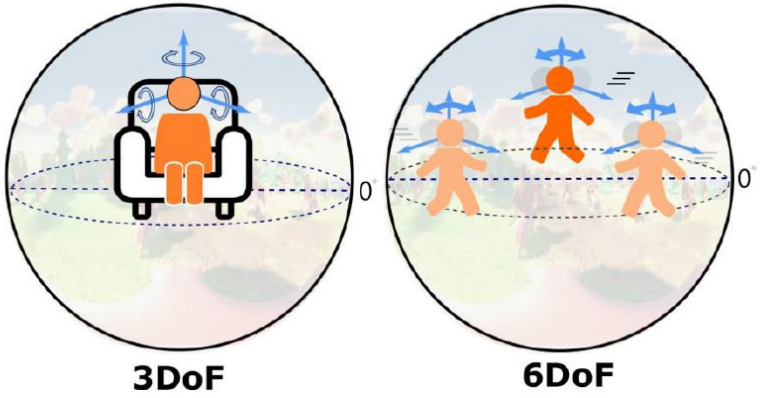
Optimizing Immersive Video Coding Configurations Using Deep Learning Approaches

Tse Hou Hung (tsehou.nthu@gmail.com)
Cooperator: Dr. Chih-Fan Hsu, Inventec
Advisor: Prof. Cheng-Hsin Hsu



Networking and Multimedia Systems Lab,
ISA, National Tsing Hua University

NMSL@NTHU
Networking and Multimedia Systems Lab

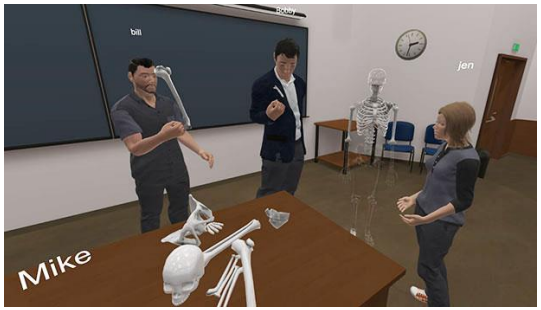
Outline

- Introduction
- Challenges
- Configuration Optimization Problem
- NN-based Configuration Optimizer
- Objective Evaluations
- Subjective Evaluations
- Summarizing Our Findings
- Conclusion

Introduction

Virtual Reality (VR)

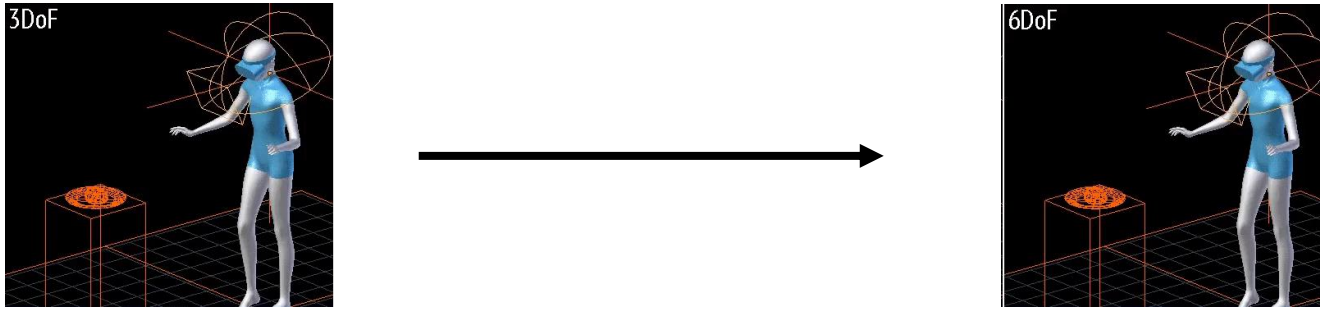
- Virtual Reality (VR) technology is thriving in recent years
- Various VR applications, e.g., healthcare, education and training, and entertainment [1] [2]
- Market report shows that VR market size value can reach USD 62.1 billion in 2027 [3]
- Industry and academia focus on improving VR experience

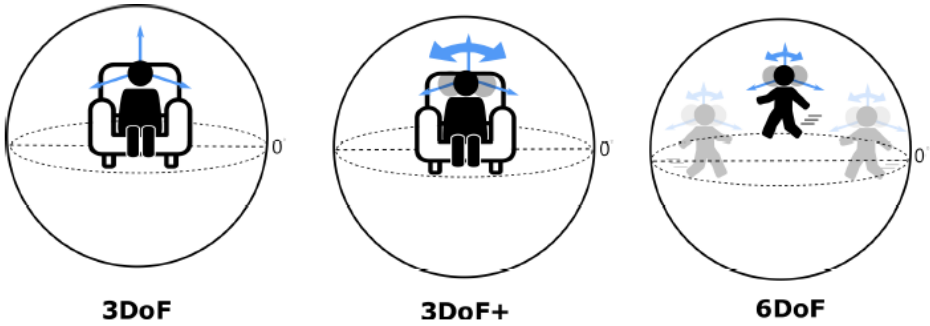


[1] Berg, Leif P., and Judy M. Vance. "Industry use of virtual reality in product design and manufacturing: a survey." *Virtual reality* 21.1 (2017): 1-17.
[2] Virtual Speech, VR Applications: 21 Industries already using Virtual Reality, https://virtualspeech.com/blog/vr-applications
[3] Grand View research, Virtual Reality Market Size, Industry Report, 2020-2027

Classify VR: Interaction Mode

- Interaction mode of VR: *3DoF* v.s. *6DoF* (Degree of Freedom)
- Today's VR content is mostly in the format of *360 video* (3DoF)
- 6DoF interaction can not achieved by single 360 video
- More descriptive 3D representations are required for enabling 6DoF VR





Video is the example video from OculusQuest,
https://www.reddit.com/r/OculusQuest/comments/bu6io9/3dof_6dof_explained/

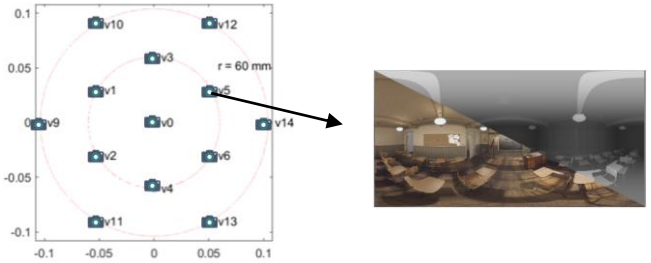
3D Representations

- Recently, MPEG release several standards for different 3D data representations
- We study the Test Model for Immersive Video (TMIV), which is a reference codec of MIV

Point Cloud

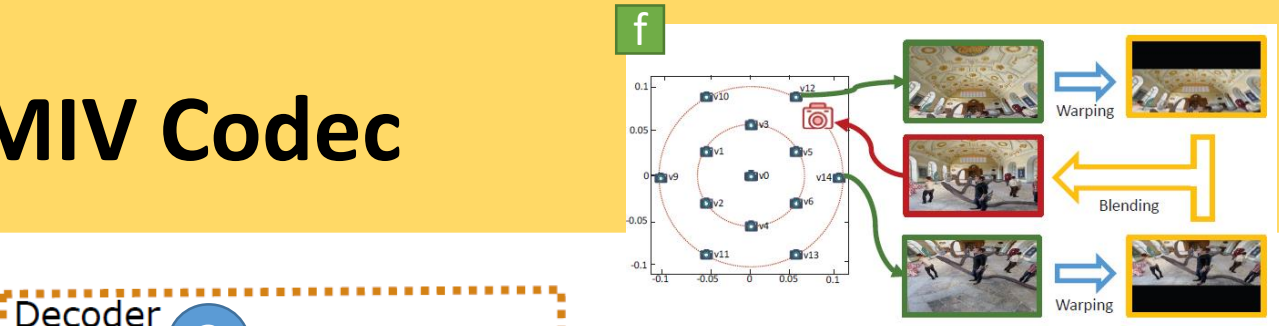
- Point Cloud Compression (PCC) standard
- Easy to manipulate
- Low rendering quality
- PCC standard focus on object

