View Prediction using Manifold Learning in non-linear feature subspace

Maliha Arif[a*], Dr. Abhijit Mahalanobis[b*],
[a,b] Center for Research in Computer Vision, University of Central Florida , 4000 Central Florida Blvd, 32816, Orlando, FL USA 407-823-2000

ABSTRACT

In this paper, we make use of a convolutional autoencoder to predict multiple unseen views of an object in the infrared domain. The dataset we use for this purpose is called 'DSIAC-ATR Image database' which has never been used before for view prediction in the non-linear feature subspace. Our method involves exploiting the underlying feature subspace – the manifold of the object - to predict an unseen view. We address a more challenging task of view prediction by working with greyscale images- the infrared images collected both during the day and night. We propose multiple architectures that not only predict how an object (a military vehicle in this case) will look like at a certain orientation but also learn to predict day or night infrared image and produce either as asked. We train our networks and show via experiments that the weights do not learn the geometry of transformation in the Euclidean space but rather in the Riemannian space. We explore the underlying feature subspace and observe that the networks learn the manifolds and thereby produce sharp, distinct and natural-looking images.

Keywords: autoencoder, view generation, non-linear subspace, infrared imagery

1. INTRODUCTION

Generating high quality natural-looking images itself is a challenging problem especially if the images are high resolution, and the task involves learning the underlying physical parameters. Natural images have been hard to reproduce mainly because of the intrinsic details they contain since those are very difficult to identify and quantify. Modern machine learning algorithms offer new methods that help improve the quality of images and at the same time generate more realistic ones. Some of these techniques involve the use of autoencoders (variational and denoising autoencoders being two of this type) and generative adversarial networks of which AC-GAN and DC-GAN are notable. Most of the recent works on image generation have been mainly focused on using synthetic datasets. [1] uses ShapeNet and produces multiple unseen views of cars and chairs using autoencoders in the non-linear subspace. [2] makes use of a concept of appearance flow and does bilinear sampling to produce an unseen view. It also works on synthetic images present in the KITTI dataset. Other than objects such as cars, one recent research [3] makes use of human models for pose estimation and prediction but again works on synthetic images in the SURREAL dataset. In contrast, our objective here is to use real world medium wave infrared images (MWIR) available in a public domain data set instead of synthetic or RGB data. In the non-linear feature subspace, predicting unseen views of objects in the infrared domain is a novel concept and one that hasn't been worked on. In the linear domain, [4] is one study that involves predicting unseen views but is able to do so at the expense of quality. We try to address these problems and predict images of infrared objects in the non-linear feature subspace by exploiting the manifold. We propose to use an auto encoder network with symmetric layers in the encoder and the decoder structure. We do not use an adversarial network as a similar problem in [1] resulted in noisy predicted images. Other than working with real world MWIR images, another challenge was the limited amount of training data available in the dataset used. To the best our knowledge, our research is the first of its kind to tackle real world infrared images for unseen view prediction with good qualitative and quantitative results.

*maliha.arif@knights.ucf.edu; phone 1 407 342-9907; amahalan@crcv.ucf.edu phone: 1 407 823-1119

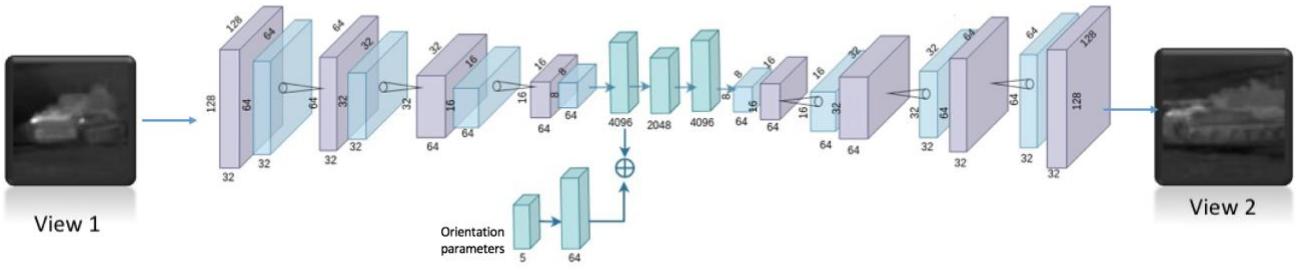

Figure.1. shows our architecture based on [1] with lesser parameters and condensed dimensions.

