EXPLOITING MULTIMODAL INFORMATION IN DEEP LEARNING

A Dissertation

by

QINBO LI

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Yoonsuck Choe
Committee Members,	Xia Hu
	Guni Sharon
	Hangue Park
Head of Department,	Scott Schaefer

May 2022

Major Subject: Computer Science

ABSTRACT

Humans are good at using multimodal information to perceive and to interact with the world. Such information includes visual, auditory, kinesthetic, etc. Despite the advancement in deep learning using single modality in the past decade, there are relatively fewer works focused on multimodal learning. Even with existing multimodal deep learning works, most of them focus on a small number of modalities. This dissertation will investigate various distinct forms of multimodal learning: multiple visual modalities as input, audio-visual multimodal input, and visual and proprioceptive (kinesthetic) multimodal input. Specifically, in the first project we investigate synthesizing light fields from a single image and estimated depth. In the second project, we investigate face recognition for unconstrained videos with audio-visual multimodal inputs. Finally, we investigate learning to construct and use tools with visual, proprioceptive and kinesthetic multimodal inputs.

In the first task, we investigate synthesizing light fields with a single RGB image and its estimated depth. Synthesizing novel views (light fields) from a single image is very challenging since the depth information is lost, and depth information is crucial for view synthesis. We propose to use a pre-trained model to estimate the depth, and then fuse the depth information together with the RGB image to generate the light fields. Our experiments showed that multimodal input (RGB image and depth) significantly improved the performance over the single image input.

In the second task, we focus on learning face recognition for low quality videos. For low quality videos such as low-resolution online videos and surveillance videos, recognizing faces based on video frames alone is very challenging. We propose to use audio information in the video clip to aid in the face recognition task. To achieve this goal, we propose Audio-Visual Aggregation Network (AVAN) to aggregate audio features and visual features using an attention mechanism. Empirical results show that our approach using both visual and audio information significantly improves the face recognition accuracy on unconstrained videos.

Finally, in the third task, we propose to use visual, proprioceptive and kinesthetic inputs to

learn to construct and use tools. The use of tools in animals indicates high levels of cognitive capability, and, aside from humans, it is observed only in a small number of higher mammals and avian species, and constructing novel tools is an even more challenging task. Learning this task with only visual input is challenging, therefore, we propose to use visual and proprioceptive (kinesthetic) inputs to accelerate the learning. We build a physically simulated environment for tool construction task. We also introduce a hierarchical reinforcement learning approach to learn to construct tools and reach the target, without any prior knowledge.

The main contribution of this dissertation is in the investigation of multiple scenarios where multimodal processing leads to enhanced performance. We expect the specific methods developed in this work, such as the extraction of hidden modalities (depth), use of attention, and hierarchical rewards, to help us better understand multimodal processing in deep learning.

DEDICATION

To my family:

My wife Anqi Bao

My parents

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my supervisor, Professor Yoonsuck Choe. I have been working with and supervised by Dr. Choe when I was a Master student since the beginning of 2014. Dr. Choe always gave us enough freedom to work on the research we are enthusiastic about, and encouraged us to try different kinds of approaches, so that we not only built up our domain knowledge but also became more independent researchers. But when I was blocked by obstacles and not able to move forward, Dr. Choe always brought insightful advise and guidance with his board knowledge in Artificial Intelligence. It is my honor to work with Dr. Choe and I will be always grateful of his guidance.

I would also like to express my gratitude to my committee members, Professor Hangue Park, Professor Guni Sharon, Professor Xia Hu for their continuous support. They brought insightful suggestions and opinions on all of my research works, so that I am able to continue improve on my research and my dissertation.

Specially, I would like to thank Professor Hye-Chung Kum and Professor Nima Kalantari for guiding and supervising me during the first two year in my Ph.D. study when Dr. Choe was temporarily on leave from Texas A&M University. Dr. Kum guided me in the first two years working on privacy preserved interactive record linkage. I still remember those days when we discussed in the meeting, ran the user study, attended the conference at San Antonio, and so on. When I decided to pursue the research that I am more enthusiastic about, Dr. Kum gave me full support. In my second year, Dr. Kalantari guided me to work on light field synthesis research for one year. Even though the time is not very long, I learnt a lot from Dr. Kalantari, and the solid work helped me to get my first Research Internship opportunity. I cannot imagine myself achieving all these without Dr. Kum's and Dr. Kalantari's help. In addition, thank you Professor Sheng-Jen Hsieh for being my co-advisor in my Master study.

In addition, I would like to thank my labmates and friends: Jaewook Yoo, Han Wang, Khuong N. Nguyen, Qing Wan, Xilong Zhou, Avinash Paliwal, and so on. We had a lot of discussions on

research and they gave me many insightful suggestions. I wish we could have more years working on research together.

Finally, I would like to thank my parents and my wife Anqi Bao for their love and continuous support, not only during my Ph.D. study but also in my life. I will love you forever.

Funding Sources

This work was funded in part by a TAMU T3 grant 246451, and by the DGIST R&D Program funded by the Korean Ministry of Science and ICT (21-IT-02).

NOMENCLATURE

MPI	Multi-plane Image
VMPI	Variable Multi-plane Image
AVAN	Audio-Visual Aggregation Network
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
RNN	Recurrent Neural Network
LF	Light Field
PSNR	Peak Signal-to-noise Ratio
SSIM	The structural Similarity Index Measure
AVFR	Audio-Visual Face Recognition
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
PPO	Proximal Policy Optimization
TRPO	Trust region policy optimization
EPI	Epipolar Plane Image
LSTM	Long Short-term Memory
MFCC	Mel-frequency Cepstral Coefficients
TAR	True Acceptance Rate
FAR	False Acceptance Rate
HAM	Hierarchical Abstract Machines
MDP	Markov Decision Process
SMDP	Semi-Markov Decision Process

HRL	Hierarchical Reinforcement Learning
MFCC	Mel Frequency Cepstral Coefficients
LPC	Linear Prediction Coefficients
PLP	Perceptual Linear Prediction

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Humans are good at using multimodal information to perceive and interact with the world. This information includes visual, auditory, kinesthetic, etc. Despite of the large advancement of deep learning using single modality in the past decades, there are relatively fewer works focus on multimodal learning. Even with existing multimodal deep learning works, most of them focus on a small number of modalities. This dissertation will investigate various distinct forms of multimodal learning: multiple visual modalities as inputs, audio-visual multimodal inputs, and visual, proprioceptive (kinesthetic) multimodal inputs. Specifically, in the first project we investigate synthesizing light fields from a single image and estimated depth. In the second project, we investigate face recognition for unconstrained videos with audio-visual multimodal inputs. Finally, we investigate learning to construct and use tools with visual, proprioception and kinesthetic multimodal inputs.

In the first task, we investigate synthesizing light fields with single RGB image and estimated depth. The goal of novel view synthesis is to synthesize an image from arbitrary target camera pose given a source image. Light field synthesis is a special problem in novel view synthesis that it focus on synthesizing small baseline novel views. A 4D light field describes light ray in 2D spacial coordinates (x, y) and 2D angular coordinates (u, v). Different from single RGB image, light field enables appealing effects such as viewpoint change, synthetic aperture, and refocusing. In the first project, we focus on synthesizing 4D light field from a single RGB image. This is a severely ill-posed problem, since the depth information is lost for a single RGB image, and depth information is crucial for view synthesis. Human can easily hallucinate novel views of a scene because human have two eyes to perceive parallax and hence can estimate the depth of the scene. Therefore we propose to use a pre-trained CNN to estimate depth of the input image, and then fuse the estimated depth with the RGB image to the view synthesis network. Experiments show that using the visual multimodal inputs consistently perform better than using only the single RGB input. In addition, we notice that using a single CNN to synthesize the novel view usually generate artifacts, because

the CNN has to learn to synthesize the novel views and hallucinate the occluded views at the same time. Therefore, we further introduce VMPI representation and two network fusion architecture to reduce the artifacts.

In the second project, we investigate face recognition for unconstrained videos. While face recognition has been well addressed for high quality images and videos, there are still many open issues for unconstrained videos. For example, in some videos such as surveillance videos, the target is far away from the camera, and in some video there are a lot of motion blur because the camera is not stable. We propose to use audio information in the video to aid in the face recognition task with mixed quality inputs. In order to achieve this goal, we introduce an Audio-Visual Aggregation Network (AVAN) to aggregate multiple facial and voice information to improve face recognition performance. To effectively train and evaluate our approach, we constructed an Audio-Visual Face Recognition (AVFR) dataset from unconstrained YouTube videos. Our AVFR dataset contains around 5.5K unconstrained videos of 1K subjects. Empirical results show that our approach with audio-visual multimodal inputs significantly improves the face recognition accuracy on unconstrained videos. The qualitative evaluation shows that, our model successfully learned to assign high weight to high quality visual frames and clear audio clips, and assign low weights to low quality visual frames and noisy audio clips.

In the last project, we focus on learning to construct tools with raw pixel input and proprioceptive (kinesthetic) input. The tool construction and use challenge requires many skills and knowledge, including sensorimotor skills, knowledge of basic physics, logic and planning capabilities, problem posing and problem solving skills, representation of sequential and hierarchical concepts, and so on. Learning this task with only raw pixel inputs is very challenging. To accelerate the training, we propose to use raw pixel image, as well as the proprioceptive and internal kinesthetic information as input. Different from previous works that only using simple 2D environment without physical simulation, we created a physical simulated environment using OpenAI gym and PyBullet. The environment includes two three-joint robot arm, two tools and one target. The goal is to reach the target object and pull it to the bottom of the environment. Because the

target is beyond the range of the arm, to successfully reach the goal, the agent must learn to construct a "T" shape tool or a "L" shape tool to pull the target to desired area. In addition, because the exploration space is very large and the reward is sparse, simply using reinforcement learning approach such as PPO [6] cannot solve the problem effectively. Therefore, we propose to use hierarchical reinforcement learning with curiosity reward. Specifically, if the agent reach a new state that it has not seen previously, then there will be some reward to encourage exploration.

The proposal is organized as follows: we investigate synthesizing light fields with single image and estimated depth in Chapter 2; in Chapter 3, we investigate face recognition for unconstrained videos; we focus on learning to construct tools with raw pixel input and proprioceptive (kinesthetic) input in Chapter 4; finally, conclusions and discussions are presented in Chapter 5.

2. SINGLE IMAGE AND DEPTH-BASED MULTIMODAL LIGHT FIELD SYNTHESIS

We propose a learning-based approach to synthesize a light field with a small baseline from a single image. This is a severely ill-posed problem since the depth information is lost, and depth information is crucial for view synthesis. We propose to use a pre-trained model to estimate the depth, and then fuse the depth information together with the RGB image to CNN to generate light field. The CNN promotes the input image into a layered representation of the scene. The layered representation is multiplane image (MPI) proposed by Zhou et al.[7] We extend the multiplane image (MPI) representation by allowing the disparity of the layers to be inferred from the input image. We show that, compared to the original MPI representation, our representation models the scenes more accurately. Moreover, we propose to handle the visible and occluded regions separately through two parallel networks. The synthesized images using these two networks are then combined through a soft visibility mask to generate the final results. To effectively train the networks, we introduce a large-scale light field dataset of over 2,000 unique scenes containing a wide range of objects. Our experiments showed that the multimodal input significantly improved

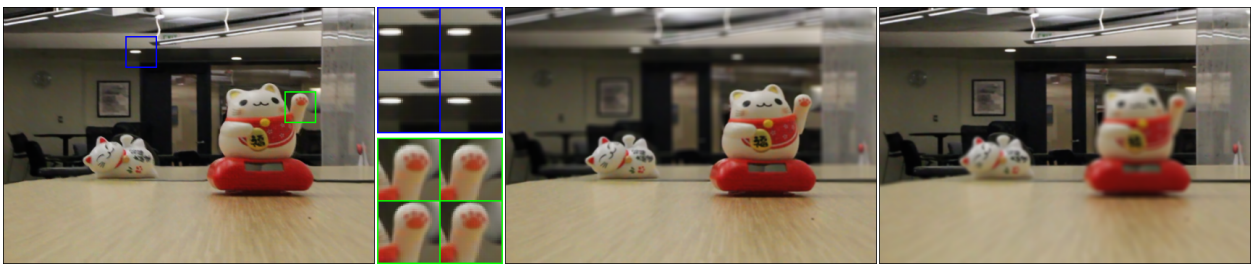


Figure 2.1: We use our learning-based techniques to synthesize an 8×8 light field from an input image, captured with a standard Canon camera. One of our synthesized corner views along with all the corner views for two insets are shown on the left. On the right, we use our synthesized light field to generate two refocused images. The image is courtesy of Wang et al. [1].

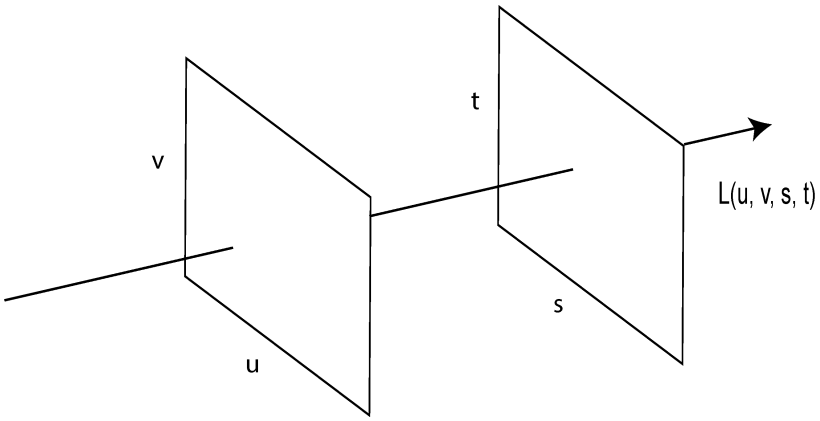


Figure 2.2: The 4D light field representation [2]

the performance over the single image input. We also demonstrate that our approach synthesizes high-quality light fields on a variety of scenes, better than the state-of-the-art methods.

2.1 Background and Introduction

2.1.1 Light Field

The standard digital camera can only captures 2D RGB images. In comparison, light field contains the spatial and angular information of all the light rays passing all the points in the space. Adelson *et al.* [8] propose a plenoptic function to describe light field as follows:

$$P = P(\theta, \phi, \lambda, t, V_x, V_y, V_z) \tag{2.1}$$

where V_x, V_y, and V_z describe all the possible points in the space, θ and ϕ describe all the possible angle, and λ, t stand for wavelength and time respectively. Because the 7-dimensional plenoptic function is complicate and difficult to capture and record, McMillan and Bishop introduce a 5-dimensional function to represent light field. Later, Levoy and Hanrahan [2] discuss a 4D representation of light field, as illustrated in fig. 2.2.

Because 4D light fields capture both the intensity and direction of light, it enables appealing effects such as viewpoint change, synthetic aperture, and refocusing. By selecting light rays in different angular resolution, we can generate an animation of different viewpoint of an object, just

Focused on foreground Focused on background

Figure 2.3: An example of refocusing.

like moving our eyes up and down, left and right to observe the object. It is also possible to change the position of the focal plane to focus on different part of the scene, which is known as refocusing. Fig. 2.3 shows an example of refocusing.

However, capturing a light field is difficult as it requires taking a set of images from different views at the same time. One way to capture light field is using a camera array [9], as shown in fig. 2.4-a. In recent years, there are commercial cameras such as Lytro (shows in fig. 2.4-b) and RayTrix to capture light fields, but they are still not as widespread as standard digital and cellphone cameras. Therefore, to make light fields widespread, our goal in this work is to synthesize 4D light fields from a single 2D RGB image.

2.1.2 Novel View Synthesis

With the rise of deep learning in recent years, the problem of single image view synthesis has received considerable attention [10, 11, 12, 13]. However, these approaches are not designed to generate structured 4D light fields with small baselines. Specifically, they are not able to resolve the ambiguity in the scale of the scene's depth, often producing light fields with incorrect scale. The only exceptions are the approaches by Srinivasan et al. [14] and Cun et al. [15]. These methods first estimate the scene geometry at the new view and use it to warp the input and reconstruct the novel view image. Unfortunately, because of the per view estimation of the scene geometry, when trained on general scenes, these methods typically produce results with incorrect depth and

(a) (b)

Figure 2.4: The image on the left shows an illustration of camera array to capture light field. The image on the right shows commercial Lytro light field camera.

inconsistent parallax.

2.1.3 Overview

To address these problems, we build upon multiplane image (MPI) representation [7] that describes a scene through a series of RGBα images at fixed depths. Once this representation is estimated for a scene, novel view images can simply be reconstructed through alpha composition of the re-projected RGB images at different layers. Although MPI has been successfully used for multi-image view synthesis [7, 16, 17], it produces sub-optimal results in our application of *single* image view synthesis with *small* baselines (See Fig. 2.8). Therefore, we propose to extend this representation by allowing the disparity[1] of each layer to be inferred from the scene. Our representation, called *variable* multiplane image (VMPI), is able to better describe a scene as the disparity of the layers are set according to the distribution of the depth of the objects in the scene.

Moreover, we observe that reconstructing the visible and occluded regions from a single image requires two different processes. The content of the visible regions exist in the input image, but the occluded areas need to be hallucinated. Therefore, we propose to handle these two types of areas

[1]Disparity and depth are closely related in a structured light field and, thus, we use them interchangeably.

through two parallel convolutional neural networks (CNN). In our system, one network handles the visible areas, while the other is responsible for reconstructing the occluded areas. Specifically, the networks estimate two sets of VMPIs using a single image and its corresponding depth, obtained with a pre-trained network. These VMPIs are then used to reconstruct two novel view images which are in turn fused using a soft visibility mask to generate the final image. We train the visible network with an $L1$ loss, but use a perceptual loss to train the occluded network to hallucinate visually pleasing content in the occluded regions.

To effectively train our networks on general scenes, we introduce a new light field dataset of 2,000 unique scenes containing a variety of objects such as building, bicycle, car, and flower. Although all of our training light fields have an angular resolution of 8×8, we propose a training strategy to supervise our network for generating 15×15 light fields. Using light fields with a fixed baseline as our training data, our system learns to infer the scale of the scene's depth from the input data. We demonstrate that our approach is able to produce high-quality light fields that match the ground truth better than the state-of-the-art methods. Overall, our main contributions are:

- We present an extension of MPI scene representation and demonstrate its advantage over the original MPI for single image view synthesis with small baseline (Sec. 2.3.2).
- We propose to handle the visible and occluded regions through two parallel networks (Sec. 2.3.3).
- We introduce a new large-scale light field dataset containing over 2,000 unique scenes (Sec.2.3.4).

2.2 Related Work

One key component of our approach is the pre-trained depth prediction network. In this section, we first give an introduction of recent development on monocular depth prediction. Then, we discuss some related works in novel view synthesis. The problem of synthesizing novel views from one or more images has been studied for several decades. Here, we provide a brief overview of the previous approaches by classifying them into two general categories: multi-image view synthesis and single-image view synthesis.

2.2.1 Monocular Depth Prediction

In the early years, most of the approaches relies on hand crafted features for depth estimation [18, 19]. Recently, almost all state-of-the-art approaches rely on deep networks. Eigen et al. [20] proposed a deep architecture with two CNNs to estimate depth: one estimates coarse depth globally and the other one refines locally. Laina et al. proposed an approach using residual network and reverse Huber loss. Wang et al. proposed to use a Hierarchical CRF to refine the depth estimated by CNN. In [21], the CRF is integrated into the CNN and their architecture can be trained in a end-to-end manner. Then they further extend their approach by using attention mechanism to integrate multi-scale features [22]. Some of the approaches use a multi-task architecture, because depth estimation is highly correlated with other tasks, including semantic labels, surface normals, intrinsics, ego-motion, and so on. Because the training data for depth estimation is difficult to acquire, There are also other methods using unsupervised or semi-supervised settings.

However, almost all the approaches above are trained on specific dataset such as NYU V2 and KITTI. NYU V2 dataset contains indoor images of shelves, table, and so on, while the KITTI dataset only contains street views, thus the methods only trained on these datasets are not able to generalize well to other scenes without training on extra data. Chen et al. [23] introduced a dataset called DIW (Depth in the wild). The DIW dataset contains 495k images collected from Filckr, each of which is labeled with a pair of point about relative depth. They proposed a relative depth loss and their method generalize well to all kinds of scenes. Xian et al. [24] further extend this direction to use web stereo images. Similarly, Li et al. [25] introduced another dataset called MegeDepth, using multi-view Internet photo collections. Want et al. [3] proposed another dataset captured by the dual-lens camera, and their approach shows better performance than the MegeDepth. We used the pre-trained model from Wang et al. to predict depth from single image, because their approach works well on different natural scenes. Then we fuse the depth with the raw image input, which will be discussed in detail later.

2.2.2 Multi-Image View synthesis

View synthesis approaches typically leverage the scene geometry [26, 27, 28, 29] to utilize the content of the input views and properly synthesize the novel view image. In recent years and with the emergence of deep learning, many approaches based on convolutional neural networks (CNN) have been proposed. [30] estimate the color and depth using two CNNs and synthesize the novel view using these estimates. [31] propose to estimate depth probability volume to reconstruct an initial image and then refine it to produce the final image. Zhou et al. [7] propose to magnify the baseline of a pair of images with small baselines by introducing multiplane image (MPI) scene representation. [16] use MPI for extreme view extrapolation, while [17] and [32] utilize it for large baseline view synthesis. We build upon the MPI representation, but use a single image as the input and tackle the specific problem of synthesizing structured light field images.

Several approaches have been specifically designed to synthesize light fields from a sparse set of input views. [33] propose to reconstruct a light field from the four corner images by breaking the problem into disparity and appearance estimations. To handle scenes with non-Lambertian effects, [34] propose to increase the angular resolution of a light field by operating on epipolar-plane images (EPI). [35] propose a similar approach, but use 3D CNN and 2D strided convolutions on stacked EPIs to avoid the pre- and post-processing steps of Wu et al.'s approach. These methods require at least four images to reconstruct a light field and are not able to work on a single image.

2.2.3 Single-Image View synthesis

A large number of methods have proposed to use CNNs to estimate novel views from a single image [36, 37, 38, 10, 39, 40], but they are only applicable to a specific object or scene. [41] propose to synthesize novel views by first predicting a flow and then using it to warp the input image. [12] estimate layered depth image representation of a scene using view synthesis as a proxy task. [11] estimate a set of homography transformations and masks and use them to reconstruct the novel view. [42] use a conditional generative adversarial network to estimate novel views by breaking the scene into foreground and background layers. To generate 3D Ken Burn effects from

a single image, [13] use their estimated depth to map the input image to a point cloud and use a network to fill in the missing areas. Evain and Guillemot [43] introduce a lightweight neural network and train it on a set of stereo images. Using an input RGB-D image, [44] reconstruct a 3D photograph through layer depth image representation. [45] introduce a point cloud based view synthesis approach from a single image without needing 3D ground truth information for supervising the network during training. While these approaches work well for their intended applications, because of the scale ambiguity of the depth, they cannot properly reconstruct structured light fields with small baselines.

A couple of approaches propose to synthesize structured light field images from a single image. [14] use two sequential networks to perform disparity estimation and image refinement. The first network estimates a set of disparities at all the novel views. These disparities are then used to warp the input image to the novel view and they in turn are refined through the second network. Their approach generate each novel view independently, so that the results are inconsistent. [15] propose to estimate a flow from the input image, depth, and the position of novel view. The flow is then used to warp the input image and reconstruct the novel view image. These approaches estimate the scene geometry independently for each view and, thus, produce results with inconsistent parallax and inaccurate depth, when trained on general scenes. In contrast, we use a unified representation of the scene geometry to reconstruct all the views and are able to produce results with high quality.

Finally, the concurrent work by [46] also propose an MPI-based approach for single image novel view synthesis. However, they mostly perform large baseline view synthesis and only demonstrate their results for small baseline light fields on the dataset of Srinivasan et al. [14], which is limited to flowers. In contrast, our goal is to develop a practical system for small baseline view synthesis with the ability to handle general scenes.

2.3 Approach

Given a single image I and the position of the novel view $\mathbf{q}$ (in u and v directions), our goal is to synthesize the novel view image I_q. Our system consists of two parallel networks that are responsible for generating results in the visible and occluded areas. To do this, each network

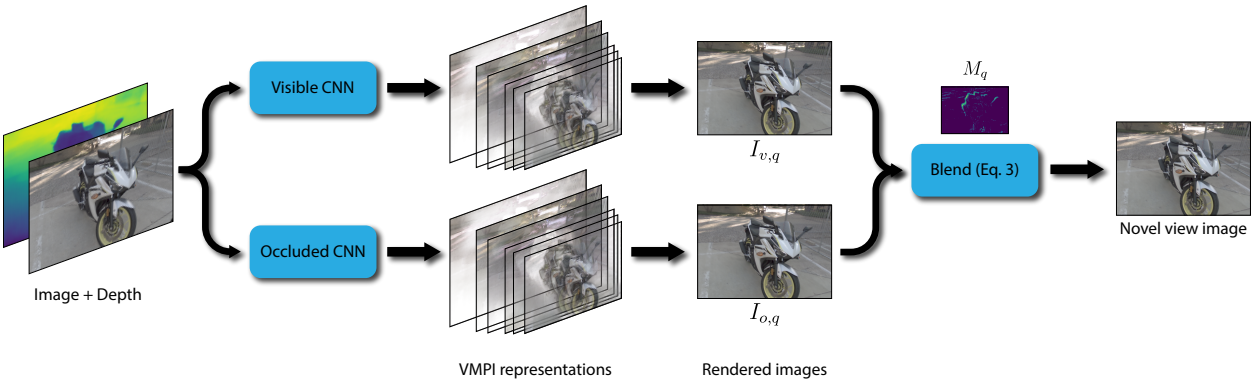


Figure 2.5: Our system consists of two CNNs for handling the visible and occluded regions. Our networks take a single image along with its corresponding depth (estimated using Wang et al.'s approach [3]) as the input and estimate our proposed VMPI scene representation. The two VMPI representations are then used to reconstruct two images, which are then combined through a soft visibility mask to generate the final image. The visibility mask is calculated using the transparency layers of the visible network's VMPI (see Eq. 2.7).

estimates a novel layered representation of the scene from the input image and its corresponding depth. These representations are then used to render two images, which are in turn combined using a soft visibility mask to generate the final image.