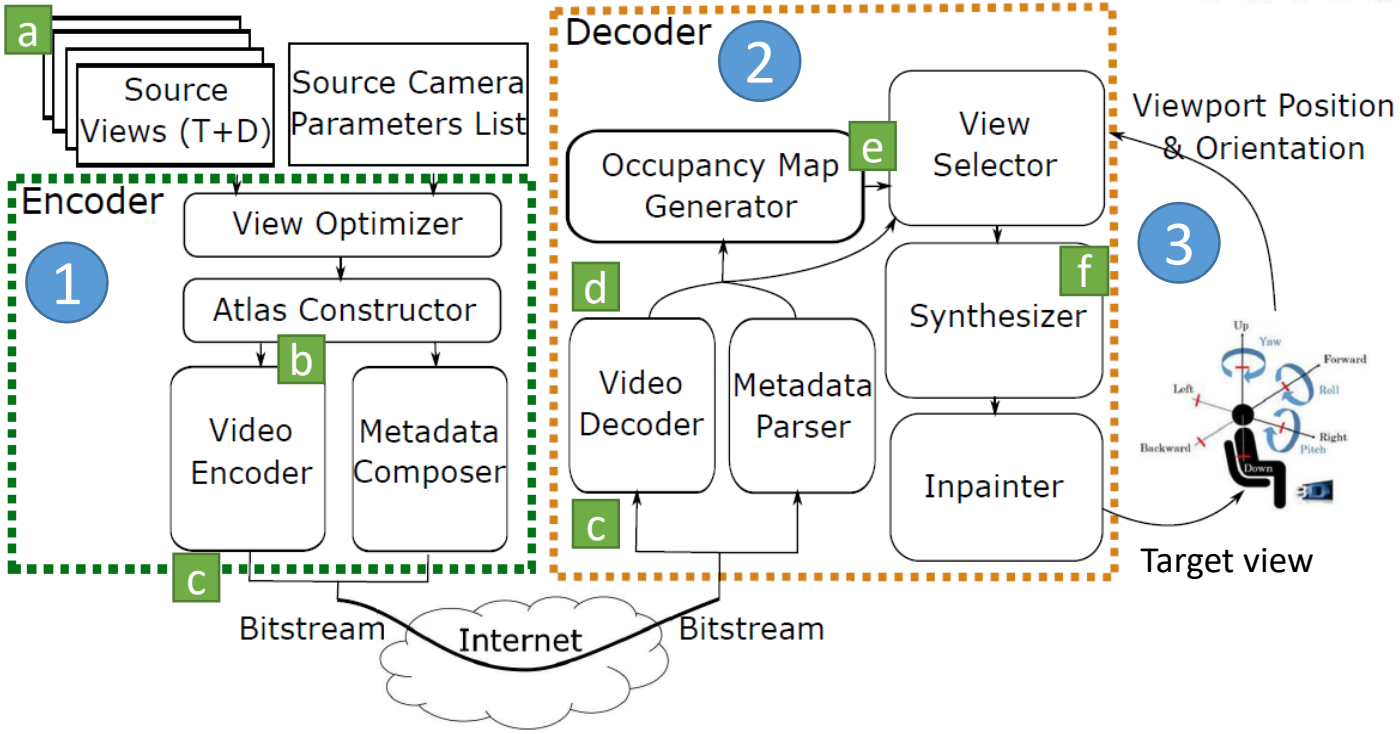
RGBD videos

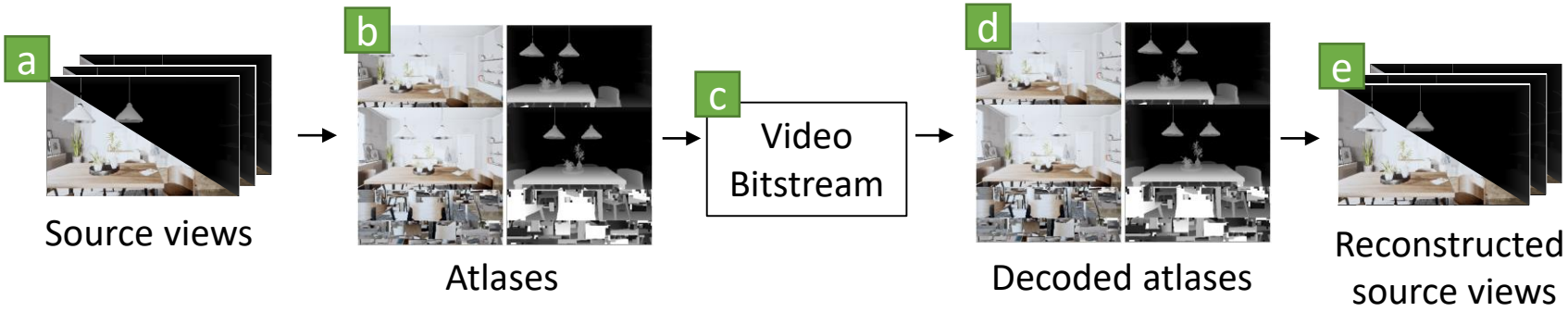


- MPEG Immersive Video (MIV) standard
- High rendering quality
- Need complex view synthesis algorithm support
- MIV standard focus on scene

Overview of TMIV Codec



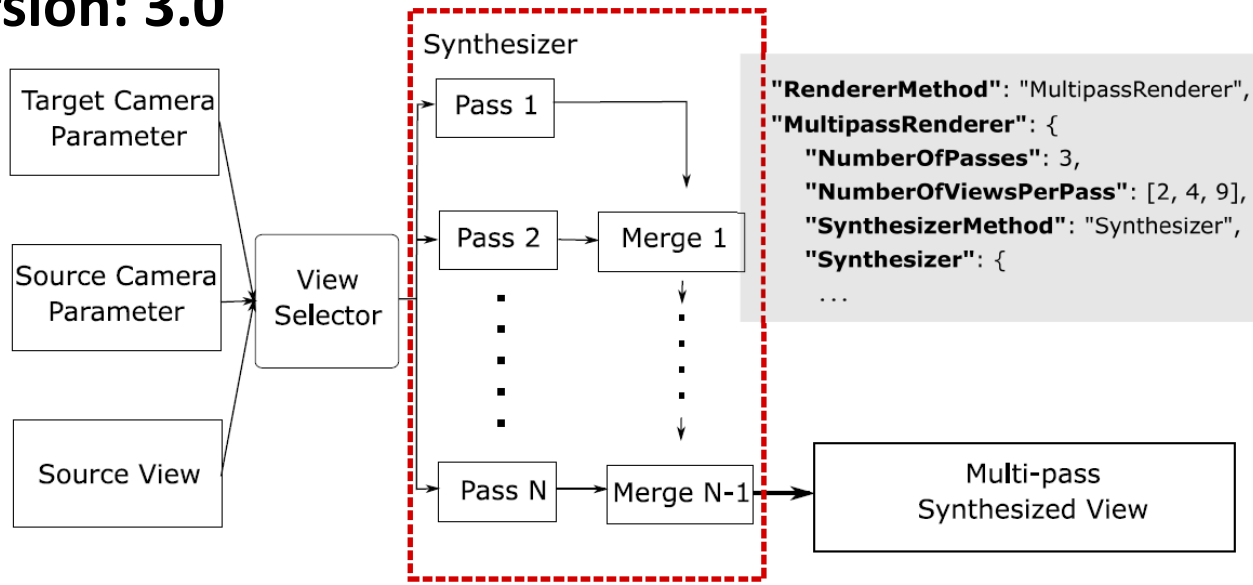




View Synthesizer in MIV

- The synthesizer in MIV decoder conducts synthesis for multiple passes, and combining the results of each pass to get final synthesized result
- The *number of passes* and *number of views per pass* are according to configuration file

