Improving user experience in free-navigation via image synthesis

Nour Hobloss

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

Nour HOBLOSS

Improving User Experience in Free Navigation via Image Synthesis

Thèse présentée et soutenue à Rennes, le le 2 Décembre 2021
Unité de recherche : IRISA
Thèse N°

Composition du jury :

Rapporteurs :	Frédéric DUFAUX	Directeur de Recherche, CNRS, CentraleSupélec
	Anissa MOKRAOUI	Professeur des Universités, Institute of Galilée Paris
Examinateurs :	William PUECH	Professeur des Universités, University of Montpellier
	Marco CAGNAZZO	Professeur, Telecom-Paris (Directeur de thèse)
	Lu ZHANG	Maître de Conférences (HDR), INSA Rennes (Directrice de thèse)
Invités :	Nicolas RAMIN	Ingénieur R&D, IRT b-com
	Jean-Yves AUBIE	Directeur du laboratoire, IRT b-com

Intitulé de la thèse :

Amélioration de l'expérience utilisateur en navigation libre via la synthèse d'image

-

Improving the User Experience in Free Navigation via the Image Synthesis

Nour HOBLOSS

UNIVERSITE BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE RENNES 1

En partenariat avec :

b com

This work has been achieved within the Institute of Research and Technology b<>com, dedicated to digital technologies. It has been funded by the French government through the National Research Agency (ANR) Investment referenced ANR-A0-AIRT-07.

Acknowledgements

I would like to thank my Ph.D. directors Marco Cagnazzo and Lu Zhang, for their guidance throughout the thesis. They helped me to be more insightful and ambitious with my research. I am incredibly thankful to have Marco as a Ph.D. director who was there from the beginning of this work, with who I learned and improved. I also thank him for his constant motivation, always encouraging, never criticizing. I also feel lucky to have Lu as a Ph.D. director for her constant help, understanding, reactivity, and motivation. I am particularly grateful to my Ph.D. supervisors Attilio Fiandrotti, Stéphane Lathuilière, and Nicolas Ramin for their deep involvement in my research. I feel lucky to have been supervised by them, and I am convinced that I could not have achieved all this work without their help. I am also grateful to professor Remi Cozot and Andrei Purica for their sincere engagement at the commencement of my Ph.D. I believe I could not have found my first research directions without their help and valuable advice. I also would like to thank my thesis committee: Frédéric Dufaux, Anissa Makraoui, William Puech, for agreeing to evaluate my work. Their constructive feedback helped me to improve the manuscript and point out new perspectives. I would like to thank my co-workers Amine, Nam Duong, and Sylvain D. for sharing their technical expertise in computer vision and machine learning throughout my thesis. I would like to thank b<>com and Telecom Paris, particularly my two labs, for providing all the necessary materials, workstations, and delightful and comfortable workspaces. I would like to thank my fellow Ph.D. Anas, Glenn, Cédric, Fatemeh, Charles, Hiba, Anthony, Pavel, Yulia, and Li for their mutual support during the challenging thesis and interesting talks about our different Ph.D. topics. I am incredibly grateful for my friends Glenn, Anas, and Cédric being there during all the phases of the thesis, listening, helping, advising, and continuously supporting. A special thanks to my friend and co-worker Patrick Gioia, who helped me make it until the end with his positive vibes, constant support, and insightful advice. I would like to thank my co-workers from b<>com who have become close friends, Cambodge, Antonin, and Nicolas D. I would like to thank my lab co-workers, David, Nicolas, Tania, Franck, and Aude. Last but not least, I would like to thank my parents Dalal and Khaled, my sister Iftikhar and my brothers Samir, Samer, and Ali, to whom I dedicate this work, without whom, I could not have achieved this work. I would never thank them enough. Finally, I would like to thank my partner Alen for his constant support, patience, and understanding throughout my thesis.

Résumé en Français

1 Introduction

La synthèse de vues est actuellement un de champs de recherche en vision par ordinateur. Elle vise à créer de nouvelles vues d'une scène capturée à partir de plusieurs points de vue. Le problème est, étant donné plusieurs images d'un objet particulier prises à partir de points spécifiques dans la scène avec des réglages et des orientations spécifiques de la caméra, essayer de créer une image synthétique à partir d'une caméra virtuelle placée à une position différente dans la même scène. Les domaines de la CVR et de l'intelligence artificielle (IA) sont impliqués dans la définition d'approches appropriées au problème qui consiste à créer une image synthétique à partir d'une caméra virtuelle. Un exemple d'application de la synthèse de vues est la Free Viewpoint Television (FTV). La FTV est un système de visualisation d'une vidéo naturelle qui permet à l'utilisateur de contrôler de manière interactive le point de vue et de générer de nouvelles vues d'une scène dynamique à partir de n'importe quelle position 3D. Dans la FTV, le centre d'attention peut être contrôlé par les spectateurs plutôt que par un réalisateur, ce qui implique que chaque spectateur peut observer un point de vue unique.

La capture Multi-View Video (MVV) change de la prise en charge partielle (généralement environ 30 degrés) à la prise en charge intégrale (360 degrés) de la scène. Par conséquent, il est possible de générer des vues stéréoscopiques appropriées pour une visualisation avec un écran 3D ou d'autres méthodes 3D. Les systèmes équipés de plusieurs caméras physiques peuvent saisir des images avec une plus grande couverture de la scène observable. Cependant, certaines régions seront occultées d'une façon permanente, quel que soit le point de vue. Un nombre plus important de caméras devrait permettre d'obtenir un résultat de haute qualité car moins d'interpolations sont nécessaires. Dans cette thèse, nous étudions les effets de transition vidéo dans une scène capturée MVV pour passer virtuellement d'un point de vue réel à un autre dans la même scène. Ainsi, lors du déroulement d'une vidéo, le spectateur ne peut s'arrêter que sur des vues réelles et faire une transition naturelle entre elles en utilisant des vues virtuelles synthétisées. La génération virtuelle de ce type de

transitions permet de réduire un nombre important de caméras dans une scène en les remplaçant par des caméras virtuelles. Elle réduit ainsi le coût de la mise en place des MVV dans une seule scène. En même temps, elle offre au spectateur une navigation interactive et naturelle d'un point de vue à un autre comme si la caméra était en mouvement. Moins le nombre de caméras réelles est élevé, moins les dépenses nécessaires à la capture du MVV sont importantes, mais plus la distance inter-caméras est grande. Comme approche à notre problème, il existe de nombreux types de transition pour joindre deux points de vue dans une scène, depuis la plus classique comme la transition *Coupure*, où un plan passe au plan suivant dans une vidéo sans aucun effet, jusqu'aux plus récentes comme les méthodes de transition basées *View Synthesis*. Les méthodes de synthèse de vues telles que le warping 3D utilisent les informations 3D de la scène (par exemple, la carte de profondeur) pour reconstruire un nouveau point de vue; Il semble donc que ces méthodes soient les plus prometteuses pour générer les transitions que nous visons et qui ont retenu notre attention dans notre étude. Cependant, les méthodes de synthèse de vue existantes souffrent toujours d'artefacts visuels dans l'image rendue finale, dus aux occultations dans la nouvelle position du point de vue.

En effet, un nombre restreint de caméras dans une scène élargit la distance à partir du point de vue virtuel, ce qui implique une multiplication des informations manquantes dans les zones non perçues par les caméras de référence et perçues par la nouvelle caméra à reconstruire. Ces contraintes entraînent un inconvénient majeur dans les méthodes proposées pour gérer le problème des grandes occultations. L'objectif principal de la thèse est d'offrir au spectateur une expérience interactive, où il peut librement changer de point de vue tout en regardant une vidéo. Le principal enjeu est de diminuer le nombre de caméras dans la scène en conservant les points de vue essentiels et en reconstruisant une transition fluide entre les caméras réelles.

Les méthodes de transition basées sur la synthèse de vues semblent être les plus naturelles et les plus réalistes par comparaison avec les méthodes plus classiques qui n'utilisent pas une reconstruction 3D de la scène. Alors que les méthodes traditionnelles de synthèse de vues ont duré plus de deux décennies, les méthodes de synthèse de vues basées machine learning ont connu un essor considérable au cours des deux dernières années, en raison des résultats exceptionnels obtenus dans ce domaine. Les faiblesses des méthodes de synthèse de vues existantes dans la littérature ont encouragé l'exploration des méthodes de synthèse de vues pour construire une transition naturelle et fluide entre deux points de vue dans une scène. Nous avons étudié les méthodes algorithmiques et les méthodes basées machine learning utilisées pour la synthèse de vues, en essayant de tirer le meilleur parti des deux approches et de développer une méthode optimale de synthèse de vues pour la synthèse de vues à grande baseline. De plus, nous gardons l'expérience de l'utilisateur au centre de notre étude puisque le spectateur est l'utilisateur final de notre application. C'est pour cette raison que nous avons réalisé une série de tests subjectifs pour évaluer les différentes méthodes de transition sur la base de jugements humains.

2 Contributions

La première partie de ce travail vise à mettre au point une nouvelle approche pour la synthèse de vues à large baseline. À cette fin, nous avons introduit de nouvelles architectures de réseaux de neurones pour l'étape de fusion des vues tout en utilisant une méthode algorithmique de pointe pour l'étape de warping des vues de référence du processus de synthèse des vues. Cette méthode vise à réduire la taille significative des filtres du réseau pour warper les images de référence, réduisant ainsi la complexité. Deux approches différentes sont étudiées dans cette partie de la thèse:

- **Une approche hybride pour la synthèse de vues à large base avec ConvNet (CNN-VB)**:

  Les Convolutional Neural Networks (ConvNets) ont été récemment utilisés [1–3] pour mettre en œuvre des architectures de synthèse de vues complètes de bout en bout, du warping de la vue de référence au blending de la vue cible tout en traitant également les occultations. Cependant, la taille des filtres de convolution doit augmenter avec la distance entre les vues de référence, ce qui rend les approches de convolution trop complexes pour les configurations à large baseline. Dans ce travail, nous proposons une approche hybride de la synthèse de vues où nous faisons d'abord un warp des vues de référence en corrigeant les occultations. Ensuite, nous développons une architecture à convolution plus simple pour mélanger les vues pré-processées. En faisant warper les vues de référence, nous diminuons la distance équivalente entre les vues de référence, ce qui permet d'utiliser des filtres convolutionnels plus petits et donc de réduire la complexité du réseau. Nous proposons également un algorithme de remplissage des trous pour combler les désocclusions dans les vues warpées. Nous montrons de manière expérimentale que notre méthode se comporte favorablement par rapport aux méthodes de synthèse traditionnelles et de convolution, tout en conservant une complexité moins importante par rapport à ces dernières.

- **Blender hybride à double flux pour la synthèse de vues à large baseligne (HDSB):** La navigation libre dans une scène nécessite de warper certaines vues de référence vers un point de vue cible souhaité et de les fusionner pour synthétiser une vue virtuelle. Les méthodes basées sur les ConvNets peuvent apprendre conjointement les tâches de warping et de blending. Ces méthodes, comme [4], sont souvent conçues pour une distance de baseline inter-caméras limitée, et des kernels plus grands sont nécessaires pour le warping si la distance de baseline augmente. Les méthodes algorithmiques peuvent gérer de grandes distances de baseline ; cependant, la vue synthétisée souffre d'artefacts au voisinage des pixels disocclus. Nous présentons une approche hybride dans laquelle les vues de référence sont warpées de manière algorithmique vers la position cible, puis fusionnées via un ConvNet. La déformation préliminaire des vues

permet de réduire la taille des noyaux convolutifs, et donc le nombre de paramètres apprenables. Nous proposons un encodeur-décodeur résiduel pour le fusionnement d'images avec un encodeur siamois pour maintenir les paramètres à un nombre faible. Nos expériences de synthèse de vues sur des séquences multivues réelles montrent une meilleure qualité d'image objective que les méthodes de l'état de l'art en raison de la diminution des artefacts dans les images synthétisées.

Les contributions de cette thèse se concentrent donc sur la qualité d'expérience de l'utilisateur pour la transition vidéo et l'ensemble de données. Premièrement, nous réalisons un dataset créatif pour la qualité de l'expérience de transition vidéo. Ensuite, notre travail vise à évaluer subjectivement les approches de synthèse de vues proposées sur 8 séquences vidéo différentes en réalisant une série de tests subjectifs.Ainsi, les contributions de cette partie sont divisées en deux segments :

- **Une base de données vidéo stéréoscopique multi-vues avec écran vert (MTF & Pandemonium) pour l'évaluation de la qualité d'expérience de la transition vidéo:**

  Nous présentons tout d'abord une base de données vidéo stéréoscopique multi-vues avec écran vert, appelée MTF et Pandemonium, destinée à être utilisée dans des applications de vision par ordinateur, en particulier pour la navigation libre, la télévision à vue libre et l'évaluation de la qualité d'expérience des transitions vidéo. Le MTF contient des vidéos full-HD de scénarios réels composés de 3 scènes. Une particularité de ce dataset est que pour comprendre son storytelling ; les utilisateurs doivent changer leur point de vue dans la scène à un moment donné. À cette fin, nous devons le plus souvent générer une transition pour joindre deux points de vue dans la même scène. Les techniques de vision par ordinateur qui permettent de telles transitions, comme les méthodes de synthèse de vues, s'appuient sur un ensemble d'images de la scène pour générer de nouvelles vues à partir de différents points de vue. Cependant, ces méthodes peuvent présenter de nombreux cas de défaillance qui entraînent des artefacts dans la transition vidéo finale restituée. Dans la plupart des tests de synthèse de vues, le contenu n'est pas conçu pour rendre la transition entre deux points de vue valide ou captivante pour les spectateurs, par exemple, ils n'ont pas besoin de faire une transition pour obtenir plus d'informations afin de mieux comprendre le contenu. Nous supposons donc que les participants jugeront sévèrement les artefacts et les imperfections de la transition effectuée. Ainsi, le MTF est censé mieux analyser l'impact visuel des artefacts persistants dans la transition finale restituée. Dans notre dataset, toutes les scènes sont enregistrées dans un studio à écran vert, qui est souvent utilisé pour ajouter des effets spéciaux et des décors lors du montage selon des besoins spécifiques. Notre jeu de données présente également une large configuration de caméras de référence, une contrainte difficile pour les techniques de synthèse de

vues. Enfin, The MTF & Pandemonium peuvent également être utilisés comme un ensemble de données complémentaires à d'autres dans la littérature dans diverses applications de vision par ordinateur, telles que la compression vidéo, le contenu vidéo 3D, l'environnement immersif de réalité virtuelle, l'estimation du flux optique...

- **MUlti-view Synthesis Enhancer (MUSE)** :

  Enfin, nous proposons une extension des solutions proposées précédemment pour la synthèse de vues. Cette approche propose d'améliorer les vues warpées pré-fusionnées en utilisant la méthode de l'encodeur Hybrid Dual Stream Blender (HDSB) puisque maintenant nous n'avons qu'une seule image d'entrée au lieu de deux. Par conséquent, le réseau ne fusionne pas les vues mais corrige les artefacts d'image provenant de l'image pré-fusionnée et améliore la qualité de l'image finale. Nous proposons également un processus d'apprentissage intra-contenu, dans lequel nous réalisons un training pour chaque contenu. La pré-fusion des textures déformées et non peintes provenant des vues latérales de référence garantit, en particulier, une fusion uniforme des zones non-lambertiennes de l'image.

3 Conclusion

Malgré les nombreuses innovations apportées aux méthodes de synthèse de vue pour générer une vue nouvelle sans artefacts, la gestion des occultations pour les caméras à large baseline reste une tâche difficile dans la communauté de la vision par ordinateur. En effet, l'introduction de technologies basées sur l'apprentissage profond a permis au domaine de la synthèse de vue de voir à nouveau la lumière après les échecs constants des approches algorithmiques traditionnelles. Cependant, la demande en calcul des méthodes de synthèse de vues basées sur l'apprentissage est encore plus importante, surtout pour des cas d'utilisation évolutifs.

L'objectif de cette thèse est donc, d'une part, d'étudier une approche de synthèse de vues légère, à la fois algorithmique et basée sur l'apprentissage, qui ne soit pas coûteuse en temps et en mémoire. Néanmoins, en même temps, cette approche peut générer un nouveau point de vue plausible pour une large base de données. D'autre part, cette thèse visait également à mettre l'utilisateur au centre des investigations, ce qui nous a conduit à une série de tests subjectifs permettant à l'utilisateur d'évaluer les différentes méthodes de synthèse de vues proposées sur des cas réels, et de faire une préférence sur un type de transition vidéo. En conséquence, nous avons été encouragés à créer une base de données plus adaptable qui est un ensemble de données stéréoscopiques multi-vues pour la qualité d'expérience des transitions vidéo.

Table of contents

List of figures

List of tables

Acronyms

3DoF	3 Degrees Of Freedom
3DoF+	3 Degrees Of Freedom Plus
6DoF	6 Degrees Of Freedom
ACR	Absolute Category Rating
ADN	Appearance Decoder Network
AI	Artificial Intelligence
ANOVA	ANalysis Of VAriance
AR	Augmented Reality
BMMCC	Blackmagic Micro Cinema Camera
CG	computer-generated
ConvNets	Convolutional Neural Networks
CTC	Common Test Conditions
CVR	Computer Vision Research
DCT	Discrete Cosine Transform
DIBR	Depth-Image-Based-Rendering
DoF	Degrees of Freedom
FTV	Free Viewpoint Television
GAN	Generative Adversarial Network
GMRF	Gauss-Markov Random Field
GVS	Generative View Synthesis
HD	High Definition

HDMI High-Definition Multimedia Interface
HDR High Dynamic Range
HDSB Hybrid Dual Stream Blender
HEVC High Efficiency Video Coding
HMD Head-Mounted Display

I2I Image-To-Image
IBR Image-based-Rendering
ITU International Telecommunication Union

LDI Layered Depth image
LTC Linear TimeCode
LTN Layered Translation Network

MIV MPEG Immersive Video
MOS Mean Opinion Score
MPEG Moving Picture Experts Group
MPEG-I MPEG-Immersive
MPI Multi-Plane Image
MR Mixed Reality
MSE Mean Squared Error
MUSE MUlti-view Synthesis Enhancer
MVD Multi-View-Plus-Depth
MVV Multi-View Video

PSNR Peak Signal to Noise Ratio
PVM Per-View Meshes

QoE Quality Of Experience

ReLu Rectified Linear Units
RVS Reference View Synthesis

SD Standard Definition
SDR Standard Dynamic Range
SFM Structure From Motion
SID Society for information display

SLERP Spherical Linear intERPolation
SUN Semantic Uplifting Network

Tanh Hyperbolic Tangent

VGA Video Graphics Array
VR Virtual Reality
VS View Synthesis
VSRS View Synthesis Reference Software
VVS Versatile View Synthesizer

XR Extended Reality

Chapter 1

Introduction

1.1 Context and challenges

1.1.1 Free Navigation

Visual media has a significant impact on our society and culture given all the services it provides (digital images, digital videos, video games, websites, social media, digital data, videos tapes...) and it can be created, viewed and broadcast on all electronic devices (TV, smartphones, computers ...). Since it requires high memory, storage and bandwidth, advances in data compression and video coding made digital multimedia distribution and streaming practically possible, allowing for billions of people to share, create, modify and store visual media via personal computers and smartphones. More specifically, International Standardization Organizations had provided Discrete Cosine Transform (DCT)-based practical video coding formats, most notably the MPEG video formats. Moreover, the evolution of visual media types has been invested heavily in terms of image resolution (Standard Definition (SD) $\rightarrow$ High Definition (HD) $\rightarrow$ 4K $\rightarrow$ 8K), image dynamic range (Standard Dynamic Range (SDR) $\rightarrow$ High Dynamic Range (HDR)), but also in realism, depth sensation and interactivity (2D $\rightarrow$ 3D $\rightarrow$ 4D). The 3D depth impression in visual media gained popularity among the users in particular in video games, cinema and home TVs. However, these systems produce 2D/3D scenes that can be only seen from one fixed point of view and do not allow the users to choose the viewpoints of the captured 2D/3D scene.

In order to further enahnce user's experience, advanced work in visual multimedia technologies and computer graphics have enabled the development of new types of visual media, such as the free viewpoint video media and the immersive media. The former allows the user to freely and interactively change his viewpoints within a multi-view captured 2D/3D scene. The latter, also referred to as Extended Reality (XR), embraces the Virtual Reality (VR), Augmented Reality (AR), Mixed Reality (MR) and 360-degree videos, where the real world is simulated in a digital world.