2. METHOD

We employ multiple methods to address the task at hand which is multi view generation on real world infrared images. Our first method is based on [1] which is an autoencoder based network using 5 layers in the encoder and decoder, having a symmetric composition. However, our network is more condensed and works with different parameters for view prediction. We create a network having 4 convolution blocks in the encoder and 4 convolution blocks in the decoder. The idea here is to force the decoder to use the latent space embedding to produce a different output (based on the parameters injected in the latent space) than the original image which was input to the encoder network. To achieve this goal

- We inject the desired orientation parameters in the embedded space and perform concatenation.
- Once we have modified the latent space embedding, we then pass this on to the decoder.

Figure 1. shows how this process was implemented. We input view 1 of the armored vehicle to the encoder, this view is what the network will use to predict a different view of the same armored vehicle at the decoder end. View 1 is an image of 128 x128 dimensions, these images have been extracted from 640 x 512 full sized images which were obtained from video sequences contained in the data set. Details for data preprocessing are under experimental setup section 3. These 128 x128 dimensional images are passed through a series of convolutional and max pooling layers. The first convolutional layer consists of 64 filters, the second convolution layer also has 64 filters, the last 2 layers having 32 filters each. Max pooling also at every step means our input of 128 x128 x3 gets reduced to 8 x8 x64 at the latent space. This latent space embedding is then reshaped into a vector of length 4096. Here at this point, our orientation parameters (our desired view characteristics) are concatenated. Our orientation parameters are as follows:

1) Day or Night target image
2) The azimuth of the target image
3) Elevation with minimal change in target image

These parameters are first processed separately in 2 fully connected layers before being concatenated with our 4096-latent space vector. After concatenation, we process the vector again using 2 fully connected layers of dimension 2048 and 4096 respectively. At this point, we have provided the network with information which tells it to produce a new output. The This concatenation of the input parameters with the latent space embedding is tricky but a significant step in the process. Decoding starts at this point, where augmented latent space encoding is sent into 4 up convolutional blocks. The 8x8x64 latent space embedding is now upsampled to return us 128x128x3 dimensional image of our desired view which is a new unseen view of the object. We use mean square error loss for this architecture. This architecture proved to be fruitful and gave us good predictions but also demonstrated some limitations. The predicted views lacked sharpness and larger angle variation in input and output views did not work well. To mitigate these problems, we devised the architecture as shown in the block diagram figure 2 below, which proved to be more complex and robust.

2.1 Skip Connections

Our second architecture which we are currently working on is shown in figure 2. It outlines our training strategy which comprises of two streams. We have 2 encoders; block 1 encodes the input view using a series of convolutions into a 4096 long embedded vector, whereas block 2 encodes View 2, which is our target view in another 4096 long embedded vectors. The insertion of orientation parameters and concatenation with the latent space encoding, and its processing via fully connected layers remains the same as in architecture 1. However, for decoding we create skip connections from block 2 to our decoder, thereby providing higher level features at each layer which adds to the activations produced by the decoder's up-convolutional layers. Skip connections are an important technique introduced in [5]. It has produced to be fruitful in a wide array of applications. For the problem at hand, previous works similar to our problem [1] [2] do not incorporate such techniques, we feel our inclusion of the technique for the aforementioned task is novel and proved to be valuable.