TMIV version: 3.0



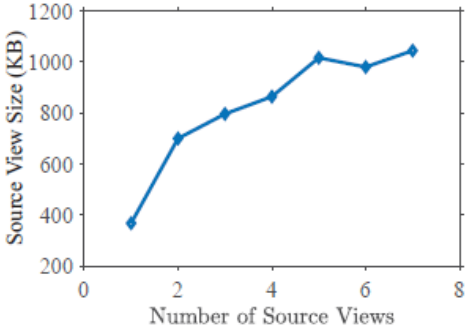
Challenges

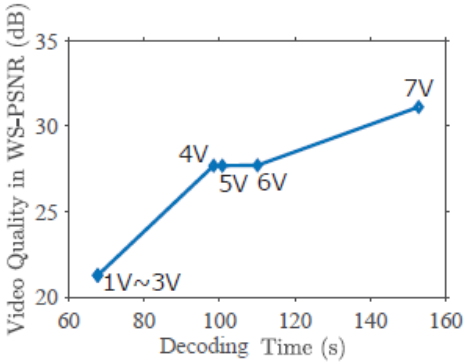
Pilot Study for TMIV

- To understand the performance of TMIV, we conduct small scale experiments to evaluate TMIV under various configurations
- Experiment 1: *vary number of views per pass*
  - Number of passes: 1
  - Number of views per pass: 1, 2, …, 7
- Experiment 2: *vary number of passes*
  - Number of passes: 1, 3, 7
  - Number of views per pass: [7], [1, 4, 7], [1, 2, …, 7]

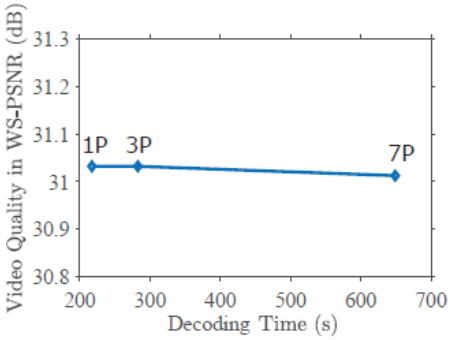
The Results of Pilot study

- In experiment 1, the results show that the *source view size, video quality, and decoding time grows along with the number of source views*





- In experiment 2, results show that the *decoding time and the number of passes are positively correlated*, but video quality and the number of passes are not.



The parameters in TMIV's configuration significantly affect the performance

Challenges

- TMIV can not determine the configuration automatically
- The pre-defined configuration can not make optimal tradeoff among
  - video quality,
  - decoding time, and
  - bandwidth consumption
- The optimal configuration may different according to scene and camera parameters
- To solve this *configuration optimization problem*, we introduce a new component, a *configuration optimizer* for the TMIV codec

Configuration Optimization Problem

Problem Statement

- Given:
  - *Source views* V
    - each view v $(v = 1,2, \dots, V)$ consists of texture (T_v) and depth (D_v)
  - V *corresponding source camera parameters* C_V
    - described by position (P_v) and Orientation (O_v)
  - *The camera parameters of target view* C_T
    - described by P_T and O_T
- Goal:
  - According to inputs, finding the optimal f^* from all F possible TMIV configurations to maximize a user-defined utility function $U(\cdot)$

$$f^* = argmax_{f \epsilon F} U(f)$$

Define F and $U(\cdot)$

- We define the configuration F with two essential parameter of TMIV
  - The number of passes N
  - The number of views per pass $r_n,\ n = 1,2, \ldots N$
- We define the utility function $U(\cdot)$ as the following

$$U(\cdot) = \frac{Q_T - Q_J}{Y_T}$$

Maximize Quality

Minimize running time

where

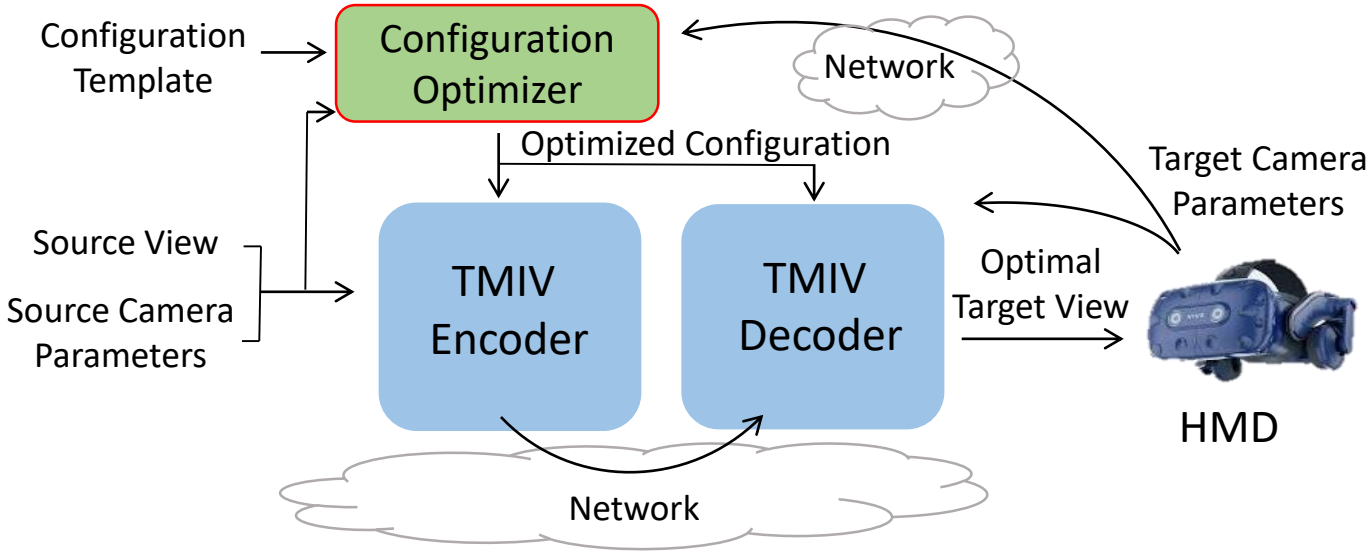
- Q_T is the video quality of the synthesized target view, and
- Q_J is the requested video quality (In our experiment, we set Q_J to 20 dB (WS-PSNR or PSNR))
- Y_T is the decoding time,

- We note that F and $U(\cdot)$ can be defined according to different scenarios

Configuration Optimizer

Configuration Optimizer

- To solve the problem, we propose configuration optimizer to generate optimal configuration f^* according to camera parameters (C_V, C_T) and source views (T_v, D_v)

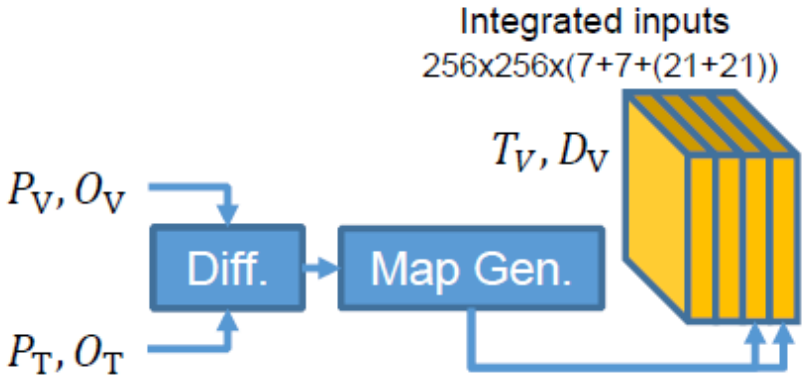


NN-based Algorithms

- Finding f^* in F is not a easy task because of the huge configuration search space (depends on C_V, C_T,T_v,and D_v)
- We utilize Neural-Network-based (NN-based) approaches to find f^*
- We treat the problem as
  - A regression problem, solving by Convolutional Neural Network (CNN)
  - A decision making problem, solving by Deep Reinforcement Learning (DRL)