The term Degrees of Freedom (DoF) refers to the movement of the head in space, or how we account for the head's position and orientation. Fig. 1.1 illustrates the three different degrees of freedom the user can have during an immersive experience : 3 Degrees Of Freedom (3DoF), 3 Degrees Of Freedom Plus (3DoF+) and 6 Degrees Of Freedom (6DoF). The first ones and simplest head-mounted display offers 3DoF which allows one to rotate the head over the three axis: Pitch, Roll and Yaw, with Pitch, tilting forward and backward on the X-axis. Yaw, turning left and right on the Y-axis, and Roll tilting side to side on the Z-axis. 3DoF+ and 6DoF come with more advanced head-mounted displays and three different movements. The 3DoF+ allows the user to move the head and the chest left and right on the X-axis, up and down on the Y-axis, forward and backward on the Z-axis, with no body displacement of the user. The 6DoF offers the swaying movement that allows the user to move left and right on the X-axis. The heave moves up and down on the Y-axis, and the surge moves forward and backward on the Z-axis, while supporting unlimited displacements of the user in the scene.

The 360° videos can be viewed with a 3DoF headset. The 3DoF+ video navigation has not yet been democratized but already mentioned [13]. The 6DoF navigation is, however, a very active research topic.

The higher the number of DoF means the more one can engage with the movement. Degrees of freedom, together with stereoscopic and spatial audio, intensify the sense of presence. The greater the DoF, the greater the experience in XR, because the greater you can suspend disbelief from the real to the virtual and engage with the content, which led us to the notion of parallax. The parallax is the horizontal shift of two homologous points of a stereoscopic pair's right and left views. The parallax effect consists in creating an illusion of depth and, thus, 3D to the user. The motion parallax refers to changes in the projective relationships between the images of objects resulting from the observer's movement in his environment.

While VR headsets may not be a standard household item yet, there is a lot of buzz around the immersive media, and the consumers are interested in experiencing it for themselves. However, the crowd is even more excited looking for both interactive and immersive contents together.

1.1.2 View synthesis

The View Synthesis (VS) is currently a study category of Computer Vision Research (CVR) that aims to create new views of a specific subject starting from several snapshots taken from given points of view. CVR and Artificial Intelligence (AI) fields are involved in the definition of suitable approaches to the problem. Fig. 1.2 illustrates the problem which is given several pictures I_1 and I_2 of a particular subject taken from specific points with specific camera settings and orientations, try to create a synthetic image I_v from a virtual camera

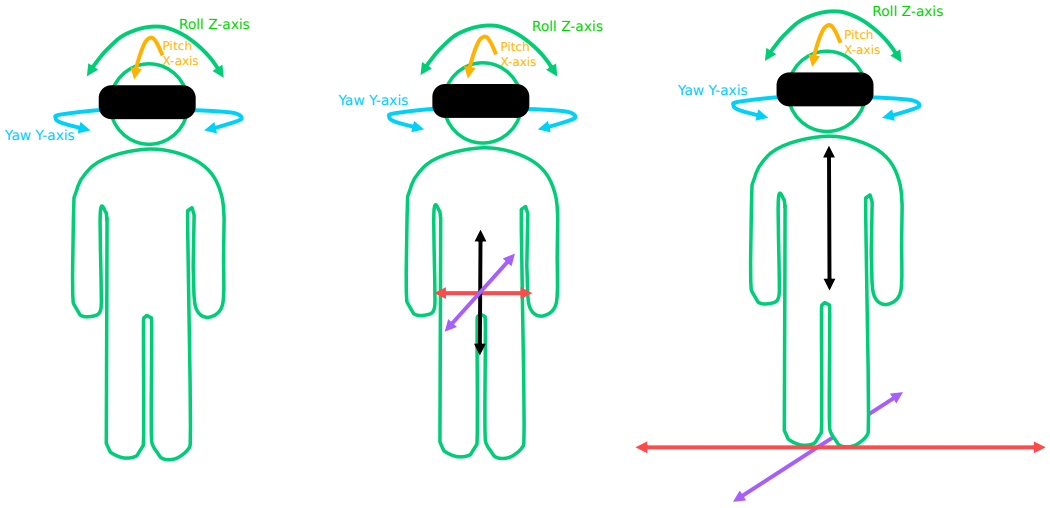


Fig. 1.1 Illustration of the different degrees of freedom (Degree Of Freedom) that a user has during his immersive experience.

placed at a different point and given settings.

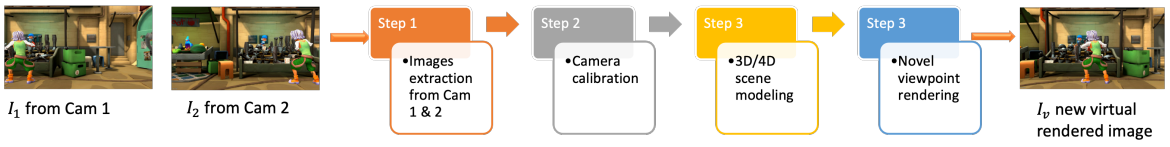


Fig. 1.2 Image rendering process

An example application of View Synthesis is Free Viewpoint Television (FTV). FTV is a system for viewing a natural video, allowing the user to interactively control the viewpoint and generate new views of a dynamic scene from any 3D position. In FTV, the center of attention can be controlled by the viewers rather than a director, implying that each viewer may be observing a unique viewpoint [14].

Several cameras are placed around the scene to obtain the views necessary to allow a high-quality rendering of the scene from any angle. Either in a studio environment [15] or an outdoor venue, such as a sporting stadium [5]. The output Multi-View Video (MVV) must then be packaged so that the data may be compressed. The users' viewing device can easily access the relevant views to interpolate new views [16] [17]. In addition to placing cameras around the scene to be captured, it is essential to estimate the geometry of the

camera set up by a process known in computer vision as camera calibration [18] [19]. The manual arrangement would be too cumbersome, so typically, an alignment is performed before capturing a test pattern used to generate calibration parameters.

MVV capture changes from partial (usually about 30 degrees) to complete (360 degrees) scene coverage. Therefore, it is possible to output stereoscopic views proper for viewing with a 3D display or other 3D methods. Systems with more physical cameras can capture images with more coverage of the viewable scene. However, certain regions would permanently be occluded from any viewpoint. A more significant number of cameras should make it possible to obtain high-quality output because less interpolation is needed.

In this thesis, we study the video transition effects in a MVV captured scenes to virtually move from one real viewpoint to another in the same scene. Therefore, while watching a video, the viewer can only stop on real views and make a natural transition between them using virtual synthesized views.

The targeted use-case provides that a user can move freely from one real view to another in a given scene. However, of course, the views we are discussing here correspond to where the cameras are placed. If we consider a scene that is shot with several cameras, as shown, for example, in Fig. 1.3 we capture a narrative scene of three actors using multi-view video setup, the views correspond to the cameras represented by the blue pictograms, and a transition happens between two adjacent cameras.

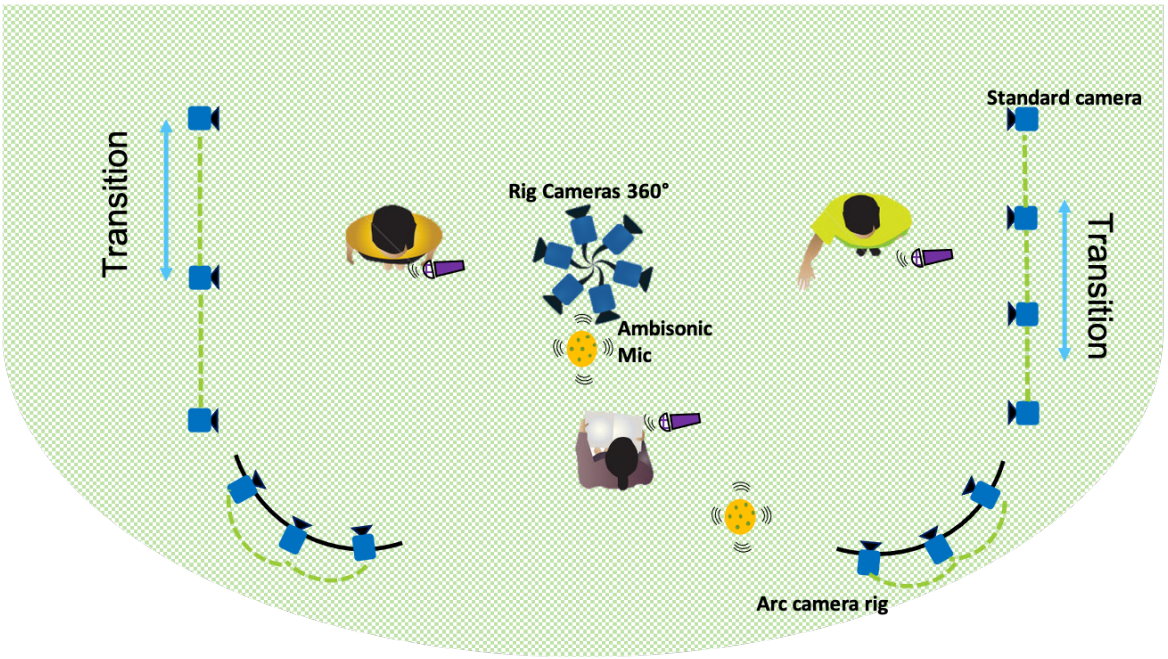


Fig. 1.3 Example of a scene map.

Generating virtually those kinds of transitions reduces a significant number of cameras in a scene replaced by virtual ones. It thus reduces the cost of the MVV setup in a single

scene. At the same time, it offers the viewer interactive and natural navigation from one viewpoint to another as if the camera was moving. The fewer real cameras we use, the fewer expenses required in the MVV capture; however, the larger the baseline is (i.e., inter-cameras distance).

Many transition types exist to join two viewpoints in a scene, as an approach to our problem, from the more classical one such as the *Cut* transition, where one shot transitions to the next shot without any effect, to the most recent one such as *View Synthesis-based* transition methods. View Synthesis methods such as 3D warping use the 3D information of the scene (e.g., depth map) to reconstruct a new viewpoint; Therefore, it seems that these methods could be the most promising to generate the transitions we are aiming at and retained our attention in our study.

In the literature, the view synthesis methods are divided into two categories: the classical algorithmic-based methods [20–23] and the learning-based methods [1, 4, 24–26]. Other work aims to improve the inpainting task of the view synthesis process to fill in the holes due to occlusions. These methods could be traditional inpainting methods [27–29] or learning-based methods [30–32]. However, these methods still suffer from visual artifacts in the final rendered image due to occlusions in the newviewpoint position. Indeed, a sparse number of cameras in a scene enlarge the baseline from the virtual viewpoint, which implies an increase of missing information in areas non-visible by the references cameras and visible by the newone to be reconstructed. Such constraints lead to a critical drawback in these proposed methods when dealing with large disocclusions.

1.1.3 Immersive and Interactive free viewpoint video

Broadcasters and content creators are so far controlling the viewing angles and positions showed to users in a video. The key benefit of free viewpoint video (FVV) is providing a new immersive user experience and interactivity that goes beyond higher image quality and higher realism. It is an advanced visual media type that allows the users to chose their desired point of view in a 3D/4D space from multi-view captured video, meaning an interactive free navigation.

Unlike computer graphics applications, FVV aims at capturing real world scenes using real cameras and creates virtually all the other possible point of views of the scene, which is very attractive to users in many applications. It allows the user, for example, to enjoy watching a DVD of a concert or an opera from different point of view freely chosen by him (his objective can focus on the singer, the band, his favorite musical instrument or maybe the audience...). As well as in post-production systems, such for a sport event like football or basketball, FVV offers the user the freedom to go anywhere in the field that he wants as a fan. It is an improved experience to watch the game whether the user is sitting in the stadium or at home, bringing him closer to the game to see different perspective even if it was not

shot by any camera. It also provides him replay angles of the game that he would never be able to see, putting him in the action with the players.

The director of Carnegie Mellon's Robotics institute Takeo Kanade, calls this technology "Virtualized reality", as opposite to virtual reality, that is based on events taking place in the real world, which are captured and processed by computer manipulation, and he said:

"*The output from these multiple cameras shooting a scene together from many angles actually can create totally new views that were not captured by any camera.*". He added, "*because our models are derived from real images, the models look much more real than typical virtual world.*"

Indeed, this technology offers a completely new way of watching an entertainment event where the user can customize the perspective from which they watch. For example Intel proposes the True View platform that delivers new views that traditional cameras can't. Fig. 1.4 illustrates dozens of small cameras installed in a ring around the venue, capturing the entire field of play. They generate massive amounts of volumetric data; the camera array is connected to Intel servers, where all data is stored and processed. Intel TrueView renders 360-degree replays, stunning freeze frames, and never-before looks at the most exciting plays.

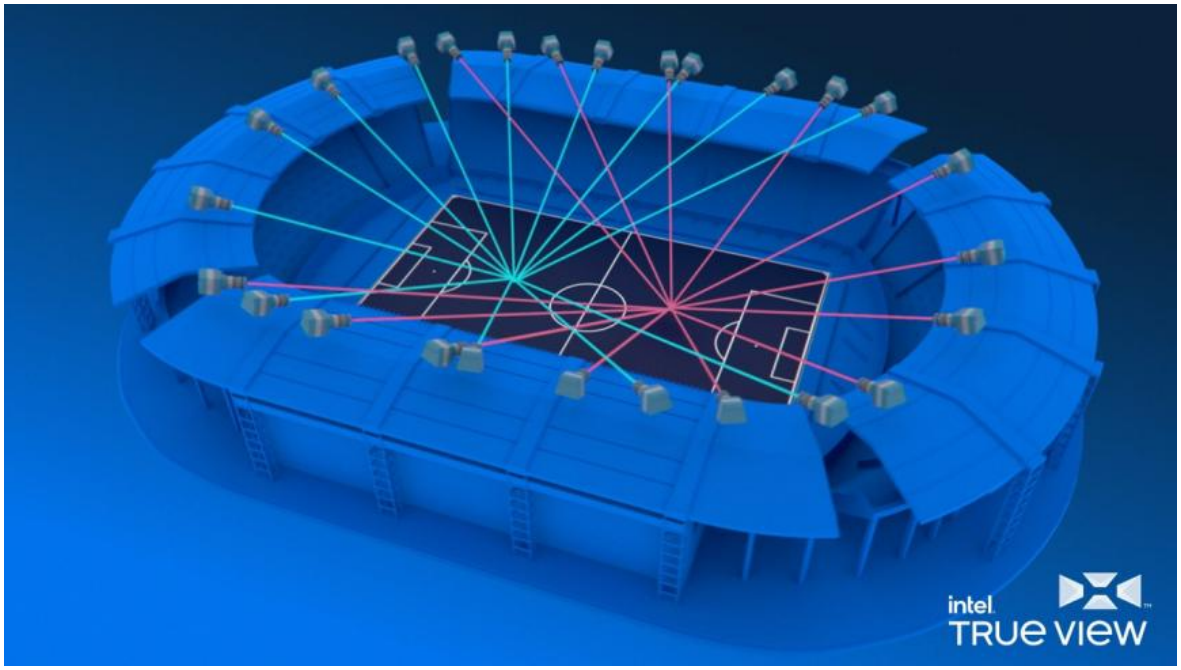


Fig. 1.4 Example of a customized entertainment event [5]

In addition, both immersive and interactive aspects are available in such applications. However, this technology is totally new and the user is not very familiar with it and thus, many questions arise. The first thing that intrigued the experts is about losing bearings in an immersive experience which can cause dizziness and motion sickness [33] for some users,

and a lot of work [34] has been already done on this topic and still going on. Content creators and image quality experts are the most involved in these studies to provide to the user an immersive experience as comfortable and pleasant as it is while watching on a flat screen. Not less than the immersion aspect, studies about interactivity experience have been taking an important place too in the community of the Quality Of Experience (QoE) studies [35]. The user has always been passive when it comes to watch any kind of entertainment show (concert, football, theater, cinema ...). The fact that the user has from now on, to control his viewing angle while watching a dynamic scene, would not necessary be unanimous [36]. Therefore, the experts work on it and their work is essentially based on subjective quality assessment tests. One interesting and very rare possible interactivity is the one applied on a narrative contents (movies, series, theater, ...) where the director has a very important role in it, as he is the one who defines what to show or not to the viewer. Offering the possibility to the viewer to change their point of view in the middle of a movie scene in a coherent and motivated way, is very challenging for a collective and coordinated work between the scenarist, the director and content creators.

1.1.4 Interactivity in Narrative content

There are two modes of interactivity in narrative content nowadays:

- choose-your-own-adventure mode.
- change-your-point-of-interest mode.

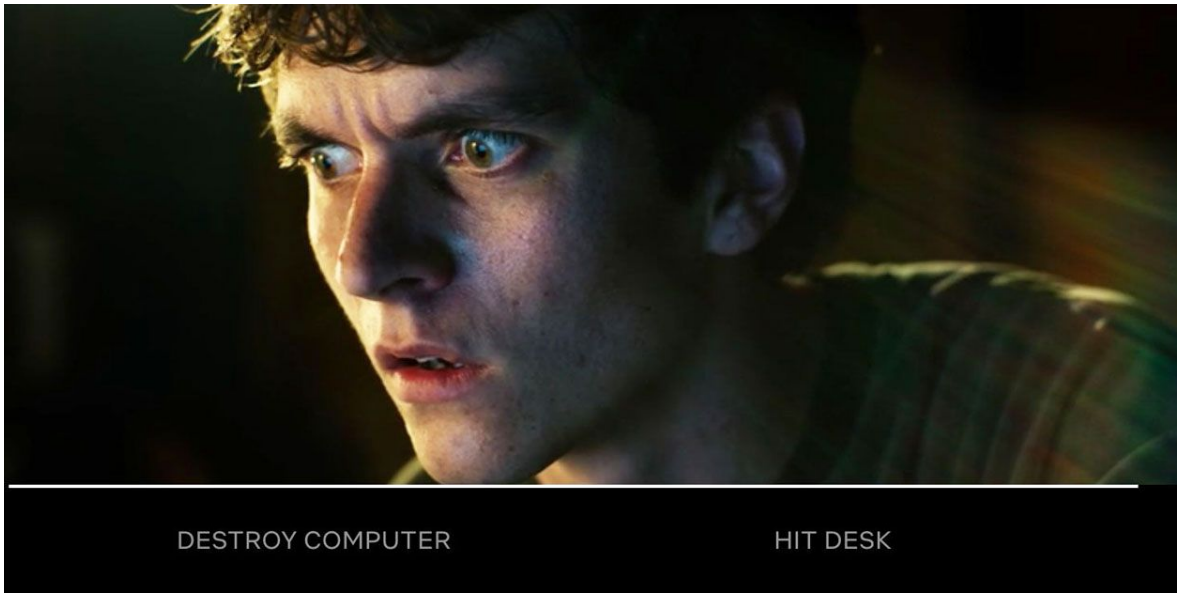


Fig. 1.5 Black mirror:Bandersnatch series [6]

Choose-your-own-adventure

Most interactive movies are a particular type of movie that blends the interactivity of the games with the narrative experience of films. This mode of interactivity does not let the viewer passive to the storytelling, but it gives them the power to make narrative decisions. However, this mode is not really new technology, since it has existed for decades as video games. Games are fully-interactive, with users making every possible decision within an experience. Some games have little to no "story", while others have narratives within them that may or may not be influenced by a player's actions. Some producers and content creators are excited about this "novelty", blurring lines between high-quality passive filmed content and consumer-decision content. For example in Fig. 1.5 Netflix wants to make more interactive TV shows. For example, in Black mirror:Bandersnatch series, the viewer in this scene can decide the character's reaction, by destroying the computer or hitting the desk. In this context, there is no "winning". Viewers have minimal control, and the quality of the outcome is entirely subjective. Furthermore, the viewer can make choices on behalf of the characters, as the story takes shape based on the viewer's choice. Each choice leads to a different adventure, so the story will be renewed each time the viewer sees the movie again.

Change-your-point-of-interest

This is the mode on which we are focusing in this manuscript, which is the rarest mode to be implemented in narrative content. In films, directors decide on the sequence of points of view in a filmed scene; depending on the scenario, he switches from one point of view to another to produce a scene that pleases the viewers. There are three different types of point of view. First, the "subjective" point of view, where one identifies with the character. In this view, the spectator does not see the actor; instead, it is his eyes.