In the following sections, we will discuss different components of our system, including the pre-trained depth prediction network, variable multiplane image representation, and two network fusion. The overview of our approach is given in Fig. 2.5.

2.3.1 Single Image Depth Prediction

To estimate the layered scene representation, our system needs to understand the geometry of the scene. In multi-image view synthesis where several images from different views are provided as the input to the system, the depth can be easily inferred through correspondences between the input images. However, in our application, only a single image is provided to the system and, thus, the geometry needs to be inferred through contextual information. Unfortunately, it is difficult to train a network to do so using limited light field training data. Therefore, we use the pre-trained single image depth estimation network by Wang et al. [3] to estimate the depth and use it along with the image as the input to our system. Figure 2.6 shows some examples of the generated depth.

Figure 2.6: The depth generated by the pre-trained single image depth estimation network

We note that the input depth is relative as there is an inherent ambiguity in the scale of the estimated depth from a single image. However, by training our system on light fields with fixed baselines, our network can infer this scale from the input image. This is in contrast to the large baseline view synthesis approaches like the one by Niklaus et al. [13], where the estimated depth is directly used to reconstruct the novel view images. Because of this, as shown in Section 2.4, Niklaus et al.'s method produces novel view and refocused images that do not match the ground truth. To prove the effectiveness of the multimodal information for synthesizing light fields, we show the numerical evaluation with and without the single image depth prediction module in Section 2.4.

2.3.2 Variable Multiplane Image Representation

Our method builds upon multiplane image (MPI) scene representation, proposed by Zhou et al. [7]. MPI represents a scene at the input coordinate frame through a set of N fronto-parallel planes at fixed disparities $D = \{d_1, \cdots, d_N\}$. Each plane L_i consists of an RGB color image C_i and a transparency map α_i, i.e., $L_i = \{C_i, \alpha_i\}$. To reconstruct the image at novel view $\hat{I}_q$, we first translate the planes to the coordinate frame of the novel view based on their corresponding disparity as follows:

$$L_{i,q}(\mathbf{p}) = L_i(\mathbf{p} + d_i\mathbf{q}) \tag{2.2}$$

where $\mathbf{p}$ is the pixel position in x and y directions, while $\mathbf{q}$ is the novel view position in u and v directions. The final novel view image can then be reconstructed by alpha blending the color images at each layer from back to front through standard over operator [47]. This can be formally written using the following recursive function:

$$I_{i,q} = (1 - \alpha_{i,q})\, I_{i-1,q} + \alpha_{i,q}\, C_{i,q}, \quad \text{where} \quad I_{1,q} = C_{1,q}. \tag{2.3}$$

where $I_{N,q}$ is the estimated novel view image $\hat{I}_q$. Note that, the transparency of the background layer α_1 is always 1 to ensure there are not any holes in the final image.

To utilize this representation in our single image view synthesis application, we can simply use a CNN to estimate the MPI of a scene from an input image and its corresponding depth. This network can be trained by minimizing the $L1$ or $L2$ loss between the rendered and ground truth novel view images. However, such a system is not able to produce satisfactory results, as shown in Fig. 2.8. When using a large number of planes ($N = 32$), the network tends to repeat the content over multiple planes and produce a blurry rendered image. This is mainly because the $L1$ loss does not sufficiently penalize blurriness to force the network to place the content on a single plane with the correct depth. On the other hand, with fewer layers ($N = 8$) the depth planes are sufficiently separated and the network can properly place the content on the correct planes. However, in cases

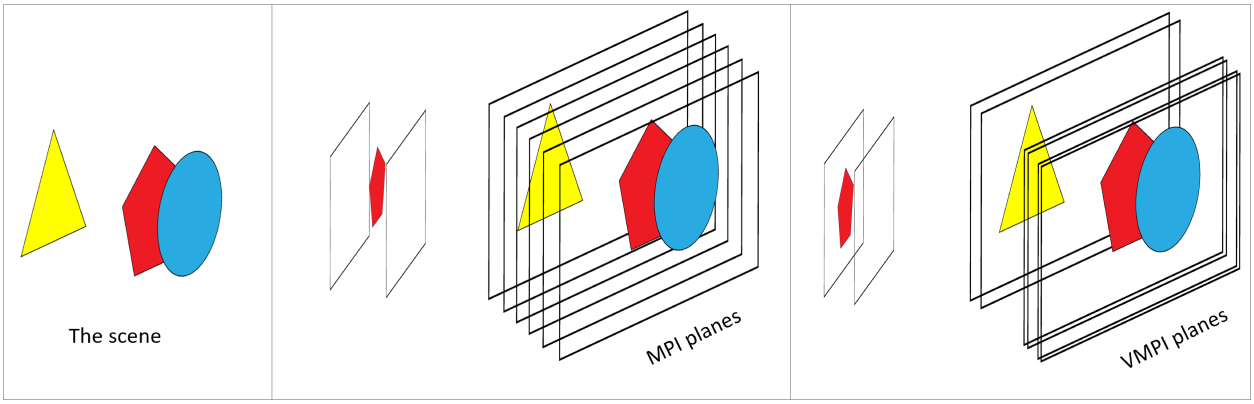


Figure 2.7: To explain the problem of MPI, we use a simple scene that contains three objects as an example. When the object locates in the middle of two plane, the network will assign this object to these two plane, and thus generating blurriness and ghosting artifacts. Therefore, we propose to extend the original MPI representation to have layers with scene-dependent disparities. The main advantage of our representation is that we can use fewer planes, but place them more accurately throughout the scene and avoid ghosting and blurring artifacts.

where an object falls in between two planes, the network places the object on multiple planes to simulate the correct depth, producing results with ghosting artifacts. Note that, as discussed at the end of this section, this problem is specific to the use of MPI in our application of *single* image view synthesis with *small* baselines.

We address this problem by extending the original MPI representation to have layers with scene-dependent disparities. In our representation, called variable multiplane image (VMPI), the disparity of each layer is inferred from the input. The main advantage of our representation is that we can use fewer planes ($N = 8$), but place them more accurately throughout the scene and avoid ghosting and blurring artifacts, as shown in Fig. 2.8. Fig. 2.7 shows the problem of MPI representation and the benefits of our VMPI representation.

To estimate the VMPI representation, we need to estimate the disparity d_i (scalar) at each layer, in addition to the color image C_i and the transparency map α_i. To do this, our network estimates a 5-channel output for each layer; three channels for RGB, one channel for α, and one channel for disparity. The final disparity of each layer is then computed by averaging the pixel values in the disparity channel of the corresponding layer. We use the estimated disparity at each layer

(instead of the fixed ones) to perform the warping in Eq. 2.2 and then blend the warped images through Eq. 2.3 to reconstruct novel view images. Note that since the estimated disparities could potentially be out of order, we first sort them before using Eq. 2.3 to blend the planes. Alternatively, we can force the network to estimate the disparities in order by introducing a penalty for out of order estimations during training.

We also experimented with estimating the disparity of each layer using a set of fully connected layers, but this strategy produced results with lower quality than our approach. This is mainly because the network needs to know the disparity of each VMPI layer to be able to appropriately estimate their RGBα values. However, in this case, the disparities are estimated by a separate fully connected branch and the main branch is unaware of the estimated disparities.

Discussion It is worth noting that existing MPI-based techniques do not face the blurriness problem, discussed in this section. This is mainly because *all* of these approaches use a perceptual VGG-based loss [7] between the rendered and ground truth novel view images to train their network. The VGG-based loss enforces the network to assign the content to one of the MPI layers, as blurriness is heavily penalized by this loss. This is even the case when using a small number of planes, as the VGG-based loss function favors slight misalignment over blurriness.

Unfortunately, when using the VGG-based loss in our application, the network tends to estimate the input image for all the novel views, i.e., $I_q = I \quad \forall q$. This is mainly because the VGG-based loss is robust to slight misalignment and favors sharpness. Since our baseline is small, the input and other novel views are relatively close. Therefore, the sharp and high-quality input image with slight misalignment produces lower error than what the network can potentially reconstruct from a single image.

2.3.3 Two-Network Fusion

The network trained with $L1$ loss to estimate VMPI representation, as discussed in the previous section, is able to produce reasonable results, but has difficulty reconstructing the object boundaries, as shown in Fig. 2.9 (Visible). In these regions, which are occluded in the input view, the CNN simply replicates the visible content, producing results with ghosting artifacts. Our main

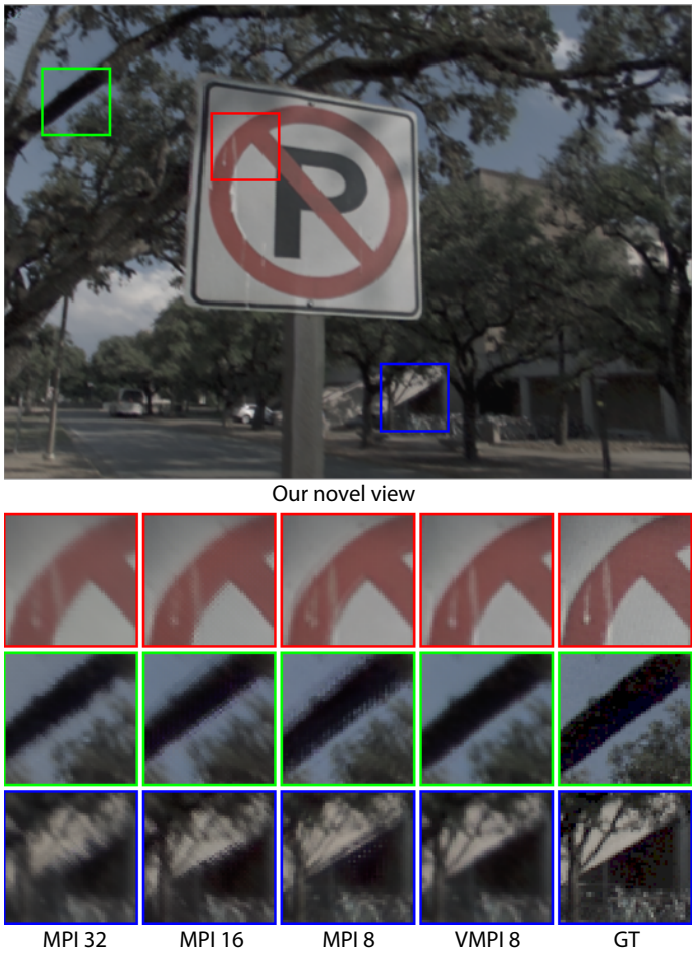


Figure 2.8: Comparison of our approach with MPI and our proposed VMPI representations. MPI with a large number of planes produces results with blurriness, while MPI with a small number of planes produces ghosting artifacts. Our VMPI representation with a small number of planes can accurately represent the scene and produce high-quality results.

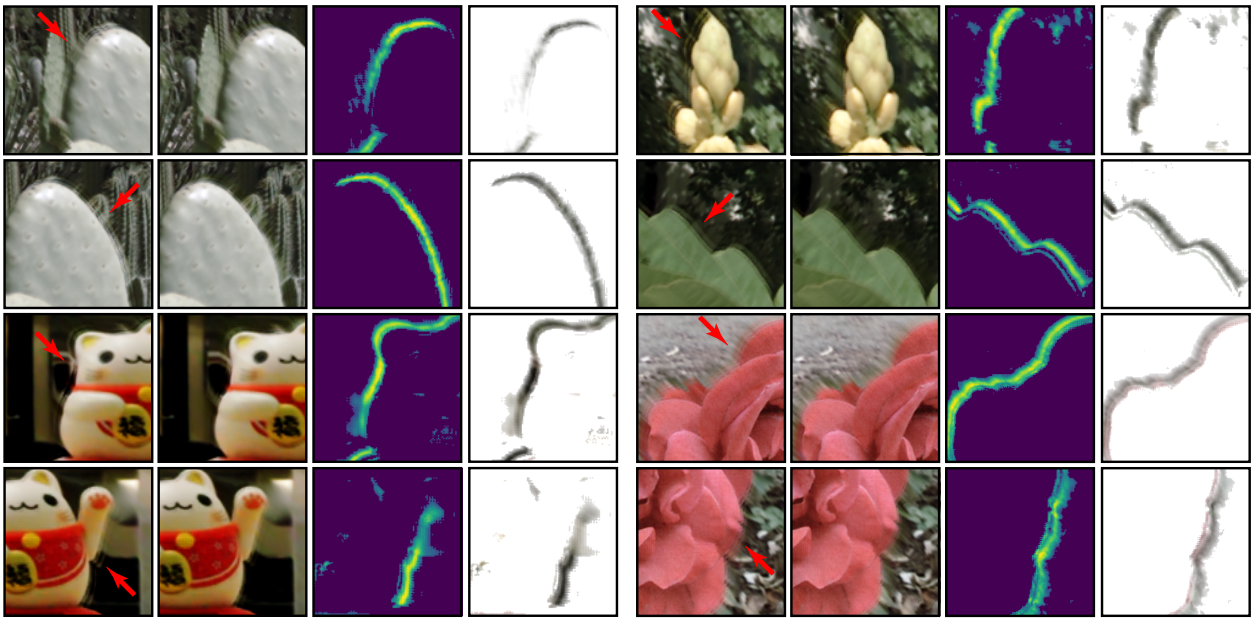


Figure 2.9: Our single visible CNN (first column) is not able to properly reconstruct the occluded regions, producing results with ghosting artifacts as indicated by the red arrows. Our system with two parallel networks (second column) significantly reduces these artifacts. For each scene, we also show the mask M_q (third column), obtained through Eq. 2.7, as well as the occluded image. Note that for better visualization, we combine the occluded image with an all white image, i.e., $M_q\mathbf{1} + (1 - M_q)I_{o,q}$.

observation is that the process of reconstructing these two areas is different. The content of the visible regions is available in the input image, but the occluded areas need to be hallucinated.

Therefore, we propose to handle these areas separately through two parallel networks (see Fig. 2.5). The visible and occluded networks first estimate two sets of VMPI representations. These VMPIs are then used to reconstruct the visible and occluded images. We then fuse reconstructed images using a soft visibility mask to generate the final novel view image. This can be formally written as:

$$\hat{I}_q = M_q \, I_{v,q} + (1 - M_q) \, I_{o,q}, \tag{2.4}$$

where $I_{v,q}$ and $I_{o,q}$ are the reconstructed images using the output of the visible and occluded networks, respectively. Moreover, M_q is a soft mask representing the visible regions, i.e., one in the

fully visible regions and zero in the completely occluded areas. This mask basically identifies the regions where the visible network has difficulty reconstructing high-quality results, since they are occluded in the input image.

We propose to compute the mask using the transparency maps of the VMPI representation, estimated by the visible network. Our high level idea is that the occluded regions are reconstructed using areas of the VMPI layers that are not visible in the input image. To compute the visibility of each VMPI layer, we first rewrite Eq. 2.3 in explicit form as follows [32]:

$$I = \sum_{i=1}^{N} \beta_i C_i \quad \text{where} \quad \beta_i = \alpha_i \prod_{j>i} (1 - \alpha_j). \tag{2.5}$$

Note that, this equation is written in the input coordinate frame, and I is the reconstructed input image using the VMPI representation. Here, β_i basically indicates how much each color image C_i is visible in the input image. To measure the visibility in the novel view, we first warp β_i to the coordinate frame of the novel view:

$$\beta_{i,q}(\mathbf{p}) = \beta_i(\mathbf{p} + d\mathbf{q}). \tag{2.6}$$

The soft visibility mask M_q can then be computed as follows:

$$M_q = \min(\sum_{i=1}^{N} \beta_{i,q}, 1), \tag{2.7}$$

where the $\min$ operation ensures that the mask is clamped to one. In the occluded regions, all the β_i's are close to zero and, consequently, M_q has a value around zero. On the other hand, in the visible areas, β_i of at least one layer is close to one. Therefore, the visibility mask in the visible areas is roughly one.

Our network architecture is similar to that of Zhou et al. [7], but with smaller number of filters in each layer (see Table 2.1). We use the same architecture for both visible and occluded networks. All the layers with the exception of the last layer are followed by a ReLU activation function and batch normalization [48]. The last layer has a tanh activation function and outputs a tensor with 40

Table 2.1: Our network architecture. Here, k is the kernel size, s is the stride, and d is the kernel dilation. Moreover, "channels" refers to the number of input and output channels at each layer and '+' indicates concatenation.

Layer	k	s	d	channels	input
conv1_1	3	1	1	4/32	input + depth
conv1_2	3	2	1	32/64	conv1_1
conv2_1	3	1	1	64/64	conv1_2
conv2_2	3	2	1	64/128	conv2_1
conv3_1	3	1	1	128/128	conv2_2
conv3_2	3	1	1	128/128	conv3_1
conv3_3	3	2	1	128/256	conv3_2
conv4_1	3	1	2	256/256	conv3_3
conv4_2	3	1	2	256/256	conv4_1
conv4_3	3	1	2	256/256	conv4_2
conv5_1	4	.5	1	512/128	conv4_3 + conv3_3
conv5_2	3	1	1	128/128	conv5_1
conv5_3	3	1	1	128/128	conv5_2
conv6_1	4	.5	1	256/64	conv5_3 + conv2_2
conv6_2	3	1	1	64/64	conv6_1
conv7_1	4	.5	1	128/64	conv6_2 + conv1_2
conv7_2	3	1	1	64/64	conv7_1
conv7_3	3	1	1	64/40	conv7_2

channels (8 VMPI layers each consisting of 5 channels for RGBα and disparity).

As discussed, we use $L1$ distance between the estimated and ground truth novel view images to train the visible network. However, we train the occluded network using the VGG-based perceptual loss [7], as implemented by Koltun and Chen [49], to produce visually pleasing results in the occluded regions.

Note that, Mildenhall et al. [17] also propose a fusion strategy, but they fuse rendered images from different views. In contrast, our approach works on a single input image and the fusion is performed between the rendered images of the same view.

2.3.4 Dataset

To effectively train our system, we collect over 2,000 light fields (See Fig. 2.11) using a Lytro Illum camera from various indoor and outdoor scenes. Our indoor scenes contain a variety of

objects such as tables, chairs, shelves, and mugs, while the outdoor scenes include objects like flowers, trees, signs, bicycles, cars, and buildings. We set the focal length as 35 mm for all the light fields, but choose the other camera parameters, such as shutter speed and ISO, manually based on the lighting condition for best quality.

We split this dataset into a set of 1950 scenes for training and 50 scenes for testing. We also use the Stanford Multiview Light Field dataset [50] for training and testing. This dataset has a set of 850 unique scenes each containing 3 to 5 light fields from different views. We use a set of 766 scenes for training containing roughly 3800 light fields. From the rest of the 84 scenes, containing around 400 light fields, we select 100 light fields for test. In summary, combining the Stanford and our dataset, we have a total of 1950 + 3800 light fields for training and 50 + 100 for testing.

The angular resolution of Lytro Illum light fields is 14×14, but we only use the center 8×8 views as the corner views are outside the aperture. To train our network, we randomly select one of the 4 corner sub-aperture images as the input. We then randomly select one of the remaining views as ground truth. With this simple strategy, we are able to supervise our system for generating light fields with angular resolution of 15×15 from only 8×8 views. Note that, this is not possible with Srinivasan et al.'s approach as their network estimates the entire 8×8 light field in a single pass. Figure 2.10 illustrates the training mechanism of our approach.

We use the approach by HaCohen et al. [51] to match the color of raw light fields to their processed version. We train our system on randomly cropped patches of size 192×192. We apply a series of data augmentations including randomly adjusting the gamma, saturation, hue, and contrast, as well as swapping the color channels to reduce the chance of overfitting.

2.3.5 Training

Training both networks by directly minimizing the loss in one stage is difficult. Therefore, we perform the training in two separate steps. In the first stage, we train the visible network by minimizing the l_1 loss between the reconstructed $I_{v,q}$ and ground truth I_q novel view images. At the end of this stage, the visible network is able to produce high-quality results in the visible areas, but produces ghosting artifacts in the occluded areas, as shown in Fig. 2.9. We use the output of

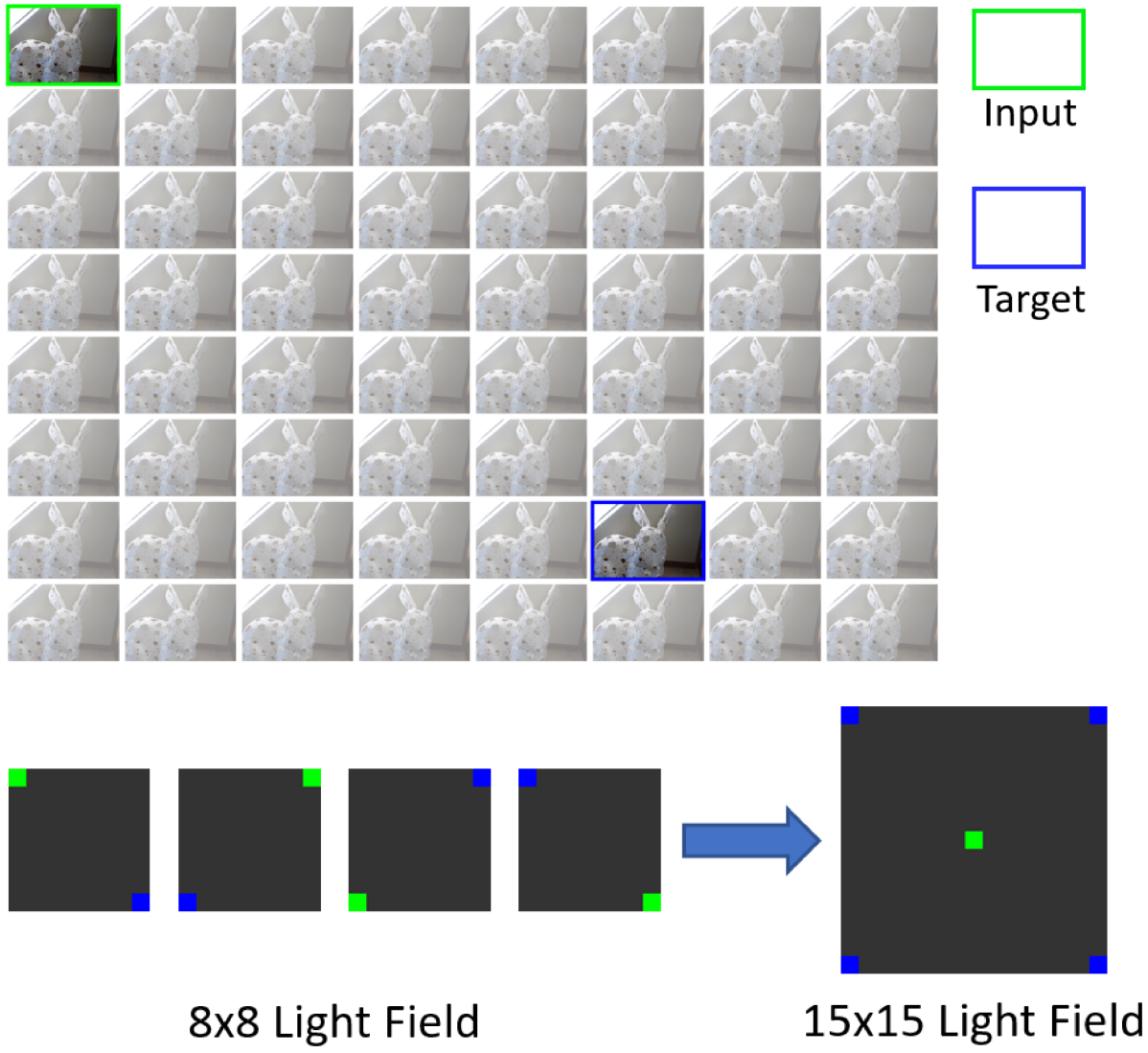


Figure 2.10: Our training mechanism allows us to generate 15×15 light fields from only 8×8 training data.

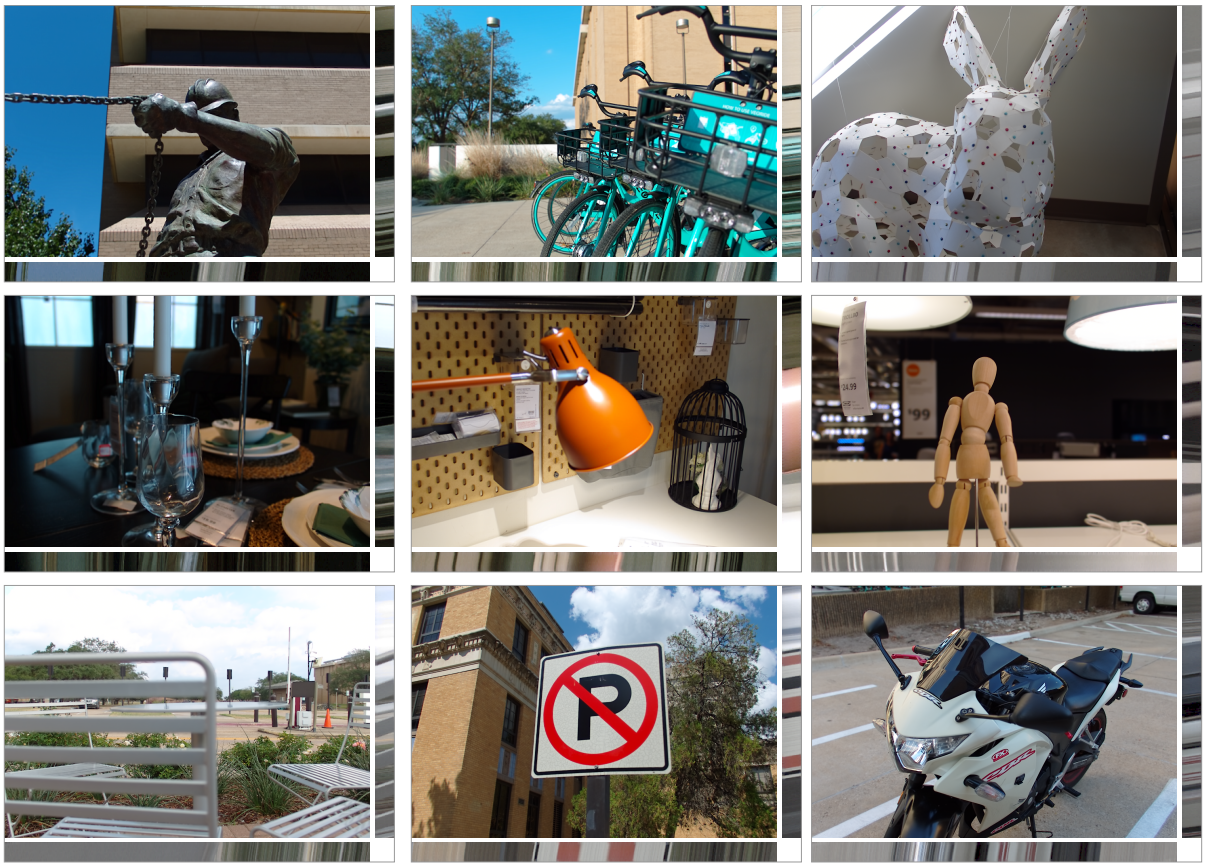


Figure 2.11: We introduce a new light field dataset containing over 2,000 unique scenes covering a wide range of objects, nine of which are shown here. We capture our dataset using a Lytro Illum camera from various locations and under different lighting conditions. The epipolar images shown on the right and below each image demonstrate the depth complexity of our scenes.