Input Preprocessing

- The input data of both NN algorithms are composed of C_V, C_T, T_v, and D_v (from 7 source views)
- We down-sample both T_v, and D_v to the same resolution of 256 × 256
- We subtract C_T from C_V, then duplicating results for 256x256 times
- The resulting inputs contain
  - 256x256x7, T_v
  - 256x256x7, D_v
  - 256x256x7x6, C_V, C_T



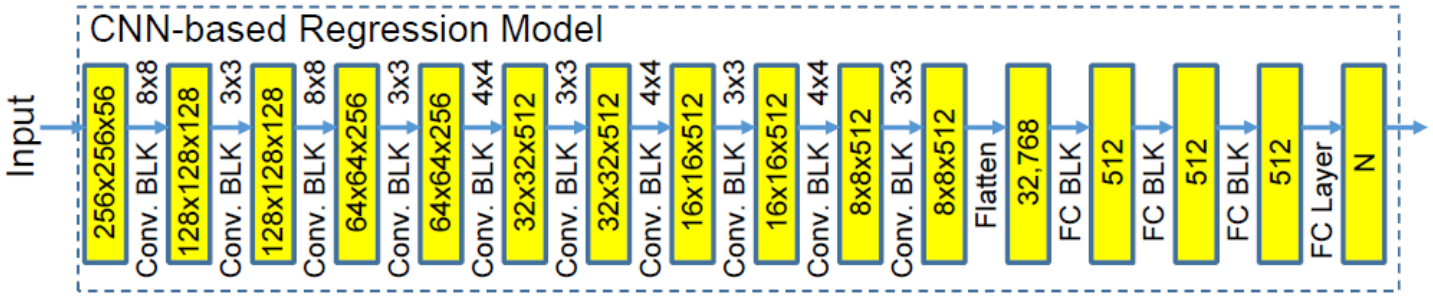
Output Post-processing

- The output of both model is a vector E, where $E = (e_1, e_2, \dots, e_N)$, and N represents the number of passes
- We set output dimension = 3 in our experiments
- E represents the number of view be added in each pass
- The number of views for each pass $r_n (n = 1, \dots N)$ is calculated by

$$r_n = \begin{cases} 0, & e_n = 0; \\ 0, & r_j = 0 \text{ and } j < n; \\ \sum_{i=1}^{n} e_i, & \text{otherwise.} \end{cases}$$

Regression Problem (Solving by CNN)

- We treat the problem as a regression problem, where the predicted target is a vector of integer
- Our CNN model include two parts
  - Convolutional network: used to extract feature
  - Fully-connected network: used to infer the results from feature
- Conv. BLK represents a convolutional layer + an ReLU activation layer
- FC. BLK represents a fully-connected layer + an ReLU activation layer



Decision Making Problem: RL Introduction

- In reinforcement learning, there are an *agent* and *environment*
- The agent performs some *action* in the environment and environment feedback *state (observation)* and *reward* to the agent
- The goal of the agent is learning how to perform actions according to states to get maximal total reward

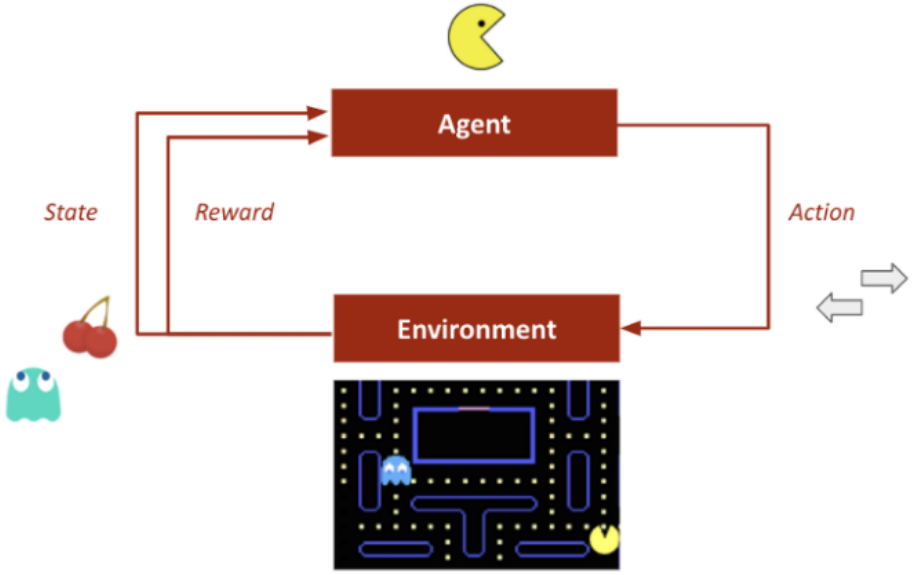
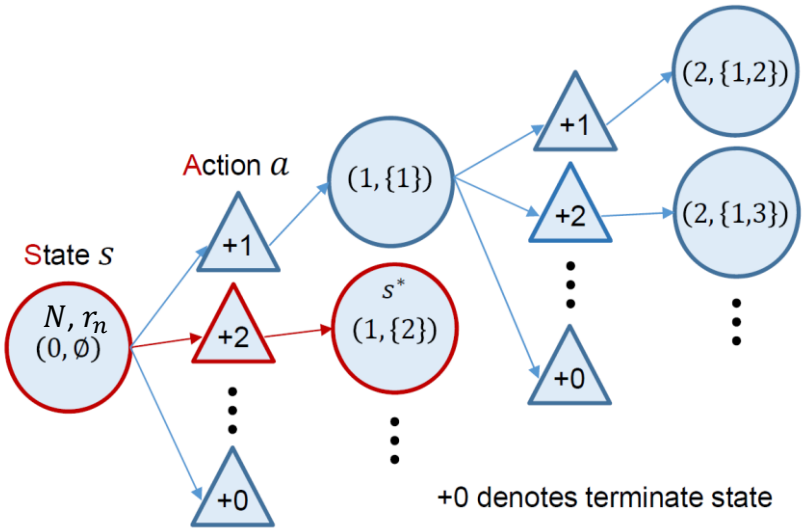


Figure from https://towardsdatascience.com/deep-q-network-combining-deep-reinforcement-learning-a5616bcfc207

Decision Making Problem (Solving by DRL)

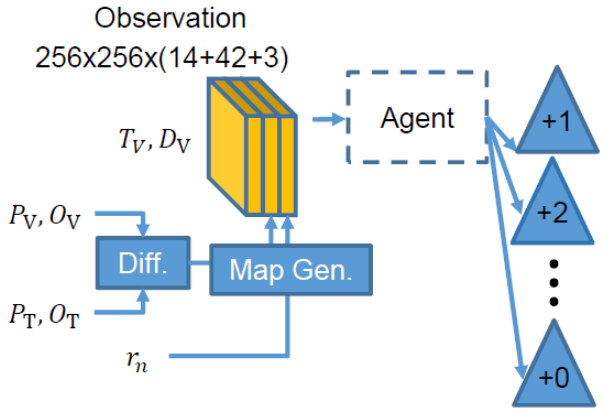
- We treat the problem as a decision making problem
- We formulate the configuration as a search space composed of **states** $s = (N, r_n, T_v, D_v, C_v, C_T)$

- **Action**: the number of extra views to reach the next state
- **Reward**: the difference of the utility values between the current state and the following state
- DRL agent start at the initial state and choose the most valuable action to move towards the optimal state



