In the following sections, we first introduce the problem formulation and denotation for motion transfer between human and anime character, and then propose a three-stage strategy that exploit structure and content information separately from real human video and anime image and finally generate the target anime video.

**A. definition of the problem.**

In this work, given an arbitrary real human video {V^r\_x} with specific motion ‘x’, and a single 2D anime image {I^a}, we aim to synthesis a video sequence {V^a\_x} in which anime character ‘a’ performing motion ‘x’. Therefore, we can clearly formulate the total objective of our project as:

F(V^r\_x, I^a) = \hat{V^a\_x}, minimalize L(\hat{V^a\_x}, V^a\_x ) (1)

where F(·) is an abstraction function of our entire system. \hat{V^a\_x} denotes the generated result, V^a\_x denotes the ground truth of anime video. L(·) is the loss function we will introduce in the following part(link?).

During the training phase, we can have full access to (V^r\_x, I^a, V^a\_x), including intermediate parameters, and apply preprocess and postprocess to the training data. Finally, we get the total well-trained models by optimizing the parameters by minimalizing the training loss. While during the testing phase, only (V^r\_x, I^a) are available and the result \hat{V^a\_x} should be outputted end-to-end.

However, ready-made pair (V^r\_x, I^a, V^a\_x), which serve as ground true in this work, cannot be collected neither online nor from real world. In detailed, we don’t have anime video performing the same motion ‘x’ as real video we collected and it is impossible to manually pair videos from the network. We address this problem in Stage one. Following other synthesis tasks(ref), we also use the intermediate feature to bridge the input and output. We will introduce the specific feature extraction and normalization method in Stage two. With the help of end-to-end neural network, we combine the features prepared to generate the final result, which will be shown in Stage three.

**B. Stage one: automatic data generation**

As introduced in part A, we must first construct dataset (V^r\_x, I^a, V^a\_x) before developing inference model while we cannot collect it directly from the real world. To fix this common problem in synthetic tasks, the previous work use either extra information to generate artificial video ground truth(ref) or original input to control output quality of different aspects(ref). In this work, we utilize the 3D anime model M^a to generate anime video V^a\_x of the collected counterpart V^r\_x. (so why we don’t use the second choice? Need to explain.) Instead of collecting 2D anime image I^a, we turn to the collection of 3D anime model M^a. According to previous 2D anime generation works(ref), extracting low-dimension information from its high-dimension version prove to be good dataset construction strategy. By dimension reduction, 2D information we need in the training phase of Eq(1) can be easily used to obtain from 3D features without any loss, as shown in the following equation:

DimReduction(M^a) = I^a, (2)

Combined with 3D modeling software which takes in anime model file M^a and motion file representing motion ‘x’, anime video ground truth V^a\_x can be generated. We choose MMD(ref) (denoted as MMD) for data generation because following OpenMMD(ref) pipeline, instead of manually configuring motion files, we can directly convert real human video V^r\_x to ‘.vmd’ file(video motion document) of anime model M^a, which can be denoted as following equation:

OpenMMD(V^r\_x, M^a) = VMD^a\_x, (3)

MMD(VMD^a\_x, M^a) = V^a\_x, (4)

where OpenMMD(·) represents the automatic pipeline introduced in (ref), MMD(·) represents the processing of software, VMD^a\_x represents the ‘.vmd’ file which contains modified motion information according to V^a\_x and M^a.

In detail, OpenMMD can be divided to four steps. 1)Use OpenPose(ref) API to do pose estimation over real human video V^r\_x frame by frame and saved as keypoints JSON files. For smoothing intermediate results as a video, we apply a Savitzky-Golay filter to output keypoints of all frames. 2) Combine all the keypoints JSON files to a continuous sequence with strong baselines for 3D human pose estimation, proposed in (ref). 3) Estimation of depth for objects, backgrounds and the moving person in the video using FCRN(ref) 4) Combine the formatted results to ‘vmd’ file VMD^a\_x, which can be directly fed to MMD for generating anime videos, originally proposed by (ref). The aforementioned four steps are integrated and operated automatically.

For MMD(·) processing, it is impossible to manually set parameters and operate the software to generate videos one by one. Therefore, by using winAppDriver(ref), a service to support Selenium-like UI test automation on Windows applications, specific script is written to automate these tedious operations.