Table 2.2: Comparison of our approach against two state-of-the-art view synthesis methods, as well as a version of their approach with the depth estimated by Wang et al. [3] as the input. We evaluate synthesized 8×8 and 15×15 light fields on three datasets in terms of PSNR and SSIM (higher is better in both cases). The best results are shown in bold. Note that, all the results are generated by training our method and both versions of Srinivasan et al.'s approach on the training set of Stanford/Ours dataset.

		8×8 LF		15×15 LF	
	Algorithm	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Stanford/Ours	Srinivasan	29.92	0.9177	NA	NA
	Srinivasan + Wang	30.03	0.9215	NA	NA
	Niklaus	24.00	0.7616	20.04	0.5835
	Niklaus + Wang	23.88	0.7864	21.02	0.6456
	Ours	**30.53**	**0.9306**	**28.52**	**0.9033**
Srinivasan	Srinivasan	24.94	0.7935	NA	NA
	Srinivasan + Wang	25.75	0.8182	NA	NA
	Niklaus	25.00	0.7639	22.06	0.6532
	Niklaus + Wang	26.38	0.8188	24.32	0.7409
	Ours	**26.78**	**0.8547**	**25.56**	**0.7990**
Kalantari	Srinivasan	26.96	0.8638	NA	NA
	Srinivasan + Wang	27.60	0.8829	NA	NA
	Niklaus	24.18	0.7948	22.04	0.6886
	Niklaus + Wang	24.94	0.8219	24.26	0.7791
	Ours	**29.03**	**0.9188**	**27.36**	**0.8841**

this network to estimate the soft visibility mask, identifying the problematic occluded regions.

In the second stage, we freeze the weights of the visible network and only train the occluded network by minimizing the VGG-based perceptual loss between the final reconstructed $\hat{I}_q$ and ground truth I_q images. Note that, we compute the loss using the final fused image instead of the reconstructed image by the occluded network $I_{o,q}$. By doing so, we ensure that the synthesized content in the occluded regions blends with the reconstructed image in the visible areas in a coherent and visually pleasing manner. Fine-tuning both networks did not improve the results and, thus, we use the trained networks after the second stage to produce all the results.

2.4 Results

We implement our approach in PyTorch [52] and train our networks using Adam [53] with the default parameters (β_1 = 0.9, β_2 = 0.999, ϵ = 1e-8). We use a learning rate of 2×10^{-4} and a

batch size of 8 throughout the training. We perform both training and testing on a machine with an Nvidia GTX 1080Ti GPU with 11 GB of memory. Throughout this section, we compare our approach against the following state-of-the-art single image view synthesis approaches:

Srinivasan et al. [14] This approach uses a network to estimate the disparity at the novel view using the input image. The disparity is then used to backward warp the input image. Finally, this warped image is passed to a second network for refinement. In the original approach both networks are trained in an end-to-end fashion on a training set containing light fields of flowers. To ensure fairness, we retrain their networks on our training dataset, described in Sec. 2.3.4. We use the code provided by the authors for all the comparisons.

Srinivasan et al. [14] + Wang et al. [3] The only difference here is that in addition to the input image, we also provide the depth, estimated using the method of Wang et al. as the input to Srinivasan et al.'s disparity estimation network. By doing so, we decouple the effect of depth on the quality of the results since our approach uses the same depth as the input.

Niklaus et al. [13] This method first uses a network to estimate the depth given an input image. The depth is then used to forward warp the input image into the novel view. Note that, since the depth is computed on the input image, backward warping is not possible. The last stage of this approach is to use another network to inpaint the potential holes in the warped image. Again, we use the source code provided by the authors for all the comparisons.

Niklaus et al. [13] + Wang et al. [3] Here, we use Wang et al.'s depth instead of the depth estimated by Niklaus et al.'s depth estimation network. Again, this comparison will decouple the effect of the depth from the quality of the results.

To ensure fairness, we train both versions of Srinivasan et al.'s approach on our training set. However, since the main component of Niklaus et al.'s approach (forward warping) is non-learning we use their provided source code for comparisons.

2.4.1 Quantitative Comparison to Other Approaches

In Table 2.2, we compare our approach against the other methods for generating 8×8 and 15×15 light fields in terms of PSNR and SSIM [54]. The PSNR stands for peak signal-to-noise

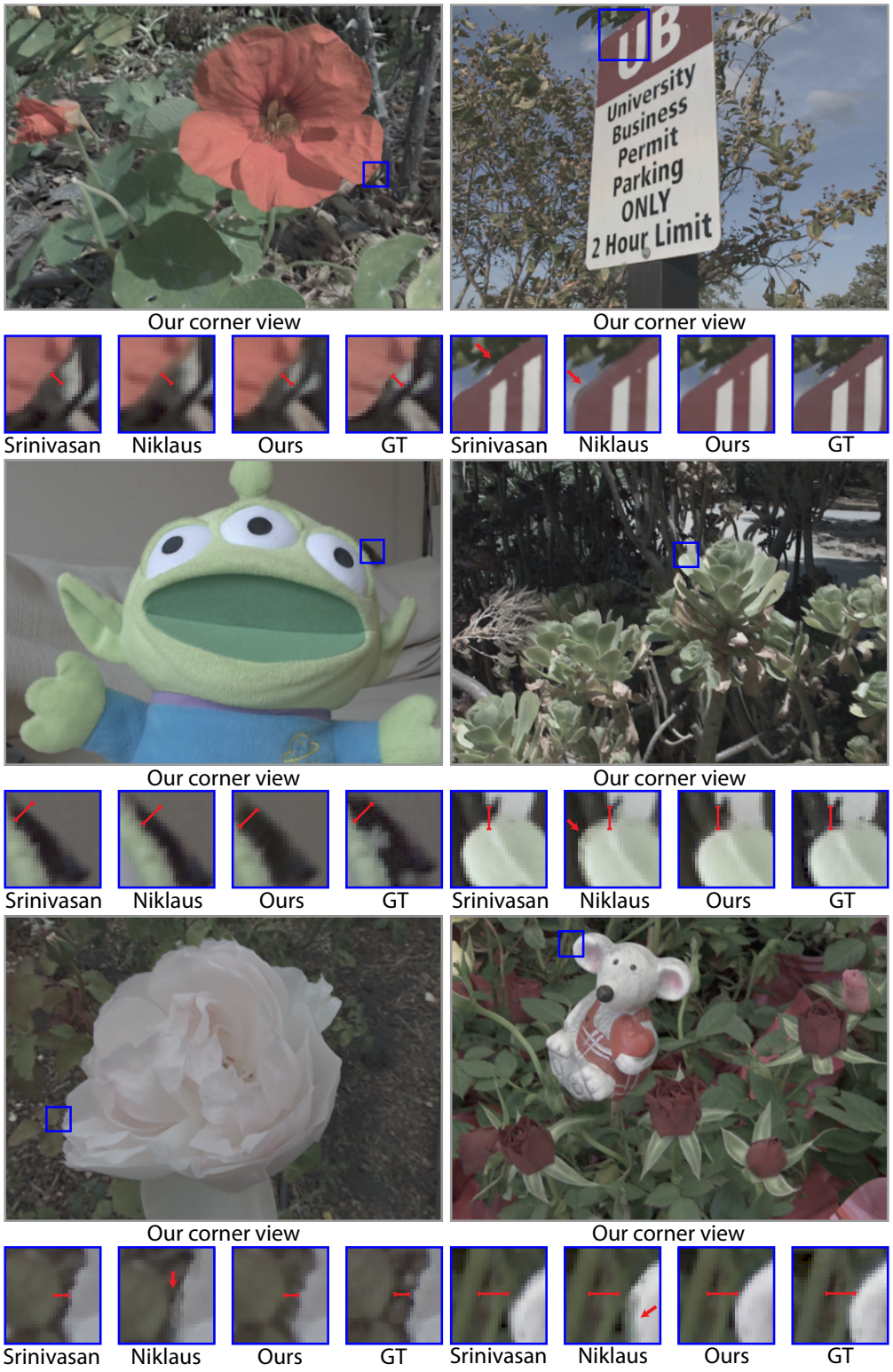


Figure 2.12: Comparison against other approaches on six challenging scenes. The red bars are of the same length and show the distance between a foreground and background object.

Table 2.3: We compare of our approach with VMPI using two-network fusion against our method with a single network as well as our technique with MPI representation with different numbers of planes. We evaluate synthesized 8×8 and 15×15 light fields in terms of PSNR and SSIM (higher is better in both cases). The best results are shown in bold.

	8×8 LF		15×15 LF	
Algorithm	PSNR↑	SSIM↑	PSNR↑	SSIM↑
MPI 8	29.57	0.9137	27.89	0.8856
MPI 16	29.39	0.9198	26.87	0.8856
MPI 32	29.28	0.9149	26.68	0.8626
VMPI 8 (Single)	30.53	0.9305	28.51	0.9033
VMPI 8 (ours)	**30.53**	**0.9306**	**28.52**	**0.9033**

Table 2.4: This table shows the result between our approach without the single image depth prediction module and our approach with the single image depth prediction module. We evaluated on the combination of the Stanford dataset and our dataset. The result shows that, our approach with depth information significantly outperforms the one without depth information.

Dataset	Algorithm	PSNR↑	SSIM↑
Stanford/Ours	Ours (without depth)	29.83	0.9125
	Ours (with depth)	35.53	0.9306

ratio, which is used to measure the quality between compressed image and the original image. Here we use it to compare the quality between the generated novel view image and the ground truth novel view image. The following equations describe how to calculate PSNR:

$$MSE = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M * N} \tag{2.8}$$

$$PSNR = 10\log_{10}(\frac{R^2}{MSE}) \tag{2.9}$$

The SSIM stands for structural similarity index. It is a perceptual metric to compare the quality between a reference image and a processed image. Here the reference image is the ground truth novel view image and we compare it with the synthesized novel view image. We refer to Wang *et al.*'s paper [54] for the process of calculating SSIM.

Note that, since the views in Srinivasan et al.'s approach [14] are hard-coded, this method can

only generate 8×8 light fields. Since the test light fields are all 8×8, we perform the numerical evaluation for 15×15 light fields by using each one of the four corner views of an 8×8 light field as the input and reconstructing the rest of the views.

We perform the comparisons on three different tests sets. Specifically, Stanford/Ours, Srinivasan et al., and Kalantari et al. [33], containing 150, 100, and 30 light fields, respectively. Compared to the other approaches, our method produces significantly better results both in terms of PSNR and SSIM. Since both approaches by Srinivasan et al. and Niklaus et al. perform better using Wang et al.'s depth in almost all cases, we use this variant of their approaches for the rest of the comparisons in the paper. However, we refer to them as **Srinivasan** and **Niklaus** for simplicity.

We also compare our approach with VMPI representation (8 planes), against ours with MPI representation using 8, 16, and 32 planes in Table 2.3. Our VMPI representation with 8 planes produces better results than all the MPI variations. The gap in quality is even larger for synthesized 15×15 light fields. See Fig. 2.8 for visual comparison of VMPI and MPI representations.

Moreover, we compare our approach against our method with just the single visible network. As can be seen, quantitatively, the differences are minor as the occluded areas are typically only small regions around occlusion boundaries. However, as shown in Fig. 2.9, our approach with the two networks significantly improves the the results qualitatively.

In Table 2.4, we show the comparison result between our approach without the single image depth prediction module and our approach with the single image depth prediction module. The goal of this ablation study is to show the effectiveness of our single image depth prediction module. We evaluated on the combination of the Stanford dataset and our dataset. The result shows that, our approach with depth information significantly outperforms the one without depth information.

2.4.2 Qualitative Comparison to Other Approaches

Here, we compare our approach against other methods on a variety of scenes. We first show comparisons on 8×8 light fields because of the restriction of Srinivasan et al.'s approach. In Fig. 2.12, we show comparison on six complex scenes captured with a Lytro Illum camera selected from the three datasets, discussed in the previous section. Here, we use the center view of a light

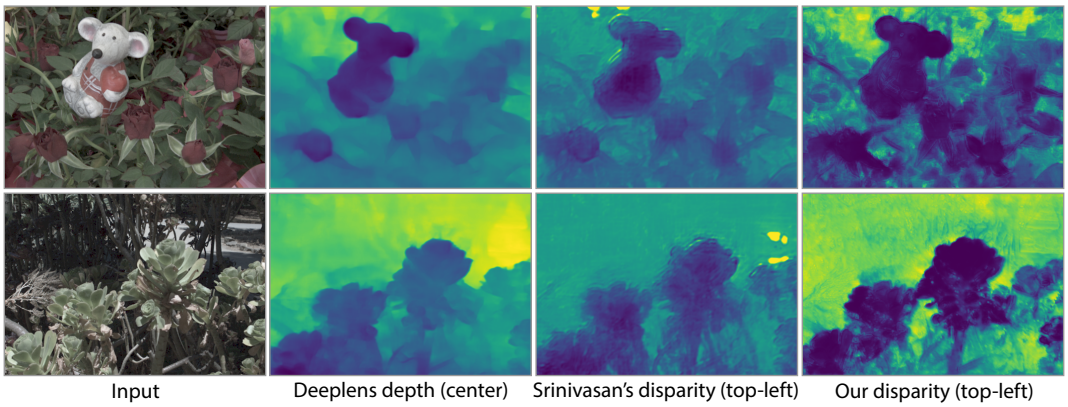


Figure 2.13: We compare our disparity at the novel view against Srinivasan et al.'s disparity and Wang et al.'s estimated depth at the center view on two scenes from Fig. 2.12.

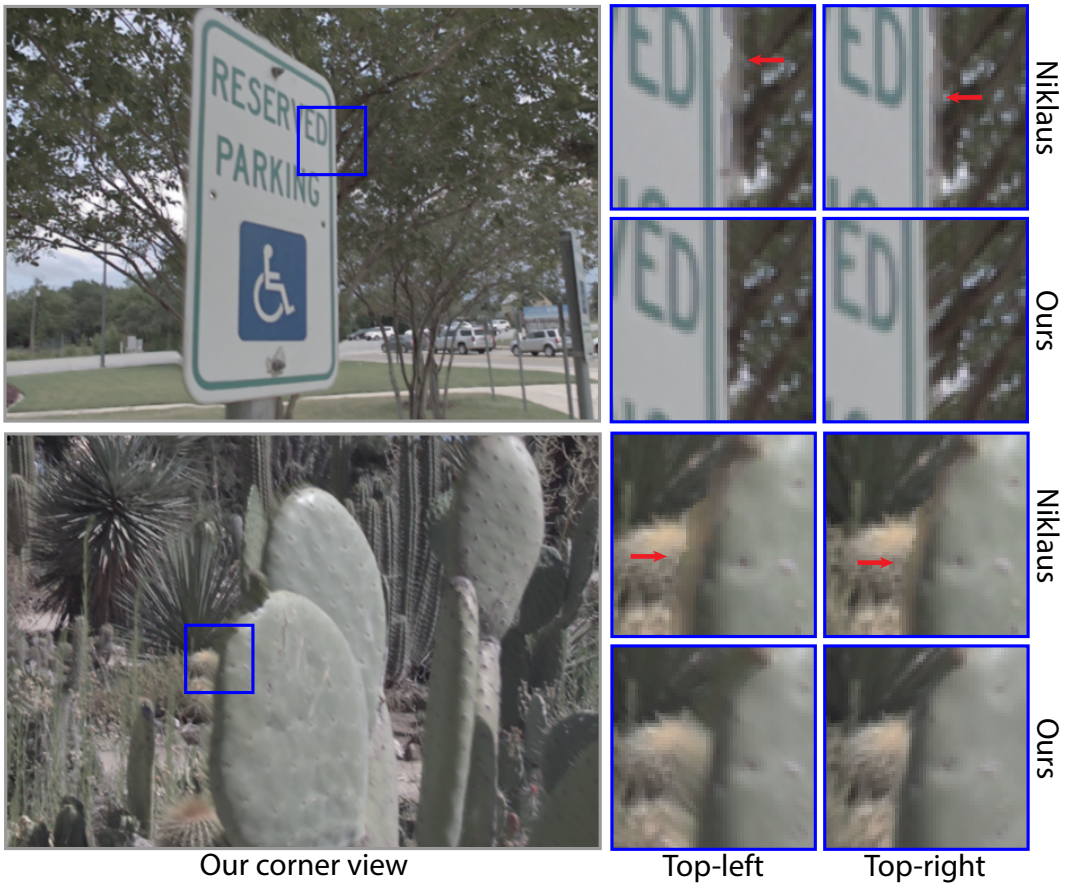


Figure 2.14: Comparison against Niklaus et al.'s method for synthesizing 15×15 light fields on two challenging scenes.

field as the input to synthesize the full light field.

Srinivasan et al.'s approach is not able to properly estimate the disparity at the novel view even with the depth as the input, creating results with incorrect parallax, as indicated with the red bars. Niklaus et al. directly use the input depth to forward warp the input image. However, this depth is relative and, thus, this approach still generates incorrect parallax in most cases, as indicated by the red bars. Moreover, it cannot properly handle the object boundaries in some cases, as shown by the arrows. Furthermore, their forward warping scheme slightly overblurs the results in some cases (see supplementary video). In contrast, our approach generates high-quality results that are reasonably close to the ground truth.

In Fig. 2.13, we compare our estimated disparity against Srinivasan et al. and Wang et al.'s depth for two scenes from Fig. 2.12 (the bottom two on the right column). As seen, our disparity computed on the VMPI layers is overall sharper than the input depth by Wang et al. [3]. This is in part the reason for Niklaus et al.'s problem around occlusion boundaries. Note that, we also generated Niklaus et al.'s results with their depth estimator, but those were of lower quality. Moreover, although both our approach and Srinivasan et al. use Wang et al.'s depth as the input in this case, our disparity is considerably better than theirs. This is mainly because their network is trained to generate a disparity map for each view by maximizing the quality of that particular view. Therefore, each novel view image during the training process, provides a supervision for only the corresponding disparity map. In contrast, we use the same representation to generate all the novel view images. So each novel view image during training, provides a supervision for the same representation.

Furthermore, we compare our approach with Niklaus et al.'s method on synthesized 15×15 light fields, in the Fig. 2.14. As seen, unlike the result of Niklaus et al., ours do not suffer from the objectionable artifacts around occlusion boundaries. We also compare our approach against the other methods on two images captured with standard cameras in Fig. 2.15. Note that in these cases, ground truth novel view images are not available. Srinivasan et al. is not able to reconstruct the top scene with proper parallax. Moreover, in their results for the scene in the bottom row, the texture

Figure 2.15: Comparison against other approaches on images captured with standard cameras. We show insets of two synthesized corner views for each image. The red lines for each method have the same length and are used to better show the distance between a foreground and a background object in the two views. Our method correctly synthesizes both the foreground and background regions without objectionable artifacts.

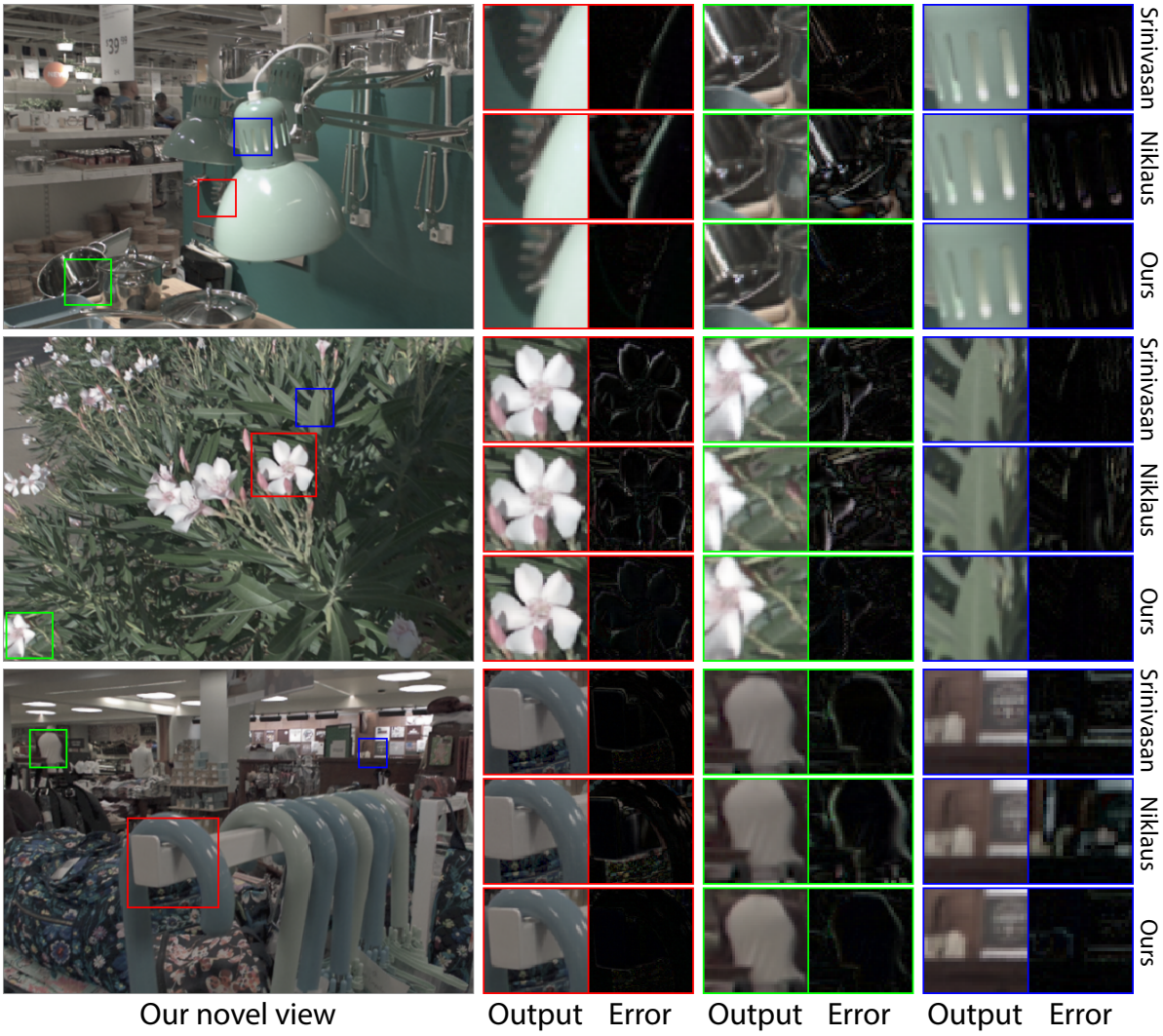


Figure 2.16: Comparison against other approaches on three challenging scenes. For each inset, we also show the absolute difference between the synthesized result and ground truth.

on the sign moves between the two views because of inaccurate disparity. Furthermore, the results of Niklaus et al.'s approach for both scenes contain ghosting artifacts around the boundaries. Our method is able to produce reasonable results in both cases.

2.4.3 Comparison to Other Methods on Refocusing

Our synthesized light fields can be used to generate refocused images, as shown in Fig. 2.18. Here, we demonstrate the ability of our approach to focus on different areas of the scene by showing two refocused images for each scene. Although Srinivasan et al. handles the foreground re-

gions, their two refocused images in the background areas have the same blurriness. This is mainly because their estimated disparity in the background areas is often inaccurate, as seen in Fig. 2.13 (see discussion in Sec. 2.4.2). Niklaus et al. produce results with incorrect defocus in both cases as they directly use the input depth which is relative and does not match the disparity of the ground truth. Our method produces considerably better results that are similar to the ground truth.

2.5 Limitations and Future Work

Single image light field reconstruction is challenging and underconstrained and, thus, our results do not perfectly match with ground truth in all cases. Despite that, our synthesized light fields are visually pleasing and contain fewer objectionable artifacts than the other approaches. Moreover, in some cases the estimated depth map by Wang et al.'s approach is inaccurate and, thus, our approach is not able to synthesized high-quality novel views. However, as shown in Fig. 2.13, our VMPI still improves upon the input depth.

Finally, our system can be used to synthesize a light field video from a standard video. This can be done by using our networks to synthesize a light field image for each frame of the video. While our synthesized light field frames are of high-quality (see Fig. 2.19), the video is not temporally coherent (see supplementary video). In the future, we would like to improve the quality of the synthesized videos, by enforcing the network to synthesize temporally coherent frames through a specially designed loss function.