Second, through "witness" point of view, we are spectators of the scene, either we watch what happens, fixed shots, or we follow the action, moving shots; the choice is that of the desired effect on the spectator: passive, he observes, explores the interior of the image, anticipates, or active: he participates in the action and is surprised.

Finally, the "narrator" point of view or simply an exterior point of view. However, this view allows the user to realize the situation by giving an overall view. The camera is led to take unreasonable positions such as an exterior view of an apartment at the top of a building, a point of view that no one ever has, a "god's eye" view. This unique status would place oneself too much on the outside, retreat from the story, and rarely being unique. However, here the viewer is just a passive spectator.

It could sound ominous, but the idea is to let the viewer participate in the "directing" of the movie by letting him change his point of view based on his interest. The main reason to change a point of interest in the scene is either the actual point of interest is not interesting anymore to the viewer, or the other point of interest in the same scene seems more attractive

to this viewer. It also could be changing a point of view of the same point of interest.

In cinemas, another point of view could be more plausible, more enjoyable, less harmful, less aggressive to the viewer depending on the scene, without losing the story events.

In theater, this option gives the viewer the power to move around in the play as he wishes. It is up to each viewer to decide how they want to see their movie or play, giving them some power. This mode will expand the number of viewers. For example, in a given brutal scene of two actors, we can keep watching the torture or change the point of interest to watch other actors' reactions to avoid such scenes to sensitive people.

To make this change of point of interest happen, we will need to generate a visual effect to link both real points of view together; we call it a video transition. In our case, this video transition is a technique used in the post-production and editing of a movie to combine two different points of view in the same scene.

Video transition: a middle run towards the 6DoF

In an immersive interactive experience, the "6 degrees of freedom (6DoF)" offers the user the possibility to be tracked not only by his head movement but also his body location as he physically moves left, right, forward, backward up, and down. The user can thus move around to freely change his location and point of view while watching a video.

The video transition is an intermediate step towards the 6DoF, where the user is not entirely free but has the choice to change his point of view using a transition that can seamlessly join two adjacent points of view of real cameras in the scene. Fig. 1.6 illustrates an example of a video transition in the same scene where a viewer can move from Camera A to Camera B to change his point of view in the scene while watching a video. A typical format for these applications is the MultiView Video composed of a set of N video sequences representing the same scene, referred to as real views, acquired simultaneously by a system of N cameras positioned under different spatial configurations. An alternative representation is the Multi-View-Plus-Depth (MVD) format [17], where the depth and texture information are used for each viewpoint.

We call the seamless video transition "a middle run towards the 6DoF", and it will be explored in this manuscript, from its creation, conception to its purpose. However, we assume that it is essential to seamlessly maintain the sense of orientation in the viewer when transitioning between video shots of a scene by relating the two- and three-dimensional space of action, else the viewer will become lost in the environment.

The *Cut* transition is when another image instantly replaces the first image. The *blend* is also overlapping, which is when the new image gradually replaces the first one. The *blur* is when a progressive and then digressive blur achieves the switch from the first point of view to the other. Furthermore, state-of-the-art computer vision allowed more advanced techniques

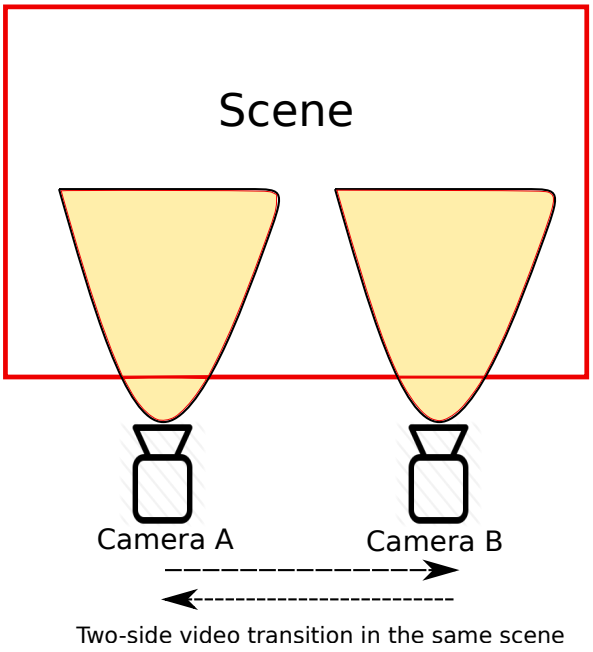


Fig. 1.6 Example of video transition in a multi-view video sequence representing the same scene.

to reconstruct the static geometry of a scene in a video, which encouraged the video-based rendering transitions. These transitions types, particularly the video-based rendering transitions, retained our attention in the rest of the manuscript.

1.2 Goals and Motivations

The work of this thesis is part of the two projects named 'INVATE' and 'VANTAGE' of the research institute b<>com, whose objective is to develop a new audio-visual technology that offers the user an immersive interactive experience putting him in the center of the action.

The main objective of the thesis is to offer the viewer an interactive experience, where he can freely change his point of view while watching a video. The main challenge is to decrease the number of cameras in the scene by retaining the essential points of view and reconstruct a seamless transition between the real cameras. View synthesis-based transition methods seem to be the most natural and realistic transitions compared to more classical methods that do not use a 3D reconstruction of the scene. While the traditional view synthesis methods lasted more than two decades, learning-based view synthesis methods acknowledged a significant rise in the past two years because of their outstanding achievements we are witnessing nowadays, in this domain.

The existing drawbacks in the literature encouraged exploring the existing view synthesis methods to build a suitable natural and seamless transition between two viewpoints in a

scene. We studied both the algorithmic and learning-based methods for view synthesis, trying to get the best out of both approaches and develop an optimal view synthesis method for wide baseline view synthesis. Moreover, we keep the user experience at the center of our study since the viewer is the end user of our application. Therefore, we conduct a series of subjective tests to evaluate different transition methods based on human judgements.

1.3 Contributions

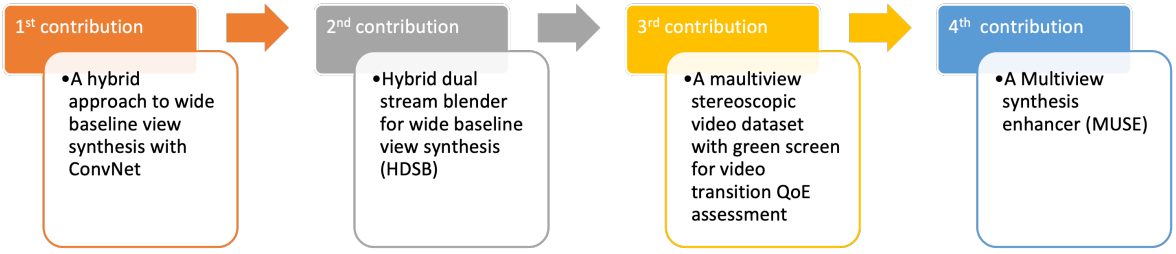


Fig. 1.7 The list of the thesis contributions

Fig. 1.7 illustrates the list of the thesis contributions.The first part of this work aims at achieving a novel approach for wide baseline view synthesis. To this end, we introduced novel neural network architectures for the view blending step while using a state-of-the-art algorithmic method for the view warping step of the view synthesis process. It aims to reduce the significant size of model filters to warp such reference images, thus reducing the complexity. Two different approaches are investigated in this part of the thesis:

- **A hybrid approach to wide baseline view synthesis with ConvNet (CNN-VB):**

  ConvNets have been recently employed [1–3] for implementing complete end-to-end view synthesis architectures, from reference view warping to target view blending while dealing with occlusions as well. However, the convolutional size filters must increase with the distance between reference views; which makes all-convolutional approaches prohibitively complex for wide baseline setups. In this work, we propose a hybrid approach to view synthesis where we first warp the reference views resolving the occlusions. Then we train a simpler convolutional architecture for blending the preprocessed views. By warping the reference views, we reduce the equivalent distance between reference views, allowing smaller convolutional filters and thus lower network complexity. We also contribute a hole inpainting algorithm to fill the dis-occlusions in the warped views. We experimentally show that our method performs favorably against both traditional and convolutional synthesis methods while retaining lower complexity concerning the latter.

- **Hybrid dual stream blender for wide baseline view synthesis (HDSB):** Free navigation of a scene requires warping some reference views to some desired target viewpoint and blending them to synthesize a virtual view. ConvNets based methods can learn both the warping and blending tasks jointly. Such methods such as [4] are often designed for moderate inter-camera baseline distance, and larger kernels are required for warping if the baseline distance increases. Algorithmic methods can deal with large baselines; however, the synthesized view suffers from artifacts near disoccluded pixels. We present a hybrid approach where reference views are algorithmic-ally warped to the target position and then blended via a ConvNet. Preliminary view warping allows reducing the size of the convolutional kernels, and thus the learnable parameters count. We propose a residual encoder-decoder for image blending with a Siamese encoder to keep the parameters low. Our view synthesis experiments on real multiview sequences show better objective image quality than state-of-the-art methods due to fewer artifacts in the synthesized images.

The contributions of this thesis focus then, on the user quality of experience for the video transition and dataset. First, we accomplish a creative dataset for video transition quality of experience. Second, our work aims to subjectively evaluate the proposed view synthesis approaches on 8 different video sequences by achieving a series of subjective tests.

Thus the contributions in this part are divided into two segments:

- **A Multi-View Stereoscopic Video Database With Green Screen (MTF & Pandemonium) For Video Transition Quality-of-Experience Assessment:**

  First, we introduce a multi-view stereoscopic video database with a green screen, called MTF, for the usages in computer vision applications, particularly for free navigation, free-viewpoint television, and video transition QoE assessment. The MTF contains full-HD videos of actual storytelling made up of 3 scenes. One particularity of this dataset is that to understand its storytelling; users must change their point of view in the scene at a given time. To this end, we usually need to generate a transition to link two points of view in the same scene. Computer vision techniques that enable such transitions, like view synthesis methods, rely on a set of images of the scene to render some new views from different viewpoints. However, these methods may have many failure cases that lead to artifacts in the final rendered video transition. In most view synthesis QoE tests, the contents are not designed to make the transition between two points of view valid or exciting for the viewers, e.g., they do not need to transition to capture more information to understand the content better. We thus assume that participants will harshly judge artifacts and imperfections in the rendered transition. Thus, the MTF is expected to analyze better the visual impact of persistent artifacts in the final rendered transition. In our dataset, all the scenes are recorded in a green

screen studio, which is often used to superimpose special effects and scenery during editing according to specific needs. Our dataset also presents a wide baseline camera setup, a challenging constraint for view synthesis techniques. Finally, The MTF & Pandemonium can also be used as a complementary dataset with others in literature in various computer vision applications, such as video compression, 3D video content, immersive virtual reality environment, optical flow estimation...

- **MUlti-view Synthesis Enhancer (MUSE)**:

  Finally, we propose an extension of the previously proposed solutions for the view synthesis. This approach proposes to enhance the pre-merged warped views using the one-encoder Hybrid Dual Stream Blender (HDSB) method since now we only have one input image instead of two. Therefore, the network does not blend the views but corrects the image artifacts emerging from the pre-merged image and improves the quality of the final image. We also propose an intra-content learning process, where we achieve one training for every content. The pre-merging of the warped and inpainted textures coming from the lateral reference views guarantees, in particular, a uniform blending of the non-Lambertian areas of the image.

1.4 Organization of this thesis

Chapter 2 includes background information essential to the development of this thesis. It is first introducing the 6DoF navigation and its applications in immersive and interactive video content. We describe the free navigation application in narrative content, discussing about its purposes. Then, we present the video transition in narrative content like a middle run towards the 6DoF and the difference with the 6DoF. Second, we introduce the view synthesis methods that allow such video transitions in narrative content, and we explain its phenomena from the capturing camera model to the depth estimation.

Chapter 3 describes the state of the art of view synthesis. This part presents an overview of the related work based on traditional techniques based on classic algorithms; moreover, we review the work based on the more recent techniques for the view synthesis that mixed the deep learning methods.

Chapter 4 describes the first contribution of this thesis that proposes a novel approach of view synthesis based on a new architecture for the view blending to generate novel view synthesis. We show preliminary results and the improvements over the state-of-the-art traditional methods for the view synthesis, and finally, we show the drawbacks and limitations of the method.

Chapter 5 describes the second contribution which is an extension of the previously mentioned view synthesis approach. This one tackles the remaining problems of the first contribution and overcomes purely algorithmic and purely learnable state-of-the-art approaches.

The proposed view synthesis method is detailed in this, where we experimentally evaluate the performance of our method, including an ablation study.

Chapter 6 describes the proposed multi-view stereoscopic video database with a green screen called MTF. An overview of the related work is first given. Then, the pipeline of creating the dataset from the storytelling to the film shooting is presented in detail.

Chapter 7 introduces the last extension of our view synthesis approach that we evaluate and compare with previous work and traditional methods using subjective tests. Thus, in this chapter, we describe the three subjective tests. The first two aim to evaluate our novel approach compared to the state-of-the-art; however, the last test describes the user preference for the view synthesis transition types.

Chapter 8 concludes this dissertation. The initial goals are first reminded, and the achieved work is then summarized. Finally, future work prospects and potential improvements are discussed.

Finally, Appendix A lists the different scenes of the dataset; in Appendix B, we list the three test instructions given to the users during the tests. Finally, Appendix C lists the publications produced during this thesis.

Chapter 2

General background on Video Navigation

The central problem addressed through the work described in this thesis is the following: "what should be the effect proposed to a user when he decides to move from one point of view to another?" This chapter gives some insights to this question regarding technical solutions of video transition.

Since most of the transition methods discussed require the definition of the camera parameters and projections functions, the camera model used and the associated projections functions and their parameters are described in Sec. 2.1. Sec. 2.2 enumerates the different video transitions techniques and details their methods of creation. Some video transition techniques require the 3D information of the scene, such as the depth map. Sec. 2.3 is dedicated to the description of the depth estimation method using a stereoscopic camera setup. Finally, we conclude the chapter in Sec. 2.4, introducing the next chapter.

2.1 Camera model

Throughout this chapter, the modeling of the image creation process within a camera is the pinhole model.

Fig. 2.1 illustrates the set of the coordinate systems (world, camera and image plane), where the camera is associated with its optical center and is marked by point C. The point M is a point in the real world filmed. The point M is projected on the image plane in Q, the intersection of the line joining the points M and C with the image plane. The point P is the principle point: it is the orthogonal projection of the optical center in the image plane. The line perpendicular to the image plane and passing through the point P is called the optical axis. The focal length f is the distance between the Camera Center and the image plane, which corresponds to the length of the segment $[CP]$. With the help of projections and changes of coordinate system, it is then possible to describe the situation in terms of linear algebra.

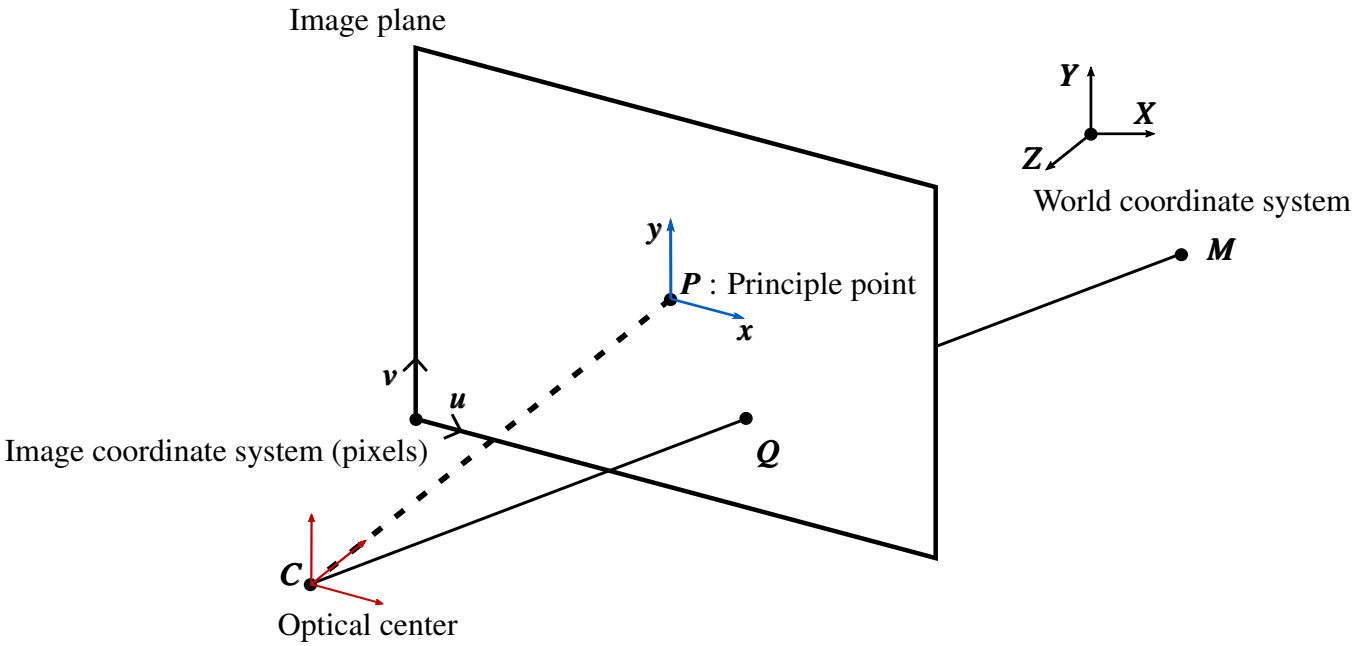


Fig. 2.1 Set of coordinate systems

Camera projections and parameters

This paragraph describes the functions used to project a real world point M position to the Projected image point Q position and obtain the parameters from one frame of reference to another. All the notations used in what follows correspond to those in Fig. 2.1 which illustrates the set of the coordinate systems (the camera coordinate system, the image coordinate system and the world coordinate system).

Fig. 2.2 shows a lateral view of the projection of a point $3D$ point M in the image plane, where it is possible to obtain the projection coordinates of the point M expressed in the camera coordinate system in the image coordinate system. Fig.2.3 shows a view from above of the projection between two coordinate systems process. We add an exponent to the coordinates of the points to specify in which coordinate system they are expressed: $.^w$ for the world coordinate system and $.^c$ the camera coordinate system.