DRL Agent

- The inputs (observation) of the agent are composed of r_n, T_V, D_v, C_V, and C_T
- We adopt *Deep Q learning*, where the agent predict the future reward Q of each action
- The agent choose the action have maximal Q



Training Procedure

- CNN algorithm
  - Learning rate: 10^{-5}
  - Adopting MSE as the loss function
  - The model takes about three hours to converge
- DRL algorithm
  - Adopting target network
  - Adopting ϵ-greedy policy
  - Adopting experience replay
  - The model takes about two days to converge

Target Network

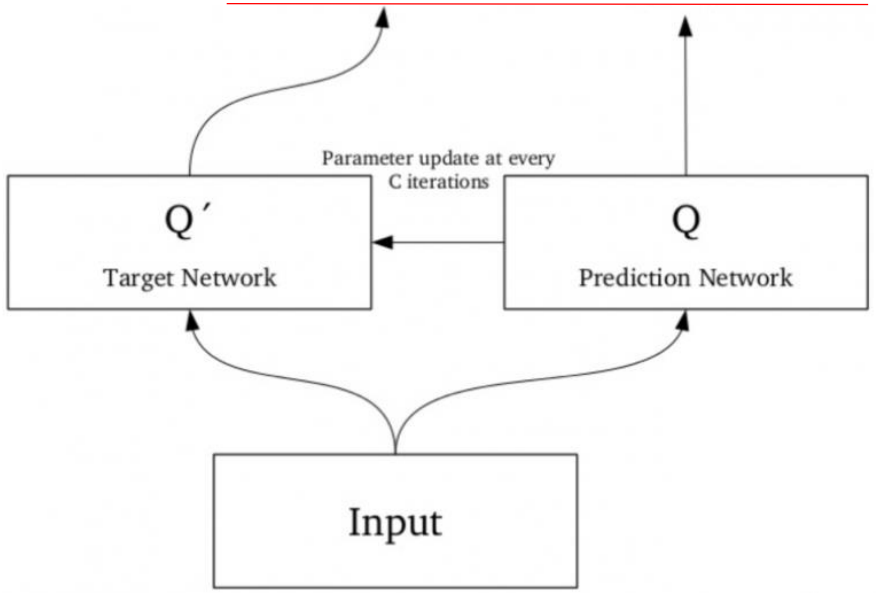
- In Deep Q learning, we have 2 neural networks model in training
- These model have same architecture but different weight
- Every m steps, the weight from prediction network are copied to target network
- Using target network approaches lead to more stability in the learning process

Original DQN:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

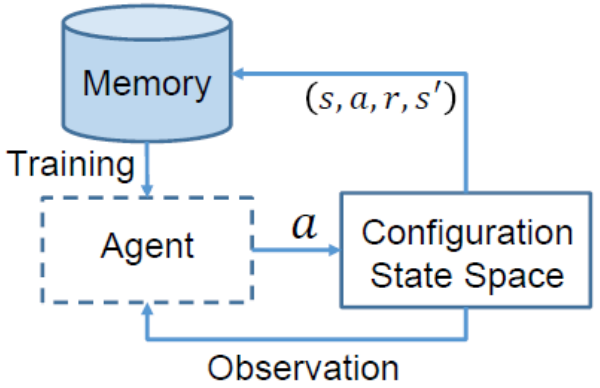
DQN + target network:

$$loss_{DRL} = (r + \gamma \max_{a'} Q(s', a'; w_i^-) - Q(s, a; w_i))^2$$



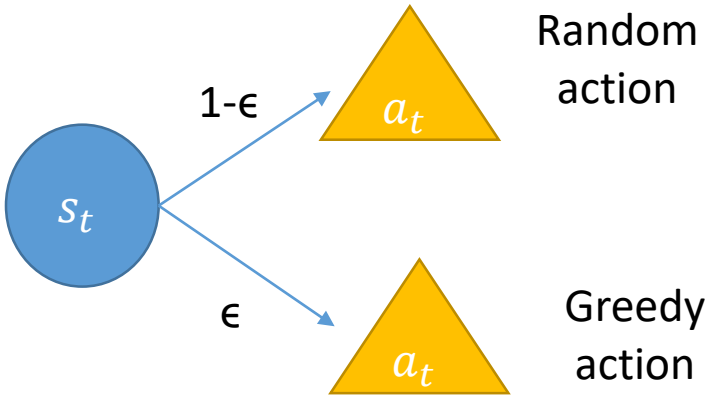
Experience Replay

- We store trajectory, which consisting of the current state s, the selected action a, the immediate reward R, and the next state s', in memory for training
- The memory is a First-In-First-Out (FIFO) queue, which queue size is 1,000
- For each training round, a data batch with 32 trajectories is randomly sampled from the memory
- Experience replay can
  - Reduces correlation between experiences
  - Increases learning speed
  - Reuses past transitions to avoid catastrophic forgetting



ϵ-greedy Policy

- The agent in DRL choose an action according to:
  - Its observation, which is output Q values of each action
  - Randomly
- In this way, agent can explore new search space to find the optimal results
- In our experiments, ϵ is increased by number of steps (~100,000: 0.5, 100,000~200,000:0.7, 200,000~:0.9)



Produce Dataset

- We run TMIV with various video sequences, r_n , and N to calculate performance under different setting
- For video sequences, we select ten video sequences from MPEG dataset
- We choose 7 videos from each video sequence as the source view
- For r_n and N, we run all of the combination when $N < 3$
- Procedure: TMIV encoder -> video encoder (h.265)->video decoder (h.265)->TMIV decoder

EquiRectangular Projection (ERP)

PerspecTive Projection (PTP)

Sequence	Projection	Resolution	Cameras	No. Frames
Classroom	ERP	4096x2048	24	300
Hijack	ERP	4096x4096	10	300
Museum	ERP	2048x2048	15	120
Kitchen	PTP	1920x1080	25	97
Painter	PTP	2048x1088	16	300
Frog	PTP	1920x1080	13	300
Fencing	PTP	1920x1080	10	250
Street	PTP	1920x1088	9	250
Carpark	PTP	1920x1088	9	250
Hall	PTP	1920x1088	9	500

Training and Testing Dataset

- In ERP video sequences:
  - Training set: all the non-source-view cameras in video sequence
  - Testing set: random camera pose from pose trace provided by MPEG
- In PTP video sequences:
  - We select all the non-source-view cameras as the target view to generate data
  - We adopt leave-one-out strategy to split training and testing set

Objective Evaluations

Experiment Setup

- Baselines:
  - Default (DEF): default configuration of TMIV
  - Optimal (OPT): configuration with optimal utility value
- Performance metrics:
  - Number of required views
  - Video quality (WS-PSNR, PSNR)
  - Decoding time
  - Optimal score ($\frac{utility\_value}{optimal\_utility\_value}$)

Qualitative Evaluations

- Most of synthesized results are similar
- However, the CNN and DRL algorithms sometimes generate noticeable distortion because of insufficient number of source views

