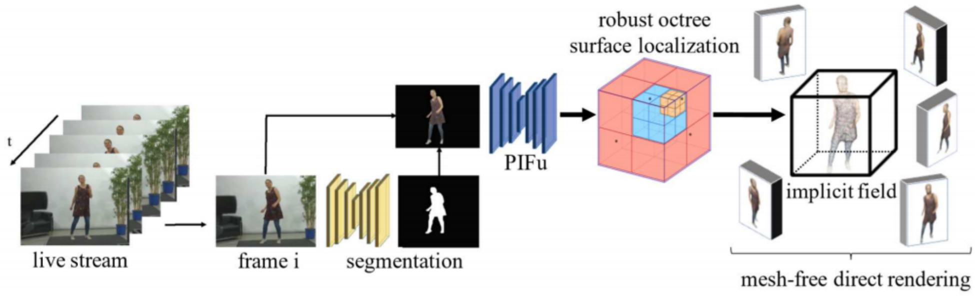
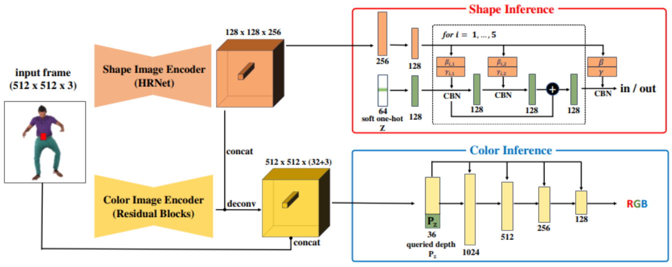


Fig. 2. System overview.

Implementation

- *Segmentation*:
  - Architecture: **U-Net** with **ResNet-18** backbone.
  - Technique: Reducing initial learning rate = **10.0** by **0.95/epoch** (**Adadelta**)
  - Trainset: https://drive.google.com/file/d/1jDUddrJlUlv5O_JAdb8qZk45EwtEqf_4/view (LIP+Web)
  - Valset: https://drive.google.com/file/d/1FPqz2P51sbnWo1K2FcowPnZCAGC1-_uY/view (LIP+Web)
  - Testset: https://drive.google.com/file/d/1gPkkqwiXKaPWLIIrF7QfvHHOu0B3zDjB/view
- *PIFu*:
  - Architecture: Modified upon PIFu.
    - **HRNetV2-W18-Small-v2** as **shape encoder** for better quality and speed
    - **6 residual blocks for color encoder (transposed convolution? *Don't really understand here*!) check out conv_arithmetic.pdf !**



  - Techniques:
    - **RMSProp** for shape inference; **Adam** for color inference; learning rate = **1e-3**
    - Batch size = **24**, sampled points = **4096/image**
    - Soft one-hot depth vector (**Soft-Z**)
    - Conditional batch normalization (**CBN**) for reducing channel size of MLP (***Don't understand***)

- Train shape inference for 5 epochs, fix it and train texture (color?) inference for another 5 epochs (***which part exactly? Do we train the encoders?***)

Segmentation (U-Net, ResNet-18 backbone)

Preprocessing (`data_builder.py`)

- Color Augmentation `T.ColorJitter(brightness=0.3, contrast=0.3, saturation=0.3, hue=0)`
- Normalization to $[-1, 1]$ for three channels
- Random Erasing `T.RandomErasing(p=0.5, scale=(0.02, 0.2), ratio=(0.3, 3.3), value=0)`
- Scaled to 256×256, preserves perspective (pad with grey)
- 50% Horizontal Flip

Architecture

TODO