The coordinates of the point $Q^c = (x,y)$ (in the camera coordinate system), projected from $M^w = (X,Y,Z)$ (in the world coordinate system) verify :

$$x = f\frac{X^c}{Z^c} \tag{2.1}$$

$$y = f\frac{Y^c}{Z^c} \tag{2.2}$$

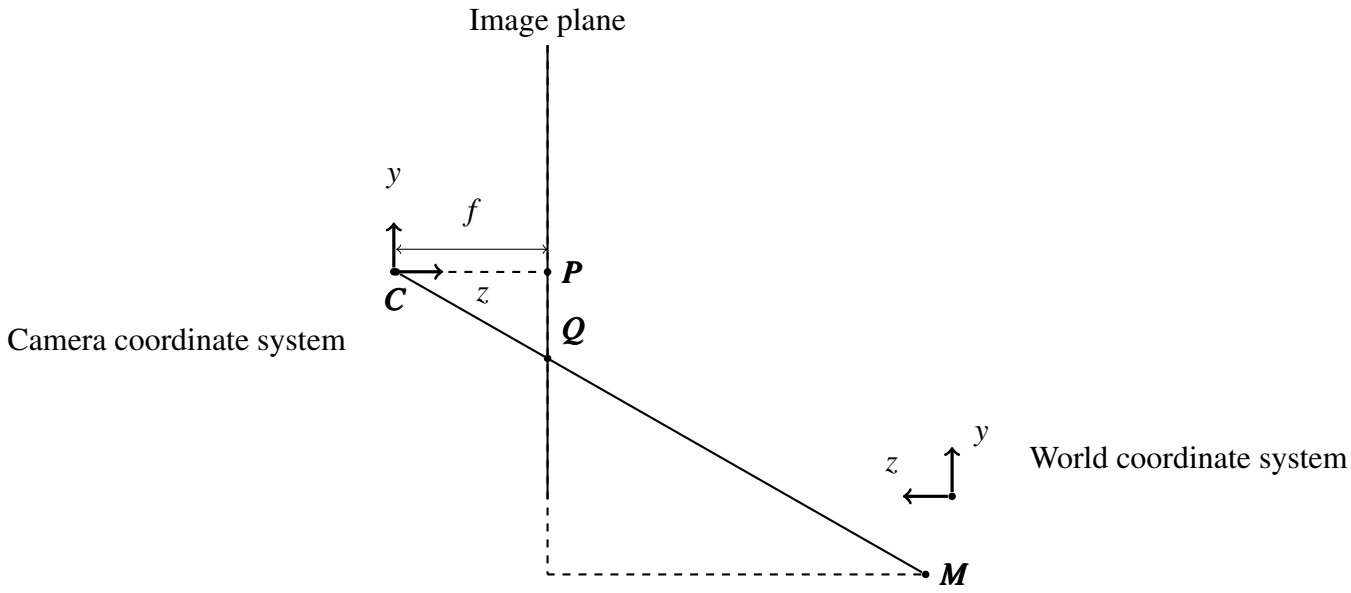


Fig. 2.2 Projection : lateral view

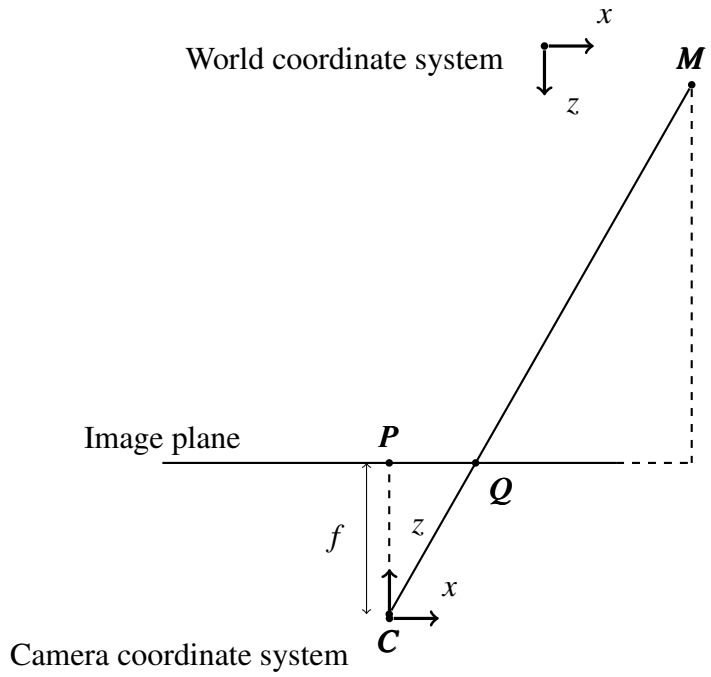


Fig. 2.3 Projection : view from above

Using the homogeneous coordinates, it is possible to put this relation in matrix form:

$$Q^c = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{pmatrix} \tag{2.3}$$

The coordinates of the point$Q^c = (x, y)$ correspond to the coordinates of the point M projected in the image plane, in the camera coordinate system. To obtain its coordinates in the image in pixels, we must now proceed to a change of coordinate system that will take into account the origin of the image plane and the pixel density of the image. We note (x_0, y_0) the coordinates of the bottom left corner of the image (usually chosen as the origin of the image coordinate) in the camera coordinate. In addition, we note k_u and k_v densities of pixels in the direction of axes u and v of the image coordinate. This data corresponds to the number of pixels contained in a unit that used for the coordinate camera.

To obtain the coordinates of $Q^c = (x, y)$ in the image coordinate system (u, v), it is necessary to perform a translation for the change of origin then a change of unit to pass into pixels. Using the homogeneous coordinates, we obtain:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} k_u & 0 & 0 \\ 0 & k_v & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2.4}$$

Thus, by combining the equations 2.3 and 2.4, we obtain the projection of a point of the camera coordinate in the pixel coordinate:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} k_u f & 0 & k_u x_0 & 0 \\ 0 & k_v f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{pmatrix} \tag{2.5}$$

The 3×3 matrix resulting from Eq. 2.5 by removing the last column is the matrix of the camera's intrinsic parameters.

The starting point for the calculations so far is the camera coordinate. There is no additional step to perform when the world coordinate system is the same as the camera. However, this is rarely the case, and the objective is to have multiple cameras. Therefore, it is essential to consider a world coordinate that is different from that of the camera.

We must then model the position of the camera in this coordinate system and its orientation concerning this coordinate system. We note thus :

$$\begin{pmatrix} t \\ 1 \end{pmatrix} = \begin{pmatrix} C_x \\ C_y \\ C_z \\ 1 \end{pmatrix} \tag{2.6}$$

the position vector of the camera in the world coordinate, in homogeneous coordinates.

R the rotation matrix of the camera in the same coordinate system.

For $M = (X_w, Y_w, Z_w)$ a point expressed in the world coordinate system. The transition from the world coordinate to the camera coordinate is done by first changing the origin and then applying the rotation. This is then expressed in matrix form as follows:

$$\begin{pmatrix} X^c \\ Y^c \\ Z^c \end{pmatrix} = R \left(\begin{pmatrix} X^w \\ Y^w \\ Z^w \end{pmatrix} - \begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix} \right) \tag{2.7}$$

$$= R \begin{pmatrix} X^w \\ Y^w \\ Z^w \end{pmatrix} - R \begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix} \tag{2.8}$$

$$= R \begin{pmatrix} X^w \\ Y^w \\ Z^w \end{pmatrix} - Rt \tag{2.9}$$

This can be expressed in homogeneous coordinates as :

$$\begin{pmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{pmatrix} = \begin{pmatrix} R & -Rt \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{pmatrix} \tag{2.10}$$

The matrix 4×4 of the equation 2.10 is the matrix of *camera extrinsic parameters*.

With the two matrices defined above, it is then possible to give the complete equation of the projection model:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} k_u f & 0 & k_u x_0 & 0 \\ 0 & k_v f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R & -Rt \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{pmatrix} \tag{2.11}$$

Let us recall the important elements defined here. We have thus :

- the matrix of intrinsic parameters which defines the parameters related to the camera sensor:

$$K = \begin{pmatrix} k_u f & 0 & k_u x_0 \\ 0 & k_v f & k_v y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.12}$$

- the matrix of extrinsic parameters which gives information about the position and orientation of the camera:

$$\begin{pmatrix} R & -Rt \\ 0^T & 1 \end{pmatrix} \tag{2.13}$$

- the projection matrix, product of the two previous ones completed with zero vectors :

$$P \sim \begin{pmatrix} k_u f & 0 & k_u x_0 & 0 \\ 0 & k_v f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R & -Rt \\ 0^T & 1 \end{pmatrix} \tag{2.14}$$

2.2 Video transition techniques

The most commonly used transition in movies is the instant transitions or *cut*, it represents a change of context. It is effective when visual displacement is significant, yet its use does not preserve the space of action. However, it is the simplest way to join two points of view. Besides the *cut*, multiple transitions exist to communicate a message and conceive impressions in the viewer's mind.

The *blend* or *blur* are commonly used in movies to represent a passage of time. They could be used between two shots in the same action scene since we have time differences between the two points of view. Nevertheless, they add nothing over a *cut* to help maintain the space of action.

Video morph or *warp* transitions are often used as a special effect in movies to transform one object into another. However, recent advances in robust feature point correspondence have allowed view change transitions as well. Warp transitions provide an alternative both to transitions that require geometry and to plane transitions. While we classify plane transitions as a subset of warps with global 2D transformations (4-point correspondence), warps can also be computed from many hundreds of points to exploit more accurate correspondence [16].

With machine graphics, we can also generate natural transitions that rely on scene geometry and virtual cameras. They sustain the space of action and sense of orientation in the viewer by rendering a perspective-correct view from virtual cameras that join both points of view. This transition also preserves as much as possible the motion of dynamic objects by projecting performing video shots onto the scene geometry.

View synthesis is the process of extrapolating or interpolating a view from other available views. It is a popular research topic in computer vision, and numerous methods have been developed in this field over the past four decades. Image-based rendering (IBR) is a virtual view synthesis technique based only on one or more images, without a prior knowledge of the 3D geometry of the scene, contrary to the technique of view synthesis by 3D

computer graphics where the 3D geometry of the scene is wholly acquired. View synthesis can be divided into three categories [37]. (Fig. 2.4) according to the major or minor use of the scene geometry: the techniques using no prior information of the 3D geometry of the scene; the techniques that require implicit information of the 3D geometry, like some pixel correspondences in the available and synthesized view, that can be computed using optical flow [16] [38]; and the techniques that require explicit knowledge of the 3D geometry such as the depth map, also called Depth-Image-Based-Rendering (DIBR) [39] [40].

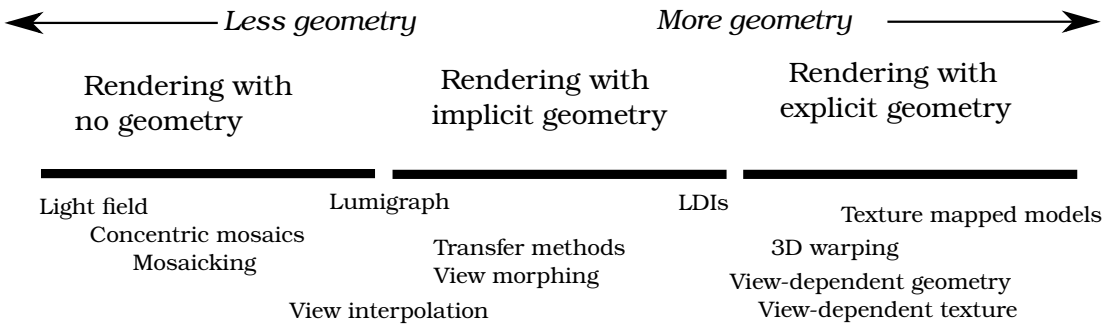


Fig. 2.4 IBR techniques based on their need for scene geometry [7]

The *light field* [41] method generates new views from arbitrarily positioned cameras in a scene, without needing any depth information or correspondence between the real images but a simple combination and sampling between them. Instead, it is a matter of interpreting the input images as 3D pieces of a 4D function. This function characterizes the flow of light diffused in a static scene with static illumination.

The *lumigraph* [42] is similar to the light field, with the same 4D parameterization and the same display principle but with an irregular sampling:

The *Layered Depth image (LDI)* [43] technique is a new image-based rendering method for objects with complex geometries. Unlike methods that represent objects by triangular meshes like most computer graphics models, LDI represents objects as an array of pixels viewed from a single camera position. Each LDI pixel is represented by its color, depth, and other properties specific to the LDI method.

We exclude these methods from our transition types to focus instead on methods that allow view change transitions, particularly for a wide baseline transition, where the displacement between the two points of view is significant.

2.2.1 View morph or warp

The concept of *view morphing* is to preserve 3D shapes [8]. The "morphs" generated to create the transition give the impression that the objects move rigidly by rotation and 3D translation between their positions in the two images. In other words, the *view morphing* effect will give the user the impression that the camera is being moved from one place to

another in the same scene. View morphing creates, by interpolation, virtual cameras at any point between the real positions of the two cameras.

This method requires geometric information about the scene and is classified as an IBR technique with implicit geometry information cf. Fig. 2.5.

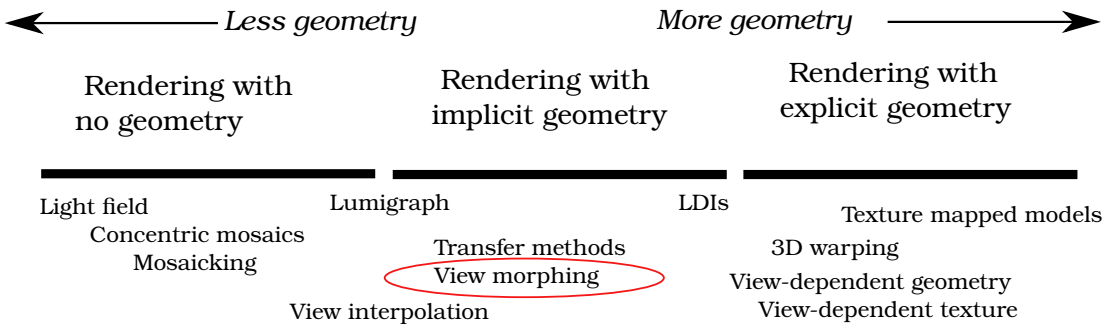


Fig. 2.5 View morph technique [7]

View morphing requires, in addition to the two original input images (I_0, I_1) and the correspondence points between them, the two projection matrices (Π_0, Π_1) of each of the cameras that captured the I_0 and I_1 images. Thus, the method uses two 2D images and attempts to perform interpolations between 3D transformations.

In the case of view morph between parallel views (I_0, I_1), for example, the intermediate generated views by view interpolation are parallel to I_0 and I_1, and consequently, the projection matrices of I_0 and I_1 are as follows:

$$\Pi_0 = \begin{bmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\Pi_1 = \begin{bmatrix} f_1 & 0 & 0 & -f_1 C_x \\ 0 & f_1 & 0 & -f_1 C_y \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

We consider that the optical center of I_0 is the origin, with a focal length f_0, and that the optical center of I_1 is translated with respect to the origin as a function of x and y (C_x and C_y), and a focal length f_1.

The linear interpolation of these two projection matrices gives :

$$\Pi_s = (1-s)\Pi_0 + s\Pi_1 \tag{2.15}$$

$$\Pi_s = \begin{bmatrix} (1-s)f_0 + sf_1 & 0 & 0 & -sf_1 C_x \\ 0 & (1-s)f_0 + sf_1 & 0 & -sf_1 C_y \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

With $s \in [0,1]$ which weights the distance between the new and the real views. The optical center of this camera position and its focal length are :

$$C_s = (sC_x, sC_y, 0) \tag{2.16}$$

$$f_s = (1-s)f_0 + sf_1 \tag{2.17}$$

From the projection matrix of the new position, we can generate a view parallel to the original views at the same position by linear interpolation of the matching pixels (cf figure 2.6)

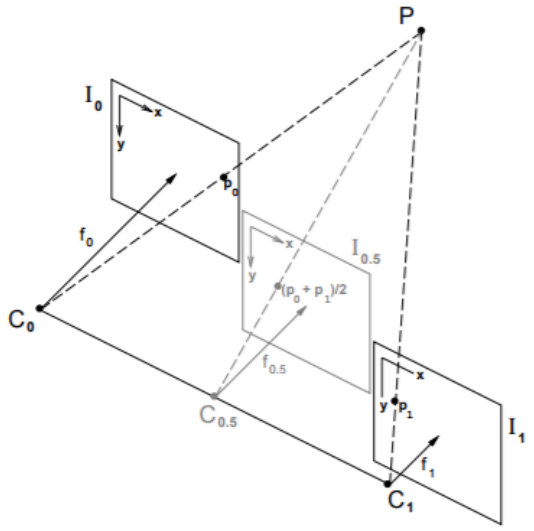


Fig. 2.6 Morphing from two parallel views I_0 and I_1. $I_{0.5}$ represents a new parallel view of the same scene [8].

2.2.2 3D warping

This method falls into the category of DIBR techniques [44]. It differs from the previous ones because it requires additional information about the scene's geometry, in this case, the depth. The method, therefore, belongs to the category of techniques with explicit knowledge of the scene (cf. Fig 2.7).

3D warping, also called *view synthesis*, is the process of interpolating or extrapolating a view from other real views. A depth or disparity map is beneficial for warping the pixel, available in the real views, to the correct position in the synthesized view.

DIBR methods rely on the geometric information of the camera to re-project a point in the reference image in the real world and then in a new image plane [45]. We consider two cameras with C_1 and C_2 optical centers, respectively, and their image planes (cf. Fig.3.19).

The projections of a real point P in each of the planes are noted by p_1 and p_2. The goal is to express p_2 as a function of p_1 and its depth. The homogeneous coordinates $(u_1, v_1, 1)$,

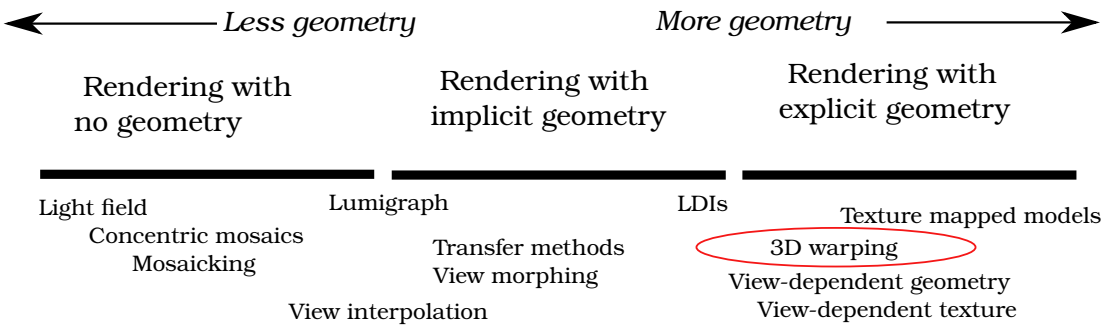


Fig. 2.7 3D warping category [7]

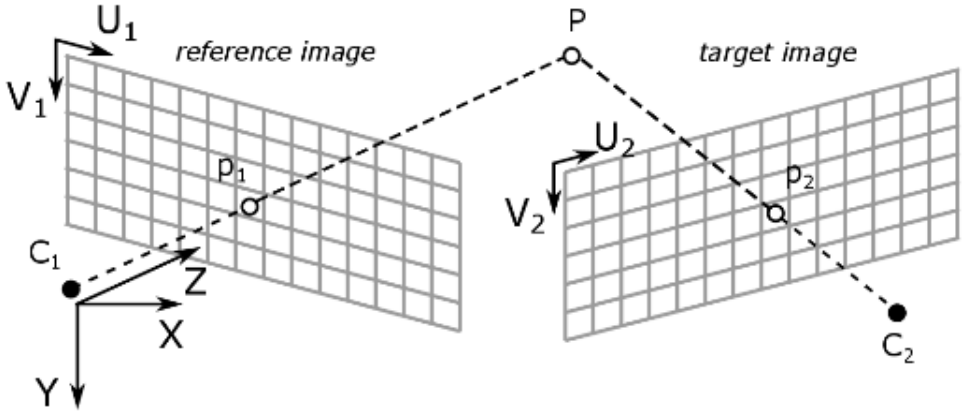


Fig. 2.8 Warping a pixel from a reference image to another target image

$(u_2, v_2, 1)$ of p_1 and p_2 can be expressed as a projection $P(x,y,z)$:

$$z_{c_1} p_1 = K_1 R_1 \begin{bmatrix} x \\ y \\ z \end{bmatrix} - K_1 R_1 C_1 \tag{2.18}$$

$$z_{c_2} p_2 = K_2 R_2 \begin{bmatrix} x \\ y \\ z \end{bmatrix} - K_2 R_2 C_2 \tag{2.19}$$

where K_1, K_2 and R_1, R_2 are 3x3 matrices of intrinsic parameters and 3x3 matrices of rotation for each camera, respectively. z_{c_1}, z_{c_2} are the z coordinates of the 3D point P in the camera coordinate, provided by the depth map. From the equation 2.18, we express the point P by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = (K_1 R_1)^{-1} (z_{c_1} p_1 + K_1 R_1 C_1) \tag{2.20}$$

Replacing the equation 2.20 in the equation 2.19, we obtain:

$$z_{c_2} p_2 = K_2 R_2 (k_1 R_1)^{-1} (p_1 + K_1 R_1 C_1) - K_2 R_2 C_2 \tag{2.21}$$

If we assume that the two cameras are identically rectified, then $K_1 = K_2, R_2 = I_3$ (no rotation between the angles of the cameras, I_3 being the identity matrix), $z_{c_2} = z_{c_1} = z, C_2 = (c_x, 0, 0)^T$ (camera 2 is positioned in the X axis in the world coordinate). In this case, the equation 2.21 becomes:

$$p_2 = p1 + K \begin{bmatrix} \frac{c_x}{z} \\ 0 \\ 0 \end{bmatrix} \tag{2.22}$$

and u_2, v_2 can be expressed such that

$$u_2 = u_1 + \frac{f.c_x}{z}, \text{ et } v_2 = v_1 \tag{2.23}$$

With f the focal length of the cameras, $u_2 - u_1 = \frac{f.c_x}{z}$ is also called the disparity and c_x is the distance between the two optical centers of the two cameras.