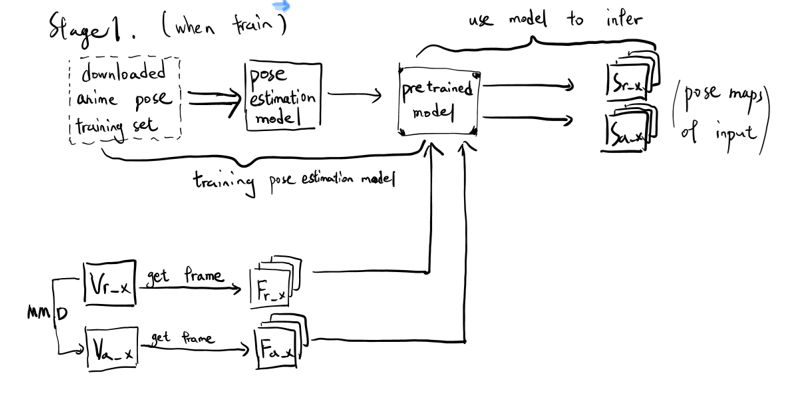
So far, we generate the pair (V^r\_x, V^a\_x) for training phase automatically and all these processing can be easily completed on personal laptop. Stage one can be formulated as the following equation:

Stage1(V^r\_x, M^a) = MMD(OpenMMD(V^r\_x, M^a), M^a) =V^a\_x, (5)

**C. Stage two: medium generation and normalization**

Medium for representation of intermediate feature is frequently used in synthesis tasks(ref) for segment the task to different phase, which makes synthesis more reasonable and effective. In this work, we actually want to extract structure feature from real human video V^r\_x and texture feature from anime image I^a as intermediate feature. The structure feature represents the motion ‘x’ for each video frame and texture feature represents the specific anime character ‘a’.

Following (ref), we use pose keypoints map to represent structure feature and denote S^a\_x pose keypoints maps of anime character ‘a’ performing motion ‘x’, S^r\_x pose keypoints maps of real person performing the same motion ‘x’. In order to extract structure feature from original input video V^r\_x and V^a\_x, we leverage two pose estimation model. Because pose estimation for single real human has achieve great performance, we simply adopt OpenPose interface to estimating real human pose, as previous works did. For anime pose estimation, we find that it performs bad if we directly apply or finetune on real human pose estimation. The feature extracted from real human and anime is quite different. Therefore, we pretrained a pose estimation model(ref) through external anime pose training set(ref). We choose this model because it shows good performance training on small datasets and our anime pose training samples are extremely insufficient even with data augmentation. After getting the pretrained models, we directly apply inference phase on frames extracted from video V^r\_x and V^a\_x, as shown in Fig1(draft)(need modified)



where F^r\_x and F^a\_x denotes frames extracted from F^r\_x and F^a\_x separately using OpenCV.

And training phase can be formulated as following:

P(V^r\_x, V^a\_x) = S^r\_x, S^a\_x, (6)

where P(·) denotes the pose estimation inference function of our pretrained model with same smoothing operation as Stage one.

It is natural to generate the predicted result \hat{V^a\_x} from its structure feature ground truth S^a\_x. However, the performance of anime pose estimation is very unstable due to the insufficient dataset, which cause strong noise if served as the training data. Directly using anime image with pose estimated will cause training to collapse. What’s more, during the testing phase, we can only have access to not the ground truth S^a\_x but an anime image input with unknown pose, which rewrites the equation 6 as:

P(V^r\_x, I^a) = S^r\_x, S^a\_0 (7)

where S^a\_0 represents the structure information of the anime character ‘a’ with unknown pose.

(not sure, should see the performance of stage3)

Therefore, we intend to set S^a\_x to normalized S^r\_x according to S^a\_x in training phase since they perform the same motion ‘x’. Due to the difference of real human and anime characters in limb proportions or distance to the camera, even if pose keypoints maps are extracted from videos performing the same motion, it is necessary to deal with the subtle offset and scale between real human and input anime character using normalization, which is also proposed in (ref). Although anime pose estimation for the whole body is unstable, the detection of neck, head and hips are always right. Based on this fact, we first align the neck keypoint of S^r\_x and S^a\_x, and then scale S^r\_x according to the upper body size of S^a\_x, which is calculated by the vertical distance between neck and hips. Finally, we rotate head and eye keypoints of S^r\_x to align with the direction of those in S^a\_x. Normalization processing of training phase can be formulated as:

N(S^r\_x, I^a) = \hat{S^a\_x}, minimalize L\_1(\hat{S^a\_x}, S^a\_x), (8) (questioned)

I^a = Sampling(V^a\_x), (9)

\hat{S^a\_x}

where N(·) denotes the normalization function which inputs structure feature and single anime image I^a, which is randomly sampled from frames of anime video V^a\_x in training phase or directly inputted in testing phase. L\_1(·) denotes the different between \hat{S^a\_x} and S^a\_x.