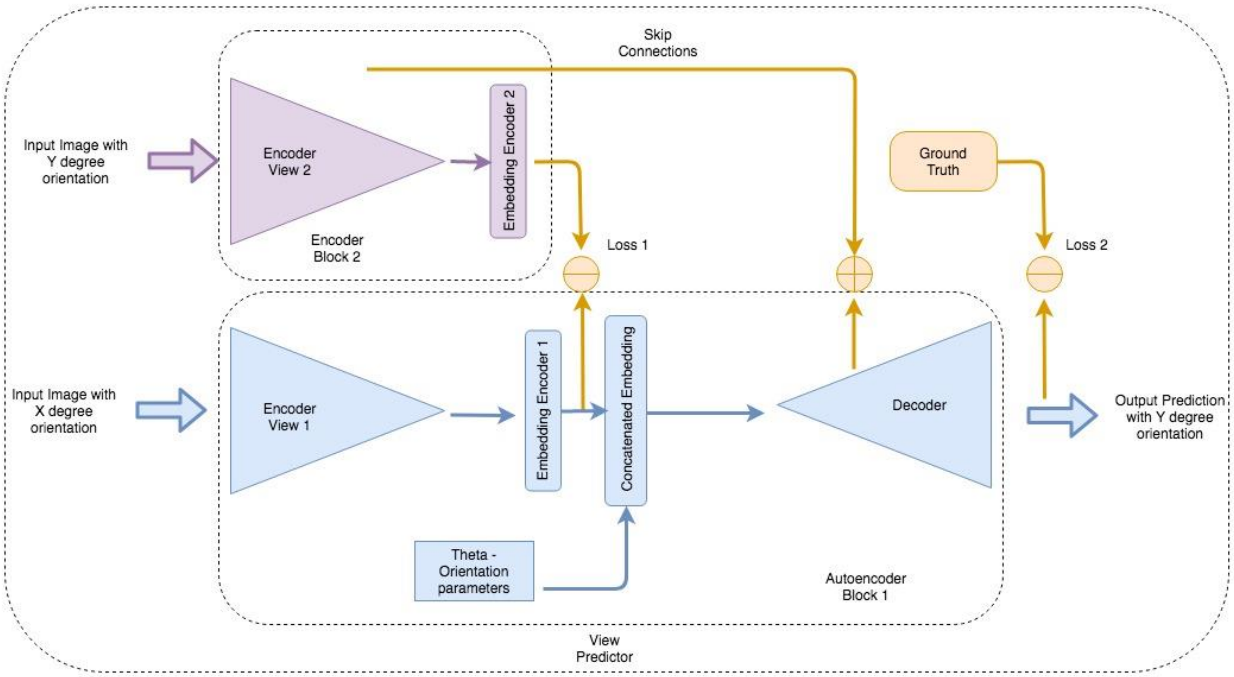

Figure 2. shows our architecture after adding skip connections and new loss function. The 2 blocks indicate two parallel streams much like a Siamese structure to understand the relation between the two inputs.

While creating a 2-stream network, we also derive our own loss function which is discussed in detail in section 2.2. This network basically takes good feature points from both views and predicts the desired view during training. The loss function makes use of both the embedded space vectors and final output view to guide the network to convergence. During test time, our goal is to only use block 1 (which is based on our architecture 1), without requiring view 2 as an input. However, this work is currently under progress, and will be reported in a future publication.

In order to train our network, we provide pairs of images that have wide angle variation in input and output views. The proposed network architecture in Figure 2 allows more freedom and variation in input and output view pairs. We get good results when we train with image pairs that over 10-degree separation in viewing angle (say 55-degree orientation for view 1, and view 2 at 65 degrees), but ask the network to predict images that are 5-degree apart (say 60-degree output for the same 55-degree input). We did not see this level of freedom when exercising our first approach based on [1] using the infrared real-world dataset. This shows our new approach is better at the problem being addressed. More details on the training technique, image pairing is in section 3 under experiments.

2.2 Loss function

We formulate the loss function considering the fact that we are using an autoencoder structure and all our information is condensed in the latent and non-linear feature subspace. We exploit this to our purpose and devise the following loss function.

$$L_e = \sum \|e_2 - e_1\|^2 \tag{1}$$

$$L_o = \sum \|y_2 - y_1\|^2 \tag{2}$$

$$L_t = L_e + L_o \tag{3}$$

$$L_{t=} \sum \|e_2 - e_1\|^2 + \sum \|y_2 - y_1\|^2 \tag{4}$$

Our loss function consists of 2 parts. Here e_1 and e_2 are the encodings of the 2 views being used during training. y_1 and y_2 are the ground truth and predicted output views respectively. We take mean of the square of the error between each pair of the 2 entities, $L_e$and L_o to find L_t. This allows us to guide our encoding to the desired viewpoint during training.