A common problem in view synthesis is areas occluded in the available views but should be visible in the virtual ones. These areas appear as holes in virtual views, also referred to as disocclusions. This problem is currently resolved by using inpainting algorithms such as [29] and [28]. Two of the most popular inpainting algorithms were developed by Bertalmio and Sapiro [27], and Criminisi et al. [46]. More recently, Pathak *et al* [47] developed the first Generative Adversarial Network (GAN) based inpainting algorithm that aims to fill missing parts in the image. This architecture is then enhanced by Yang *et al* [48].

Concerning the calculation of the depth map for such a DIBR method, we studied the method that consists of estimating each viewpoint's depth from a pair of cameras placed side by side I_1 and I_2.

2.3 Depth estimation

The information of the depth of each reference view is essential for the view synthesis. Indeed, the more precise the depth value is, the more accurate the pixel's position in the 3D world is. A classical way to find an approximation of the depth for a reference view is to use another camera placed at a small distance and estimate the offset of the pixels between them. This shift, also called disparity, is inversely proportional to the depth.

The disparity d (in pixels) is a function of the focal length of the camera f (in pixels),

the camera-object distance z (in meter) and the distance between cameras b (in metre) :

$$d = \frac{b.f}{z}. \tag{2.24}$$

The maximum disparity d_{max} (resp. minimal d_{min}) is calculated for the nearest (resp. farthest) object in the scene :

$$d_{max} = \frac{b.f}{z_{min}} \tag{2.25}$$

$$d_{min} = \frac{b.f}{z_{max}} \tag{2.26}$$

The amplitude of disparity a is given by

$$a = d_{max} - d_{min} \tag{2.27}$$

Therefore, we can find the depth map of the reference view in question. Thus we compute the optical flow of each reference view.

In the case of parallel I_1 and I_2 input images, we can compute the optical flow or the disparity, which is only different from zero in the x-axis. This allows us to use the simplified equation to compute the depth map:

$$D = f \cdot B/d \tag{2.28}$$

where D is the depth of the actual view, f is the focal length obtained in the camera parameters, B is the distance between the two stereo cameras, and d is the horizontal disparity between the two cameras.

Fig. 2.9 illustrates an example of the result of depth map estimation on real stereo images.

2.4 Conclusion

In this chapter, we classified all the video transition techniques from the basic ones to the recent ones.

Despite that, the view synthesis with DIBR methods still suffers from artifacts and imperfections in the final rendered image due to the disocclusion problem; the work for improving view synthesis methods is still relevant now more than ever. Notably, the improved DIBR solution seems to bring the most promising and natural video transition to our problem with improved depth maps obtained with ConvNets.

Fig. 2.9 Example of depth map obtained after computing the optical flow computed on real stereo captured images I_1 and I_2.

In the following, we enumerate all the state-of-the-art methods for view synthesis and their drawbacks. Furthermore, we aim to improve the existing view synthesis methods to obtain the most plausible rendered image quality to attain the required transition quality.

Chapter 3

State of the art of Depth-based View Synthesis techniques

In this chapter we describe the state of the art of the view synthesis techniques and we highlight the relative limitations. View synthesis methods can be roughly divided into algorithmic-based methods and learning-based methods. Sec.3.1 is dedicated to the algorithmic-based view synthesis methods. In Sec.3.2 we discuss the traditional and the learning-based image inpainting methods. In Sec.3.3 we describe in detail the learning-based view synthesis methods. Finally, in Sec.3.4 we recapitulate on the limitations of existing view synthesis methods that motivate our work.

3.1 Algorithmic-based View Synthesis methods

3.1.1 View Synthesis Reference Software (VSRS)

The MPEG proved a considerable interest in MVD formats for their capacity to support 3D video applications. Therefore, this new activity focuses on developing a 3D extension of the High Efficiency Video Coding (HEVC) [49] video coding standard. They developed as well an experimental framework in order to conduct the evaluation experiments [9]. This framework defined a View Synthesis Reference Software (VSRS), which supports several new rendering techniques.

The VSRS algorithm [50] takes two reference views, and two depth maps as inputs to generate a synthesized virtual view. The software has two main modes referred to as "General mode" non-parallel camera setups (convergent camera rig) and "1D mode" for 1D parallel setup (cameras aligned perpendicularly to their optical center). The intrinsic and extrinsic camera parameters are required.

In the general mode they use a "reverse warping" algorithm, which is reported to give a higher rendering quality [9]. First, the left and right depth maps D_L and D_R are warped to the

target virtual view position giving the warped depth maps D_{Lw} and D_{Rw}. The highest depth value (closest to the camera) handles the occlusions; usually, the depth values are reversed quantized from 0 to 255 such that the highest value in the depth map corresponds to the lowest depth of the scene [19]. To fill small holes in the warped depth maps D_{Lw} and D_{Rw}, we apply a median filter, giving D^f_{Lw} and D^f_{Rw}. A binary mask is maintained for each view to keep track of larger holes caused by disocclusions. D^f_{Lw} and D^f_{Rw} are then used to warp the texture views T_L and T_R to the virtual view position, giving T_{Lw} and T_{Rw}. If available, holes in one of the warped views are filled with non-hole pixels from the other warped view. This gives T^f_{Lw} and T^f_{Rw}, blended to form one new image.

The blending can be a weighted average according to the distance of each view to the virtual viewpoint (Blending-On mode), or it can simply consist in taking the closest view to the virtual viewpoint and discarding the other (Blending-Off mode). The binary masks of each view are merged at this stage. The remaining holes are filled at the final stage of the algorithm by propagating the color information within the region boundaries. Fig. 3.1 illustrates the rendering process in the general mode of VSRS.

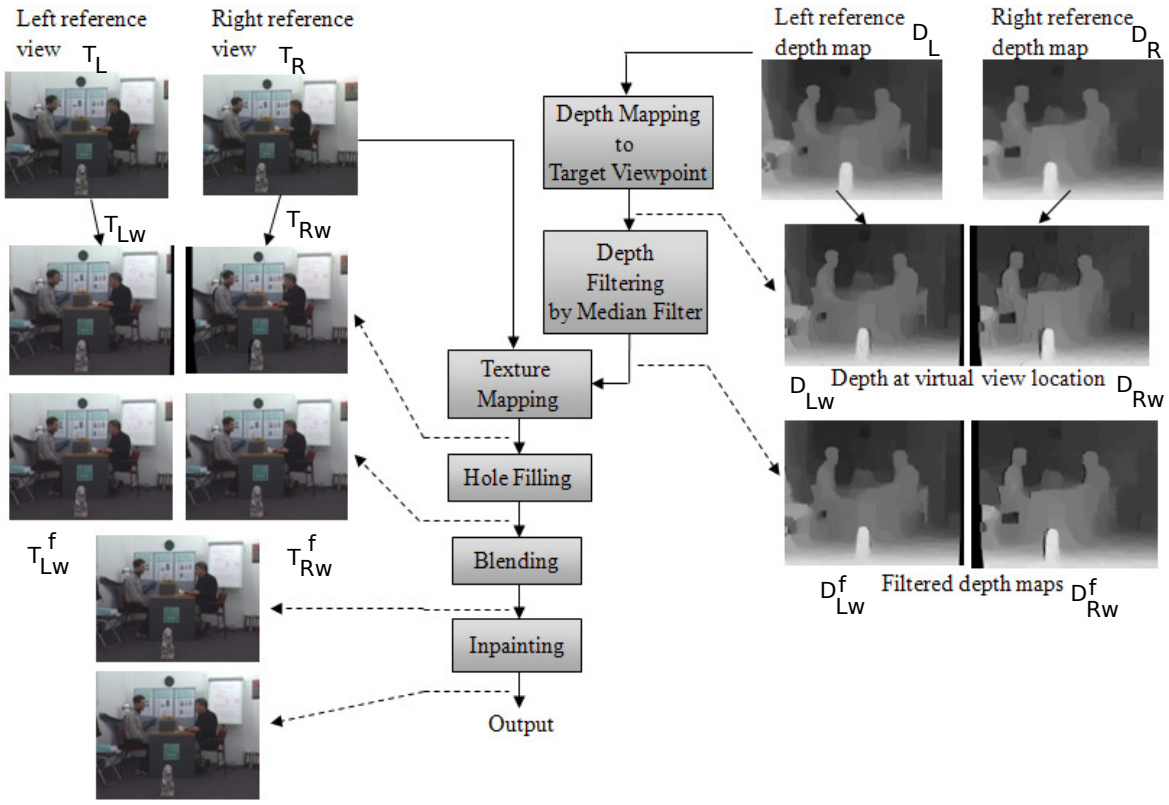


Fig. 3.1 Flow diagram for View Synthesis Reference Software (VSRS) general mode [9]

The 1D mode of VSRS works a bit differently. In this mode, the camera setup should be 1D parallel. This allows making several simplifications to the warping process, which

is reduced to a simple horizontal shift. First, the color video is up-sampled for half-pixel or quarter pixel accuracy. Second, a "CleanNoiseOption" and "WarpEnhancementOption" withdraw warping unreliable pixels. Finally, the process gives two warped images, two warped depth maps, and two binary masks from the left and right reference views. Each pair is then merged together. When a pixel is warped from both the left and the right reference views, the final pixel value is the pixel closest to the camera or an average of the two. The remaining holes are filled by propagating the background pixels into the holes along the horizontal row. Finally, the image is downsampled to its original size.

Fig.3.2 illustrates the rendering process in the 1D mode of VSRS.

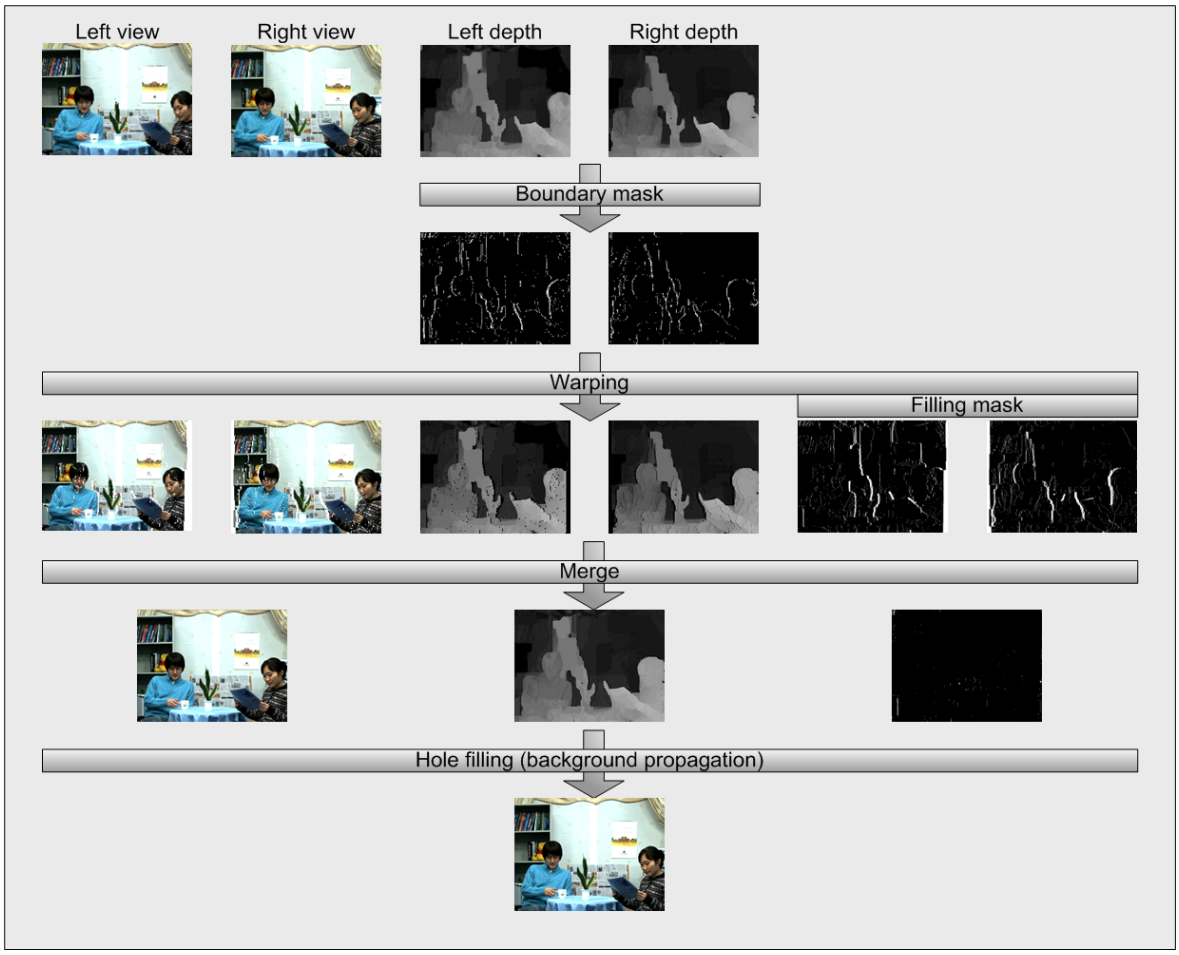


Fig. 3.2 Flow diagram for View Synthesis Reference Software (VSRS) 1D mode [9]

3.1.2 Reference View Synthesis (RVS)

The Reference view synthesis was created for the 3DoF+ investigations [51] by the MPEG. The 3DoF+ 360°videos are positioned between 3DoF and 6DoF, which can realize the motion parallax with relatively simple virtual reality software in head-mounted displays.

The Reference View Synthesis (RVS) is a view synthesis software that takes multiple input views to deal with a large baseline between the input views. The warping process of RVS consists in computing the floating-point floating-point image coordinate map, then, Unproject input image coordinates, which is the reference coordinate system to world coordinates. A single affine transformation $x \rightarrow R_x + t$ is applied to make the virtual camera the reference coordinate system, and the world coordinates are projected to the virtual image. The input view is divided into triangles with the centers of the pixels as peaks. Finally, the triangles are warped using the affine transformation before being filled with interpolated colors. The colors are the tri-linear interpolations between every three vertices of the triangle. Discontinuities between objects creating disocclusions and tangential surfaces may lead to triangles with very elongated shapes. They will not be kept in the final result, as they get eliminated during the blending phase.

All the synthesized images corresponding to each input view will be blended. The depth of a pixel and the shape of the triangle it belongs to determine whether a pixel is good quality. An exemplary pixel quality has low depth and belongs to a triangle with a regular shape. Concerning the inpainting step, it is straightforward where the color of the nearest pixel using a Manhattan distance are propagated.

3.1.3 Versatile View Synthesizer for 6DoF Immersive Video (VVS)

VVS [21] is a later reference software for view synthesis released by the MPEG superseding VSRS as a result of a challenge for 6DoF of the MPEG-Immersive (MPEG-I) group.

VVS is conceptually similar to VSRS; however, it accounts for the compression artifact on view textures and depth maps at the receiver side. Namely, VVS improves the precision of the backward texture warping and better preserves edges via a conditional depth blending process, up-sampling the reference textures, and using reliability maps that indicate which pixels are safe to be warped to the target view position. Unlike VSRS, VVS prioritizes foreground pixels during warping and warps "triangles" instead of "points" to generate fewer occlusion artifacts in the warped view followed by a series of hole inpainting steps.

Fig. 3.3 illustrates the processing chain implemented for generating synthesized views T^w (and associated depth map D_w) from the texture maps T^i and depth map D^i is associated to each reference viewpoint. The different treatments carried out are summarized here:

- Reference view ranking: reference views are ranked from "closest" to "farthest" according to an original metric. The farthest maps are not considered if the number of reference views set is limited.

- Pre-processing of depth and texture maps: this step includes the generation of resampled depth maps D^i, and the generation of spatially oversampled texture maps T^i.

- The classification of pixels likely to be incorrectly projected: this step includes the generation of a binary mask B^i_{map} of the pixels located at the border of the objects of the scene and whose depth is often severely estimated.

- Computation of the depth map associated with the virtual view (D^i_w): projection of the depth map points on the virtual view, triangular interpolation, and computation of the map A^i of the areas not observed by the view i.

- Conditional mixing of depth maps and computation of the merged map D^i_{wm}.

- The generation of the texture map T^i_w associated with the source view i: for each pixel of the view to be synthesized, a 3D point can be estimated and projected on the oversampled source view where the pixel value of T^i_w can be interpolated. The value of T^i_w is 0 if the point is projected outside T^i.

- Merging texture and depth maps: the associated texture and depth maps T^i_w and D^i_{wm} are merged to obtain the maps T_w and D_w.

- *Inpainting* of the unobserved areas: compute the new texture map T_w and the mask of the "imagined" pixels I_w.

- Temporal filtering: a non-motion compensated temporal average restricted to "imagined" pixels is used.

- Spatial filtering: The final T_w texture map is obtained by local correction of object boundary pixels and "imagined" pixels (using a 5x5 Gaussian filter for these areas).

VVS outperforms many view synthesis methods in reason of the high quality of the warped reference views.

It is worth noting that [21] compares favorably VVS, in terms of Peak Signal to Noise Ratio (PSNR), to the reference software VSRS and RVS based on Common Test Conditions (CTC) selected by the MPEG Immersive Video (MIV) group.

3.1.4 Rendering-algorithmic-based techniques in the literature

Chaurasia *et al.* [52] introduced the depth synthesis and local warps for image-based navigation method. Their objective is to provide a plausible free-viewpoint navigation for complex geometry in everyday scenes (such as vegetation, vehicles.) captured with a simple digital camera, where large regions are poorly reconstructed. Their main contribution is a two-step approach (*depth-synthesis* and *Local Shape-Preserving Warp and Rendering*) applied on over-segmented multiple input images. The over-segmentation creates super-pixels of homogeneous color content preserving depth discontinuities. The *depth-synthesis* algorithm

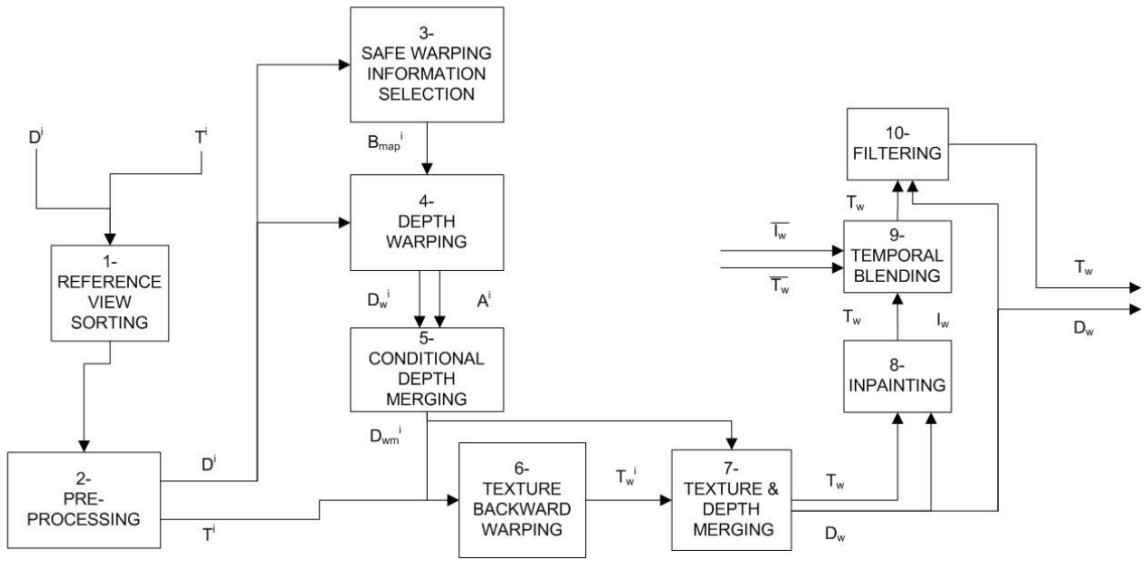


Fig. 3.3 Flow diagram for VVS [10]

is a pre-processing step to non-locally fill in poorly reconstructed super-pixels depth from "similar" super-pixels of the image. The poorly reconstructed super-pixels are identified by the over-segmentation and the projected depth. Three super-pixels are selected in color and spatial proximity as best super-pixel matching the poorly reconstructed super-pixel depth. This area is then filled by interpolation of the depth of these best matching super-pixels. The *Local Shape-Preserving Warp and Rendering* step aims to avoid visible artifacts in rendering due to the non-local warping process. Therefore a local shape-preserving warp is performed on each super-pixel individually. However, this method has several drawbacks since the depth synthesis algorithm fails to fill the poorly reconstructed super-pixels in more complex scene geometry (e.g., tree) and for dynamic content (e.g., people). The over-segmentation also does not capture very thin structures. Finally, their hole-filling approach results in blurring and artifacts for wide baseline view synthesis.