DEF

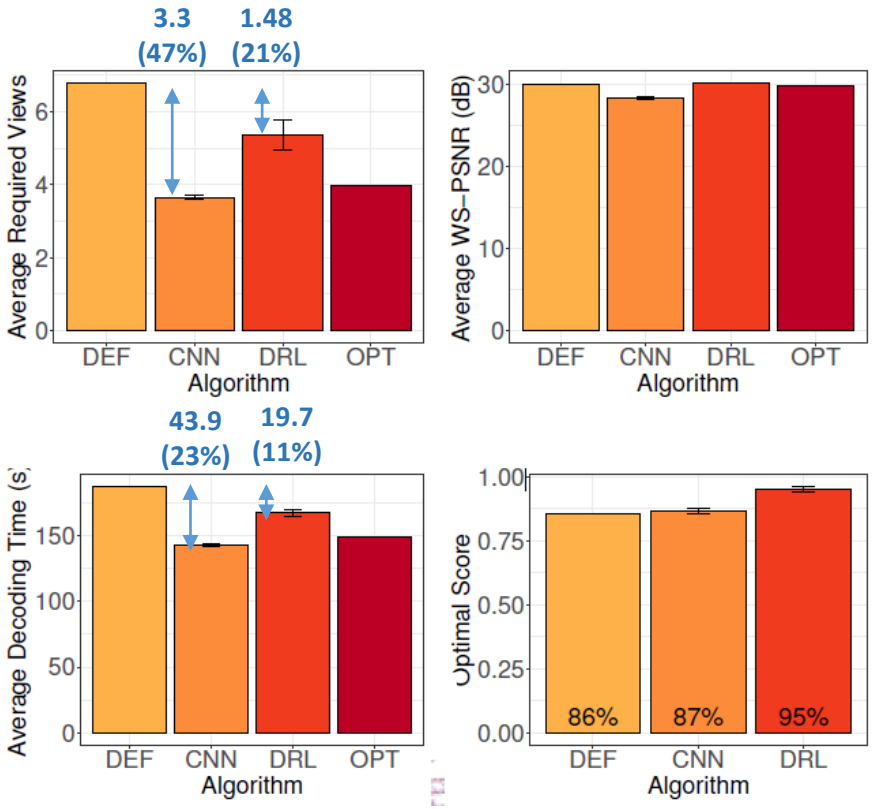
CNN

DRL

OPT

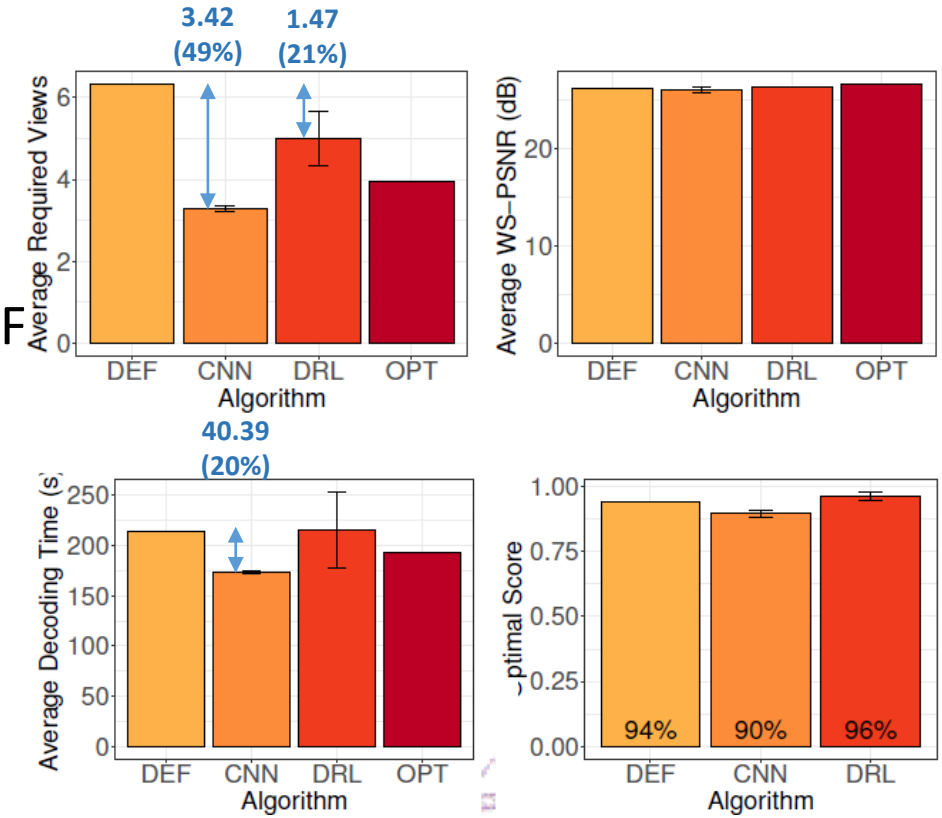
Evaluation Results (ERP, Training Set)

- Our algorithm require fewer views and decoding time
- The difference of quality between our algorithms and baseline is small (std.: 0.5)
- In optimal score, our algorithms achieve higher score, and DRL achieve large improvement
- Overall, our algorithms outperform baseline in ERP training set, and DRL achieve the best performance



Evaluation Results (ERP, Testing Set)

- The results of testing set show the trend similar with training set
- In optimal score, CNN algorithm achieve lower optimal score than DEF
- Overall, our algorithms required fewer views than DEF, and DRL has better performance than other algorithms

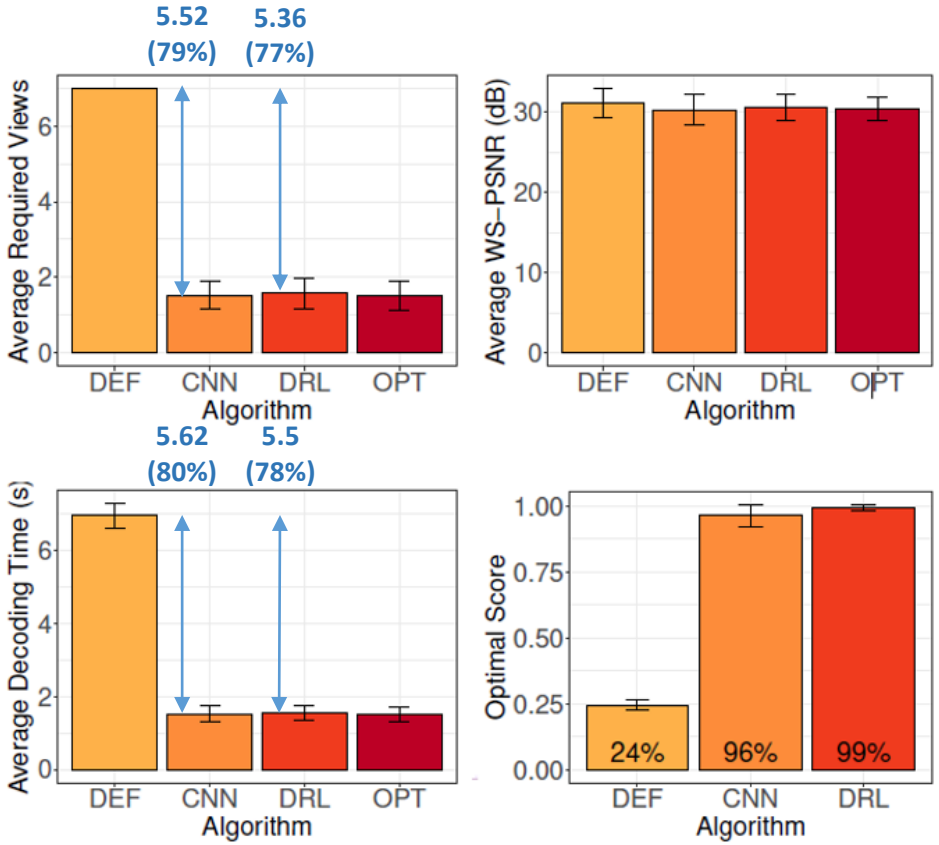


Summary, in ERP video sequences

- **DRL algorithm perform well for various user positions and orientations.**
- **CNN algorithm results in inferior performance when facing new user positions and orientations.**