In stage two, input the ground truth pair (V^r\_x, V^a\_x), we generate the predicted structure feature \hat{S^a\_x} of the target anime video V^a\_x. Stage two can be formulated as the following equation:

Stage2(V^r\_x, V^a\_x) = N(P(V^r\_x), Sampling(V^a\_x)) = \hat{S^a\_x}, (10)

**D. Stage three: anime video generation by GAN**

Because of performance of GANs applied in generation tasks prove outstanding in recent works, especially synthesis and style transfer tasks(ref), our video synthesis method is based on a modular generative neural network presented in (ref) designed for synthesizing a image of person and a desired pose.

The model takes an input tuple of input image, output image and their keypoints from pose estimation. It first segments the input image into foreground and background layers and then further segments the character’s body into different parts and represents them in different layers. These layers are actually texture feature of input image, which allows to be modified independently. Then it calculates a similarity transformation fit using the keypoints maps of input images and applies it on each body part. Finally, merging body parts and refine the appearance of final output using a U-Net(ref) structure generator, training with a combination of feature and adversarial losses.

In original GAN setup, the generator network G plays a minimax game against discriminator D. The generator try its best to synthesize plausible images to deceive the discriminator D to judge it as ‘real’ images(ground truth) by learning differences from ground truth data. On the other hand, he discriminator D must discern between “real” (ground truth) images and “fake” images produced by the generator. The two networks are trained in turn and make each other better. Through this adversarial training, the results produced by the generator will be realistic and detailed. We formulate our adversarial loss as:

L\_GAN(G,D) =

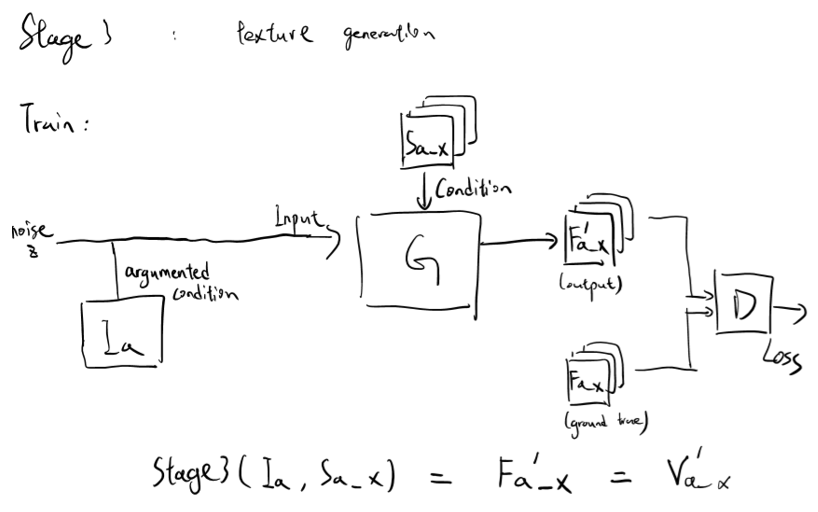
E\_\hat{S^a\_x},\hat{F^a\_x},F^a\_x [logD(F^a\_x,\hat{S^a\_x})+log(1-D((\hat{F^a\_x},\hat{S^a\_x)], (11)

TODO start here, VGG loss

In this work, during train phase, the input are two frames sampled from the same anime video V^a\_x and corresponding keypoints maps \hat{S^a\_x} generated in Stage two. In this way, we train a model that can synthesis anime image given a reference anime image I^a and target pose keypoints map. By processing frame-by-frame, we finally generate frames \hat{F^a\_x} of target anime video from the given reference anime image I^a and keypoints maps \hat{S^a\_x} representing target motion ‘x’. The processing can be formulated as following:

poseWarper(\hat{S^a\_x}, I^a) = \hat{F^a\_x}, minimalize L2(\hat{F^a\_x},F^a\_x), (13)

where poseWarper(·) represents our synthesis network inference function, L2(·) is the combination of feature and adversarial losses.



In stage three, input structure feature \hat{S^a\_x} and single anime image I^a, we generate the predicted anime video \hat{V^a\_x}, which is our overall target. Stage three can be formulated as the following equation:

Stage3(\hat{S^a\_x}, I^a) = \hat{V^a\_x}