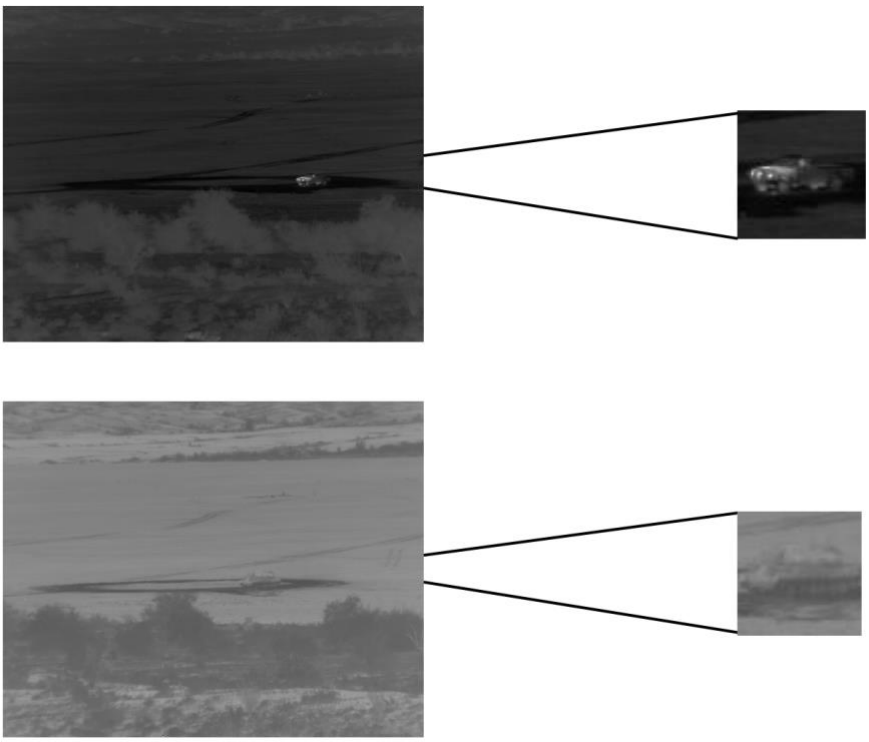

Figure 3. shows an illustrative image from the MWIR data set. The full images are the frames that we had in the dataset and these were captured at 987 m range, the closet available in the dataset. The chips extracted on the right are the one we fed to our network for training. The top row shows an example of a PICKUP during the night and the bottom row shows an example of a ZSU23 during the day.

3. EXPERIMENTS

We train our network on the Defense systems information analysis center- CSAIC ATR Dataset which is a dataset containing 207 videos in the arf format. There is a lot of preprocessing involved before we are able to extract the image / object chips that we use in our training. The videos in the arf format are first converted to avi format before being separated into frames. The frames are of 640 x 512 dimension as shown in figure 3. The videos are of 10 different categories, as shown in table 1. Each armored vehicle can be seen moving around in two 360 degree circles at nine ranges which lie between 1000 -5000 m distance. We select the images which are under 1000 m range for the view prediction experiments. These have well resolved features which allow us to compare how well the view prediction technique works. The 1000 m images are then used to extract only the object chips of size 64 x 64 which can be seen in figure 3. The real images make the task very challenging unlike synthetic images which are rendered for the research being done in [1] [2] [3]. Previous researchers have used large number of synthetic images produced in a controlled environment with desired angles, azimuth, elevation, degree and range which is of course not available when we work with real world IR images. Since we work with data only from the 1000 m range, we end up with training images ranging from either 0-180 degree in azimuth or 180-360 degree in azimuth for each video per class. Hence, we work with 9 classes x 180 images/class, or a total of 1620 images.

Name	Target description
Pickup	Civilian Pickup
SUV	Civilian Sports Utility Vehicle
BTR70	Armored Personnel Carrier
BRDM2	Infantry Scout Vehicle
BMP2	Armored Personnel Carrier
T72	Main Battle Tank
ZSU23	Anti-Aircraft Weapon
2S3	Self-Propelled Howitzer
MTLB	Armored Reconnaissance Vehicle
D20	Towed Howitzer

Table 1. shows the categories in the infrared dataset. Other than SUV and Pickup truck, all other vehicles are tracks and very similar in structure making their prediction more challenging.

3.1. Training pairs

The training process depends a lot on the different pairs we create for the input view and what we want the output to look like. Our dataset is organized in a way that for each orientation, there are 4 images within 1/4$_{th}$ degree of that view. We train architecture 1 using these minimal variations in degree and get a minimum loss of 0.0010. The 2$_{nd}$ network architecture is much more powerful and we use it to train with image pairs having 5-10-degree angle variation. Furthermore, the training loss is very low and hence we end up with realistic sharp looking images as shown in figure 4.
We train our network using adam optimizer and learning rate of 1e-4. We get convergence at 12 epochs. We use a batch size of 64 and our own loss function. Results for our experiments are shown in figure 4. We try variations by asking the network to give night image when given day image and vice versa.