Approaches that use 3D reconstructions such as [52] require additional steps to fill in missing depth from the 3D reconstruction and suffer from hard tears at depth boundaries and occlusions. Penner *et al.* [23] propose a soft 3D representation that retains partial depth ambiguity to handle complex cases plausibly, and their geometry aims to be linearly interpolated smoothly into virtual views, with occlusions and image edges being explicitly modeled. Therefore they developed a 3D soft reconstruction and a view synthesis algorithm that computes depth maps using a fast local stereo method that provides visibility estimates to perform per-pixel view selection and improve depth edges. Then, the soft 3D reconstruction consists of volumetric vote volumes that model uncertainty from stereo and depth refinement (voting) operations. This representation is refined iteratively by providing a soft visibility estimate for per-pixel view-selection in a subsequent stereo pass. Finally, they use the exact soft rep-

resentation directly to model ray visibility and occlusion during view-synthesis. However, this method may work well for a small baseline camera setup; it still suffers from visible artifacts for wide baseline viewpoint reconstruction.

Lin et *et al.* [53] assumes that these methods mentioned do not entirely solve the fundamental problem of how to achieve real-time 3D video generation. Therefore, to simplify the generation process of virtual viewpoint images, the proposed algorithm uses the principle of a reverse search for matching to fast generate multi-view 3D images directly from 2D images. It is also reported to help reducing operation time and memory usage and lower hardware cost. The method consists of calculating the parallax range of the original image, then generates a warped image through a reverse mapping, containing information for eight different viewpoints. Finally, the image modification can be made. However, there is no significant visual improvement reported comparing to the classical DIBR methods, especially for wide baseline cases.

Prakash et *el al.* [54] targeted a new view synthesis algorithm that focuses on scenes with wide-baseline capture, allowing free-viewpoint navigation for far from the input views at interactive frame-rate. Their work first identifies the common Image-based-Rendering (IBR) artifacts and combines the strengths of different algorithms to discover a good balance in the speed/quality tradeoff. Their contribution lies in proposing a new multi-view image harmonization algorithm that reduces color differences in diffuse regions in wide-baseline captured scenes. The color differences are due to different camera parameters (exposure, white balance, etc.) or lightning that may change during capture; we call these regions view-independent effects. At the same time, they tend to preserve the view-depend effects, such as glossy material appearance, which are identified and re-injected in the harmonized input images. This results in modified input images with diffuse regions that can be seamlessly blended in IBR while preserving view-dependent content. Then they propose a blending algorithm for Per-View Meshes (PVM) and analyze the depth errors of view-dependent meshes to propose a new filtering approach based on the clustering of the depth of the different meshes, followed by spatial filtering. Finally, they analyze the potential of different rendering algorithms by identifying regions where the global mesh is sufficient by estimating geometric uncertainty and use the new filtering per-view mesh IBR algorithm for the uncertain region. This final hybrid IBR method provides a quality/speed tradeoff. Despite that this method combines the strength of three different algorithms in a novel hybrid algorithm without a neural network, it stills suffers from drawbacks that result in artifacts or ghosting in the rendered image whenever the method assumptions are violated.

Scalable inside-out image-based rendering [55] is a new system that allows users to convert a room in their house into virtual computer graphics. They can navigate interactively and freely move in a photorealistic scene where the reflections and highlights come and go accordingly. Everything in this reconstruction subsequent image-based rendering pipeline is computed automatically when footage of a scene is captured. The user shoots in all directions

using a regular camera and a standard color plus depth camera like the Microsoft Kinect. The system discards the frames that it does not need, so more is better during capture. To build a 3d wireframe mesh of the scene, they explored many algorithms designed to merge color, images depth, or both. Ultimately, they adopted a standard structure from a motion algorithm based on interest points to register the color images into a sparse 3d point cloud. The depth images are then fused onto a scaffolding. The payoff of this approach is that footage can be captured by non-experts and leads to reconstructions with coverage of the whole room. Such a global reconstruction agrees with all the footage; however, some details visible in just subsets of images are then lost in the process. For improved image-based rendering, they compute additional local geometry from just nearby views. While the local reconstruction may not agree with all the other footage of the scene, it does align more closely with details in a specific image. For the interactive rendering, their image-based rendering is designed to cope with large scenes while still providing good quality at an interactive rate. However, an increasing number of input images affects quality and performance. Their primary method uses meshes simplification enabling interactive framerate with hundreds of images on a PC. Nevertheless, considering all input views rendering causes the frame time to be linear in the number of the input images. For better scalability, they built a spatial scalability tiling and prioritization process. This helps larger scenes run at interactive rates and makes the system usable on more modest hardware. The quality difference with and without tiling is minimal.

Early works, such as that of Xiao et *al.* [56], propose a scalable bit allocation scheme, where a single ordering of depth and texture packets is derived. Furthermore, depth packets are ordered based on their contribution to the reduction of the synthesized view distortion. Another method for improving the synthesis's quality is applying a non-linear transformation to the depth maps [22]. Specifically, the depth range of points in the background is compressed, such that these points would have the same or slightly different depths. This reportedly reduces holes in the synthesis. The transformation depends on the depth map histogram. Objective gains are not presented, but a visible improvement is noticed on the shown images.

3.2 Disocclusion inpainting methods for Depth-based View Synthesis

A common issue with all the above methods is recovering occlusions in the reference views, i.e., areas occluded in all references, showing up in the synthesized view as visual artifacts. Resolving occlusions is a challenge per se, and several approaches have been proposed, mainly based on hole inpainting [19, 24–26, 57–59].

3.2.1 Traditional image inpainting methods

Traditional image inpainting, are mainly divided into diffision-based methods [27], [60], [61] and patch-based methods [62], [46], [63].

Diffusion-based image inpainting mainly spreads the pixel information around the broken hole in the image and synthesizes new textures to fill the hole. Unfortunately, the information around the hole usually restricts reconstruction; it is difficult to learn from distant information reasonably and lack a high-level semantic understanding of the image, making it challenging to restore meaningful texture structure in the disappeared area. Moreover, as the distribution distance of pixel information around the hole increases, the larger the hole is, the less effective pixel information will be obtained in the center. So the traditional diffusion-based method is more suitable for structure texture background inpainting and removal of small objects in the image. However, the effectiveness of large-hole restoration for natural scene objects with complex textures in real life is limited.

The patch-based image inpainting assumes that the degraded area and visible area of the image have related content. It searches for the best matching similar patch in the visible area of the image and then mimics the information to fill the missing area at pixel level. The traditional method [64] usually requires immense computing power to calculate the similarity score between patches. PatchMatch [63] reduces the high memory and computational cost of the search process by using a fast nearest-neighbor field algorithm. Furthermore, it shows a particular practical advantage in image editing applications. However, more often, the content of the image loss area may be a completely independent small individual or disorganized biased damage. Thus, at this time, the traditional patch-based method may become challenging to handle.

In exceptional cases, it may not be possible to find a patch similar to the missing area in the image. Therefore, some researchers have proposed image inpainting between images, which mainly refers to searching for an image with similar semantics to the target damaged image in the existing image, and then select appropriate patch information for porting and acquiring. James Hays *et al.* [65] used million-level data to find the most similar image of the target inpainting image. They then extracted corresponding area information to complete the degraded image. This method can better repair degraded images when there is a large amount of image data in a particular field. However, it also often means that a large amount of field data collection and the best match search need to be carried out [66], [67], [68]. So in the real scene, such methods have fewer applicable scenarios, and the application range is relatively limited.

To minimize disocclusion holes in the synthesized novel view, Thatte *et al.* [69] proposed a statistical model that predicts the likelihood of missing data in synthesized images as a function of the viewpoint translation. Li *et al.* [70] employed multiple views to synthesize output virtual views. They propose a scheme of selective warping of complementary views

developed by locating a small number of valuable pixels for hole reduction. In [58], multiple reference views are warped and combined to generate a blended image. Other methods use the neighbor pixel color or the depth information to extrapolate or interpolate the occluded pixels [43, 57], or by pre-processing the warped depth maps [71]. In [31], Luo *et al.* extract the foreground objects in reference images and synthesize the background to be used to fill holes in the synthesized view, as they consider that occluded pixels have the same patterns as the background. Yao *et al.* [72] exploit the temporal correlation of texture and depth information to generate a background reference image, used to fill the holes associated with the dynamic parts of the scene.

Bi *et al.* [73] assume that filling the holes generated due warping process using only texture image has low efficiency, that is why they proposed a hole filling approach with the corresponding depth maps. Their main contribution is that after warping the reference views using a classical DIBR 3D warping method, they layer the complex depth scene into simple scenes and fill holes in these layered depth scenes using the approach based on patch sparsity in the layer. Namely, the priority of holes is calculated for each simple scene of a layer. Due to the strong structure of hole edges, the priority of patch in hole edge might be too high, which will result in filling an order from foreground to background. Similarly, Satapathy *et al.* [74] work to enhance the disparity maps estimated from stereo pairs and depth maps. First, they deal with small missing pixels in every degraded depth map by exploiting a non-local extension of the Gauss-Markov Random Field (GMRF) prior. Second, they handle the large occlusions, using an over-segmentation on the corresponding RGB frame using superpixels. These superpixels are overlaid on the degraded depth map wherein large missing regions are inpainted by optimizing an objective function formulated using the non-local GMRF prior.

For a better hole filling, Lee *et al.* [75] maintain the depth structure of occlusions and perform morphological operations on images and the background depth levels. They use DIBR to image warping with a weighted gradient vector flow. The edge detection algorithm allows the image-labeling method to find the edges of large holes; thus, the gradient vector flow direction is used to find color and depth information of the edges of large holes. An n by n weighted gradient vector flow is created for the labeled large holes of the images and the maps, and the weighted gradient vector flow is used to process the images and the maps. Finally, a median filter is used to fill the holes. Similarly, Gao *et al.* [76] claim to subjectively overcome deep and non-deep learning methods significantly with their disocclusion filling approach. They first use the Gaussian mixture model to extract the background from the image. Then, they propose a refined foreground depth temporal correlation approach that recovers the background frame-by-frame based on depth information. These two methods are used to chose the adaptive pixels background for a good filling.

3.2.2 Learning-based image inpainting methods

The emergence of generative models such as GAN [77], [78], allowed the image inpainting method based on generative model to learn the high and low-frequency feature information of the degraded image visible area and learn the consistency of image structure and texture at the high-level semantic level . This can be done by adding different constraints to generate novel and reasonable features to complement the degraded area. Therefore, in recent years, various deep learning image inpainting models based on the generative network have been the hot direction for many researchers to improve.

Pathak *et al.* [47] proposed an image inpainting network called context-encoder, which employs unsupervised feature learning induced by context-based pixel prediction to large-hole image inpainting. Overall, architecture is a simple encoder-decoder. The encoder extracts feature representation of the input image, and the decoder expands the compressed feature map step by step to restore to the size of the original picture. Convolutional layers cannot directly connect all locations within a specific feature map. Thus, the encoder composed of convolutional layers is no way for information to propagate directly from one corner of the feature map. Therefore, a fully connected layer with groups based on stride one convolution to propagate information cross channels method is proposed as the intermediate connection between the encoder and the decoder to propagate information within activities of each feature map. Context-encoder adopts reconstruction loss (L2) and adversarial loss to handle both continuities within the context and multiple modes in the output. The reconstruction loss is responsible for capturing the overall structure of the repaired area and the consistency with the surrounding visible area; the adversarial loss predicts the repaired area looks real. Thus, the best possible inpainting results can be generated by maintaining the balance of them. Context-encoder can understand image semantics to a certain extent and predict pixels according to information around the hole to generate new content. At that time, it was a very cutting-edge image inpainting technology.

Researchers have proposed a globally and locally consistent image completion method [79] to solve context-encoder defects inspired by context-encoder [47]. For example, we can only process static low-resolution images. The mask area must be located in the center of the image. The complete area cannot maintain local consistency with the surrounding area. The network uses two auxiliary context discriminators for training. The global discriminator network takes the entire image as input. The local discriminator network takes only a small region around the completed area as input to ensure the global and local semantics of the restored image, respectively. Dilated convolution [80] is used in the middle four layers of the complete network to increase the receptive field of the extracted features. Thus, images of any size can be completed, and new textures and objects can be generated according to local and global structural-semantic information.

Image-level inpainting methods [46], [63], [79] generally lack understanding of the high-

level semantics of the image. Indeed, searching for similar patches in the visible area of the image and copying them to the damaged area for texture synthesis to fill the damaged area cannot produce semantically reasonable results. Moreover, even if generative models [47], [30] can enhance semantic consistency of repair region, stacked constructions and poolings to a certain degree can cause image resolution details to be over-smooth and lack visually realistic. Zeng et al. [81] proposed a Pyramid-context Encoder network (PEN-Net). It consists of a pyramid-context encoder, multi-scale decoder, and an adversarial training loss. It can fill missing regions at both image-level and feature-level for improving capability in image inpainting. The main innovations are as follows: A transfer network is introduced to learn the affinity in a high-level feature map between the damaged and visible areas. Then convert visible area-related features into low-level higher resolution feature maps according to the patch affinity weight to fill the missing content through ensuring image restoration's visual and semantic coherence. A multi-scale decoder with deep supervision of pyramid loss and adversarial loss is proposed. Similar features learned by the transfer network and latent features are decoded together to obtain corrected images through skip connections. This design can not only make training converge fast but also can make the test more realistic.

Previous image inpainting methods are often limited to low-resolution image inpainting, typically smaller than 1 K. Yu and Lin et al. [82] proposed a high-resolution image inpainting method to remove large target blocks without a trace. The whole inpainting process is divided into two steps. In the first step, the coarse inpainting result of a low-resolution image is obtained using the coarse cascade to fine network structure. Then, in the fine inpainting step, the confidence map of the repair result is introduced to assist iterative correction of the unsatisfied area in obtaining the fine inpainting result. The second step is with a guided inpainting upsampling network to generate an inpainting image given the first step inpainting result. The guided upsampling network consists of two shallow networks, one branch for learning patch similarity by patchGAN discriminator [83] and the other for image reconstruction.

Meng-Li *et al.* [32] want to generate a 3d photo from one single RGBD input. To achieve this, they propose a 3 step pipeline to identify a fill-in realistic content in the regions occluded in the input. Given a single RGB image as input, first the depth edges are identify, then from each detected depth edge they generate a context and the synthesis region. Finally they use a local context to inpaint the depth edges in a synthesis region and then use the inpainted depth edges as a structured guidance to synthesis color and the depth values. This approach can generate 3d photos in a wide variety of settings. It is used for personal photos, or to reanimate historical moments.

In a nutshell, image inpainting technology has become an essential branch in the field of vision research. Deep learning image inpainting based on generation network gradually becomes the mainstream method. Researchers have continuously innovated and made significant progress in generation model selection, network structure design, the introduction of prior guidance, discriminator optimization, loss function optimization. However, several

problems still need to be solved, such as improving the inpainting effect of complex textures, large holes, and high-resolution images.

3.3 Artificial intelligence for View Synthesis

Over the years, a number of learning-based approaches to view synthesis relying on ConvNets have been proposed. Early approaches to view synthesis mainly addressed the problem of temporal frame rate upsampling, where missing pixels are interpolated temporal or spatial neighbors.

Niklaus *et al.* [20], for example, propose a convolutional architecture capable of joint motion estimation and pixel synthesis for temporal up-sampling. Liu *et al.* [3] propose a convolutional architecture that output is a 3D voxel flow field, used to sample the original input video with a volume sampling function to synthesize the final frame. The work in [12] deals with large object motion or occlusion by explicitly detecting occlusion leveraging the depth information to warp the temporally adjacent frames. Such approaches assume that the equivalent camera displacement between consecutive frames of the same view is small. So, they are intrinsically unsuited to the free viewpoint scenario we address due to the large baseline distance between cameras. Regmi *et al.* [84] use a homography as a pre-processing step to map the images between reference views and guide a generative adversarial networks to inpaint the missing regions in the novel view. However, their method mainly addresses the problem of generating images across street and aerial views.

Later on, view synthesis methods designed especially for free-navigation have been proposed. These methods are divided into two categories:

- the methods that use deep learning as complementary to their view synthesis algorithmic approach
- the methods that use deep learning for all the view synthesis steps.

3.3.1 Novel view synthesis using deep learning

Zhou *et al.* [11] give reference source image with the pose provided in their proposed solution for novel view synthesis. Then they take the second source image, which is a similar viewpoint, and they do a plane sweep of this image to get various information throughout the different depths of the image. This is fed into a neural network that extracts a background color image and blending weights. These blending weights are used to create representations of the image at different depth values. These different blending weights and background colors are blended to create an MPI representation such as in Fig. 3.4. A multi-plane image is an RGB-A version of an image at different depth levels or disparity levels. The disparity is

the inverse of the depth, so as something has a farther depth, it will have a smaller disparity, and if something has a shallower depth, it will have a more significant disparity. These planes describe fronto-parallel scenes, so essentially the same scene with being parallel at different depths. The novel view synthesis is created using these RGB-A images and basic camera parameters, allowing for novel viewpoints to be created through homography-based calculations along with alpha compositing, which allows the objects in the images to have a particular value allowing for closer images to have a more significant alpha value and farther images to a lesser alpha value.

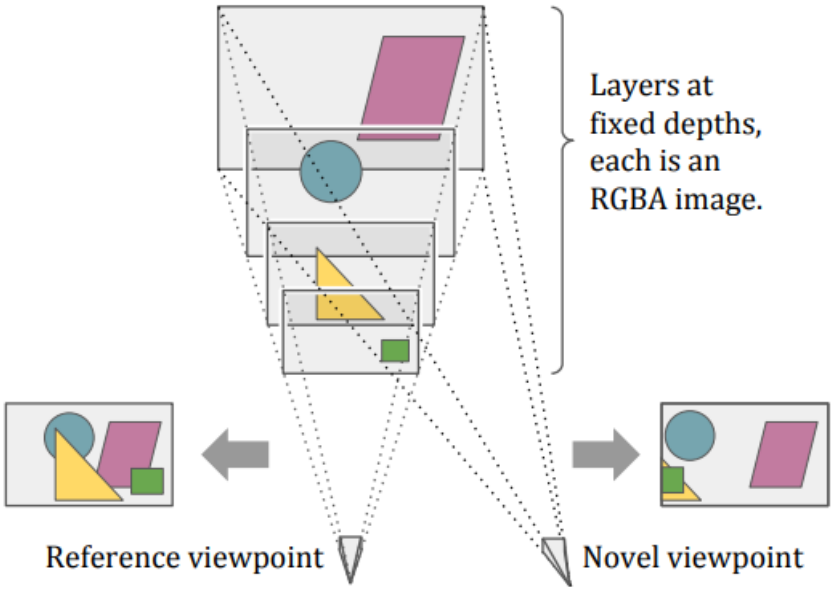


Fig. 3.4 An illustration of the (MPI) representation. An MPI consists of a set of fronto-parallel planes at fixed depths from a reference camera coordinate frame, where each plane encodes an RGB image and an alpha map that capture the scene appearance at the corresponding depth. The MPI representation can be used for efficient and realistic rendering of novel views of the scene [11] .

The following related work is the DeepMVS: Learning multi-view in stereopsis for Huang *et al.* [85]. This network is a three-part process. The first part is a patch-matching network that finds similarities with different images, all from the same objects but from different viewpoints. The next part is an intra-volume aggregation which takes the features from these different patches and aggregates them into one common feature map. The feature maps can then be aggregated into a single volume aggregation, which gives a total summary of the image depth within the input image. The output is a disparity map which is just a map that describes the different depths within the image.

- Patch-matching : it takes a reference image along with single patches from a plane

sweep volume and the camera pose of those images. The camera poses can be given manually or they can be defined using a structure from motion algorithm and COLMAP. This patch-matching algorithm extracts a 64 channel feature from both patches and then concatenates the two patches into a single feature of a 4 channel patch. This process is repeated for all the images at all different depth levels (in this paper they use a 100 depth levels, which is considered as a middle-ground for memory consumption and speed).