2.6 Conclusion

We present a learning-based approach for synthesizing a light field from a single 2D image. We do this by introducing a new layered scene representation, called variable multiplane image (VMPI), where the disparity of each layer is inferred from the input image. Moreover, we propose to handle the visible and occluded regions separately, through the two parallel networks, to reduce ghosting artifacts in the occluded areas. In our system, each network synthesizes a novel view image and the two images are fused using a soft visibility mask to produce the final image. We introduce a light field dataset containing over 2,000 unique scenes for training. We show through

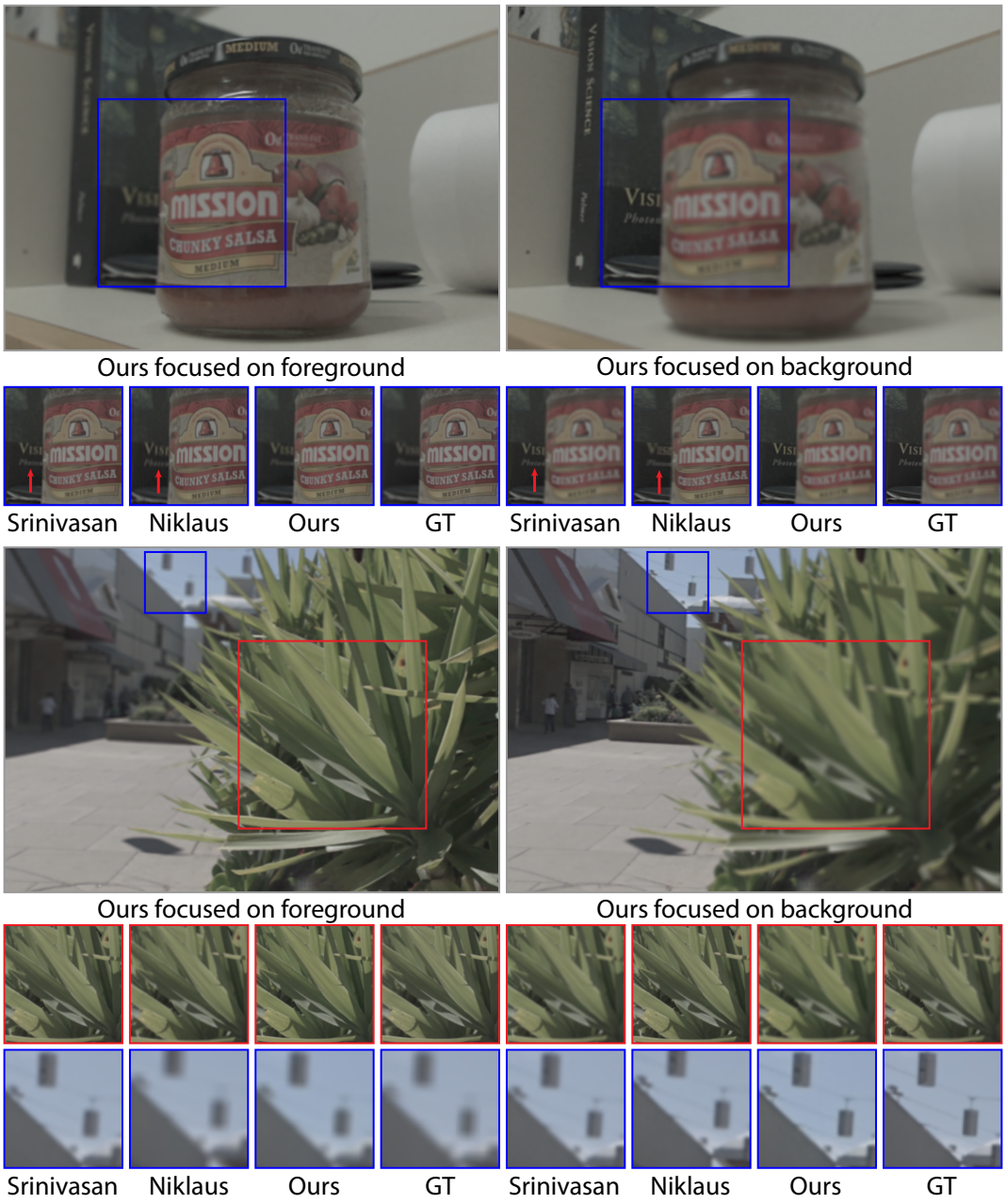


Figure 2.17: Comparison of our refocusing results against the other methods. Our method produces results that are closer to the ground truth than the other methods.

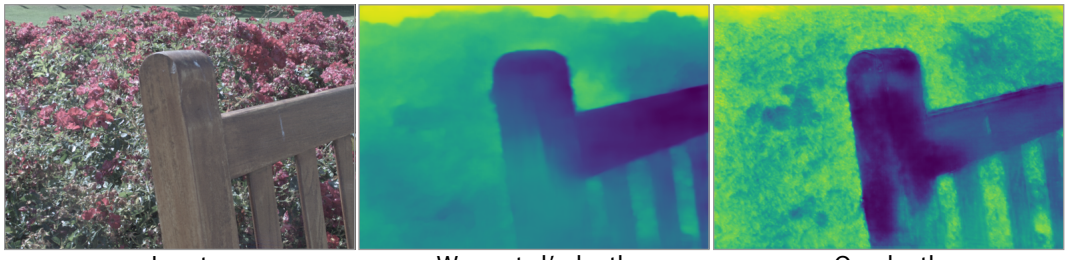


Figure 2.18: Here, the input depth by Wang et al. [3] is inaccurate. Our method improves upon this input depth, but still cannot correctly estimate the disparity of the wooden bench. See supplementary video for the synthesized views and comparison to the other methods.

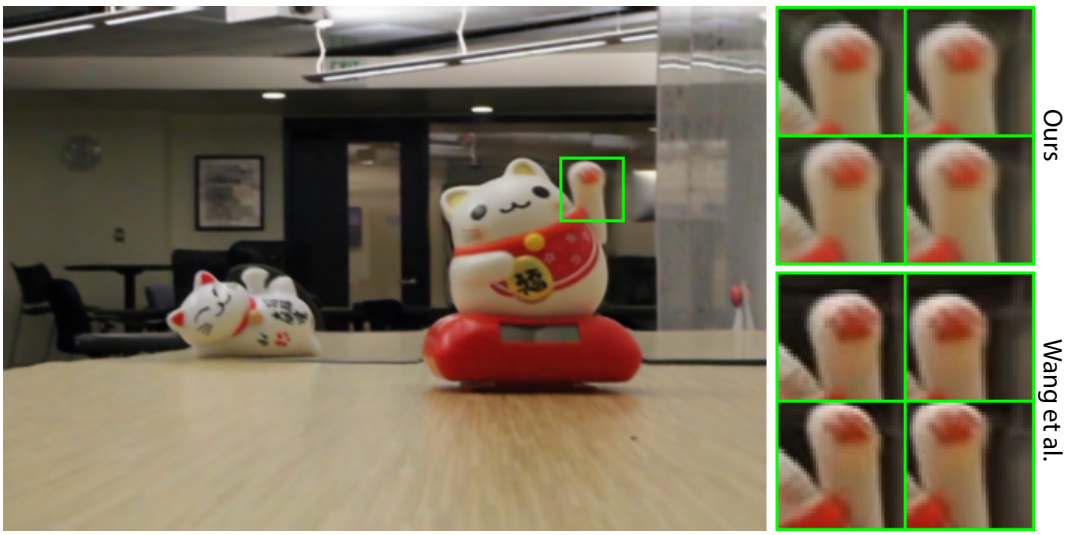


Figure 2.19: On the left, we show a corner view of our synthesized light field frame from an input 2D video. On the right, we show an inset of all our synthesized corner views and compare them against Wang et al.'s approach [1]. Note that, the approach by Wang et al. uses an additional light field video with a low frame rate as the input. Despite that, we produce light field frames with comparable quality. However, our light field video shows slight flickering as we synthesize each frame independently (see supplementary video).

extensive experiments that our approach produces better results than the other methods on view synthesis and refocusing.

3. AUDIO-VISUAL MULTIMODAL FACE RECOGNITION

With the continuing improvement in deep learning methods in recent years, face recognition performance is starting to surpass human performance. However, current state-of-the-art approaches are usually trained on high quality still images and do not work well in unconstrained video face recognition. We propose to use audio information in the video to aid in the face recognition task with mixed quality inputs. We introduce an Audio-Visual Aggregation Network (AVAN) to aggregate multiple facial and voice information to improve face recognition performance. To effectively train and evaluate our approach, we constructed an Audio-Visual Face Recognition dataset. Empirical results show that our approach significantly improves the face recognition accuracy on unconstrained videos.

3.1 Introduction

Face recognition has received considerable attention in the past decades. With the rise of Deep Convolutional Neural Network (DCNN) and large scale face recognition datasets such as MS-Celeb-1M [55] and MegaFace [56], face recognition performance has started to exceed human performance. Recently, video face recognition has attracted increasing attention. Potential applications of video face recognition include content analysis of online videos, and human identification in surveillance videos.

However, directly using the representations learned from still images does not work well on unconstrained videos. This is because unconstrained videos often contain a lot of low-quality videos, due to the bandwidth limitation of online videos and hardware constraints in surveillance videos. In addition, the view angle and head pose variance is often very large. The noisy frames due to transition and motion blur could further undermine the performance.

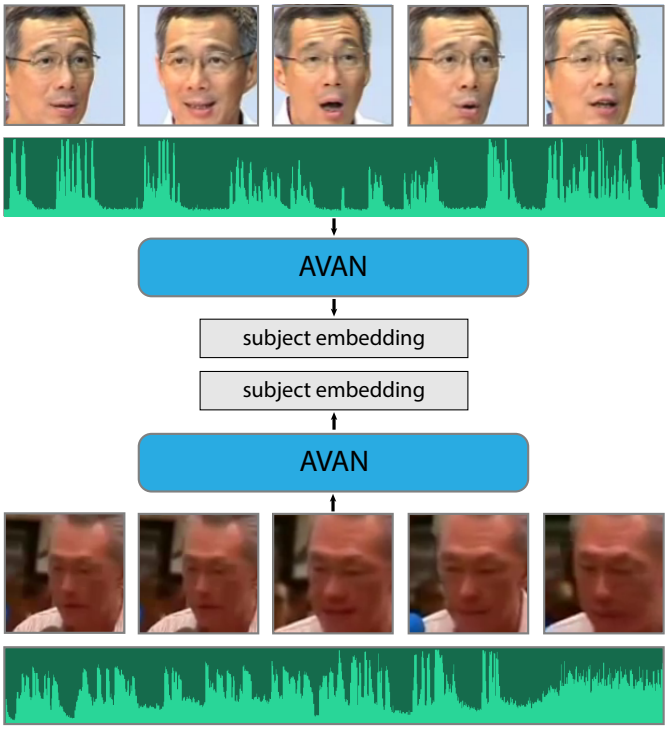


Figure 3.1: The Audio-Visual Aggregation Network (AVAN) uses both the visual information and the audio information to generate a feature embedding for the subject, to be compared to that of others.

In this paper, we propose to utilize the audio information from the video to aid in the face recognition task, as describes in figure 3.1. Our motivation is that we can use the audio information to increase the face recognition performance when the subject is speaking in the video where the video quality is low. We found that there are a large amount of such videos online. Modern surveillance cameras nowadays also support audio.

We collected and curated a large dataset of unconstrained videos where the subject is actively speaking. To the best of our knowledge, this is the first dataset of this kind. We also propose an Audio-Visual Aggregation Network (AVAN) to aggregate visual embeddings and audio embeddings to improve video face recognition performance.

We first use CNNs pre-trained on large-scale image face recognition datasets to learn the visual embedding, and CNN pre-trained on large-scale speaker recognition datasets to learn the audio embedding. The AVAN then uses LSTMs and fully connected layers to calculate attention weights

to aggregate the visual embeddings and audio embeddings.

While it is not new to use attention module to aggregate visual embeddings, we believe that we are the first to introduce the use of attention to aggregate speaker embeddings. AVAN then learns to aggregate the fused visual embedding and fused audio embedding to generate the final embedding for the subject. Experiments show that our approach improves the face recognition performance significantly.

Overall, our contributions are as follows:

- We propose an approach to aggregate visual embeddings and audio embeddings respectively, and aggregate the resulting embeddings into the final embedding.
- To effectively train and evaluate our approach, we constructed a large speech-video-matched audio-visual person recognition dataset, which is a first of its kind.
- Experiments show that our approach significantly improves the person recognition accuracy.

3.2 Related Works

In this section, we will first discuss various works related to face recognition. The face recognition works is classified by image-based face recognition, video face recognition, and audio-visual face recognition. Then, we will briefly discuss recent advancements in speaker identification.

3.2.1 Image face recognition

Recent advances in deep convolutional neural networks (CNN), such as Inception [57] or ResNet [58], bring discriminative power in face recognition and significantly improve its performance. For image classification tasks, softmax-based cross entropy loss, being widely used, is very effective in CNN's feature extraction.

However, large intra-class variations existing in face recognition reduce the effectiveness of softmax. To tackle this limitation, later, many novel losses are proposed in order to enlarge the inter-class difference and reduce intra-class fluctuation for better face recognition. L-Softmax [59] is the first one to encourage intra-class compactness and inter-class separability. FaceNet [60] and

Triplet Network [61] use the triplet loss, a variant mutated from a contrastive learning perspective [62], to impose Euclidean margin in discriminating features.

SphereFace [63] and Additive Margin [64] proposed a training loss based on trigonometric functions to learn angle dis/similarities existing in the face features. These angular-based loss functions were further improved in CosFace [65] and ArcFace [4] to enhance the CNN's ability in learning geometric attributes from face images. Besides, adding an exclusive regularization term [66] could also extend angular distance and facilitate telling apart differences in the identities.

3.2.2 Video face recognition

Major categories of methods on video face recognition task can be classified into geometry-based methods and deep-learning-based methods. Conventionally, geometry-based methods include learning on a Euclidean metric [67, 68] or learning on a Riemannian metric [69, 70].

Both of these methods try to form a mapping that projects an image from the input space to a feature space, which aims at making the image features more geometrically separable. The differences between the two are largely due to the space where the metrics are defined. In the Euclidean metric, the distance is defined across the input space itself. However, in the Riemannian metric, the distance is defined on a manifold embedded in the input space. This way, Riemannian metric can more easily represent the geometric structure inherent in the data.

Deep learning methods [71, 72, 73, 74] intend to distill an embedding by aggregating frames from a video sequence, with the knowledge acquired from trainable convolutional kernels, so that they are more discriminable. Applying attention algorithms could efficiently improve a CNN's performance on a video [75]. Other deep learning models like RNNs also have been found to be experimentally effective in video face recognition [76].

3.2.3 Audio-visual face tasks

There are several papers on combined audio-video face recognition from the conventional machine learning perspective [77, 78, 79]. However, few studies [80] explore this area using deep learning methods. One of the potential reasons might be the lack of a comprehensive audio- and

video-labeled dataset.

Audio-visual resources related to different tasks are abundant (i.e., not addressing face recognition). These include a lip reading dataset [81] that decodes text from a video frames containing a speaker's mouth movement, audio-video fusion dataset [82] that presents a deep learning approach to connect audio with the face in unstructured video to better identify multiple speakers, and speech separation [83] that proposes an audio-visual model to isolate a single speech signal from a mixture of sounds.

3.2.4 Speaker identification

The overall procedure of speaker identification includes data collection, data preprocessing, feature extraction, and classification.

The performance of speaker identification heavily dependent on the quality of the dataset. Godfrey *et al.* [84] introduced a dataset called SWITCHBOARD, which contains 2,430 data from 543 speakers. Panayotov *et al.*[85] proposed the LibriSpeech dataset that has been widely used in speaker identification. VoxCeleb is another large scale (153,516 data from 1,251 speakers) dataset proposed by Nagrani *et al.* [86]. Then, an augmented dataset VoxCeleb2 is proposed by Chung *et al.*[87]. There are also speaker identification datasets in languages other than English, for example, XiaoDu [5] and THCHS30 [88] in Chinese, CENSREC-4 [89] in Japanese, SIVA [90] in Italian, and so on. Table 3.1 summarizes the information of these datasets. Note that this is only a partial list of the speaker identification datasets. We refer to Jahangir *et al.*'s survey [91] for a more comprehensive list.

The data preprocessing includes silence removal, framing, endpoint detection, dimension reduction, and so on. Silence removal is usually the first step in data preprocessing. The speech signal is likely to contain silence at the beginning, in the middle, and at the end of the speech. Removing these silence duration can not only reduce computation complexity but also improve generated features in the following steps. Framing is to segment the continuous audio signal into discrete segments. The segmentation may contain overlap in order to generate better features. Endpoint detection is to detect when the signal ends and separate the signal from the background

Dataset	Size	Speakers	Language
Switchboard	2,430	543	EN
LibriSpeech	11,373	251	EN
VoxCeleb	153,516	1,251	EN
XiaoDu	89,227	11,558	ZH
THCHS30	13,389	40	ZH
CENSREC-4	8,440	110	JA
SIVA	>2,000	671	IT

Table 3.1: Example of some speaker identification datasets.

noise. Assume the background noise to be white noise and it follows the normal distribution:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2) \tag{3.1}$$

Then, if any audio signal satisfy $(|x-\mu|)/\sigma <= 3$, they should be background noise and we can remove them from the speech signal.

In feature extraction, spectral features are widely used as the first step. In our approach, we chose Mel Frequency Cepstral Coefficients (MFCC) as the spectral feature. Other spectral features include Linear Prediction Coefficients (LPC) and Perceptual Linear Prediction (PLP), etc. Because of recent break-through in deep learning, many state-of-the-art approaches use deep networks to extract features for speaker identification. Convolutional Neural Network (CNN) based approaches include Lukic *et al.* [92], Li *et al.* [5], Imran *et al.* [93], Dhakal *et al.* [94], and An *et al.* [95], etc. Recurrent Neural Network (RNN) based approaches include Larsson *et al.* [96] and Jung *et al.* [97]. Other deep learning based speaker identification approaches include Restricted Boltzmann Machine (Ghahabi and Hernando [98]) and Deep Audoencoder (Tirumala and Shahamiri [99]).

3.3 Existing Datasets and Their Limitations

3.3.1 Image-based face recognition

The problem of image-based face recognition has been well studied. The success of image-based face recognition is largely because of large-scale face recognition datasets. Examples include the Labeled Faces in the Wild (LFW) [100], AgeDB [101], CASIA-WebFace [102], MS-Celeb-1M [55], MegaFace [56], etc. The LFW dataset focuses on an unconstrained face recognition task. It contains 13,233 images of 5,749 people. AgeDB manually collected images in the wild annotated with accurate age. The CASIA-WebFace dataset semi-automatically collected 500K images of 10K celebrities from the Internet. The MS-Celeb-1M is one of the largest datasets that contains about 10M images from 100K people (now discontinued due to concerns). The MegaFace dataset collected 1M images of 690K subjects from Flickr.

3.3.2 Video-based face recognition

Compared with the image face recognition datasets, video face recognition datasets are difficult to collect and hence smaller in terms of the size. The YouTube Face dataset (YTF) [103] contains 3,425 videos of 1,595 subjects collected from Youtube. The Point and Shoot Face Recognition dataset (PaSC) [104] collected a variety of videos with respect to the distance to the camera, alternative sensors, frontal versus not-frontal views, and varying location. UMD Face dataset [105] consists of still images as well as video frames. In terms of the video frames, UMD Face dataset is one of the largest video face recognition dataset. However, it is currently unavailable. IJB-C dataset [106] contains 11k videos from 3,531 subjects. However, the faces in the video frames of IJB-C dataset are not labeled.

3.3.3 Multimodal person identification

The iQIYI-VID dataset [107] is a multi-modal person identification dataset. The videos in this dataset contains face, full body, and audio. However, it does not guarantee that there is an active speaker in the video, or that the subject is the main speaker.

3.3.4 Limitations of existing datasets

Our goal is to collect a video face recognition dataset where the subject is actively speaking, and use the audio information to improve the face recognition accuracy. The YTF, PaSC, and UMD Face datasets only contain video frames without the audio. The IJB-C and iQIYI-VID datasets contain audio, however, they do not label the audio such that there is an active speaker or the active speaker is the subject. The VoxCeleb dataset and our AVFR dataset are the only two dataset with correlated audio and visual information. However, the VoxCeleb dataset is not suitable to be used as low-quality video face recognition task. This is because the VoxCeleb dataset is collected in a fully automatic way, that one key step is to use CNN to detect celebrity faces. Compared with the VoxCeleb dataset, our AVFR dataset is collected in the semi-automatic manner, so that our AVFR dataset contains a lot of low-quality videos, for example, the video on column 2 row 3 in Figure 3.2. A relevant dataset is the AV Speech dataset. Instead of face recognition, it is proposed for isolating a single speech signal from a mixture of sounds. It contains around 300K video clips with labeled face, and there is an active speaker in each video clip. However, all the video clips for one subject comes from the same video, therefore, there is not enough intra-class variance for each class.

3.3.5 Proposed audio-visual dataset

To overcome these shortcomings, we collected and curated a large video face recognition dataset, the Audio-Visual Face Recognition dataset (AVFR), to address the shortcomings in the existing data. Section 3.4.1 provides details of our data collection and curation methods. Table 1 summarizes the related face recognition datasets in comparison to our AVFR dataset.

3.4 Approach

In this section, we will describe our data collection method and the audio-visual embedding with attention.

Image datasets	#Identity	#Images	Labeled face	Labeled audio
LFW	5,749	13,233	Yes	NA
AgeDB	568	16,488	Yes	NA
CASIA	10k	500k	Yes	NA
MS1MV2	85K	5.8M	Yes	NA
MegaFace	690K	1M	Yes	NA
Video datasets	**#Identity**	**#Videos**	**Labeled face**	**Labeled audio**
YTF	1,595	3,425	Yes	No
PaSC	2,802	265	Yes	No
UMD	3,107	22,075	Yes	No
iQIYI-VID	4,934	172,835	No	No
IJB-C	3,531	11,779	No	No
VoxCeleb	7,363	172,976	Yes	Yes
AVFR	1,000	5,534	Yes	Yes

Table 3.2: This table shows the comparison of different face recognition dataset. **AVFR** is the dataset we collected for this study. Note that the video-based data sets are significantly smaller in size than the image-based ones.

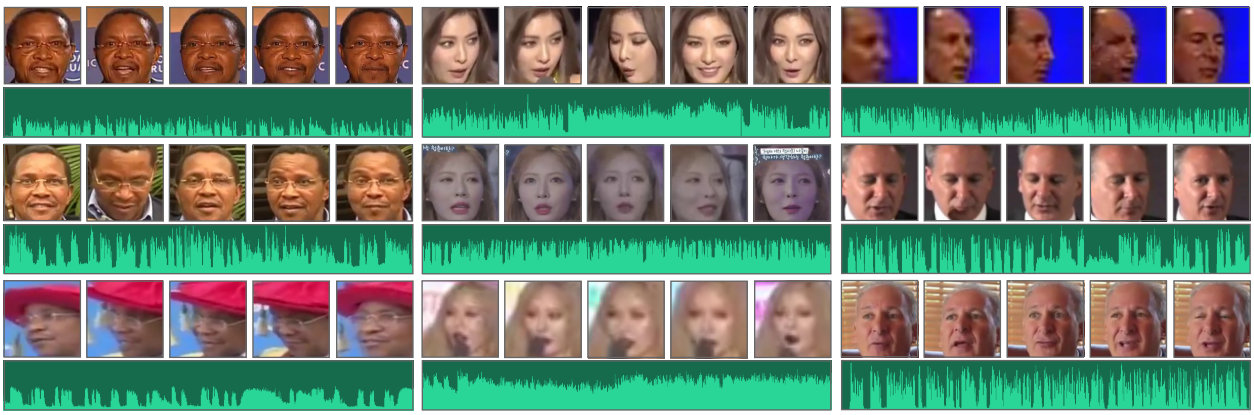


Figure 3.2: We selected some examples from our AVFR dataset. Each column shows three videos from the same subject. As it is can be seen, the intra-class variance is very large. The variance comes from lighting, head pose, age, occlusion, makeup, watermark, and so on. Our dataset also covers videos with different quality. The low quality videos are due to low resolution, or the camera being far from the subject.

3.4.1 Collection of the AVFR Dataset

Our AVFR dataset contains around 5.5K unconstrained videos of 1K subjects. In the following "subject" means the main speaker in the videos. Our goal is to collect a large video dataset, where each subject has multiple videos and the subject is the main speaker in each of the video. We started by creating a list of 2,801 celebrities, covering different ages, gender, occupations, nationality, and so on. Then, we searched and downloaded 40 videos from YouTube for each subject. In this step, we downloaded 58,270 videos of the 2,801 subjects in total.

We added the search terms "speech" and "interview" in addition to the subject's name to search for the videos, however, this did not guarantee that the results satisfy our requirements. There were a lot of videos in which the subject is not speaking or is not the main speaker, or even the subject is not in the video. Therefore, in the next step we selected the video clips where the subject is present and actively speaking.

To aid in this labelling process, we extracted MFCC with a sampling rate of 2 seconds, and generated audio embeddings for each audio clip. We used k-means to cluster the audio clips into two categories: one is the audio of the subject and the other is not. We then manually selected the audio clips that belong to the subject. In this step, we selected 5,534 videos from 1,000 subjects.

Note that even when the subject is the main speaker in the video, there still could be other subjects in the video. We also labeled the face for the subject in the video. We used a facial landmark detection approach, MTCNN [108], to detect faces from the videos. We detected faces one frame per second, and then manually labeled the face belonging to the target subject. The detected face image was then resized to 112×112. In total, we detected 230,718 face images.

To summarize, the data collection flow was as follows:

1. Collect list of celebrity names (the "subject"): 2,801

2. Search for name + ("speech" or "interview"): 58,270 videos found

3. Reject samples where the main speaker in the video clip is not the subject:

- Extract the MFCC of the audio track, and use k-means clustering to group the videos into "main speaker" vs. "not main speaker". Promptly reject large number of videos based on this.

4. Reject samples where there are people other than the subject in the video clip:

   - Use MTCNN to detect faces in the video, and reject those with multiple faces.

5. Manually curate the rest: Final dataset contains 5,534 videos of 1,000 subjects.

Figure 3.2 shows some examples from our AVFR dataset. We will release our AVFR dataset upon publication of this paper.

3.4.2 Embedding: Overview

Given a video where the target subject is actively speaking in the video, our goal is to generate a feature embedding for use in identity recognition. Our approach first generates visual feature embeddings for each video frame, and audio feature embeddings for each audio clip. Then, we use attention module to aggregate the visual embeddings and audio embeddings respectively. The attention module for the visual embeddings and audio embeddings share the same architecture, but do not share the weights. Finally, we add a fully connected layer to fuse the visual embedding and audio embedding. Figure 2.5 illustrates our overall system architecture.

3.4.3 Audio and Visual Embeddings

We use pre-trained CNNs to generate audio embeddings from audio clips and visual embeddings from video frames. These CNNs are trained on large-scale image face recognition dataset and speaker recognition dataset. These datasets are usually much larger than video face recognition dataset.