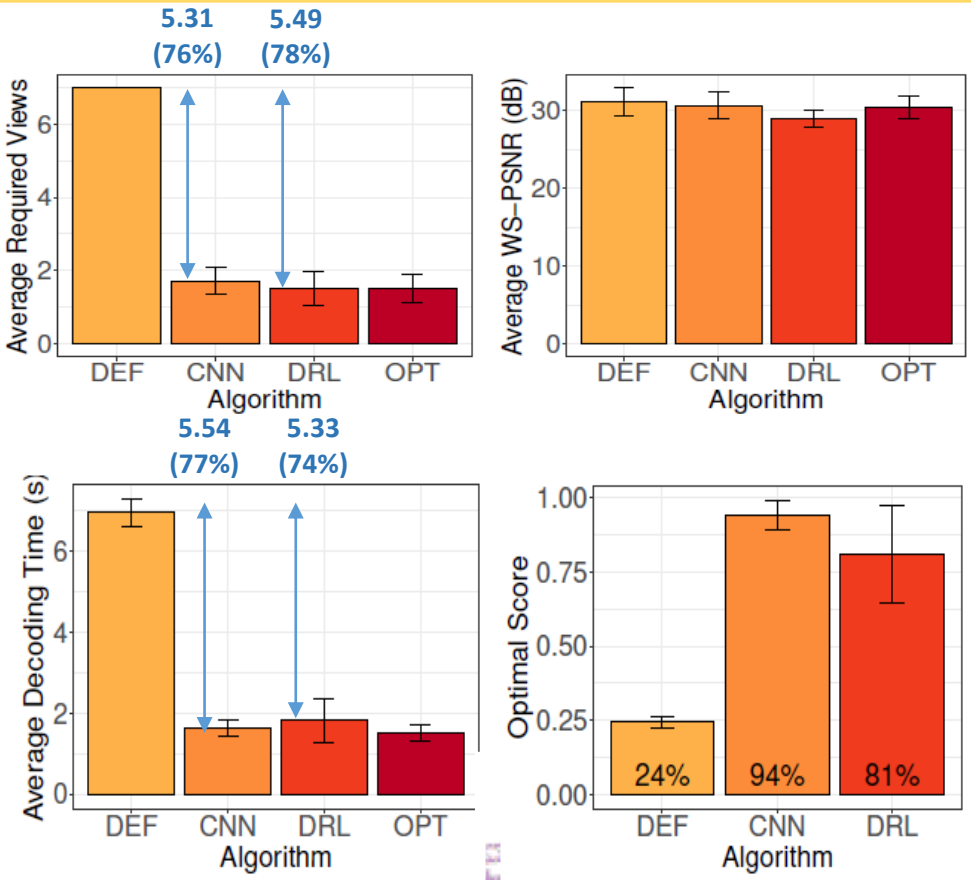
Evaluation Results (PTP, Training Set)

- Our algorithms require fewer source views and less decoding time than DEF algorithm
- They also achieve comparable quality to baselines (Std.:0.37)
- In optimal score, our algorithms outperform DEF
- Overall, the performance of our algorithms better than DEF, and it very close to OPT



Evaluation Results (PTP, Testing Set)

- Similar to training set, our algorithm outperform DEF algorithms in required view, decoding time, and optimal score
- DRL achieve lower optimal score than CNN, and it have higher variance compared to the results in training set
- Overall, our algorithms outperform DEF, and the performance of DRL shows higher variance in the testing set

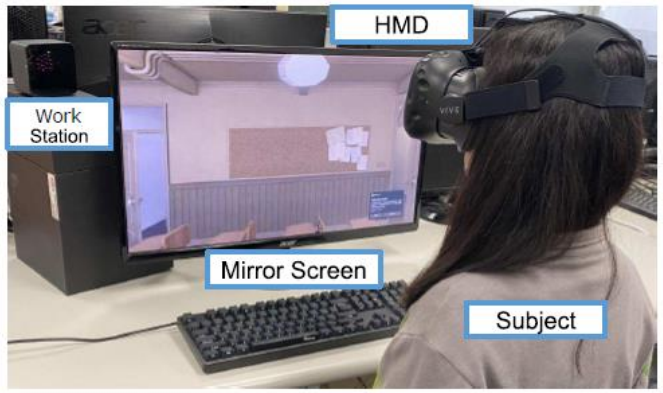


Summary, in PTP video sequences

- **The CNN and DRL algorithms perform well for various video sequences and user positions and orientations**
- **The CNN algorithm leads to more stable performance with new video sequences**

Subjective Evaluations

Experiment Setup



- We conduct subjective evaluations on both ERP and PTP video sequences to study the perceived quality of our algorithms and baselines
- For ERP video sequences,
  - We choose 6 camera parameter settings from each video sequences (3 from training set, 3 from testing set)
  - We ask user watch synthesized results through HMD
  - 6 camera parameter settings x 3 video sequences (18 rounds)
- For PTP video sequences,
  - We choose 4 camera parameter settings from each video sequences
  - We ask user watch synthesized results through 27'' 2D monitor
  - 4 camera parameter settings x 7 video sequences (28 rounds)
- In each round, subject have to rank the synthesized results from different algorithms
- We recruit 23 subjects for both ERP and PTP experiments

Sample Synthesized Results

DEF

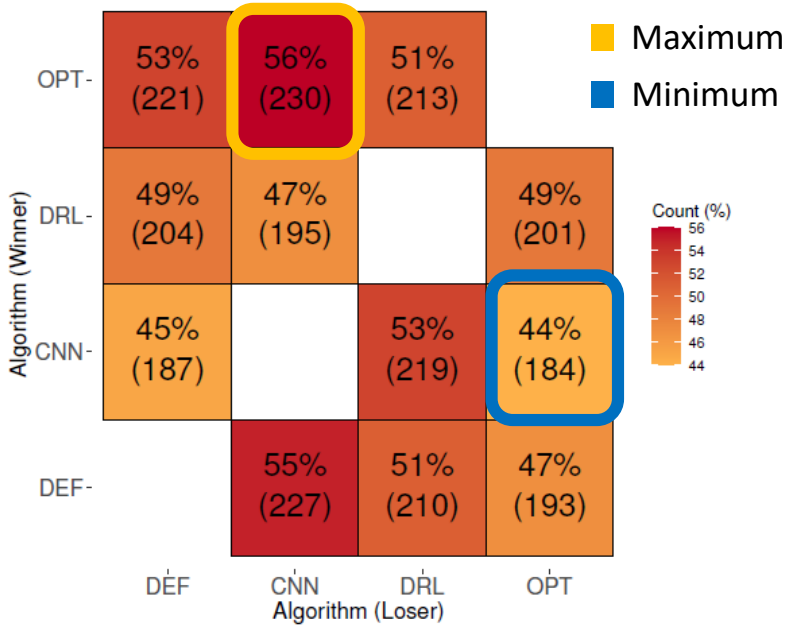
CNN

DRL

OPT

Evaluation Results (ERP)

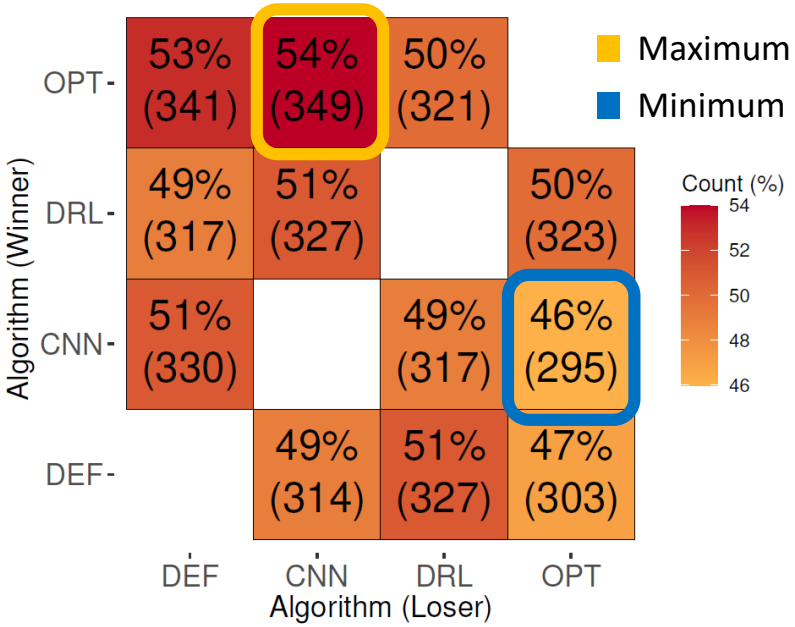
- The pairwise comparisons show no obvious superior algorithm
- We model the pairwise comparisons and the ranking results using the Bradley-Terry and Plackett-Luce model
- The estimated coefficients of both models range between -0.1 to 0.09 and the corresponding p-values of the coefficients are greater than 0.15
- Overall, the perceived quality of our algorithms and baseline are similar