- The intra-volume feature aggregation is a UNet architecture that also encodes features from a vgg-based network. It takes neighboring images and concatenates the four-channel patches that were taken from the previous network. While encoding and then decoding, the features from the vgg-based architecture encoded at every level of the encoding process, they are all concatenated on, and then once the channels are then decoded, the output at 800 channel volume, which essentially describes the disparity within the whole image.

- The inter-volume feature aggregation: After having a volume representation of all these scenes, all these scenes can be then aggregated into one scene. This is done by using max-pooling because it allows for the network to have an arbitrary amount of images and an arbitrary image order. We have the essential features within the disparity map, so we look for the maximum values. Finally, we downsample using standard convolutional layers once these values are aggregated, and we get a D-channel pixel map.

To further improve the quality of the results, they apply the Fully-Connected Conditional Random Field (Dense-CRF) [86] to the raw disparity predictions. The use of DenseCRF encourages the spatially close pixels and with similar colors to have closer disparity predictions.

Choi *et al.* [87] try to propose a new solution for extreme view synthesis. Extreme views are defined as having a small number of input sources as little as two in this network. They want to be able to synthesize new views that are varying substantially from the original viewpoints. In addition, they are trying to generate synthesized images that create disocclusion by viewing certain objects from different viewpoints. Finally, they want to increase the baseline magnification of these images, allowing for more explicit images. Therefore, the proposed solution is to estimate depth probability volumes using the DeepMVS network. These depth probability volumes are just the disparity maps. Once all the depth probability maps are available for all the source images, they can fuse them and warp them with the novel new depth. This is done with a simple affine transformation. Once the synthesized image is created, there will be artifacts and holes within the image as expected when we have a novel

viewpoint. Thus, they propose a refinement network that takes this image and clears it up at the patch level.

This network overcomes the [11], and [23] methods; however, their refinement network does not hallucinate pick pixels. It only looks at patches from different viewpoints and finds similarities between them, and refine the actual image. In addition, the novel views are affected by depth quantization artifacts, meaning that it could be challenging to define the depth values along the edges, which entails creating artifacts.

Flynn *et al.* [25] present a new technique for photo realistic view synthesis. Using a set of sparse input views, the method uses the learned gradient descent to generate a MPI, which is, as we said before, a representation of the scene that allows real-time rendering to new viewpoints with standard graphics hardware. MPI is a representation that can model complex appearance effects, including transparency reflections and even volumetric effects. To render an MPI to an image on a new viewpoint, their MPI images are first warped and then composited using a standard alpha blending. To create those MPI, Flynn *et al.* [25] use the learned gradient descent. This technique combines the best features of deep learning and direct optimization. Traditional deep learning relies on optimization only during the training to find the network weights. However, learn gradient descent models inverse problems such as MPI generation as unrolled optimizations at inference time. This is achieved by replacing the simple gradient descent update rule with the ConvNets. The idea is that the ConvNets can learn to take larger steps than the standard gradient descent update rule and reduce overfitting by constraining the generated model (the MPIs) to lie on a plausible manifold scene. These gradient computations and update blocks are stacked repeatedly, creating a form of the recurrent network. They train a whole unrolled network by rendering the generated MPI to a held-out view. The network showed to make significant improvements at each iteration requiring only a few steps. They show that the gradients provided to the ConvNets an intuitive interpretation in terms of visibility clues. Effectively later iteration uses visibility to find a better solution in occluded regions. Their method correctly reconstructs complex geometry such as trees, and thin structures, and transparent surfaces; they overcome [23], and [11] methods.

Volker *et al.* [88] build upon [25] approach with the concern of creating a lightweight network that can be trained on a limited dataset. They mainly address view synthesis from large baseline light fields by turning a sparse set of input views into an MPI. Because available datasets are limited, they propose a lightweight network that does not require lengthy training. Unlike DeepView [25], they do not estimate color layers but only encode the scene geometry within the MPI alpha layers, which is a segmentation problem. Then, they implement a recurrent refinement, where the same network loops on successively enhanced MPIs. This allows a lightweight model that is successfully trained on a reduced dataset consisting of a few dozen light field sequences.

Broxton *et al.* [13] (Google) proposed a pipeline that turns the traditional photography

and video into something more immersive 3d video, that led to look around the object and move freely without uncomfortable limits on the head motion. Google calls this new technology "Immersive light field video with layered mesh representation," a system capable of capturing, reconstructing, compressing, and rendering high-quality immersive light field views at a bandwidth low enough to be streamed over a regular Wi-Fi. In the new system, wild field of view scene can be recorded playback to move around within the video after it has been captured. The system can capture challenging content such as reflective surfaces. The authors use a custom low-cost 46 time-synchronized camera mounted to an acrylic dome called a camera rig to record an immersive light field. From this data, they will produce 6 degrees of freedom volumetric video with a wide 80cm viewing baseline, 10 pixels per degree of angular resolution, and a wild field of view at 30fps video framerate. Even if the cameras are placed 18cm apart on average, the system can reconstruct objects as close as 20cm to the camera rig. A machine-learning algorithm developed by DeepView [25] is used to combine the producing from each camera into a 3d representation of the scene being recorded. They replaced the deep freeze underlying multiple images with MPI representation with a collection of spherical shells better suited for representing panoramic light field content. The data is then processed to reduce many shell layers to a small fixed number of RGB+D layers without significant loss in visual quality. The system layered mesh representation is a series of concentrated layers with semi-transparent texture rendered from back to front, bringing the scene vividly and realistically to life and resolving the issue of synthesizing viewpoint that was never captured by the camera in the first place. This enables the user to experience a natural range of head movement, as they will explore light field video content. The resulting RGB-alpha depth channel in a layer is compressed using conventional texture analyzing video compression technology. The final compressed representation is lightweight and can be rendered on mobile via VR/AR platforms or in a web browser. The user will be able to stream light field video content over a typical fast internet connection. However, a drawback of this method is that it is complex dis-occluded regions sometimes appear blurry; This results from the following approximations: First, due to GPU memory constraints, they train with a local cluster of only seven cameras, although they still perform inference with all 46 cameras. Thus, during training, the network has a more limited ability to peek around objects and learn to distinguish foreground and background content. Second, there is a loss of quality when converting to layered meshes. However, this method allows only a restitution in 3DoF and not in 6DoF.

In [26] Hedman *et al.* follows a similar architecture to InsideOut [55] described in 3.1.4, but enhances realism to the improved per-view geometry and deep blending. Their crucial novelty is the introduction of a deep blending method for IBR that synthesizes each novel view after having learned to compute the *blending weights* to combine the relevant input photos. The input to their method is a set of input photographs of a scene. First, they calibrated the cameras using Structure From Motion (SFM). They use per-view meshes for

rendering. The goal is to provide high-quality per-view depth map refinement and generate compact meshes that respect occlusion edges as much as possible. To allow a per-frame interactive rendering loop that includes ConvNets evaluation, they use the U-net architecture and generate a fixed set of inputs to the ConvNets. For the rendering, they build on InsideOut [55] which at each output pixel selects a variable number of input photos to blend into a final image. The rendering loop ranks these per-pixel selections to generate a fixed number of mosaics blended into the novel view. Each pixel of the first mosaic contains the color value of the best-selected pixel. The second mosaic contains the color value of the second-best and so on. Pixels are ranked according to an IBR cost. The blending step of IBR aims to compensate for geometric and visibility errors and view-and image-dependent effects. They trained a convolutional neural network to generate *Blending weights* that are then used to combine warped contributions from different input images. The network architecture has a receptive field of 63 pixels at the bottleneck, giving it the power to correct artifacts that are at most this size. This network requires a fixed number of inputs, which are five images in this paper, one global mesh renders, and 4 source-image mosaics, that need to be blended. Similar to InsideOut, their rendering algorithm uses a spatial acceleration data structure to select, at each output pixel, pixels in multiple input images that contribute to the novel view. However, this method degrades and creates blur in regions with missing geometry or where difficult decisions need to be made.

Neural radiance fields or Nerf [89] is a new method for representing complex scenes that achieves state-of-the-art results of view synthesis. Given a set of input images of a scene, they optimize a volumetric representation of the scene as a vector-valued function defined for any continuous 5d coordinate consisting of a location (x, y, z) and view direction. The scene representation is parametrized as a fully connected deep network that takes each single 5d coordinate and outputs the corresponding volume density and view-dependent emitted RGB radiance at that location.

Then, they use techniques from volume rendering to composite these values along a camera array to render any pixel. This rendering is fully differentiable, so it is possible to optimize the scene representation by minimizing the error of rendering all camera rates from a collection of standard RGB images.

Wizadwongsa *et al.* [90] want to capture the real world and turn it into a digital 3d model. Given a sparse set of input images from different viewpoints, about 10 to 30 images are turned into a scene representation that allows real-time rendering on standard graphics hardware. This representation, or what is called Nex MPI, is a multiplane image but with significant improvements that allow modeling view-dependent effects. After reconstructing the Nex-MPI, we can use it to generate new images from novel viewpoints. This method aims to reproduce challenging objects in the scene, like a transparent glass, the specular highlights on a spoon, or the reflections of food plates. Their main contributions are a new representation that can render the scene with challenging view-dependent effects in real-time.

This is done by re-parameterizing each pixel as a combination of view-dependent neural basis functions. Then, they introduce a mixed modeling strategy that uses both implicit and explicit representations and achieves the best visual quality scores across all the significant metrics on three benchmark datasets. Finally, they introduce a new dataset called shiny, which contains scenes with highly challenging effects such as the rainbows reflections on CDs. The representation is based on MPI, and in order to synthesize these novel views, MPI uses a set of semi-transparent planes stacked together along a viewing frustum. This technique is capable of real-time rendering; however, it cannot model view-dependent effects because each pixel is represented with RGBα, which is the same for all viewing directions. To model view-dependent effects, the color of each pixel needs to be a function of both the spatial location xyz and the viewing direction ϕ, θ. The key in this method is to factorize this 5d function into the sum of products between k_n and h_n:

$$RGB(x,y,z,\phi,\theta) \approx \sum_{n=0}^{N} k_n(x,y,z)h_n(\phi,\theta) \tag{3.1}$$

where k_n depends on the location and h_n depends on the viewing direction. This technique is closely related to the problem of light field factorization. In this work, they propose to use neural networks to learn both k_n and h_n showing that it outperforms other fixed basis functions. In the original MPI, each pixel is represented by RGBα values; the Nex-MPI each pixel is represented by α k_0 to k_n. These k's are coefficients of the basis functions, and each k has three dimensions corresponding to the RGB channels. Finally, they can evaluate this linear combination and produce the final view-dependent RGB value given the location and the viewing direction. This method achieves higher PSNR than Nerf [89] and can successfully recover fine detail while Nerf is struggling. Importantly, it is possible to render the scene with over 60 frames/sec. In terms of megaflops for 1 pixel, Nerf uses 200 while Nex uses 0.16, which is about a thousand times faster. However, the model cannot fully reproduce the most complex scenes in the Shiny dataset with extremely sharp highlights. Training MPI still takes a long time and may require more input views to replicate view-depend effects.

Yoon *et al.* present a new method to synthesize a novel view from dynamic scenes using a monocular camera. Given images captured from a dynamic scene, the goal is to synthesize the images at arbitrary views and times. The key idea is to combine the strengths of stereo-based and learning-based depth to estimate a scale and variant and complete depth map. Using this depth map they render a photorealistic virtual view by warping and refining the images. The method enables various applications such as space-time navigation, bullet time effects, and customized cinemagraphs. It is applicable to general dynamic content.

3.3.2 End-to-end View Synthesis networks

PixelNerf [91] is a new framework for predicting neural radiance fields from one or a few images in a feed-forward manner. PixelNerf can seamlessly incorporate multiple observations such that given two views, it can output a more accurate representation. Furthermore, PixelNerf models the entire scene to be trained on more complex settings with more than one object without modification. PixelNerf can be applied to real scenes from the DTU dataset without test time optimization. In contrast, because Nerf [89] is unable to exploit prior knowledge common across scenes and purely relies on multi-view consistency, it performs poorly when only three input views are available. PixelNerf equips the volumetric Nerf representation with fully convolutional image features to learn priors across the scenes. The input image is first encoded into a pixel-aligned. Then, they render points along with a target array. Finally, for each 3d point, they query the feature grid at the projected pixel coordinate in the input image and pass the image feature along with the coordinates of the point in view space into the network to get color and opacity. This is done for every point along the target ray then volumetrically rendered to the pixel value in the novel target view. When multiple input views are available, every input image is encoded into a feature grid. The multiple features are processed in parallel and aggregated into the final color and opacity. The authors claim that this method operates in view space and does not require mask supervision. It can also perform wide baseline view synthesis on real complex scenes from the DTU dataset using 88 training scenes. PixelNerf can obtain some reconstruction from only one view and outputs better results with more observations. In contrast, Nerf [89]fails to achieve good results with very few views. Despite being trained on three input views, their method also improves slightly with more views. Note that all the results shown in this paper are the direct output of the network without test time optimization. Whereas the training on Nerf takes 14 hours for each scene with a set of input views. Ultimately this approach bottlenecked by the availability of large-scale wide baseline multi-view datasets, limiting the applicability to ShapeNet and DTU.

In [92] they presented work on Generative View Synthesis (GVS) from single view semantics to novel view images. Semantic image synthesis techniques, for instance, convert a semantic map to a realistic image. Artists and the general public have well-received these methods. The GVS problem consists in rendering an input 2d semantic map from an arbitrary camera poses. A generative view synthesis approach takes 2d semantics as input and renders a scene from multiple novel views. GVS is, thus, related to image-to-image translation and monocular view synthesis problems. A straightforward GVS approach could be devised by combining state-of-the-art image-to-image and monocular view synthesis methods. However, the input semantics are first translated to a photo-realistic image. The translated image is rendered from arbitrary camera poses using a monocular view synthesis method. [92] claims that such combination results in unsatisfactory outputs. Thus, they

propose the GVSNet, which renders novel views by lifting the input semantics into a hybrid layered 3d semantic representation and performing image-to-image translation in this lifted semantic space. Semantic lifting from 2d to 3d is a key idea that enables this method to produce novel views that are semantically consistent with the semantic input map. GVSNet has three main sub-networks: Semantic Uplifting Network (SUN), Layered Translation Network (LTN), and the Appearance Decoder Network (ADN). SUN is trained to convert the 2d semantics into a multiple plane image semantics MPI-S. This method uses a hybrid model where scenes semantics are compressed into few-layer semantic images. This allows our model to capture the semantics of all surfaces in the scene without wasting memory. The lifted semantics representation are converted into a layered appearance using LTN. Layered appearance features are then converted into an MPI appearance via an association tensor predicted by SUN. The MPI appearance model contains high dimensional features at each pixel location on every plane. Therefore, projecting the MPI appearance to novel views gives per-pixel appearance descriptors at each pixel location in the novel view. ADN is applied at the end to convert the projected appearance descriptors into color values. Projecting high-dimensional features allows for the propagation of neighborhood information from the source camera to novel views. This local neighborhood information is crucial in mitigating rendering artifacts from multi-plane discretization of the 3d space. The results show that this method can synthesize more geometrically coherent views while preserving semantic structures present in the input semantics.

Riegler *et al.* [93] propose a method that erects a geometric scaffold and then uses a recurrent mapping and blending network to render new views of the scenes. First, they use SFM to estimate the camera parameters for the input images. Next, they run a multi-view stereo to get an initial 3d reconstruction of the scene. Delaunay-based surface reconstruction is used to obtain 3d proxy geometry. This proxy geometry assists the mapping of image features from input views to new target views. A recurrent network that integrates information from multiple source images performs the processing of image features in the target view. Each source image is first encoded using a convolutional network. The encoded features are then mapped to the target view using the proxy geometry. The mapped features are decoded by another convolutional network that produces estimated colors and confidence values in the target view. Estimated colors for multiple nearby source images are integrated into a coherent prediction. The results show that this method overcomes Nerf [89]; however, there is no temporal consistency as they synthesize images frame-by-frame. Thus synthesized videos exhibit temporal instability. Second, the pipeline will produce visible artifacts if the 3D model used for mapping misses large parts of the scene or has gross outliers.

[4] propose a method for performing view synthesis from a single image. It is trained end-to-end using a new differentiable point cloud rendered. It achieves impressive results for single image view synthesis. Given a single scene image, it learns a model to address what would happen if we changed viewpoint, such as moving forward and then turning left. To

solve this problem, their model has to learn about two aspects. First, it must understand the 3d structure of the scene to manipulate the pixels. The model must also understand the context in order to inpaint missing regions. To solve this problem, they use a latent point cloud representation. Given an image, they pass it through a network to obtain features at each pixel location. The depth map is also predicted at each location used to create a point cloud of features. Then they apply the available camera transformation to view the point cloud at the new viewpoint. Using ideas from traditional graphics approaches, they introduce the differentiable point cloud renderer, which renders these features at the new viewpoint. This gives a set of projected features, but it will have holes and potentially artifacts from where the point cloud does not project, such as regions in the new image that are not visible in the original one should not have projected features. In order to fix this, they use a generator network that is based on GAN models in order to fill in the missing regions. A set of losses is applied: discriminator loss and $L2$ loss, and a perceptual loss. At training time, the supervision is the input image, the rotation, the translation, and the target image, and no ground-truth depth is used. The depths and features networks are trained end-to-end using the differentiable point cloud renderer. At test time, the target image is not needed for the loss; they only take an image and desired new viewpoint as input. This is passed to a network to generate a new image of the scene at that new viewpoint. The results show that the model can generate viewpoints for indoor and outdoor scenes. However, approaches such [4] for single-view synthesis make small camera transformations, such as rotation by a few degrees. The following method is a single-view synthesis approach that aims at expanding the possible camera changes to include *large* transformations.

Rombach *et al.* [94] present a probabilistic approach to Novel View Synthesis based on transformers, which does not require explicit 3D priors. They synthesize plausible novel views that exhibit high fidelity given a single source frame and a camera transformation. They demonstrate that by modeling interactions between far-flung regions and the source image and target images, transformers can implicitly represent the required geometric transformation without requiring hand-engineered operations. Therefore, they propose learning a probabilistic model for view synthesis that appropriately considers the uncertainties inherent in the task. They analyze the need for explicit 3D inductive biases in transformer architectures and find that transformers make it obsolete to explicitly code 3D transformations into the model. Instead, they can learn the required transformation implicitly. Finally, they found that the benefits of providing geometric information in the form of explicit *depth maps* are relatively small. They also investigated the ability to recover an explicit depth representation from the layers of a transformer, which has learned to represent the geometric transformation implicitly and without any depth supervision. The experiments show that no such geometric priors are required and that the transformer is capable of implicitly learning 3D relationships between images. Furthermore, this approach outperforms the state of the art (notably [4]) in terms of visual quality while covering the full distribution of possible realizations.

3.4 Conclusion

To generate the most natural transition between two viewpoints in a scene, we studied possible computer vision methods that allow such transition. The one that retained our attention is the view synthesis with 3D warping. The 3D warping process is divided into three stages. The view warping stage is responsible for warping the reference views in the scene to the target novel viewpoint. The view blending stage blends the warped reference views altogether to obtain one final view. Finally, the view inpainting stage takes care of filling the holes generated by discolusions. Therefore in this chapter, we enumerate the most effective state-of-the-art view synthesis methods divided into three categories: the classic algorithmic-based view synthesis methods such as [21] [23]. The tradition algorithmic-based [28] [27] and learning-based [31] [30]inpainting state-of-the-art methods. Finally, the learning-based view synthesis state-of-the-art methods [25] [26] [4].

Recent research is mainly based on deep learning methods, used either such end-to-end view synthesis architectures [4], or such a booster for one or multiple view synthesis stages such as for blending [26] or inpainting [31]. The learning-based view synthesis methods accelerated and condensed the work in this field. They are achieving encouraging, outstanding results in a domain that has been challenging dealing with for the past two decades. However, these methods focused mainly on improving the quality of the visual rendered image novel view. Regardless of the number of input images and the computational and memory cost, the spectacular improvement is mainly performed on small baselines from the target viewpoint and many input viewpoints. Therefore, such methods are often designed for moderate inter-camera baseline distance, and larger kernels are required for warping if the baseline distance increases. Those methods can deal with large baselines; however, the synthesized view suffers from artifacts near disoccluded pixels.