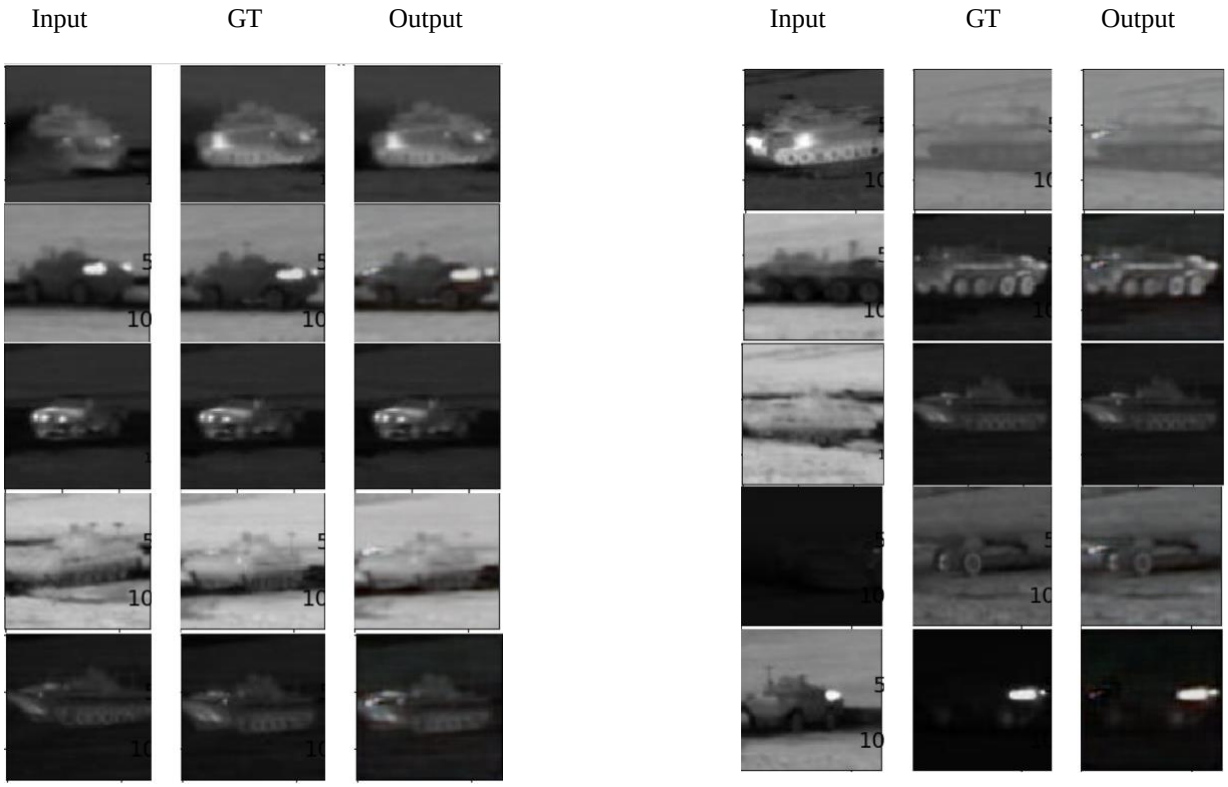

Figure.4. shows our network's predicted images as compared to ground truth. We can see how well the network understands our orientation parameters and when asked to get a day image or a night image, it does exactly that. The first set of pictures produces unseen views during the same time of the day. The second set of images however shows the network's transition from day to night image and vice versa.

3.2. Recursive testing

We perform another interesting experiment with our model. We use the predicted views in a recursive sequence and insert them as input to the network asking for view 1 as output. This view was the one used to create the other view 2 now being used as input. Results can be seen in figure 5. The network produces some sharp images again which shows that it was able to learn the features in the manifold subspace really well. This further shows how exploiting the manifold feature subspace is useful for tasks such as view morphing.