		DEF	CNN	DRL	OPT
Bradley-Terry	Coefficient	0.00	-0.10	-0.07	0.08
	p-value	N/A	0.16	0.30	0.24
Plackett-Luce	Coefficient	0.00	-0.07	-0.05	0.09
	p-value	N/A	0.45	0.58	0.31

Evaluation Results (PTP)

- The pairwise comparisons show no obvious superior algorithm
- The estimated coefficients of both models range between -0.003 to 0.16
- The OPT algorithm is statistically superior to the DEF algorithm, but this result can be neglected
- Overall, the perceived quality of our algorithms and DEF are similar



		DEF	CNN	DRL	OPT
Bradley-Terry	Coefficient	0.00	-0.003	0.04	0.10
	p-value	N/A	0.96	0.52	0.06
Plackett-Luce	Coefficient	0.00	0.07	0.02	0.16
	p-value	N/A	0.34	0.76	0.02

Summary of Our Findings

Summary

- Our evaluations results show that our algorithms require *fewer source views and less computational resources* to deliver *comparable video quality and perceived quality* compared to the baseline algorithms
- According to our recommendation, our algorithms can reduce the number of views and decoding time and maintain similar video quality

Type	Dataset	Rec.	Compared to DEF			
			No. of Views	Video Quality	Decoding Time	Utility Value
ERP	Training (Seen Camera Parameters)	DRL	79%	100%	89%	+9%
	Testing (New Camera Parameters)	DRL	79%	101%	100%	+2%
PTP	Training (Seen Video Sequences and Camera Parameters)	DRL	23%	98%	22%	+75%
	Testing (New Video Sequences and Camera Parameters)	CNN	24%	98%	23%	+70%

Conclusion and Future Direction

Conclusion

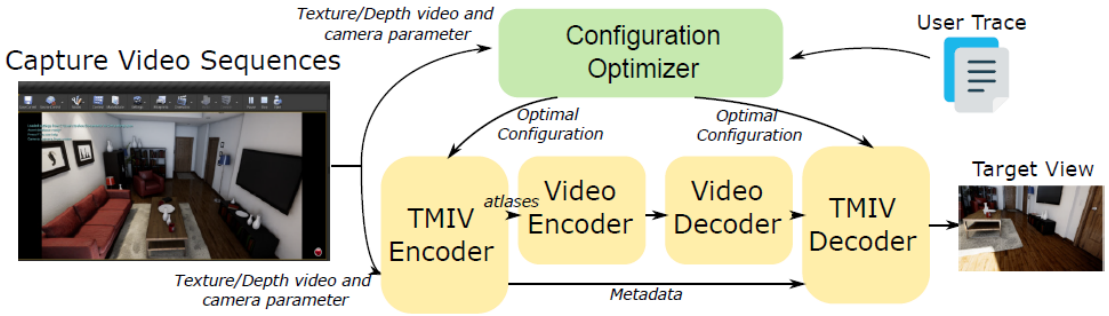
- In this thesis, we propose two ML-based configuration optimization algorithms (CNN and DRL algorithms)
- Our algorithms work with user-specified utility function and computer best configuration for TMIV
- We conduct both objective and subjective experiments to evaluate our algorithms
- Our experiment results show that our algorithm can reduce the resource requirement and achieve comparable quality to default configuration

Future Directions

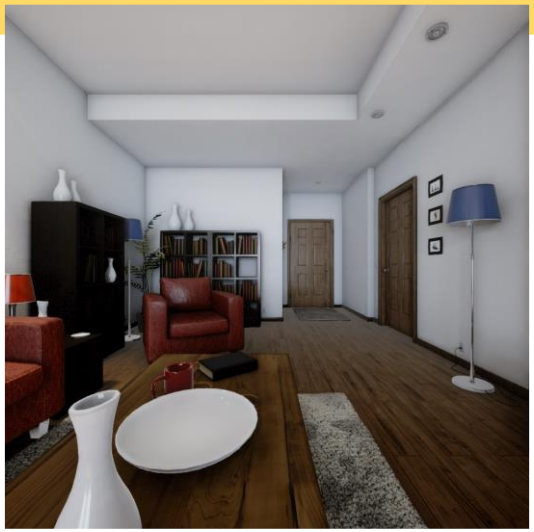
- Several system challenges need to be addressed, such as
  - The inference frequency of our algorithms
  - The performance under diverse and dynamic network and system condition

Use Case: Real Estate Virtual Tour

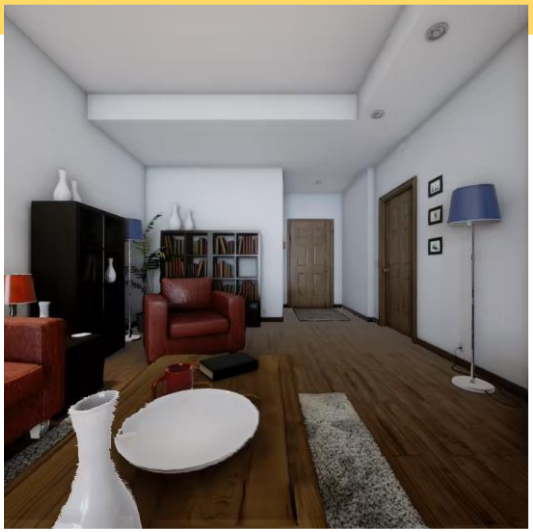
- To show the feasibility of our algorithms, we build a capturing system to capture more video sequences from various scenes
- We build our system based on Airsim, which is an open-source drone simulator
- Our system inputs camera parameters (e.g., position, rotation, FoV, and resolution), and outputs the video sequence and corresponding camera parameter in the required format of MIV codec
- The experiment results in the new dataset are similar to previous experiments



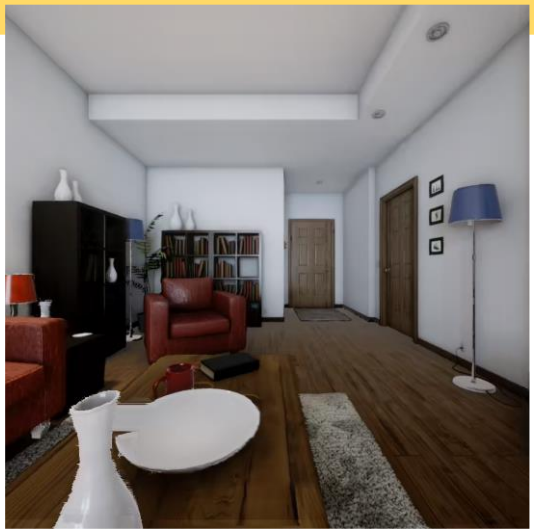
Sample Synthesized Results



Ground Truth



DEF



CNN

DRL

OPT

Thank you

Tse-Hou Hung (tsehou.nthu@gmail.com)

Publications:

- **C. Hsu, T. Hung, and C. Hsu, "Optimizing immersive video coding configurations using deep learning: A case study on TMIV," ACM Transactions on Multimedia Computing, Communications, and Applications, June 2021, Accepted to Appear.**
- C. Fan, T. Hung, and C. Hsu, "Modeling the user experience of watching 360◦ videos with head-mounted displays," ACM Transactions on Multimedia Computing, Communications, and Applications, April 2021, Accepted to Appear.