To generate visual embeddings, we first use MTCNN [108] to detect face from the video frame, and then use the similarity transformation using the facial landmarks generated by the MTCNN. The face images are then resized into 112×112. The visual embedding CNN is trained on the

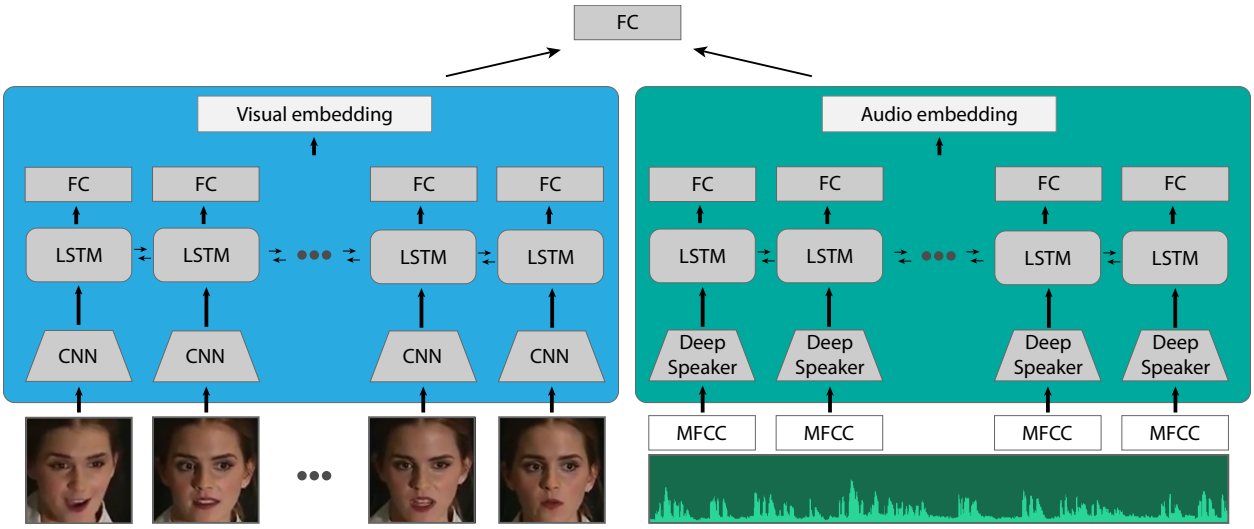


Figure 3.3: This figure shows the overall architecture of our approach. We use both the video frames and audio as input to generate the embedding for the subject. We use pre-trained CNNs to calculate visual embeddings from video frames and audio embeddings from audio clips. We then use attention module to aggregate the visual embeddings and audio embeddings. The attention module is a recurrent neural network followed by fully connected layers. The attention module of the visual embeddings and audio embeddings share the same architecture. The aggregated visual embedding and audio embedding are then fused into the final embedding.

MS1MV2 dataset with the ArcFace loss. The visual embedding network will generate visual embeddings VF = $\{vf_1, vf_2, ..., vf_N\}$, where N is the number of frames.

We use the pre-trained DeepSpeaker [5] model to generate the audio embeddings. The DeepSpeaker model is trained on LibriSpeech dataset with Triplet loss. The raw audio is firstly converted into MFCC with a sampling interval of 2 seconds. The procedure for calculating MFCC includes frame blocking, windowing, discrete Fourier transform (DFT), Mel-scale filter band, logarithm of magnitude, discrete cosine transform (DCT), and Mel-frequency Cepstral Coefficients. Fig. 3.4 illustrates the overall procedure of calculating MFCC. Then, the audio embedding network will generate audio embeddings AF = $\{af_1, af_2, ..., af_M\}$, where M equals to the length of the 2-second audio clips.

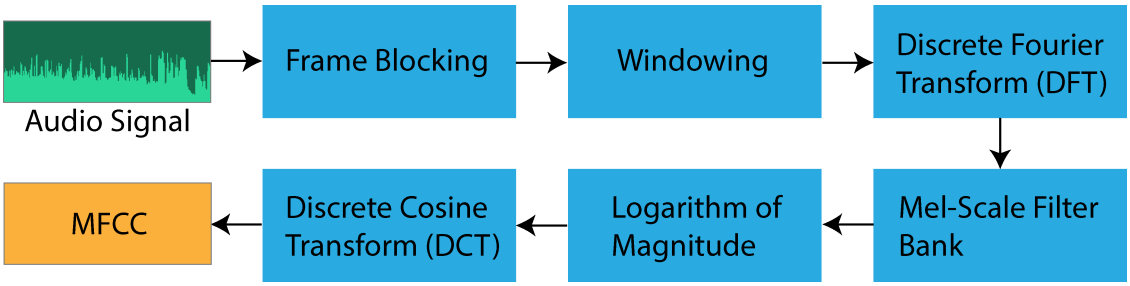


Figure 3.4: The procedure for calculating MFCC.

3.4.4 Audio-Visual Attention Module

From the previous step, we generated the visual embeddings from video frames and audio embeddings from audio clips. To calculate the final embedding from the visual embeddings and final embedding from the audio embeddings, simply using mean pooling does not work well. For the visual embeddings, there are a lot of noisy frames due to the motion blur, occlusion, undesired head pose, and so on. For the audio embeddings, there also exists noisy intervals from the audio, such as applause, acclamation, and other background noise. Even when there is no noise from the audio, we believe that there is certain interval of the audio can better represent the subject.

Therefore, we propose to use an attention module to generate weights to aggregate the visual embeddings and the audio embeddings. Similar to Gong et al.'s [76] method, we use RNN to generate the attention weights to aggregate embeddings. While it is not new to use attention mechanism to aggregate facial embeddings, to the best of our knowledge, we are the first to introduce using attention mechanism to aggregate speaker embeddings.

The architecture of the attention module is shown in figure 3.3. Both of the attention modules share the same architecture, so we use the audio embedding attention module as an example to describe the attention module. We use the feature embeddings AF = $\{af_1, af_2, ..., af_M\}$ generated from DeepSpeaker as input to the attention module. The input is processed by a bidirectional LSTM. The output from the LSTM layer is followed by fully connected layers to generate value scores V = $\{v_1, v_2, ..., v_m\}$, where v_i represents the importance value for the i_{th} audio clip. The fully connected layers for each time frame share the architecture as well as the weights. Finally,

the value scores V are normalized by a softmax layer to generate the weight w. The final audio embedding of the subject is then calculated as:

$$e_a = \sum_{i}^{M} w_i \cdot af_i. \tag{3.2}$$

Algorithm 1 describes our audio attention module. The visual attention module is very similar to the audio attention module, so it is not repeated here (the only difference is that MFCC feature is not used, and instead of DeepSpeaker, ArcFace is used for the embedding of individual images).

We use the triplet loss to train the attention module:

$$L = \sum_{i}^{N} (\|e_i^a - e_i^p\|^2 - \|e_i^a - e_i^n\|^2 + m), \tag{3.3}$$

where e_i^a is the embedding of anchor sample, e_i^p is the embedding of the positive sample, and e_i^n is the embedding of the negative sample. The parameter m represents the margin penalty, and in our experiment we set it to be 3.

After the visual embeddings and the audio embeddings are aggregated, we can concatenate the embeddings to represent the subject. For the face verification task, we further add fully connected layers to generate weights to concatenate the embeddings E:

$$E = w \cdot e_v + (1 - w) \cdot e_a. \tag{3.4}$$

Different from the attention module that takes a single subject's embeddings as input, this fully connected layer takes embeddings of a pair of subject as input, and the generated weight is shared by the two subjects. Our intuition is that, using visual embeddings only has a higher performance than using audio embeddings only, because speaker recognition is generally a more challenging task than face recognition.

However, when the image quality of the video is very low, the audio information is more reliable than the visual information. Therefore, the final fully connected layers can adjust the

weights for the visual embeddings and the audio embeddings. However, our experiment shows that the last fully connected layer does not improve the performance, which we will discuss in section 5.

Algorithm 1: Pseudo-code of audio attention

Input: The extracted audio track from the video
$mfcc$ =extract_mfcc($audio$)
$embeddings$ =DeepSpeaker($mfcc$)
$hidden$ =initialize_hidden($batch\_size$)
$lstm\_outs, hidden$ =lstm($embeddings, hidden$)
$v = []$
for $lstm\_output$ in $lstm\_outs$:
    v_i =fc($lstm\_output$)
    $v.append(v\_i)$
w =softmax(v)
$output$ =sum($w \times embeddings, dim = 1$)
Output: The aggregated audio embedding

3.5 Experiments and Results

In this section, we describe in detail our experimental setup and the results.

3.5.1 Experiments

We implemented our approach in pyTorch [52]. The DCNN for visual embeddings outputs 512-D embeddings, and the DeepSpeaker model also outputs 512-D embeddings. For the attention module, we used the single layer bi-LSTM with 128 hidden units, followed by a single fully connected layer to output the value score, and then a softmax layer. The attention module for visual embedding and audio embedding share the same architecture. To train our model, we used the Adam optimizer [53] with default parameters. We set the learning rate as 2×10^{-4} and batch size as 32.

The network is trained with an Nvidia GTX 1080Ti GPU with 11 GB of memory. We randomly split our AVFR dataset into 90% of data for the training set and 10% of data for the test set.

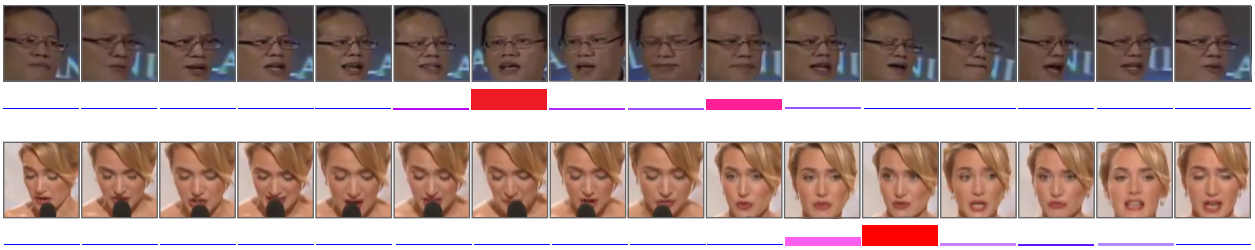


Figure 3.5: This figure shows a qualitative evaluation of our visual attention module. As it is can be seen, our model assigns high weights (tall red bars) for the high quality frames, and low weights for the low quality frames due to the undesired head pose, occlusion, etc.

3.5.2 Results

We evaluated our approach with the standard face verification task, and compared our approach with three baseline methods as described below:

- **ArcFace** [4] only uses the video frames to generate feature embedding for the video. It is a pre-trained ResNet-50 network, trained on MS1MV2 dataset with ArcFace loss. It generates a 512 dimensional embedding for each video frame. Then, we use mean fusion on all the video frames to get the final embedding for the video.

- **DeepSpeaker** [5] only uses the audio to generate feature embedding for the video. It is a pre-trained DCNN trained on LibriSpeech dataset with triplet loss. It generates a 512 dimensional embedding for each audio clip. Then, we use mean fusion on all the audio clips to get the final embedding for the video.

- **ArcFace+DeepSpeaker** uses the ArcFace to generate a 512-D embedding from the video frames, and uses the DeepSpeaker to generate a 512-D embedding from the audio. Then, it simply concatenates these two embedding into a 1024-D embedding for the video.

- **Ours-Visual (ArcFace + Attention)** uses the ArcFace to generate 512-D embedding for each video frame. Then, it uses attention module discussed in section 4 to fuse the embeddings to generate the final embedding for the video.

- **Ours-Audio (DeepSpeaker + Attention)** uses the DeepSpeaker to generate 512-D embedding for each audio clip. Then, it uses attention module discussed in section 4 to fuse the embeddings.

- **AVAN** is our approach as discussed in section 4.

We run the experiments with 10-fold cross validation and report the True Acceptance Rate (TAR) when False Acceptance Rate (FAR) equals to 0.1 and 0.01, and the best accuracy. Table 2 shows the quantitative evaluation on our approach and other baseline approaches. As it is can be seen, our approach outperform the baseline methods significantly. When using the video frames as input only, our attention module improves the accuracy from 94.89% to 95.63%. When using the audio information only, our attention module improves the accuracy from 87.47% to 88.12%. Approaches using the video frames as input only has a higher performance than using the audio as input only. When using both the video frames and the audio as input, our AVAN achieves an accuracy of 96.99% on the testing set. As for ablation study, Table 2 also shows that using both visual and audio inputs outperforms using visual inputs only and using audio inputs only.

Method	Input Type	TAR@FAR$= 0.1$	TAR@FAR$= 0.01$	Accuracy
ArcFace [4]	Image	95.53 ± 0.68	90.76 ± 0.87	94.89 ± 0.64
DeepSpeaker [5]	Audio	86.23 ± 0.77	56.29 ± 5.54	87.47 ± 0.42
ArcFace+DeepSpk.	Image+Audio	97.79 ± 0.77	93.02 ± 1.17	95.97 ± 0.43
Ours-Visual	Image	95.93 ± 0.94	92.47 ± 0.63	95.63 ± 0.47
Ours-Audio	Audio	86.83 ± 0.94	46.85 ± 5.19	88.12 ± 0.51
AVAN (Full model)	Image+Audio	$\mathbf{98.90 \pm 0.40}$	$\mathbf{95.12 \pm 0.98}$	$\mathbf{96.99 \pm 0.29}$

Table 3.3: We compare our approach with ArcFace [4] and DeepSpeaker [5] on standard face verification task. We report the best accuracy and TAR when FAR = 0.1 and 0.01. We ran all experiments for 5 times and report the average value and std. As it can be seen, our approach (AVAN) outperforms other approaches significantly.

Figure 3.5 illustrates the qualitative evaluation of our visual attention module. The height of the histogram is proportional to the weights assigned by our model. The color of the histogram

also represents the weight: red represents high weight while blue represents low weights. It can be seen that our model assigns low weights for the low-quality frames.

For example, in the first row of figure 3.5, the head pose of the subject in the beginning frames and ending frames are not facing the camera, therefore, our attention module assigns low weights to these frames. In the second row, the head pose in the beginning frames are also not desirable, and there are occlusions in these frames. Our attention module also assigns low weight to these frames.

Another finding is that the attention module usually assigns high weights for some high quality frames, instead of distributing the weights to all the high quality frames. We think it is easier for the model to learn to identify some of the highest quality frames, instead of finding all of them and assigning average weights for them.

Figure 3.6 shows a qualitative evaluation of our audio attention module. The meaning of the histogram is the same as in figure 3.5. It is noteworthy that audio is the only input at this stage. The reason that we include the video frames with the audio in the figure is to better explain the content of the audio. In the top row of figure 3.6, the video begins with background music, followed by the speech given by the subject. Our model assigns very low weights to the beginning of part of the audio.

In the second example, the video starts with the subject's speech, and then the audience start to applaud. Then the subject speak for 2 second and the audience start to laugh while the subject is speaking. The video ends with the subject's speech without noise. Our model assigns very high weight for the beginning of the audio, and then low weights for the middle of the audio, and lastly some weights for the end of the audio.

For the face verification task, we further add fully connected layers to fuse the visual embedding and audio embedding aggregated by the attention modules, as described in section 4. Our intuition is that the model can learn to adjust the weight base on the quality of the video frames to further improve the accuracy. However, our experiments show that it does not improve the performance: the model simply assign 0.5 to both the visual embedding and the audio embedding. We think that

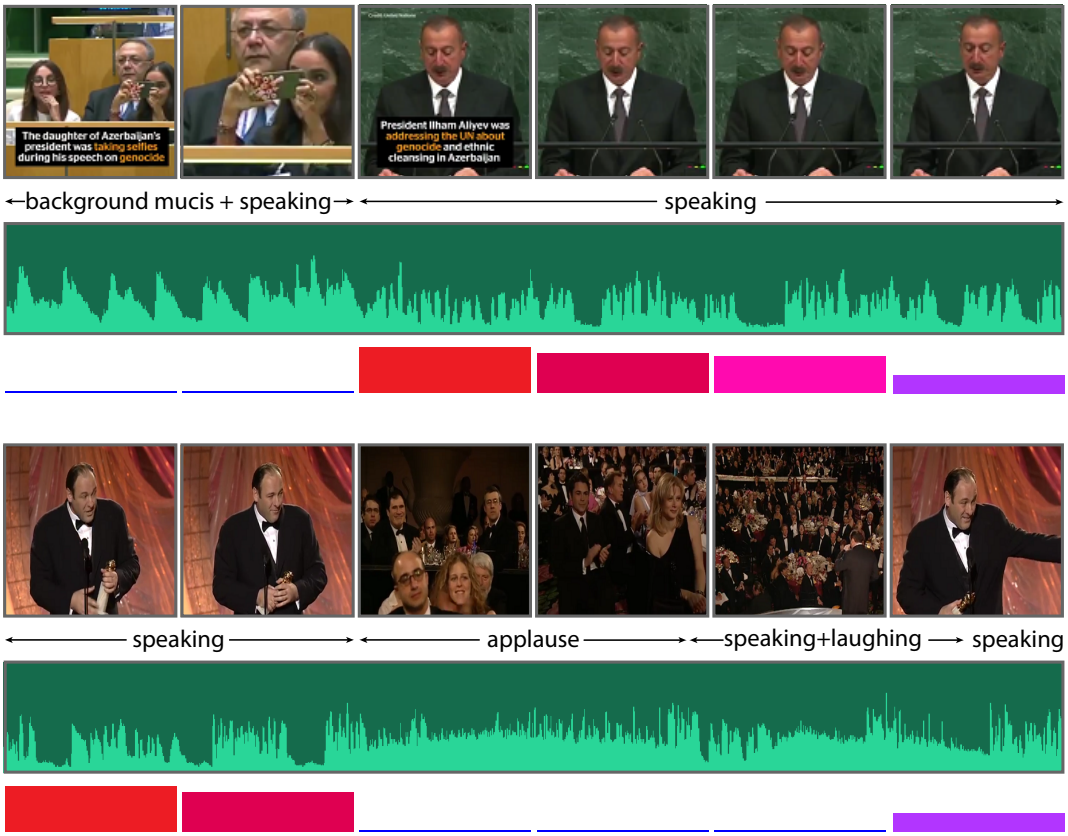


Figure 3.6: A qualitative evaluation of our audio attention module. In the first example, the video begins with background music. In the second example, there is background noise such as an applause and laughter from the audience, interleaved with the subject's speech. Our attention module assigns low weights when the quality of the audio is low.

this is because our dataset is not large enough to enable the model to learn such knowledge. We leave this as one of our future works.

3.6 Discussion and Conclusion

Our approach assumes that in the input video, the subject is the main speaker. To make our approach applicable to the general videos, one of the future works is to add another module to identify the active speaker in the video. This can be learned from the lip movement and facial expression in the video frames together with the audio.

Our AVFR dataset is the first audio-visual person recognition dataset, but it is not as large as some other video face recognition datasets such as the UMD Face dataset. Therefore, another

further work is to increase the size of the AVFR dataset.

Our approach may be ideal for applications such as few-shot learning in a noisy environments (e.g. in an automobile). Modern high-end automobiles may include driver identification based on face recognition, but the lighting condition and other environmental factors may introduce a lot of noise. The combined use of video (not just an image) and audio, as we demonstrated in this paper, can greatly improve the robustness in scenarios like this. The embeddings learned from our method can easily be applied in few-shot learning algorithms.

In summary, the main contributions of this work are two-fold: (1) we collected and curated a matched-video-speech audio-visual data set for video face recognition, which can serve as a valuable resource for the research community, and (2) we showed that attention-based fusion for each visual and audio embeddings, followed by an audio-visual fusion, again using attention, leads to superior performance compared to recognition based on vision or audio separately and without attention.

4. VISUAL, PROPRIOCEPTIVE AND KINESTHETIC MULTIMODAL CONTROL FOR TOOL CONSTRUCTION AND USE

The use of tools in animals indicates high levels of cognitive capability, and, aside from humans, is observed only in a small number of higher mammals and avian species. Constructing novel tools is an even more challenging task. Constructing and using tools requires detailed understanding of the causal physical properties of the environment, curiosity, planning, and learning through trial-and-error. In this chapter, we investigate learning to construct tools with deep reinforcement learning in physical simulated environment. We propose to solve the task using multimodal input: the RGB image of the environment, proprioception input, and kinesthesia input. The rest of this chapter is organized as follows: in section 4.1, we introduce some background knowledge of reinforcement learning, and some previous works related with learning to use tools, including one of our previous work "Emergence of Tool Use In an Articulated Limb Controlled by Evolved Neural Circuits"; we then introduce our proposed approach in section 4.2; we show our experiments and results in section 4.3, and discuss the future works and conclusion in section 4.4.

4.1 Background and Related Works

4.1.1 Reinforcement Learning

Reinforcement learning is an area in machine learning where the agent takes actions to maximize cumulative reward. There are a lot of real world problems can be solved by reinforcement learning. For example, controlling robot's body, learning autonomous driving, playing strategy games such as Go, and so on.

Two key components in reinforcement learning are the agent and the environment. The agent generates an action to interact in the environment, and the environment provides an reward and the next observation to the agent, so that the agent can generate its next action. Fig. 4.1 illustrates the loop between the agent and the environment. Generally, reinforcement learning is formalized as Markov Decision Process (MDP), including states, actions, transitions, and rewards:

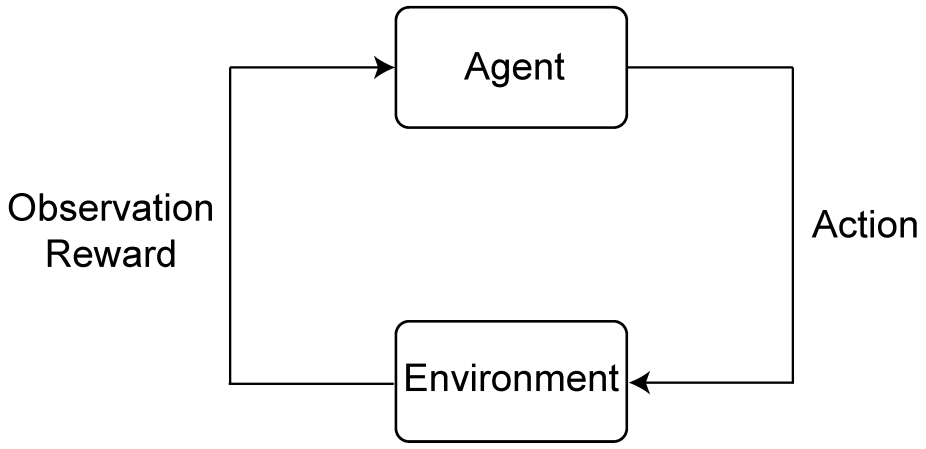


Figure 4.1: The loop between the agent and the environment in reinforcement learning

- The states S of the environment is a set $s^1, ..., s^N$ that represent the state of the environment.
- Actions A are used by the agent to interact with the environment.
- The transition probability $P(S_{t+1}|s_t, a_t)$ describes the transition between states.
- The reward function R defines the reward for the agent when reach some states.

Given an MDP, a policy π is a function that maps the state S to an action A for the agent to interact in the environment. The goal of reinforcement learning is to find the best policy π that maximize the cumulative reward of the agent. Generally, we use the discounted reward to define the cumulative reward:

$$R = E[\sum_{t=0}^{\infty} \gamma^t r_t]. \tag{4.1}$$

In order to link the policy to the optimal reward, the value function $V^\pi(s)$ under policy π is defined to describe the expected reward when being in state s and follow the policy π in the future. Similarly, we can define the value of a state-action pair $Q^\pi(s, a)$. Let π^* denote the optimal policy, then it can be proved that the optimal solution $V^*(s)$ satisfy:

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V^*(s')]. \tag{4.2}$$

Then, the optimal policy π^* can be formulated as:

$$\pi^*(s) = \arg\max_{a} \sum_{s \in S} T(s, a, s')[R(s, a, s') + \gamma V^*(s')]. \tag{4.3}$$

To solve the Markov Decision Process, there are generally two types of methods: model-based methods and model-free methods. The purpose of this section is to introduce background knowledge to understand the rest of this chapter, therefore, we mainly focus on model-free methods. In the rest of this section, we will introduce policy gradient method, actor-critic method, and Proximal Policy Optimization (PPO).

4.1.1.1 Reinforcement Learning Algorithms

There are a lot of reinforcement learning algorithms, which can be classified by tabular solution methods or approximate solution methods, model-based methods or model-free methods, designed for discrete space problems or continuous space problems or both, etc. Here we only review some reinforcement learning algorithms that are suitable to our tool construct and use task, including policy gradient, actor-critic, trust region policy optimization, and proximal policy optimization.

There are a lot of reinforcement learning algorithms try to learn the values of actions to select actions. Policy gradient algorithms try to learn the policy directly without learning the value function. Some policy gradient algorithms may still learn the value function, but the action selection is not based on the value function. As a result, policy gradient methods are ideal for the problems where the state and action space are continuous. We use $\pi(a|s, \theta)$ to represent the policy parameterized by θ, and $J(\theta)$ denotes the performance function:

$$J(\theta) = \sum_{s \in S} d^{\pi}(s) V^{\pi}(s). \tag{4.4}$$

where $d^{\pi}(s)$ is the stationary distribution under policy π. Then, policy gradient methods update

the parameters with:

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}. \tag{4.5}$$

It is not straightforward to calculate the gradient because it dependents on both the policy and the stationary distribution of the policy. The policy gradient theorem simplify the calculation because it allow us avoid calculating the partial derivatives of the stationary distribution:

$$\nabla J(\theta) = \sum_s \mu_\pi(s) \sum_a q_\pi(s,a) \nabla_\theta \pi(a|s,\theta) \tag{4.6}$$

Algorithm 2: Monte-Carlo Policy-Gradient method (REINFORCE)

Input: parameterized policy, action space, state space, parameter space
Generate an episode followed by the policy
For each step in the episode:
  G = return from step t
  $\theta = \theta + \alpha \gamma^t G \nabla_\theta \ln \pi(A_t|S_t,\theta)$

Algorithm 2 describes the Monte-Carlo Policy-Gradient (REINFORCE) method, which is one of the policy gradient methods.

To reduce the variance of the vanilla policy gradient method, Actor-Critic method learns a value function and use it to assist the policy update. There are two roles in Actor-Critic method: the actor, whose role is to maintain and update the policy to select an action; the critic, whose role is to estimate a value function with respect to the policy. Algorithm 3 describes the Actor-Critic algorithm [109].