3.3. Classification scores

Lastly, we perform classification. We train a classifier using real world images for 8 classes and switch to predicted images only for training and testing for 1 class. We even try using predicted images for this class only for training and then testing using real world images. We get a good classification score for all classes highlighting the good quality of images our network is producing and how well it is learning the manifold. We fine-tune VGG-16 weights and build our classifier on top of it. We use 500 images per class in an 80:20 ratio for training and testing.

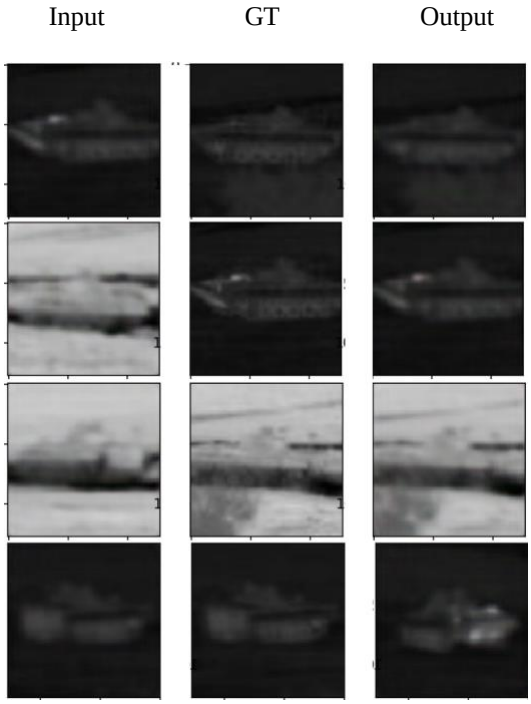

Figure 5. shows the results of our recursive experiments. Input view here is our previously predicted output image which is now sent in the network as a recursive loop to see what the network can make of it now and what else can it predict. The output view is what it is able to predict from the generated input.

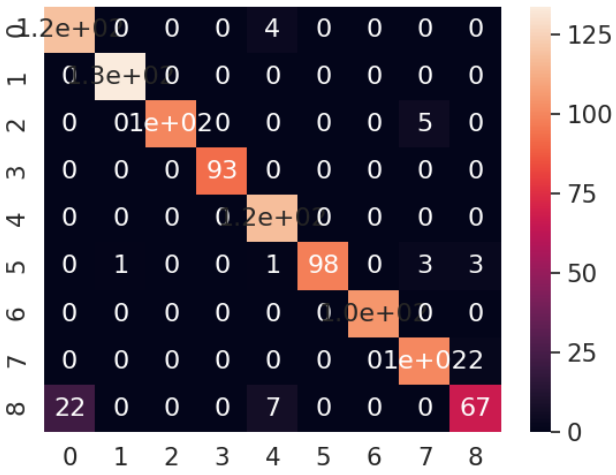

Figure 6. Shows our classification scores. The numbers in the diagonal represent the number of images correctly classified. These vary depending on classification scores per class and number of test images available. 5th class shown here is the one based on our network's predicted images only. We use the predicted images for training and testing one class and our VGG based classifier is able to classify our predicted images equally well as the real world actual images with minimal incorrect classifications.

4. Future works:

We plan on using t-SNE for predicted and real-world images to see how well our approach approximates the class manifolds. We are currently working on modifying the architecture so that view 2 is not used during our testing when experimenting with our architecture, in figure 2. Further, we want to exploit the latent space and observe how it responds to variation in orientation parameters which include but are not limited to day, night and azimuth information. We would also want to explore what the network is learning in terms of object classification and therein the manifold.

Acknowledgement: The authors gratefully recognize the support of Leonardo DRS for funding parts of this research.

REFERENCES

[1] Maxim Tatarchenko, Alexey Dosovitskiy, Thomas Brox, "Multi-view 3D Models from Single Images with a Convolutional Network", European Conference on Computer Vision (ECCV), 2016
[2] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, Alexei A. Efros, "View Synthesis by Appearance Flow", Computer Vision and Pattern Recognition (CVPR), 2017
[3] Shi Jin, Ruiynag Liu, Yu Ji, Jinwei Ye, Jingyi Yu, "Learning to Dodge A Bullet: Concyclic View Morphing via Deep Learning", European Conference on Computer Vision (ECCV) ,2018
[4] Abhijit Mahalanobis, Phil Berkowitz, Mubarak Shah, "View Morphing using Linear Prediction of subspace features"
[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", Computer Vision and Pattern Recognition (CVPR), 2015