Schulman *et al.* [110] propose TRPO (Trust region policy optimization) and demonstrate its robust performance. TRPO updates parameters by taking large steps, and at the same time it enforces a KL divergence constraint to avoid performance collapse. PPO (Proximal Policy optimization) [6] has the same intuition as TRPO while being less complicate by using a surrogate objective. Both

Algorithm 3: Actor-Critic algorithm

Input: policy parameter θ,state-value parameter w, action space, state space

$Repeat$:

$S = environment.reset()$

$I = 1, done = False$

$while\ !done$:

$A \sim \pi$

$S', R, done = environment.step(A)$

$\delta = R + \gamma\hat{v}(S', w) - \hat{v}(S, w)$

$w = w + \beta\delta\nabla_w\hat{v}(S, w)$

$\theta = \theta + \alpha I\delta\nabla_\theta log\pi(A|S, \theta)$

$I = \gamma I$

$S = S'$

End

TRPO and PPO are on-policy algorithm and can be used in discrete space problems and continuous space problems. Algorithm 4 describes the proximal policy optimization algorithm. Because of the state-of-the-art performance, we use PPO to train the agent, which will be elaborated in the following section.

Algorithm 4: Proximal Policy Optimization (PPO) algorithm

Input: policy parameter θ, value function parameter ϕ

For k in 0,1,2...:

Collect trajectories D_k using policy π

Collect rewards R_t

Compute advantage estimates $\hat{A}_t$

$\theta_{k+1} = \arg\max_\theta \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} \min(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)))$

$\phi_{k+1} = \arg\min_\phi \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} (V_\phi(s_t) - \hat{R}_t)^2$

End

4.1.1.2 Hierarchical Reinforcement Learning

Many real-world tasks are very challenging, requiring multiple steps of decision making, while the reward are sparse. Learning to construct and use tools is one of these challenging problems.

Hierarchical reinforcement learning decompose the task into different level of abstraction. In the past decade, the realm of hierarchical reinforcement learning has advanced profoundly. Here we briefly discuss recent advancement in hierarchical reinforcement learning.

One of the most well-known formulation of hierarchical reinforcement is option introduced by Sutton *et al.* [111]. An option consists of a triplet $< I, \pi, \beta >$, where π is a policy, β is a terminal condition, and I is an input set $I \subseteq S$. An option can be seen as an action at the higher level. The implementation of options is simple and it can increase the convergence speed of training. Parr and Russell [112] proposed an approach called hierarchical abstract machines (HAM). Similar to option, HAM investigates the theory of Semi-Markov Decision Process (SMDP). However, HAM are complex to implement and thus does not have many applications. MAXQ value function decomposition is another hierarchical reinforcement learning algorithm proposed by Dietterich [113]. MAXQ decomposes the value function into two components: one is the total expected reward of the action-state pair, and the other is the total reward expected parent task. Compared to option, MAXQ directly decompose the task into sub-tasks and the policy for the sub-tasks can be reused. However, the learned hierarchical policy is not guaranteed to be optimal. Dayan and Hinton [114] proposed a framework called Feudal Reinforcement Learning. In feudal reinforcement learning, there is a manager to assign subtask to lower-level workers, and the works learn to execute these subtasks. The manager observes the state of the environment at a higher level, while the workers focus on a subgoal and observe in the original state space.

In our tool construction and use task, the agent need to learn a sequence of steps to achieve the goal, for example, pick up the tools, construct the tool parts into a novel tool, use the novel tool to achieve the goal. Each of the steps can be seen as a subtask. Therefore, we use the feudal reinforcement learning approach to train the agent. Here we continue discuss some recent approaches under the feudal reinforcement framework.

Kulkarni *et al.* [115] propose to use two Deep Q Network (DQN) [116] for the manager and worker. Learning policies in multiple levels at the same time leads to non-stationary training. To address this issue, Nachum *et al.* [117] proposed Hierarchical Reinforcement Learning with

Off-policy Correction (HIRO). Levy *et al.* [118] proposed Hierarchical Actor-Critic (HAC) and claimed that HAC is more efficient than HIRO when learning multiple levels of policies. Vezhnevets *et al.* [119] proposed a end-to-end differentiable model called FeUdal Networks (FuNs) that use dilated LSTM for the manager. Their experiments showed that FuNs improved long-term credit assignment in ATARI games and some memory tasks.

In addition to hierarchical reinforcement learning, we use curiosity reward to encourage exploration. As one of the related works, Burda *et al.* [120] propose random network distillation to add an exploration bonus, where the bonus is the error of a neural network predicting the future states of the environment. They show that random network distillation achieves outstanding performance on hard exploration Atari games.

4.1.2 Tool construction and use

Tool construction ans use require but not limited to the following abilities and skills: sensorimotor skills, logic and planning abilities, curiosity, problem posing and problem solving skills, etc. Apart from human, only a small number of animals exhibit the tool use behavior. For example, Capuchin monkey can use stone as a tool to crack nuts [121], and crows can use sticks to reach targets beyond their reach [122]. Constructing novel tools is a more challenging task than using a simple tool. Previously, it is believed that such behavior belongs to human only. However, recently scientists observed that crows can construct tools from multiple objects [123]. In this study, crows need to use stick to reach food far from their reach, but there are no long sticks to achieve this goal. The crows then figured out to build a long tool with provided combinable elements to reach the food without any help.

4.1.2.1 Emergence of Tool Use In an Articulated Limb Controlled by Evolved Neural Circuits

One of our previous work [124] investigates how the capability to use tools can spontaneously emerge in an evolved neural circuit controller for a two degree-of-freedom articulated limb. The goals of the study were to find minimal and indirect fitness criteria for the emergence of tool use, and to analyze the properties of evolved neural circuits that permit tool use.

The task was to reach a target object that may or may not be within reach of the limb, with or without the tool. We evolved the neural circuit controller by gradually augmenting the network topology, illustrated on the left side of figure 4.2. The neural evolution algorithm is called "NeuroEvolution of Augmenting Topologies" (NEAT) proposed by Stanley and Miikkulainen [125]). NEAT is based on the idea of complexification, where behavioral complexity is achieved through complexifying neural circuit topology. NEAT has been used in various learning environment such as games and has been shown to be effective in evolving complex, non-trivial behavior [126, 127]. In NEAT, the chromosome encodes neurons and their connections separately, as well as the connection weights. Neurons and connections can be added or removed to change the network topology, thus the chromosome has a variable length. Mating of chromosomes with different network topology is achieved through the use of a quantity called "innovation number" that indicates the evolutionary origin of a particular gene. Innovation numbers allow only compatible genes to mate (i.e., genes that have the same ancestral origin). Another unique mechanism of NEAT is "speciation" that helps freshly changed topology to be preserved despite the initial plunge in fitness. The rest of the algorithm is similar to other neuroevolution or evolutionary algorithms: (1) instantiation of phenotype from genotype, (2) testing in the task environment, (3) calculating fitness, (4) selection and reproduction.

Our aim in selecting the fitness criteria was to find measures that do not directly dictate the evolved tool-use behavior (i.e., minimal and indirect). We used three basic fitness criteria and tested different combinations of them to evolve the neural circuit controller for the limb. For each controller, by the end of 100 trials, the following quantities were calculated: (1) D: distance between the end effector and the target; (2) S: steps taken to reach the target; and (3) T: tool pick-up frequency. Different combinations of the fitness criteria elements were tested: D, S, DS, DT, ST, and DST. Multiplication ("$\times$") was used when combining the fitness criteria elements (e.g., DS means D $\times$ S). In addition, neural circuits evolved with and without recurrent connections were compared ($SST = S^2T$ and $SST_{noRecur} = S^2T_{noRecur}$).

Each two degree-of-freedom articulated limb was given 100 trials to perform the target reaching

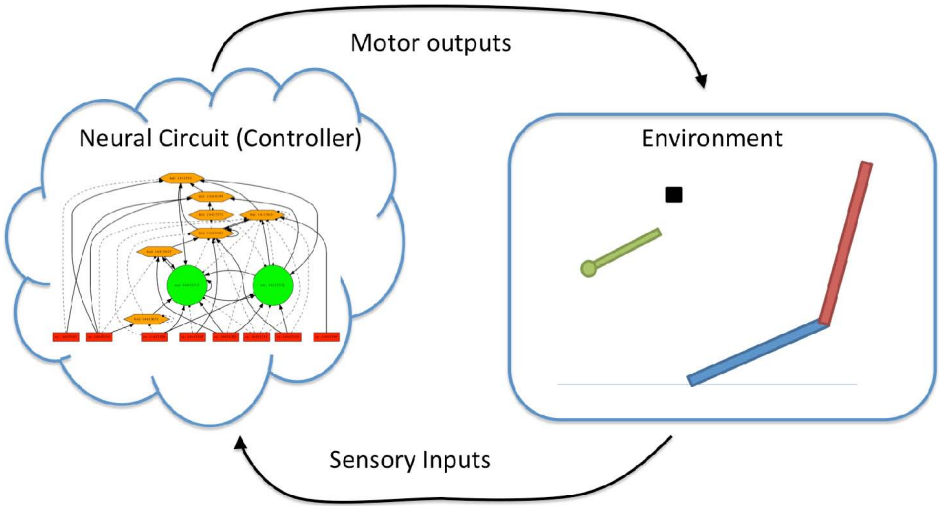
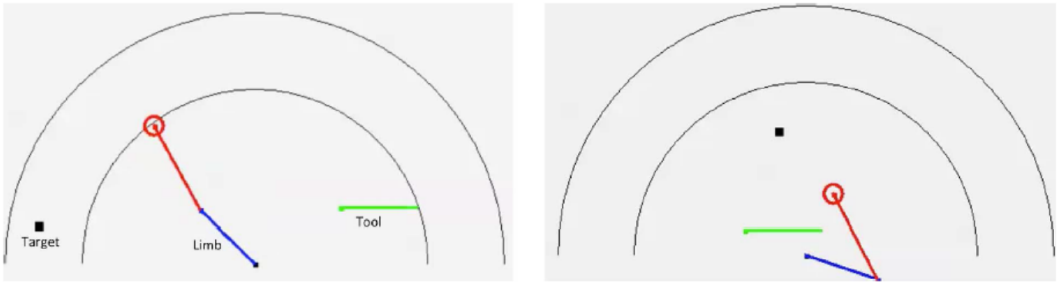


Figure 4.2: The framework of our previous work "Emergence of tool use in an articulated limb controlled by evolved neural circuit"

task. In each trial, the limb was allowed to move up to 500 time steps (maximum time step). For each trial, the target and the tool were placed at random locations. The distance of the target could be either within the reach of the limb or not (the tool was always within the reach of the limb; see Fig. 4.3). In the latter case, the limb must use the tool to reach the target (otherwise it will fail). If the limb successfully reached the target, the current trial ended and the number of time steps taken was recorded. If the agent failed to reach the target, the distance from the end effector to the target was recorded. We evolved the the neural circuit (the limb controller) for 120 generations with a population size of 100. To evaluate the performance, after the 120 generations were done, we picked the best limb controller and repeated 10 times a set of 100 test trials (total 1,000 trials). With this, we compared the performance and the trends throughout the generations, as well as the network topology characteristics. For all experiments, the within-reach and out-of-reach conditions were balanced (50% each).

Fig. 4.4 shows the network topology of evolved circuits under different fitness criteria. The one that included tool pickup frequency (T) in its fitness showed a much simpler network while maintaining the same level of performance. In Fig. 4.5, limb movements are shown as time-lapse images. Time is encoded as the pixel intensity, where darker means more recent state. The three



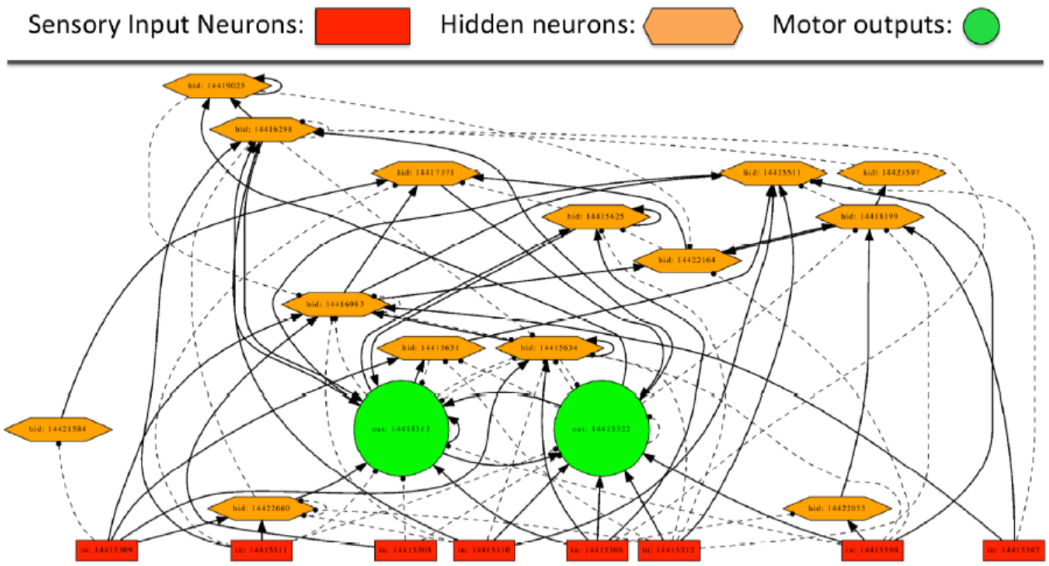
(a) Target out of reach (a) Target within reach

Figure 4.3: Example task conditions. The environment consists of two degree-of-freedom articulated limb, a target object, and a tool. all on a 2D plane. Note that the tool's initial location are variable. The two half-circles indicate the original reach (inner) and the reach with tool use (outer).

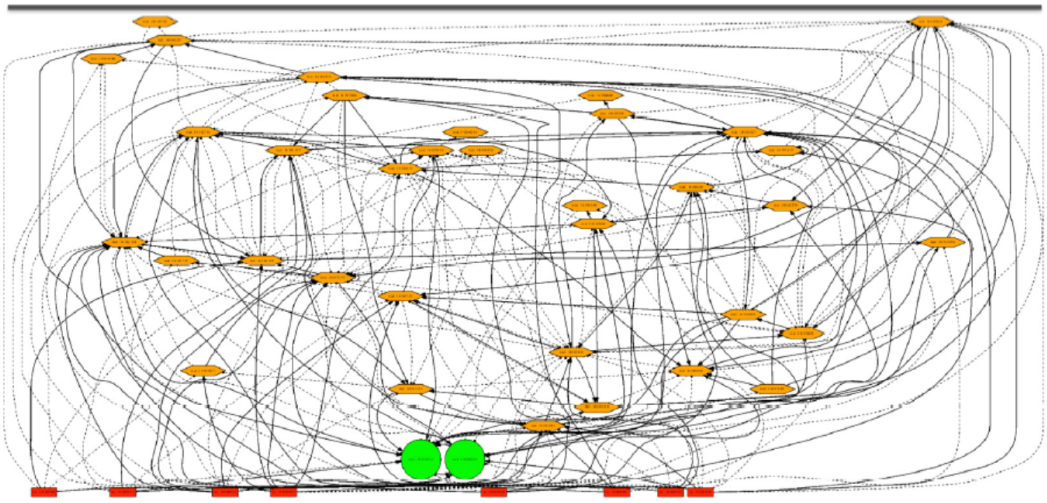
images in the top row show examples of successful movements. Starting position is marked as "s", tool pick up event as "t", and ending position as "e". Target location is marked as black square. The limbs (blue and red lines) in the three images first move towards the tool (green line with a circle at the handle) to pick it up, and then the extended limb moves towards the target. The trajectories of the end effectors are shown as black solid curves. In the bottom row, the three images show examples of unsuccessful movements, where the three movements of the limb could not reach the target.

When measuring the performance of the different fitness criteria, comparing the raw fitness scores is not fair because different fitness criteria produce different fitness score scales. Therefore, we used success rate to compare the performance between different criteria. Success rates are the percentage of successful target reach events during the 1,000 test trials. For each fitness criterion, we selected the best neural circuit from the last generation and then ran 1,000 trials. The entire process of evolution and testing was repeated four times (n = 4).

In general, DT, ST,and DST showed superior success rates than their counterparts that did not use the tool pickup frequency criterion T (D, S, and DS). Within the same group (within DT, ST, DST or within D, S, DS), the success rates were similar (t-test, n = 4, $p > 0.1$ in all cases). However, differences across the two groups were statistically significant (t-test, n = 4, $p < 0.01$ for all cases). The fitness criteria only using D (distance) or S (number of steps) did not



(a) Evolved neural circuit with fitness = S²T



(b) Evolved neural circuit with fitness = DS

Figure 4.4: Examples of evolved neural circuits. (a) Evolved neural circuit with fitness criterion S^2T (speed square and tool pick-up frequency). (b) Evolved neural circuit with fitness criterion DS (distance and speed).

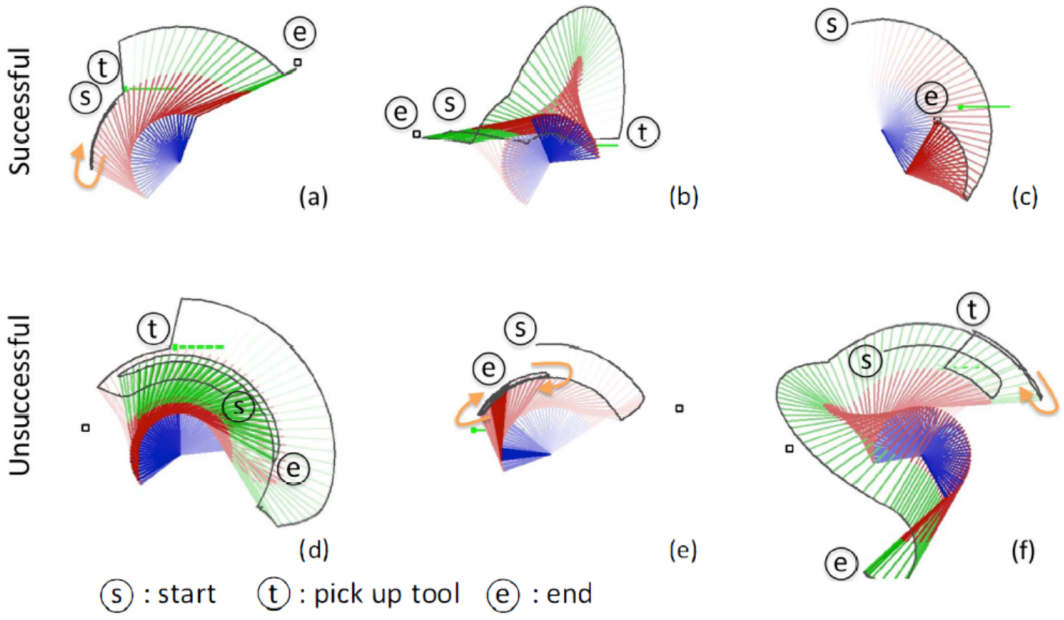


Figure 4.5: Time-lapse images of representative limb movements. The three images in the top row show examples of successful movements. In the movements of (a) and (b), the limbs (blue and red lines) first move towards the tool (green horizontal line) to pick it up, and then move towards the target. In the movement of (c), the limb moves directly toward the target without picking up the tool. The trajectories of the end effectors are shown as black curves, and parts of the trajectories that may be hard to keep track of are annotated with orange arrow. The three images in the bottom row sow examples of unsuccessful movements, where the limbs failed to reach the target.

show good performance (around 50%), neither did the combination DS. However, combining with T (tool pick-up frequency) boosted the performance (DT, ST, around 70%). This indicates that giving reward for simply picking up the tool, even without any further implications given, could lead to the emergence of adaptive tool use. Related observation was reported in an experimental study. Amant and Wood [128] noted that in the experiment in Visalberghi and Limongelli [129], capuchin monkeys frequently achieved tasks requiring tool-using ability, by quickly trying many inappropriate strategies with tools (e.g., pick up something and wield it without a clear plan). In most cases, the fitness criterion S (number of steps) seems to be useful (since high S fitness also means target has been reached, early). The limbs evolved with S (such as ST and DST) reached the target directly without picking up the tool, if the distance to the target is within the limb's reach. For the fitness criteria without S but with T (such as DT), we observed that the limb mostly tried to get to the tool first even when the target is within reach. As mentioned above, about 50% of the trials were within reach condition and the other 50note that even in cases where the success rates are around 50%, the tool was used to reach out-of-reach targets. That means performance of about 50was used successfully to reach out-of-reach targets, and cases where within-reach targets were not reached. For example, the success rates for the beyond-reach trials were D (17.78%), S (31.66%), and DS (28.65%) for the fitness conditions lacking T; and DT (93.70%), ST(90.47%), DST(94.07%) for the conditions with T. In sum, D/S/DS conditions did utilize the tool, although less frequently than DT/ST/DST.

We compared neural circuits evolved with and without recurrent connections. The S^2T fitness criterion was used to evolve the neural circuit. In the first case recurrent connections were allowed (S^2T), and in the second case only feedforward connections were allowed ($S^2T_{noRecur}$). First, we compared the success rates of the best chromosomes for the two conditions (S^2T: mean = 80.68%; $S^2T_{noRecur}$: mean = 52.58%). The difference was statistically significant (t-test, n=4, $p < 0.01$). Next, we compared the maximum fitness scores, average fitness scores, the number of total neurons, and the total degrees (the total number of connections) throughout the generations. In Fig. 4.6, top-left and top-right panels show the maximum fitness scores and the average fit-

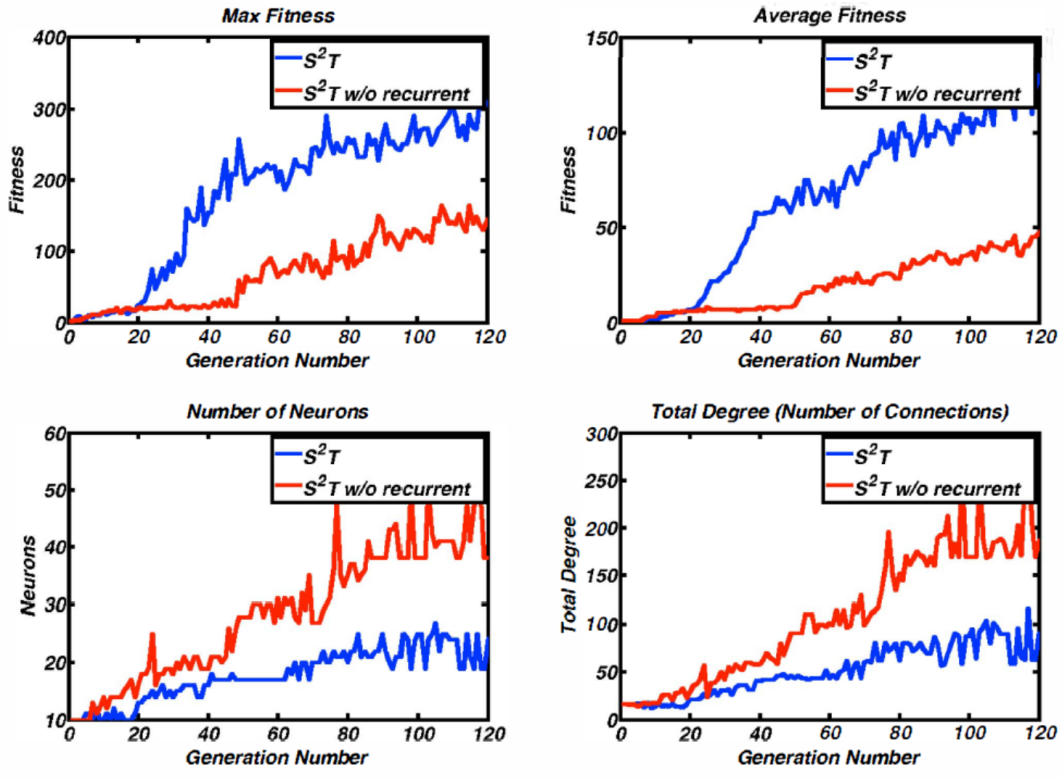


Figure 4.6: Fitness over generations and network size, with or without recurrent connections. S^2T with (blue line) and without recurrent connections (red line) are compared. Top-left: max fitness scores through the generations. Top-right: average fitness scores. Bottom-left: number of total neurons. Bottom-right: number of total degrees (number of connections) in the neural circuits through the generations.

ness scores throughout the generations when evolving the neural circuits. The one with recurrent connections shows better performances throughout. However, interestingly, the number of neurons and connections in the neural circuits shows the opposite trend, as shown in the bottom-left and the bottom-right panel. The neural circuit without recurrent connections has more neurons and more connections even though the controller (neural circuit) shows significantly lower performance than the one with recurrent connections. This observation emphasizes the importance of recurrent connections in evolving neural circuit controllers with continuous action and feedback loop. Recurrent connections in neural networks can provide memory of the past. Also, recurrent connections can provide the ability to predict future internal dynamics, which is an important ability for neural circuit controllers, especially for challenging control tasks [130].

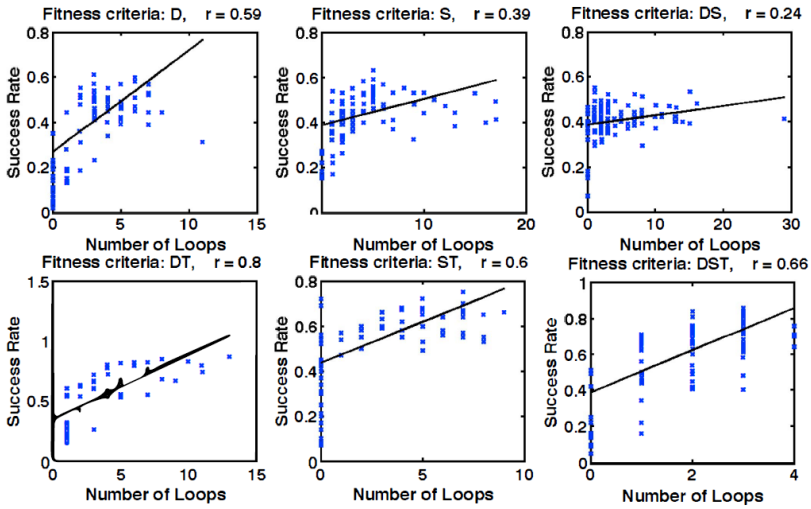


Figure 4.7: Correlations between the number of recurrent connections (self loop + 2 hops + 3 hops) and success rates. The data points (blue "x"), the linear regression lines, and correlation coefficients (r) are plotted, for the six fitness criteria. For each fitness criteria, the 120 best neural circuits for each generation (generation 1 to 120) were analyzed to see the correlation between the number of recurrent connections and success rates (average of 1,000 trials each). The total number of cycles (number of loops) is found to be positively correlated with success rate. Note that the correlation is even higher for those that included T (tool pick-up frequency) in the fitness.

Also, we analyzed the correlation between (1) the number of recurrent connections and (2) success rates under different fitness conditions (Fig. 4.7). Self loop, 2-hop, and 3-hop loops were counted as recurrent connections. The six fitness criteria compared earlier were analyzed (D, S, DS, DT, ST, and DST). For each fitness criterion, the best neural circuits from each generation (total of 120 per condition, generation 1 to generation 120) were analyzed to see the correlation between the number of recurrent connections and success rates (same as before). The correlation plots show a trend that the fitness criteria with comparably high correlation coefficients (such as DT, ST, and DST, the ones with T [tool pick-up frequency]) have high success rates in Fig. 4.8. This indicates that the recurrent connections affect positively the performance of the evolved neural circuits (or vice versa). However, the number of recurrent connections themselves may not be critical for the performance as long as recurrence is allowed. Instead, how they are connected could be more important: recurrent connections that are able to predict future internal dynamics could result in neural circuits with higher success rates.

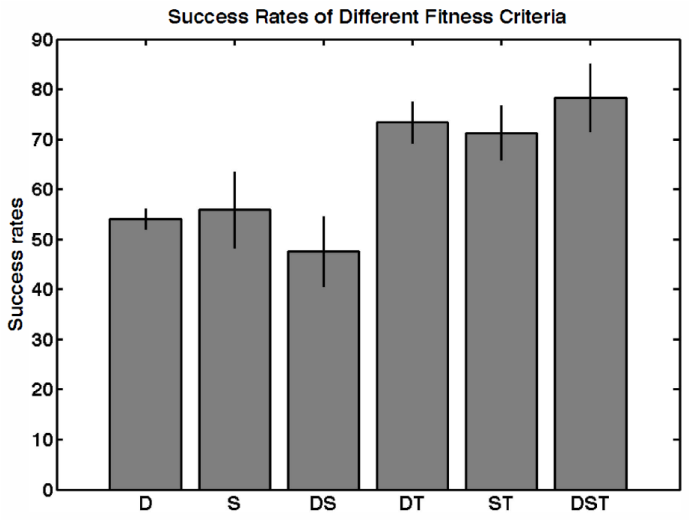
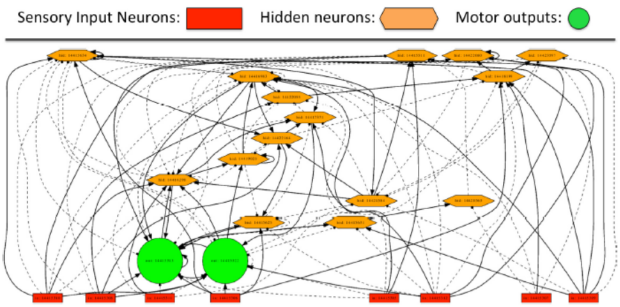


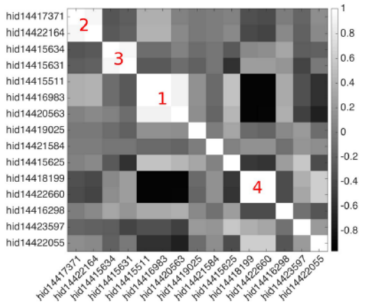
Figure 4.8: Average success rates of controUers based on different fitness criteria. Four sets of experiments (evolution of the neural circuit controller followed by testing) were ran for each fitness criterion. In general, fitness criteria that included T (tool pick-up frequency) did significantly better than those that did not include T (t-test, n = 4, $P < 0.01$). Results with S2T were similar (data not reported here). Note that the target was placed within the limb's reach only 50% of the time during the trials. See text for details.

Fig. 4.9 shows time series values of the fifteen hidden neurons of an evolved neural circuit. In the time series correlation matrix (Fig. 4.9 (b)), we can observe four groups of neurons that are highly correlated. The four clusters are cluster 1 (hid14415511, hid14416983, hid14420563), cluster 2 (hid14417371, hid14422164), cluster 3 (hid14415634, hid14415631), and cluster 4 (hid14418199, hid 14422660). As we can see in Fig. 4.9 (c) and (d), the groups of neurons respond (change activity) to the behavioral events (1), (2), (3), and (4). At the event (1), the limb changes the movement towards the tool, and at (3), it changes towards the target. Tool pickup happens at the event (2), and the limb reaches the target at (4). Cluster 3 responds to (1) and (3), while cluster 4 to (2) and (4). There are single neurons that also exhibit interesting behavior, e.g., hid14416298 (third row from the bottom of Fig. 4.9 d showing rapid oscillation between events (2) and (3)).

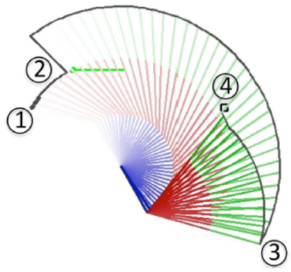
In summary, our results indicate that simple, indirect criteria such as distance to target, number of steps to reach target, and if the tool was fetched was enough to give rise to tool using behavior. Quantification of recurrent loops in the neural circuit topology showed that better controllers have



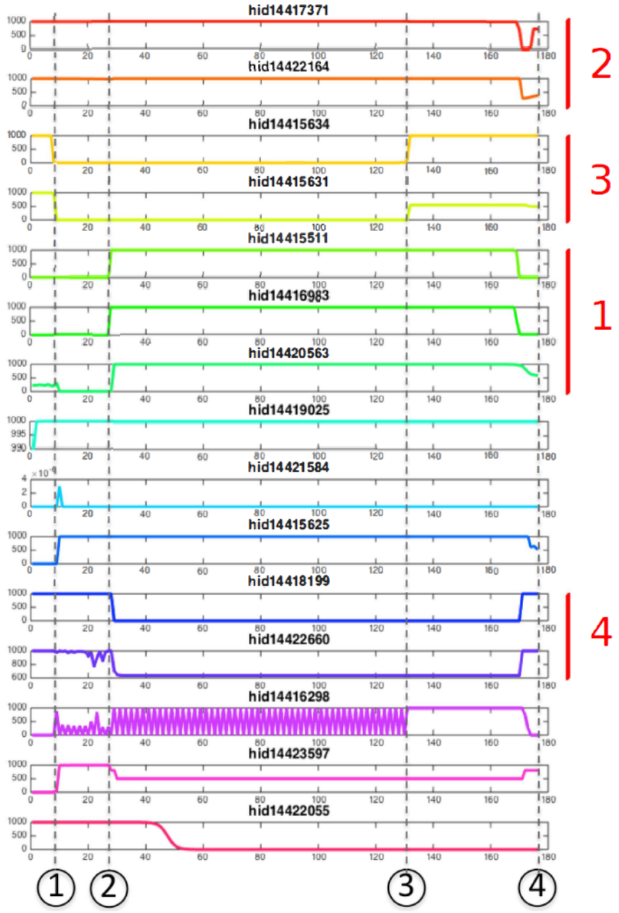
(a) Evolved neural circuit with fitness=DST.



(b) Time series correlation matrix of the fifteen hidden neurons



(c) Limb movement (time-lapse image)



(d) Time series values of the fifteen hidden neurons

Figure 4.9: Internal dynamics of fifteen hidden neurons, their correlations, and matching limb movements. See text for details.

more such loops. Furthermore, if recurrent loops are prohibited, the task performance greatly degraded and the evolved circuit had much more neurons than their recurrent circuit counterpart.

4.1.2.2 Other Related Works

Following our work "Emergence of Tool Use In an Articulated Limb Controlled by Evolved Neural Circuits", Wang *et al.* [131] investigated the dynamics of the recurrent neural network evolved by the NEAT algorithm. They analyzed the behavior and the neuronal activation of the RNN, and showed the correlation between the behavior of the limb and the internal dynamics of the RNN. They claim that the successful and fail trails can be explained by the existence of fixed points and limit cycles . Then, Reams and Choe [132] investigated the emergence of tool construction in the reaching task under similar environment. They showed tool construction and use behavior can be evolved by Neural Evolution algorithm with board fitness criteria. Nguyen *et al.* [133] created a physical simulated environment to investigate the emergence of tool use, and proposed to use proprioceptive and kinesthetic inputs to accelerate the training.

4.2 Approach

Similar to Nguyen *et al.* [133], we created a physical simulated environment using OpenAI gym and PyBullet, illustrated in Figure 4.10. The environment includes two robot arms, two tools and one target. Formally, we define the robot arm as the three-joint arm that directly belongs to the robot's body, and we define the tool as the stick that can be picked up by the robot to achieve some goals. To simply the problem, we add special joints (yellow joints showed in fig. 4.10) to the robot hand and the tool. The special joints can be connected to each other upon contact. The goal of the agent is to move the target to the bottom of the environment.

4.2.1 Multimodal Input

Instead of using only raw pixel image as input, we use the image and proprioceptive/kinesthetic feedback as input, similar to Nguyen *et al.*'s method [133]. To be specific, we use the following input:

- Vision input: RGB image of the entire environment

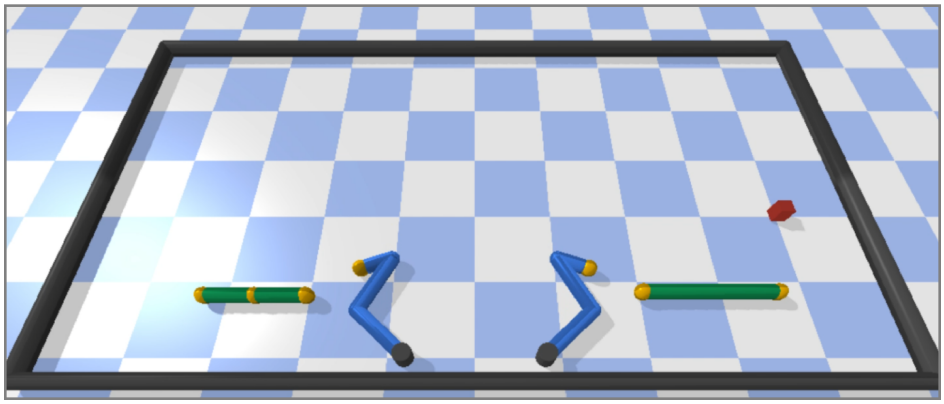


Figure 4.10: The tool construction environment build with PyBullet and following the OpenAI gym protocol. The environment is surrounded by a rectangle arena (black). There are two robot arms (blue), two tools (green), and one target (red). The joints marked as yellow are special joints that can be connected together upon contact. The goal for the agent is to move the object to the bottom of the environment.

- Proprioception input: the position of all the agent's joints
- Kinesthesia input: the velocities of all the agent's joints

4.2.2 Hierarchical Reinforcement Learning

In order to achieve the goal, the agent need to complete a sequence of non-trivial sub-tasks, for example, pick up the tool on the left using the left hand, pick the tool on the right using the right hand, construct a "T" shape tool by connecting one end of a tool to the middle of another tool, drag the target to the desired area. Nguyen *et al.* investigated simply using tools to drag the target with one arm, and they used manually engineered multi-step reward function. To avoid using the complicate manually designed reward function, directly using reinforcement learning algorithms cannot solve the problem effectively, because the exploration space is very large and the reward is sparse.

We propose to use hierarchical reinforcement learning with curiosity reward. Specifically, our model consists of a manager and many workers. The manager learns to plan the "macro steps" such as "pick up the two-joint tool", "pick up the three-joint tool", etc. The workers learn to execute the macro steps generated by the manager. If the macro step lead to a new state that the agent has not

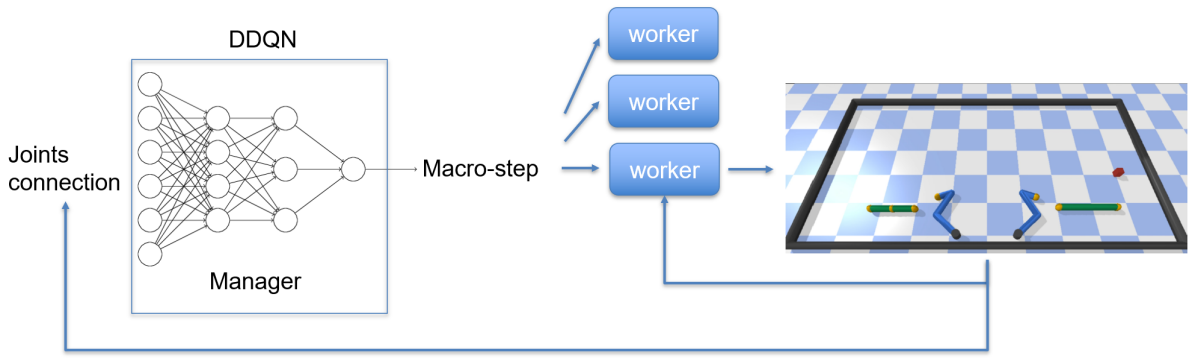


Figure 4.11: The overall architecture of our approach.

seen previously, then there will be some reward for the manager. Finally, there will also be a reward for the manager if the goal is reached. As for the macro step, if we define them manually, we have to enumerate all the possible steps. In addition, if we were given a novel task, we might have to re-define the macro steps. Therefore, we define the macro step in a more general way: we define the macro step as minimizing the distance between two salient objects, where the salient objects in our environment are the joints (yellow sphere shown in figure 4.10) and the target. Because the final goal is to move the target to the bottom of the environment, reducing the distance between the target and the bottom is another macro step. In addition, we add two macro steps that both the robot hands can release the tools they holding.

The action space for the manager is all the pairs between all the salient objects, and the action space for the worker is the control of the two three-joint robot arm. We use DQN [134] to train the manager and PPO [6] to train the worker. Algorithm 5 describes the details of training the manager with DQN. In Algorithm 5, for each macro step generated by the manager, a worker will be trained with PPO (Algorithm 4) to try to accomplish the macro step. Fig 4.11 illustrates the overall architecture of our approach.

There is one issue to be addressed for the macro step: what should be the initial environment state for the macro step? For the first macro step, the initial state is the environment's initial state. But what is the initial state for the following macro step? For example, if the first macro step is "pick up the tool on the left side" and the second macro step is "connect the end of one tool to the

middle of another tool", then what should be the initial state of the second macro step? We can save all the states of the environment (all positions of the objects, joint's velocity, joint's connection status, etc.) at the end of the previous step and use it to be the initial environment state to train the next macro step. However, the initial state from the previous macro step is not fixed. For example, if the previous macro step is "pick up the tool on the left side", then in the initial state of the next macro step, the left hand should holding the tool, but it could be at any position. Therefore, we run the previous macro step for k times and save all the environment's state. When training the next macro step, we randomly pick one environment's state to resume as the initial state. In our experiment, we set $k = 5$.

Algorithm 5: Pseudo-code of training the manager

Input: Raw pixel of the environment and the internal kinesthetic inputs
$manager = DQN(observation, action\_space)$
$target\_manager = DQN(observation\_space, action\_space)$
$state = env.reset()$
$for\ i\ in\ (1, number\_of\_iterations)$:
  $action = manager.act(state)$
  $next\_state, reward, done, info = env.step(action)$
  $replay\_buffer.push(state, action, reward, next\_state, done)$
  $state = next\_state$
  $if\ done$:
    $state = env.reset()$
  $if\ len(replay\_buffer) > batch\_size$:
    $loss = compute\_td\_loss(batch\_size)$
  $if\ i\%100 == 0$:
    $update\_target\_model(manager, target\_manager)$

4.2.3 Neural Network Architecture

We trained the manager using DDQN (Double DQN) algorithm. The neural network structure for the manager is three fully connected layers with a hidden size of 128 for each layer. A Relu activation layer is followed by each layer except for the last layer. The input to the manager is the connection status of all the joint in the environment, for example, whether the left hand is holding

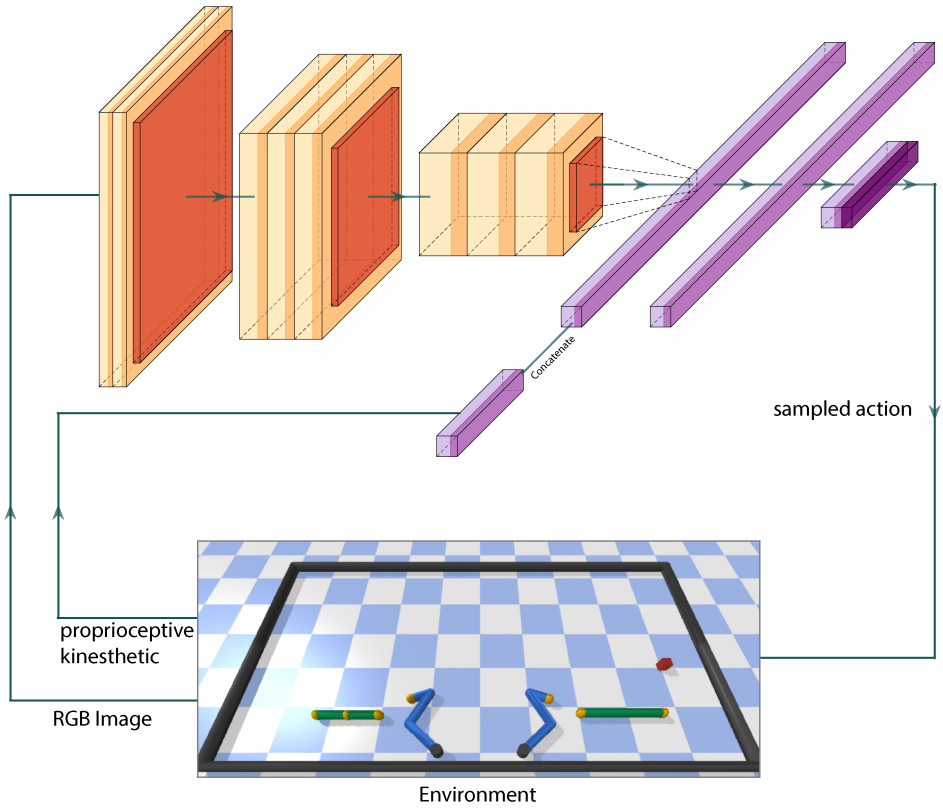


Figure 4.12: The overall network architecture of the workers. This figure only shows the actor network of the workers. The critic network shares the same network architecture with the actor network, except the the last output layer only has 1-dimensional output.

a tool, whether two tools are connected to each other, and so on. The output of the manager is two joints to be connected, for example, the right hand of the agent and the tool on the right hand side, the middle joint of one tool and the left joint of another tool, etc.

We trained the workers using Proximal Policy Optimization (PPO) algorithm. In PPO, we use the same network architecture for the actor and the critic. The overall network structure of the workers is illustrated in fig. 4.12. We use the 2D RGB image as well as the 1D proprioceptive/kinesthetic feedback as input. We first use three convolutional layers to process the RGB image of the environment. The first convolutional layer consists of 32 kernels with kernel size of 8 and stride of 4. The second convolutional layer consists of 64 kernels with kernel size of 4 and stride of 2. The third convolutional layer consists of 64 kernels with kernel size of 3 and stride of 1. Each of the convolutional layer is followed by a Relu activation function. Then, we

flatten the feature into 1-dimensional vector and concatenate it with the 1-dimensional proprioceptive/kinesthetic input. The concatenated feature is then fed to two fully connected layers with a hidden unit size of 512. A Relu activation layer is followed by the first fully connect layer. Both the actor and the critic share the same network architecture and parameters. The only difference is that the output dimension for the actor is 6 representing the control signal for the robot joints, while the output dimension for the critic is 1 representing the value. Fig. 4.12 shows the details of the actor network and the critic network.

4.3 Results

We implemented our approach in PyTorch [52]. We use the DDQN algorithm to train the manager and use Adam optimizer [53] with default parameters (β_1 = 0.9, β_2 = 0.999, ϵ = 1e-8). The replay buffer size is set to 1000 and the batch size is set to 4. For every 100 iterations the target model is synchronized with the current model.

We use the PPO algorithm to train the workers. The resolution of the image of the environment is (200, 360). We resize the image to (84, 84) and stack 4 consecutive frames before feeding to the neural network. The maximum steps for each episode is 4,000. We set the gamma to be 0.99, and use the Adam optimizer with default parameters.

We also evaluated using RGB image as input only. The network structure is the same as fig. 4.12, except that the feature vector from the last layer of CNN does not concatenate with the proprioceptive and kinesthetic input.

We ran all the training and evaluation on a machine with a NVIDIA GeForce RTX 2080 Ti GPU with 11G memory.

The manager successfully learnt a sequence of sub-tasks to solve the task: (1) pick up the tool on the right; (2) pick up the tool on the left; (3) connect the two-joint-tool with the middle joint of the three-joint-tool to construct a "T" shape tool; (4) use the "T" shape tool to drag the target to the bottom. Fig. 4.13 shows the smoothed reward for each sub-tasks. The overall success rate of our approach is 48%, while the success rate when using RGB image as input is only 20%. Table 4.1 describes the success rate of the agent to finish each of the sub-task. As it is can be seen from

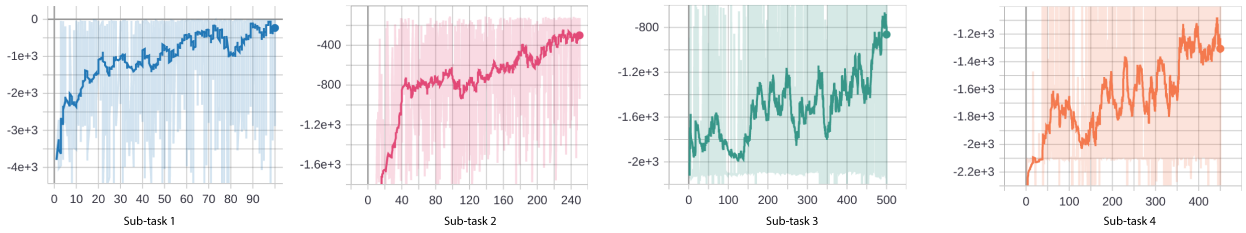


Figure 4.13: The smoothed reward for each sub-tasks. The Y-axis stands for the reward and the X-axis stands for epochs.

Task	vision only	vision/proprioceptive/kinesthetic
sub-task 1	76%	99%
sub-task 2	90.8%	94.9%
sub-task 3	82.6%	92.6%
sub-task 4	35.1%	55.2%
Overall	20%	48%

Table 4.1: Comparison of success rate between using vision input only and using vision, proprioceptive, and kinesthetic input.

the table, the success rate using vision/proprioceptive/kinesthetic input is much higher than the success rate using vision input only. Table 4.2 also shows the number of steps taken by the agent to achieve each of the sub-task. Similarly, the agent when using vision/proprioceptive/kinesthetic input use less steps to achieve the sub-tasks, except for the last sub-task which they use similar number of steps.

Fig. 4.14 illustrates one of the successful trials of constructing tool and reach the target. As can be seen from the figure, without given any predefined knowledge, the agent learned to construct a novel tool to drag the object. The agent first reaches the tool on the right side with its right arm (frame 3), and then reaches the tool on the left side with its left arm (frame 4), and then constructs a "T" shape tool (frame 6), and finally uses this tool to drag the object to the bottom (frame 7-8).

Fig. 4.15 shows more results.

	Vision only		full input	
Task	steps	std	steps	std
Sub-task 1	53	113.6	21.3	1.5
Sub-task 2	166.6	171.3	32	17.2
Sub-task 3	245.3	197.5	145.5	138.5
Sub-task 4	234.1	129	257.6	136.4
Overall	2923.2	2090.1	1488.5	1176.2

Table 4.2: Comparison of number of steps needed to achieve the goal between using vision input only and using vision, proprioceptive, and kinesthetic input.

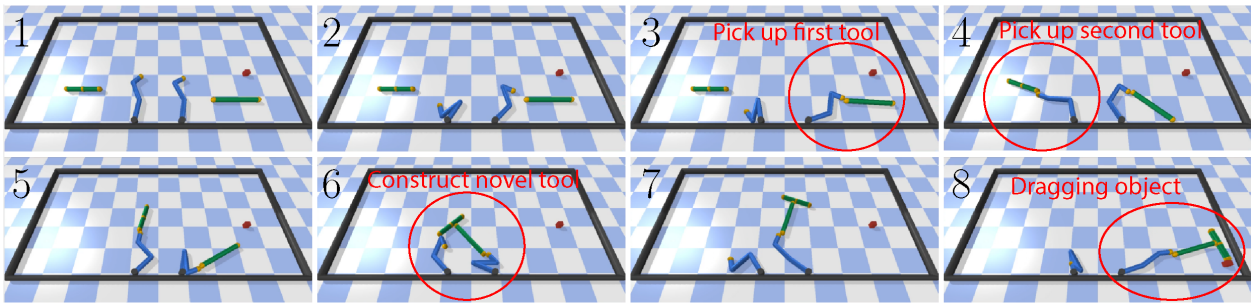
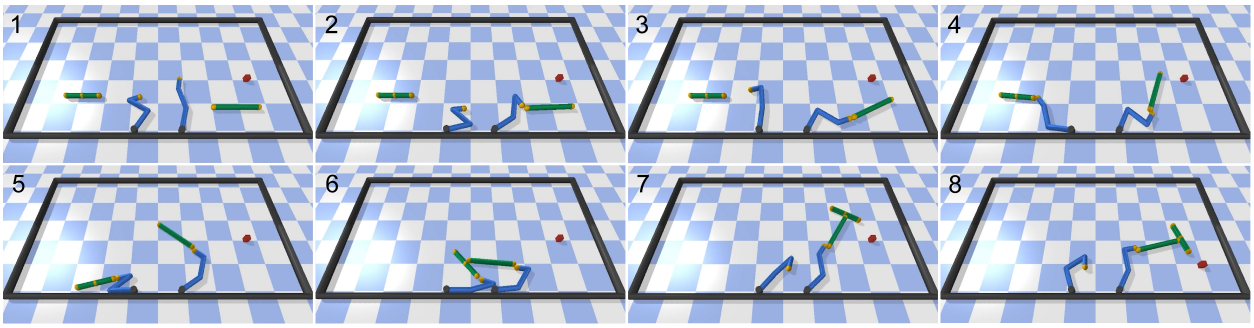


Figure 4.14: Controlling two jointed arms to construct a "T"-shaped tool for dragging an object (red cube) to the bottom of the screen.

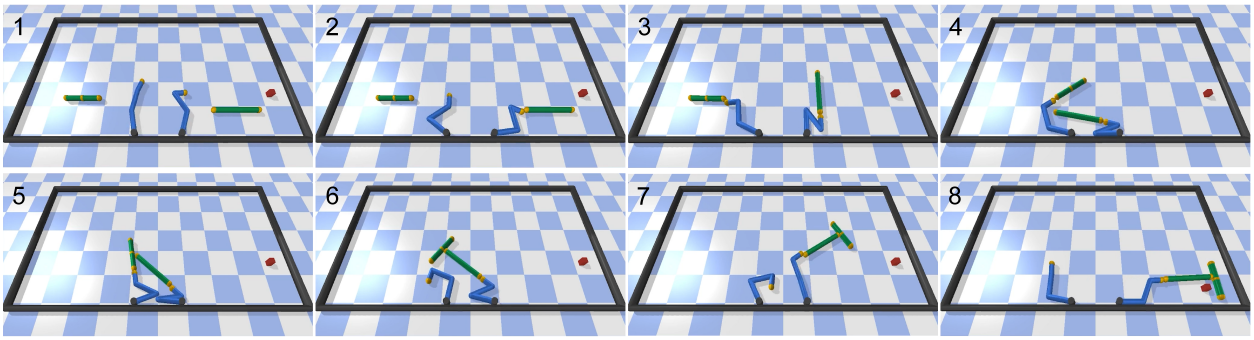
4.4 Discussion & Conclusion

The limitation of this work is that there are some simplification, assumptions and restrictions of the environment we made to simplify the problem, which might not be very realistic. For example, we simplify the robot's hand to be a joint; we assume the joint can be connected upon contact; and we initialize the target at a certain range of location. Therefore, one of the future work is to release these assumptions and restrictions of the environment. Although our environment is built in a 3D world, it is still a 2D environment. Another future work is to extend the environment to be a full 3D environment, similar to the IKEA Furniture Assembly environment [135].

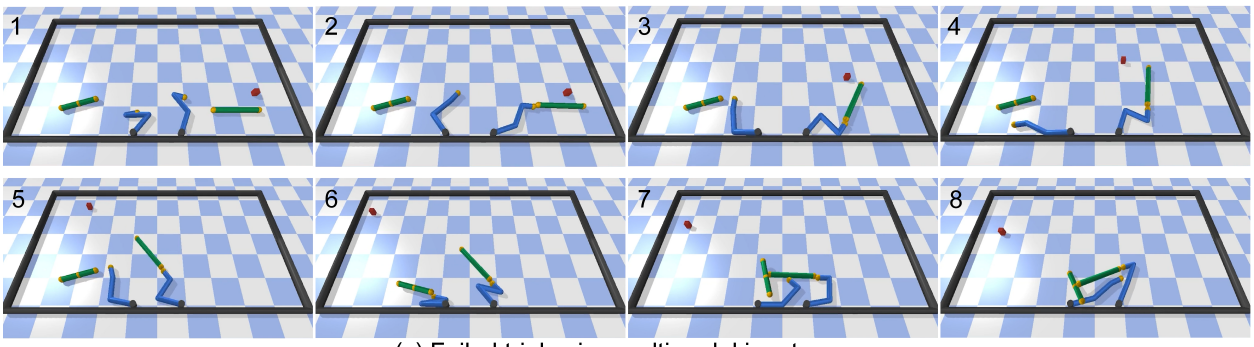
As conclusion, we introduced a physical simulated environment for tool construction ans use. We proposed an approach to use raw pixel input as well as proprioceptive/kinesthetic input to learn to construct and use novel tools. Different from the previous works [133], we do not use



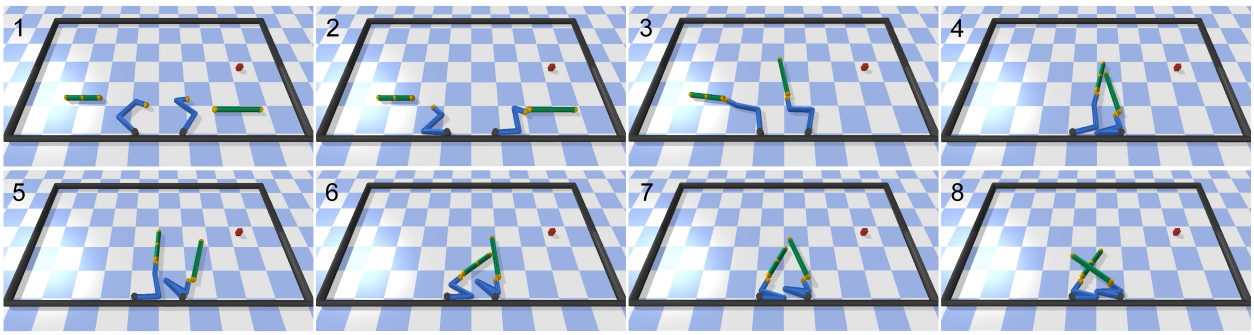
(a) Successful trial using multimodal input



(b) Successful trial using vision input only



(c) Failed trial using multimodal input



(d) Failed trial using vision input only

Figure 4.15: (a) Successful trial using multimodal input. It takes 1700 steps to complete the task for this trial. (b) Successful trial using vision input only. It takes 3400 steps to complete the task for this trial. (c) Failed trial using multimodal input. The agent accidentally pushed the target away when constructing "T" shape tool, making the task unsolvable. (d) Failed trial using vision input only. The agent struggle to construct the "T" shape tool without proprioceptive and kinesthetic input.

manually designed reward function. We propose to use hierarchical reinforcement learning with curiosity reward to accelerate the training. In our approach, the manager proposes and assigns sub-goals to workers and the workers then learn to achieve the sub-goals. We design the sub-goals as minimizing the distance between some salient objects to encourage behaviors such as picking up tools, connecting two tools to build a new tool, and so on. Generally, the definition of the sub-goals allows us to explore the environment at a higher level: to reach and grab some objects and to construct novel objects. We believe that this strategy to design sub-goals to encourage certain behaviors and accelerate exploration can also be applied in other Reinforcement Learning problems. Our experiments show that the agent can successfully learn a sequence of sub-tasks to construct a novel tool to achieve the goal. In addition, adding proprioceptive/kinesthetic input can accelerate the training and lead to a better performance.

5. CONCLUSION

In this project, we investigated three different problems with multimodal deep learning.

The first problem is learning to synthesize 4D light field from a single image. In this task, our key contributions include: (1) we found that using pre-trained depth prediction network to generate estimated depth and fusing the estimated depth with the 2D image input can significantly improved the light field synthesis performance, (2) we extended the MPI scene representation and demonstrated its advantage over the original MPI for single image view synthesis with small baseline, (3) we proposed a two parallel network architecture to handle the visible and occluded regions to reduce artifacts, (4) we introduced a new large-scale light field dataset containing over 2,000 unique scenes.

In the second task, we focused on face recognition for unconstrained videos. We proposed to use audio information in the video to aid in the recognition of low quality videos. In order to achieve this goal, we introduced an Audio-Visual Aggregation Network (AVAN) to aggregate multiple facial and voice information to improve face recognition performance. To effectively train and evaluate our approach, we constructed an Audio-Visual Face Recognition dataset. Overall, the key contributions are: (1) proposing an approach to aggregate visual embeddings and audio embeddings for unconstrained video face recognition, (2) a large speech-video-matched audio-visual person recognition dataset, which is a first of its kind, (3) experiments show that our approach significantly improves the person recognition accuracy.

Finally, we investigated learning to construct and use tools with deep reinforcement learning. We proposed to use raw pixel image as visual input, as well as proprioceptive and kinesthetic inputs to accelerate the training. In addition, because the exploration space is very large and the reward is sparse, we proposed to use hierarchical reinforcement learning with curiosity reward. Our experiments showed that our approach can successfully achieve the goal with the multimodal input. The contributions for this research includes: (1) proposing an approach to use visual, proprioceptive (kinesthetic) multimodal inputs to construct and use novel tools, (2) using hierarchical

reinforcement learning with curiosity reward to accelerate training, (3) introducing a physically simulated environment for novel tool construction and use research.

In summary, we investigated different approaches, strategies, and architectures to improve the performance by using multimodal input in three different scenarios. We expect the specific methods developed in this work to help us better understand multimodal processing in deep learning.

REFERENCES

[1] T.-C. Wang, J.-Y. Zhu, N. K. Kalantari, A. A. Efros, and R. Ramamoorthi, "Light field video capture using a learning-based hybrid imaging system," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 133, 2017.

[2] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 31–42, 1996.

[3] L. Wang, X. Shen, J. Zhang, O. Wang, Z. Lin, C.-Y. Hsieh, S. Kong, and H. Lu, "Deeplens: Shallow depth of field from a single image," *ACM Transactions on Graphics (TOG)*, vol. 37, pp. 245:1–245:11, Dec. 2018.

[4] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2019.

[5] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, vol. 650, 2017.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[7] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," *ACM Transactions on Graphics (TOG)*, vol. 37, pp. 65:1–65:12, July 2018.

[8] E. H. Adelson, J. R. Bergen, *et al.*, *The plenoptic function and the elements of early vision*, vol. 2. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of . . . , 1991.

[9] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy, "High performance imaging using large camera arrays," in *ACM SIGGRAPH*

2005 Papers, pp. 765–776, 2005.

[10] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg, "Transformation-grounded image generation network for novel 3d view synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3500–3509, 2017.

[11] M. Liu, X. He, and M. Salzmann, "Geometry-aware deep network for single-image novel view synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4616–4624, 2018.

[12] S. Tulsiani, R. Tucker, and N. Snavely, "Layer-structured 3d scene inference via view synthesis," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 302–317, 2018.

[13] S. Niklaus, L. Mai, J. Yang, and F. Liu, "3d ken burns effect from a single image," *ACM Transactions on Graphics (TOG)*, vol. 38, Nov. 2019.

[14] P. P. Srinivasan, T. Wang, A. Sreelal, R. Ramamoorthi, and R. Ng, "Learning to synthesize a 4d rgbd light field from a single image," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2243–2251, 2017.

[15] X. Cun, F. Xu, C. Pun, and H. Gao, "Depth-assisted full resolution network for single image-based view synthesis," *IEEE Computer Graphics and Applications*, vol. 39, pp. 52–64, March 2019.

[16] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely, "Pushing the boundaries of view extrapolation with multiplane images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 175–184, 2019.

[17] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Transactions on Graphics (TOG)*, vol. 38, pp. 29:1–29:14, July 2019.

[18] K. Karsch, C. Liu, and S. B. Kang, "Depth transfer: Depth extraction from video using non-parametric sampling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2144–2158, 2014.

[19] L. Ladicky, J. Shi, and M. Pollefeys, "Pulling things out of perspective," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 89–96, 2014.

[20] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, (Cambridge, MA, USA), pp. 2366–2374, MIT Press, 2014.

[21] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe, "Multi-scale continuous crfs as sequential deep networks for monocular depth estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5354–5362, 2017.

[22] D. Xu, W. Wang, H. Tang, H. Liu, N. Sebe, and E. Ricci, "Structured attention guided convolutional neural fields for monocular depth estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3917–3925, 2018.

[23] W. Chen, Z. Fu, D. Yang, and J. Deng, "Single-image depth perception in the wild," in *Advances in Neural Information Processing Systems*, pp. 730–738, 2016.

[24] K. Xian, C. Shen, Z. Cao, H. Lu, Y. Xiao, R. Li, and Z. Luo, "Monocular relative depth perception with web stereo data supervision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 311–320, 2018.

[25] Z. Li and N. Snavely, "Megadepth: Learning single-view depth prediction from internet photos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2041–2050, 2018.

[26] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis, "Depth synthesis and local warps for plausible image-based navigation," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 3, pp. 1–12, 2013.

[27] P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf, "Casual 3d photography," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–15, 2017.

[28] P. Hedman and J. Kopf, "Instant 3d photography," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–12, 2018.

[29] E. Penner and L. Zhang, "Soft 3d reconstruction for view synthesis," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, p. 235, 2017.

[30] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, "Deepstereo: Learning to predict new views from the world's imagery," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5515–5524, 2016.

[31] I. Choi, O. Gallo, A. Troccoli, M. H. Kim, and J. Kautz, "Extreme view synthesis," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7781–7790, 2019.

[32] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker, "Deepview: View synthesis with learned gradient descent," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2367–2376, 2019.

[33] N. K. Kalantari, T.-C. Wang, and R. Ramamoorthi, "Learning-based view synthesis for light field cameras," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 193, 2016.

[34] G. Wu, M. Zhao, L. Wang, Q. Dai, T. Chai, and Y. Liu, "Light field reconstruction using deep convolutional network on epi," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6319–6327, 2017.

[35] Y. Wang, F. Liu, Z. Wang, G. Hou, Z. Sun, and T. Tan, "End-to-end view synthesis for light field imaging with pseudo 4dcnn," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 333–348, 2018.

[36] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Single-view to multi-view: Reconstructing unseen views with a convolutional network," *CoRR*, vol. abs/1511.06702, 2015.

[37] J. Yang, S. E. Reed, M.-H. Yang, and H. Lee, "Weakly-supervised disentangling with recurrent transformations for 3d view synthesis," in *Advances in Neural Information Processing Systems*, pp. 1099–1107, 2015.

[38] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, "Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision," in *Advances in Neural Information Processing Systems*, pp. 1696–1704, 2016.

[39] K. Rematas, C. H. Nguyen, T. Ritschel, M. Fritz, and T. Tuytelaars, "Novel views of objects from a single image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 8, pp. 1576–1590, 2016.

[40] K. Olszewski, S. Tulyakov, O. Woodford, H. Li, and L. Luo, "Transformable bottleneck networks," *arXiv preprint arXiv:1904.06458*, 2019.

[41] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, "View synthesis by appearance flow," in *European Conference on Computer Vision*, pp. 286–301, Springer, 2016.

[42] H. Dhamo, K. Tateno, I. Laina, N. Navab, and F. Tombari, "Peeking behind objects: Layered depth prediction from a single image," *Pattern Recognition Letters*, vol. 125, pp. 333–340, 2019.

[43] S. Evain and C. Guillemot, "A lightweight neural network for monocular view generation with occlusion handling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–14, 2019.

[44] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang, "3d photography using context-aware layered depth inpainting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8028–8038, 2020.

[45] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson, "Synsin: End-to-end view synthesis from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7467–7477, 2020.

[46] R. Tucker and N. Snavely, "Single-view view synthesis with multiplane images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 551–560, 2020.

[47] T. Porter and T. Duff, "Compositing digital images," in *ACM Siggraph Computer Graphics*, vol. 18, pp. 253–259, ACM, 1984.

[48] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[49] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1511–1520, 2017.

[50] D. G. Dansereau, B. Girod, and G. Wetzstein, "LiFF: Light field features in scale and depth," in *Computer Vision and Pattern Recognition (CVPR)*, IEEE, June 2019.

[51] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski, "Non-rigid dense correspondence with applications for image enhancement," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, p. 70, 2011.

[52] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[54] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[55] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," in *European conference on computer vision*, pp. 87–102, Springer, 2016.

[56] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard, "The megaface benchmark: 1 million faces for recognition at scale," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4873–4882, 2016.

[57] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[59] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks.," in *ICML*, vol. 2, p. 7, 2016.

[60] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[61] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*, pp. 84–92, Springer, 2015.

[62] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 539–546, IEEE, 2005.

[63] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 212–220, 2017.

[64] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[65] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5265–5274, 2018.

[66] K. Zhao, J. Xu, and M.-M. Cheng, "Regularface: Deep face recognition via exclusive regularization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1136–1144, 2019.

[67] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," *Advances in neural information processing systems*, vol. 17, pp. 513–520, 2004.

[68] N. Quadrianto and C. H. Lampert, "Learning multi-view neighborhood preserving projections," in *ICML*, 2011.

[69] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian detection via classification on riemannian manifolds," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 10, pp. 1713–1727, 2008.

[70] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa, "Statistical computations on grassmann and stiefel manifolds for image and video-based recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2273–2286, 2011.

[71] C. Herrmann, D. Willersinn, and J. Beyerer, "Low-resolution convolutional neural networks for video face recognition," in *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 221–227, IEEE, 2016.

[72] K. Sohn, S. Liu, G. Zhong, X. Yu, M.-H. Yang, and M. Chandraker, "Unsupervised domain adaptation for face recognition in unlabeled videos," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3210–3218, 2017.

[73] Y. Rao, J. Lin, J. Lu, and J. Zhou, "Learning discriminative aggregation network for video-based face recognition," in *Proceedings of the IEEE international conference on computer vision*, pp. 3781–3790, 2017.

[74] J. Zheng, R. Ranjan, C.-H. Chen, J.-C. Chen, C. D. Castillo, and R. Chellappa, "An automatic system for unconstrained video-based face recognition," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 2, no. 3, pp. 194–209, 2020.

[75] J. Yang, P. Ren, D. Zhang, D. Chen, F. Wen, H. Li, and G. Hua, "Neural aggregation network for video face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4362–4371, 2017.

[76] S. Gong, Y. Shi, and A. K. Jain, "Recurrent embedding aggregation network for video face recognition," *arXiv preprint arXiv:1904.12019*, 2019.

[77] T. Choudhury, B. Clarkson, T. Jebara, and A. Pentland, "Multimodal person recognition using unconstrained audio and video," in *Proceedings, International Conference on Audio- and Video-Based Person Authentication*, pp. 176–181, Citeseer, 1999.

[78] A. Albiol, L. Torres, and E. J. Delp, "Fully automatic face recognition system using a combined audio-visual approach," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 152, no. 3, pp. 318–326, 2005.

[79] X. Tang and Z. Li, "Audio-guided video-based face recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 7, pp. 955–964, 2009.

[80] G. Sell, K. Duh, D. Snyder, D. Etter, and D. Garcia-Romero, "Audio-visual person recognition in multimedia data from the iarpa janus program," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3031–3035, IEEE, 2018.

[81] Y. M. Assael, B. Shillingford, S. Whiteson, and N. De Freitas, "Lipnet: End-to-end sentence-level lipreading," *arXiv preprint arXiv:1611.01599*, 2016.

[82] K. Hoover, S. Chaudhuri, C. Pantofaru, M. Slaney, and I. Sturdy, "Putting a face to the voice: Fusing audio and visual signals across a video to determine speakers," *arXiv preprint arXiv:1706.00079*, 2017.

[83] A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, and M. Rubinstein, "Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation," *arXiv preprint arXiv:1804.03619*, 2018.

[84] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 1, pp. 517–520, IEEE Computer Society, 1992.

[85] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210, IEEE, 2015.

[86] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[87] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[88] D. Wang and X. Zhang, "Thchs-30: A free chinese speech corpus," *arXiv preprint arXiv:1512.01882*, 2015.

[89] M. Nakayama, T. Nishiura, Y. Denda, N. Kitaoka, K. Yamamoto, T. Yamada, S. Tsuge, C. Miyajima, M. Fujimoto, T. Takiguchi, *et al.*, "Censrec-4: Development of evaluation framework for distant-talking speech recognition under reverberant environments," in *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[90] M. Falcone and A. Gallo, "The" siva" speech database for speaker verification: Description and evaluation," in *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, vol. 3, pp. 1902–1905, IEEE, 1996.

[91] R. Jahangir, Y. W. Teh, H. F. Nweke, G. Mujtaba, M. A. Al-Garadi, and I. Ali, "Speaker identification through artificial intelligence techniques: A comprehensive review and research challenges," *Expert Systems with Applications*, vol. 171, p. 114591, 2021.

[92] Y. Lukic, C. Vogt, O. Dürr, and T. Stadelmann, "Speaker identification and clustering using convolutional neural networks," in *2016 IEEE 26th international workshop on machine learning for signal processing (MLSP)*, pp. 1–6, IEEE, 2016.

[93] A. S. Imran, V. Haflan, A. S. Shahrebabaki, N. Olfati, and T. K. Svendsen, "Evaluating acoustic feature maps in 2d-cnn for speaker identification," in *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, pp. 211–216, 2019.

[94] P. Dhakal, P. Damacharla, A. Y. Javaid, and V. Devabhaktuni, "A near real-time automatic speaker recognition architecture for voice-based user interface," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, pp. 504–520, 2019.

[95] N. N. An, N. Q. Thanh, and Y. Liu, "Deep cnns with self-attention for speaker identification," *IEEE Access*, vol. 7, pp. 85327–85337, 2019.

[96] J. Larsson, "Optimizing text-independent speaker recognition using an lstm neural network," 2014.

[97] J.-W. Jung, H.-S. Heo, I.-H. Yang, H.-J. Shim, and H.-J. Yu, "Avoiding speaker overfitting in end-to-end dnns using raw waveform for text-independent speaker verification," *extraction*, vol. 8, no. 12, pp. 23–24, 2018.

[98] O. Ghahabi and J. Hernando, "Restricted boltzmann machines for vector representation of speech in speaker recognition," *Computer Speech & Language*, vol. 47, pp. 16–29, 2018.

[99] S. S. Tirumala and S. R. Shahamiri, "A deep autoencoder approach for speaker identification," in *Proceedings of the 9th International Conference on Signal Processing Systems*, pp. 175–179, 2017.

[100] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," 2008.

[101] S. Moschoglou, A. Papaioannou, C. Sagonas, J. Deng, I. Kotsia, and S. Zafeiriou, "Agedb: the first manually collected, in-the-wild age database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 51–59, 2017.

[102] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.

[103] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *CVPR 2011*, pp. 529–534, IEEE, 2011.

[104] J. R. Beveridge, P. J. Phillips, D. S. Bolme, B. A. Draper, G. H. Givens, Y. M. Lui, M. N. Teli, H. Zhang, W. T. Scruggs, K. W. Bowyer, *et al.*, "The challenge of face recognition from digital point-and-shoot cameras," in *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pp. 1–8, IEEE, 2013.

[105] A. Bansal, A. Nanduri, C. D. Castillo, R. Ranjan, and R. Chellappa, "Umdfaces: An annotated face dataset for training deep networks," in *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 464–473, IEEE, 2017.

[106] B. Maze, J. Adams, J. A. Duncan, N. Kalka, T. Miller, C. Otto, A. K. Jain, W. T. Niggel, J. Anderson, J. Cheney, *et al.*, "Iarpa janus benchmark-c: Face dataset and protocol," in *2018 International Conference on Biometrics (ICB)*, pp. 158–165, IEEE, 2018.

[107] Y. Liu, B. Peng, P. Shi, H. Yan, Y. Zhou, B. Han, Y. Zheng, C. Lin, J. Jiang, Y. Fan, *et al.*, "iqiyi-vid: A large dataset for multi-modal person identification," *arXiv preprint arXiv:1811.07548*, 2018.

[108] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[109] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[110] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

[111] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[112] R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines," *Advances in neural information processing systems*, pp. 1043–1049, 1998.

[113] T. G. Dietterich, "Hierarchical reinforcement learning with the maxq value function decomposition," *Journal of artificial intelligence research*, vol. 13, pp. 227–303, 2000.

[114] P. Dayan and G. E. Hinton, "Feudal reinforcement learning," in *Advances in Neural Information Processing Systems* (S. Hanson, J. Cowan, and C. Giles, eds.), vol. 5, Morgan-Kaufmann, 1993.

[115] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," *Advances in neural information processing systems*, vol. 29, pp. 3675–3683, 2016.

[116] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[117] O. Nachum, S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," *arXiv preprint arXiv:1805.08296*, 2018.

[118] A. Levy, G. Konidaris, R. Platt, and K. Saenko, "Learning multi-level hierarchies with hindsight," *arXiv preprint arXiv:1712.00948*, 2017.

[119] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *International Conference on Machine Learning*, pp. 3540–3549, PMLR, 2017.

[120] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," *arXiv preprint arXiv:1810.12894*, 2018.

[121] E. B. Ottoni and P. Izar, "Capuchin monkey tool use: overview and implications," *Evolutionary Anthropology: Issues, News, and Reviews: Issues, News, and Reviews*, vol. 17, no. 4, pp. 171–178, 2008.

[122] A. H. Taylor, B. Knaebe, and R. D. Gray, "An end to insight? new caledonian crows can spontaneously solve problems without planning their actions," *Proceedings of the Royal Society B: Biological Sciences*, vol. 279, no. 1749, pp. 4977–4981, 2012.

[123] A. M. P. von Bayern, S. Danel, A. Auersperg, B. Mioduszewska, and A. Kacelnik, "Compound tool construction by new caledonian crows," *Scientific reports*, vol. 8, no. 1, pp. 1–8, 2018.

[124] Q. Li, J. Yoo, and Y. Choe, "Emergence of tool use in an articulated limb controlled by evolved neural circuits," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2015.

[125] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[126] K. O. Stanley, B. D. Bryant, I. Karpov, R. Miikkulainen, *et al.*, "Real-time evolution of neural networks in the nero video game," in *AAAI*, vol. 6, pp. 1671–1674, 2006.

[127] J. Gauci, K. O. Stanley, *et al.*, "A case study on the critical role of geometric regularity in machine learning.," in *AAAI*, pp. 628–633, 2008.

[128] R. S. Amant and A. B. Wood, "Tool use for autonomous agents.," in *AAAI*, pp. 184–189, 2005.

[129] E. Visalberghi and L. Limongelli, "Acting and understanding: Tool use revisited through the minds of capuchin monkeys.," 1996.

[130] J. Kwon and Y. Choe, "Predictive internal neural dynamics for delay compensation," in *2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pp. 443–448, IEEE, 2010.

[131] H. Wang, Q. Li, J. Yoo, and Y. Choe, "Dynamical analysis of recurrent neural circuits in articulated limb controllers for tool use," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 4339–4345, IEEE, 2016.

[132] R. Reams and Y. Choe, "Emergence of tool construction in an articulated limb controlled by evolved neural circuits," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 642–649, IEEE, 2017.

[133] K. N. Nguyen, J. Yoo, and Y. Choe, "Speeding up affordance learning for tool use, using proprioceptive and kinesthetic inputs," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2019.

[134] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[135] Y. Lee, E. S. Hu, and J. J. Lim, "IKEA furniture assembly environment for long-horizon complex manipulation tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.