In the following chapters, our work focuses on developing solutions that deal with large baselines view synthesis while mixing the advantages in algorithmic-based warping method with the benefits of ConvNets on improving the final rendered image quality. Our objective is to propose new solutions for large baselines to the target viewpoint position, with the minimum costs in terms of ConvNets learnable parameters, computational and memory storage, and reducing the number of the input reference images to two inputs.

Chapter 4

A hybrid approach to wide baseline View Synthesis with ConvNet

This chapter proposes a hybrid approach to view synthesis where we first warp the reference views resolving the occlusions. Then we train a simpler convolutional architecture for blending the preprocessed views. By warping the reference views, we reduce the equivalent distance between reference views, allowing smaller convolutional filters and thus lower network complexity. Later on, we improved our architecture by adding more convolutional layers while keeping the learnable parameters low and thus the complexity.

The rest of this paper is organized as follows. Section 4.1 reviews the relevant method challenges and introduces the related background. Section 4.2 describes the proposed view synthesis method. Section 4.3.1 experimentally evaluates the performance of our method. Section 4.3.4 describes the upgraded version of our method and shows the experimental results. Finally, Section 4.4 concludes the paper.

4.1 Context and challenges

View synthesis is the process of generating a virtual view of a scene by leveraging a set of neighbor reference views [95]. This tool would enable special services such as free navigation in a scene [18]. Methods based on Depth-image-based-rendering perform virtual view synthesis exploiting the scene's geometry where the virtual view is synthesized from two or more adjacent references. Even though multiple different approaches to DIBR exist, in most cases, first, the reference views are warped to the target view position using the depth maps and then are blended to synthesize the virtual view. Occlusions around the object edges, noise, or errors in the depth maps, can yield visible artifacts in the virtual views, demanding complex disocclusion filling or mitigation techniques before the images are blended.

Many inpainting and hole-filling methods are proposed to cope with occlusions in syn-

thesized views. For instance, in [96, 97] multiple warped input images are combined to generate one blended image. Nevertheless, such methods cannot avoid minor artifacts, and some holes are still present after blending. Some methods are based on interpolation or extrapolation of the hole regions with neighboring pixel color or depth information as [43] and [98], or by pre-processing the warped depth map like in [99]. Luo *et al.* [31] assume that black holes areas have similar patterns as background. Therefore, they extract foreground objects in the reference image and use the reconstructed and synthesized background to fill holes in the virtual desired image. However, the performance of such methods depends on content-specific parameters such as scene geometry, texture complexity, or baseline distance between cameras.

Recently, several methods based on ConvNets [100] have been proposed for image synthesis tasks such as video temporal interpolation or image super-resolution. Such methods [2], [3] [1] attempt to perform all the above steps with a learning-based approach. However, even in the case of moderate baseline (i.e., inter-camera distance), typical multiview sequences would require large convolutional kernels. Thus, it will be increasing the complexity of the above methods and making the network prone to overfit to the training data, demanding, in turn, large training sets to achieve satisfying generalization capabilities. Of course, this problem is only worse in the case of a large baseline.

In the following, we propose a hybrid approach to view synthesis that relies on DIBR for preliminary view warping and a ConvNets for view blending. Preliminary, we warp the reference views to the target position to minimize the disparity between views. Next, we handle occlusions around foreground object edges with a simple yet effective hole filling scheme. Finally, we blend the warped, disoccluded views with a lightweight ConvNets simple enough to be trainable in a fully supervised way. Compared to the DIBR-based state-of-the-art references, the proposal shows better image quality thanks to reduced artifacts such as ghosting, despite the moderate-to-large disparity between views.

4.2 The view synthesis approach

Our proposed method for view synthesis is schematized in Fig. 4.2, whereas Fig. 4.1 exemplifies the synthesis process. In a nutshell, first, we warp the reference views to the target position to reduce the camera baseline. Then, we resolve the occluded pixels, and finally, the two views are blended using a convolutional network.

4.2.1 View warping

As a first step, we warp the left and the right reference views T^L and T^R to the position corresponding to the target view to synthesize exploiting the relative depth maps D^L and D^R. For simplicity, in the following, we assume that cameras are arranged in a mono-dimensional

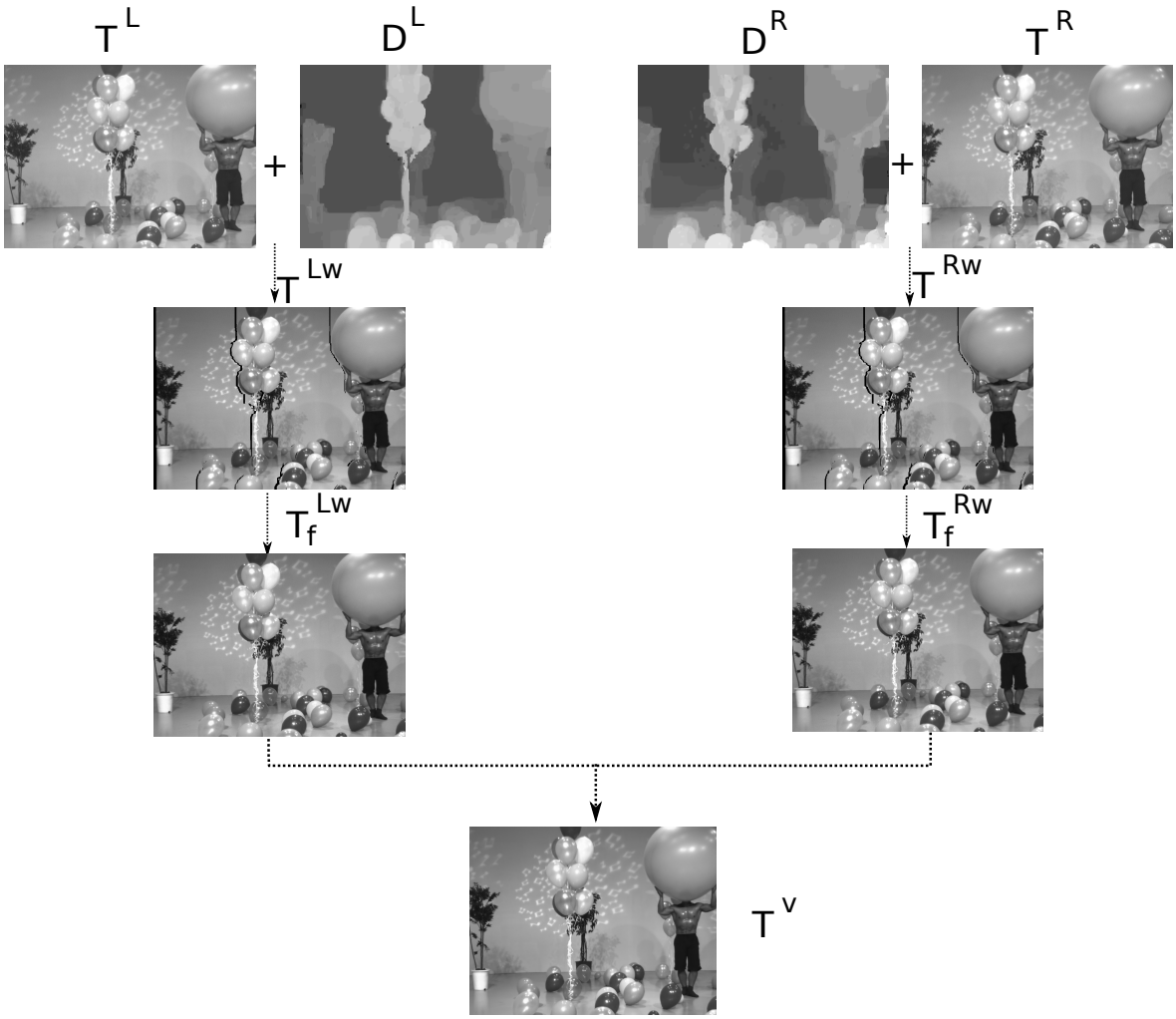


Fig. 4.1 Typical view synthesis process: the reference views T^L, T^R are warped to the target view position as T^{Lw}, T^{Rw}; occlusions-free warped views are indicated as T_f^{Lw}, T_f^{Rw}; the target blended view is indicated as T^V.

setup, i.e., arranged as a linear array whose geometry is known. Under these assumptions, the horizontal displacement of each pixel of each reference view to the target view can be computed as

$$d = \frac{f \cdot b}{z}, \tag{4.1}$$

f is the camera focal length, b is the baseline distance between the reference view and the target view, and z represents the pixel depth. The warping process has a sub-pixel precision of 1/4 pixel. That is, we first upsample the two reference images four times their resolution. Then, we shift each pixel to its new position and downsample the warped image to its original resolution. Fig. 4.1 illustrates the output of this process, consisting of the two reference views (T^L, T^R) warped to the target view position (T^{Lw}, T^{Rw}).

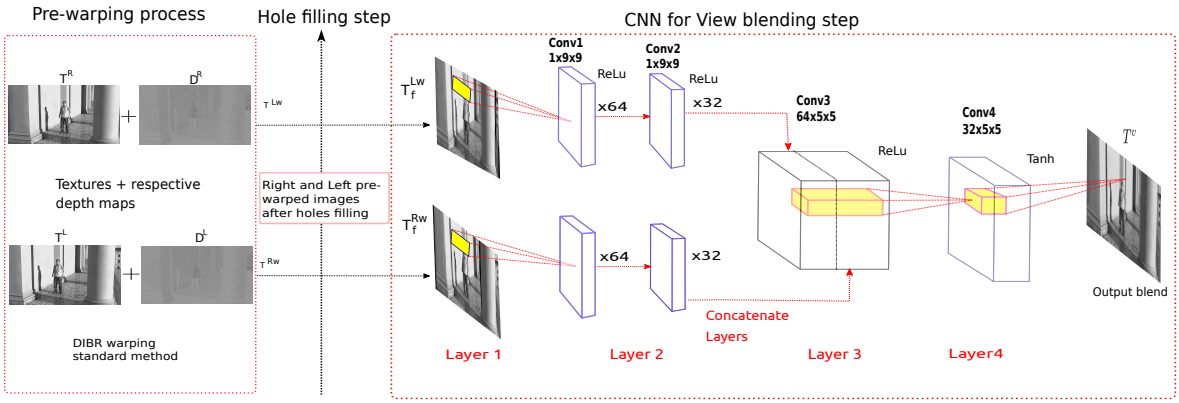


Fig. 4.2 The proposed method for view synthesis: the original views are pre-warped to the target position (left), occlusions are resolved via hole-filling (center) and the disoccluded views are blended using a convolutional network.

4.2.2 Hole filling

Due to the warping process of the reference view towards the target view, some visible area in the target view are invisible in the reference view and thus, they are represented as black holes in the warped views where each disoccluded pixel at a given position (r,c), $T^{Lw}(r,c)$ is set to zero. In Fig. 4.2, occluded pixels in the warped views (T^{Lw}, T^{Rw}) are represented as black holes (occlusions are typically located to the side of foreground objects edges). The hole inpainting process fills in holes in textures (T^{Lw} and T^{Rw}) generated during warping as a result of pixel disocclusions. Namely, hole inpainting takes as input the left and right warped textures (T^{Lw} and T^{Rw}) and produces as output the corresponding warped, filled, textures (T_f^{Lw} and T_f^{Rw}) respectively. First, every disoccluded pixel at the position (r,c) in T^{Lw} (i.e, every $T^{Lw}(r,c) == 0$) is filled with the value of the co-located pixel at the position (r,c) in T^{Rw} if the latter is not occluded (*i.e* $T^{Rw}(r,c) \neq 0$). Then, the same procedure is followed to fill T^{Rw} using non-occluded pixels from T^{Lw}. However some pixels may still be occluded at this point when the new area in the target view is invisible in both left and right reference views. For example, Fig. 4.3 illustrates the case of a camera of a reference view warped from right to left.

To fill the remaining holes in T_f^{Lw} and T_f^{Rw}, we use for each remaining disoccluded pixel in T_f^{Lw} and T_f^{Rw} its non-occluded neighboring pixels. Therefore, we assume that the disoccluded pixels may appear only to the left or the right of foreground objects of the scene regarding the warping direction. Accordingly, we assign to each disoccluded pixel the median value of its five left or right neighboring valid pixels. Considering the remaining disoccluded pixels in T_f^{Lw} to be filled, each disoccluded $T^{Lw}(r,c)$ value is filled with the median value of the following five neighboring valid non-occluded pixels: $T_f^{Lw}(r-1,c), T_f^{Lw}(r-1,c+1)$, $T_f^{Lw}(r,c+1), T_f^{Lw}(r+1,c+1)$ and $T_{f.}^{Lw}(r+1,c)$. A pseudo-code of this algorithm applied on the left warped image T^{Lw} is in Alg. 1.

In a nutshell, to fill holes in the left warped view, we leverage the knowledge that the holes will appear on the right side of the foreground objects, and we want to fill the missing information using the background. Thus, we sweep the image row after row, from left to right, filling holes pixel by pixel using the median over five right not occluded neighbours. We use the same technique to fill holes in the right warped image but instead we sweep the image from right to left, since the holes appears on the left of the foreground objects, using the median over the left five not occluded neighbors. Combining the two steps together, our technique allows reasonable results with large holes.

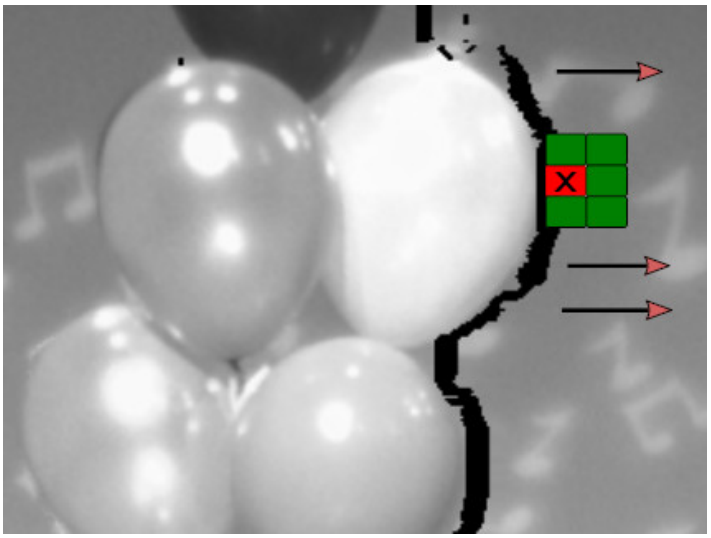


Fig. 4.3 Disoccluded holes representation in warped left image. The occluded pixel in red is resolved from right side neighboring pixels in green.

4.2.3 CNN-based View blending (CNN-VB)

The network comprises four convolutional layers where the first two are duplicated to operate on the two input images independently. The two layers are 64 and 32 filters, respectively, both sized $W \times W$, and project the two warped images over 32 feature maps for the left warped view and 32 for the right view. The third and fourth convolutional layers include 16 and 1 filters, respectively sized $W \times W$, and combine the 32 feature maps extracted from the left view and the 32 feature maps extracted from the right view into the desired intermediate view. For simplicity, here we assume that we deal with single-channel (i.e., grayscale) images.

Proper padding is employed wherever necessary to guarantee that the convolution operator does not reduce the size of the output feature map concerning the layer input image or feature map. Concerning the activation functions, all convolutional layers are followed by Rectified Linear Units (ReLu) except for the last, followed by a Hyperbolic Tangent (Tanh)

function. The network is designed to input and output images with approximately zero-mean, and pixel intensity bounded in the [-1, +1] interval. Not shown in the picture, we added a batch normalization layer after each convolutional layer to speed up the network convergence at training time.

The network takes two images as input and predicts additional information exploiting sub-pixel scale differences between input images. Namely, besides the enhancements related to super-resolution, the network also learns to correct object edges, deal with non-matching pixels (e.g., foreground in one warped image and background in another), and infer missing information from the surrounding areas.

Notice that the filter size W affects the maximum camera baseline distance that the network can handle. Therefore, we will experimentally find an appropriate W value for the considered test material in the following section.

4.2.4 Training procedure

The above network is trained end-to-end in a fully supervised way as follows. Training data consist in triplets of images $(T_f^{Lw}, T^C, T_f^{Rw})$ where T^C is the view to synthesize and T_f^{Lw} and T_f^{Rw} are the pre-warped, hole-filled reference views generated as above. From each triplet of images, we crop at a random position the three co-located 64×64 patches $(t_f^{lw}, t^c, t_f^{rw})$. Each patch is normalized, so pixel values lie in the $[-1,1]$ interval and have approximately zero mean.

Next, we randomly flip each patch as a form of data augmentation. In our experiments, such a method was shown very influential towards reducing the likelihood of overfitting on the training data. However, only a vertical flip could be applied due to the disparity between the left and right images. Therefore, a horizontal flip should be coupled with a Left-Right image switching. The initial values of the weight s of all convolutional layers but the output layer are drawn from a $\mathcal{N}(0, 0.01)$. Concerning the output layer, we applied a Xavier initialization [101]. For all layers, biases are initialized at zero. For each pair of reference patches (t_f^{lw}, t_f^{rw}) provided in input, the network learns to minimize the quadratic error between the network output t^v and the ground truth t^c. Formally, we minimize the loss function

$$L(w, t^v, t^c) = \frac{1}{n} \sum_{i=1}^{n} (t_i^v - t_i^c)^2, \tag{4.2}$$

where t_i^v and t_i^c are the i-th pixel of t^v and t^c, respectively.

We train our network following the standard practice of back-propagating the gradient of the above error function with respect to the learnable parameters w and updating the parameters using the Adam algorithm [102] for determining the actual update step-size. Our experiments revealed that a reasonable trade-off between performance and convergence time could be achieved using batches of 128 patches and a learning rate of 0.0001. The training

procedure ends after the above loss function measured over a validation set distinct from the training set stops decreasing.

4.3 Experimental results

In this section, after describing our experimental setup, we evaluate our method over several multi-view video sequences having moderate-to-large baselines.

4.3.1 Experimental setup

We experiment using the seven multiview sequences summarized in Tab. 4.1 and provided by Nagoya University [1], Poznan University and Technicolor. Such sequences account for various content types with natural or artificial light, complex or straightforward objects motion, and different video resolution. All sequences are captured using multiple cameras arranged in a linear array except for Painter captured using a 2D array. For each view of each sequence are made available also the corresponding depth map, either acquired or computed. All sequences are available in uncompressed YUV colorspace, and we consider only the luminance channel Y in our experiments. From each sequence, we select two reference views (T^L, T^R) and one target view T^C so that T^C is adjacent to both T^L and T^R. For the sole Painter sequence captured using a 2D camera array, we extract three different target views from three different camera rows, bringing the total number of experimental setups to 9.

Next, we pre-warp and disocclude T^L and T^R using the method described in the previous section. Then, from each sequence we extract at random 10k triplets of 64×64 co-located patches $(\widetilde{t}^l, t^c, \widetilde{t}^r)$ as described in the previous section. Each sequence is alternatively reserved for testing for a total of 10k testing patches. In contrast, the other eight sequences are used for training and validation purposes, for a total of 70k training patches and 10k validation patches. The results below referred to the network trained from scratch over the eight train sequences and tested on the left-out sequence. Our experimental testbed is implemented in python language for hole filling and patch extraction using the PyTorch for neural network deployment. All networks were trained and tested over an NVIDIA RTX2080 GPU.

[1] http://www.fujii.nuee.nagoya-u.ac.jp/multiview-data/

Sequence	Characteristics	Setup	Resolution	(T^L, T^C, T^R)	(T^L, T^R)
Balloons	Indoor, colorful	1D	1024x768	(1,3,5)	20 cm
Kendo	Indoor, white smoke	1D	1024x768	(2,4,6)	20 cm
Newspaper	Indoor, people	1D	1024x768	(2,4,6)	20 cm
PoznanStreet	Outdoor, nature	1D	1088x1920	(2,4,6)	55 cm
PoznanHall	Indoor, building hall	1D	1088x1920	(5,6,7)	27.5 cm
Lovebirds	Outdoor, nature	1D	1024x768	(4,6,8)	15 cm
Painter-1	Indoor, art studio	2D	2048x1088	(0,1,3)	14-21 cm
Painter-5	"	"	"	(4,5,6)	"
Painter-9	"	"	"	(8,9,10)	"

Table 4.1 The seven multiview video sequences used in our experiments (all sequences are 100 frames, average camera baseline is 24 cm).

4.3.2 Preliminary Experiments

As a preliminary experiment, we explore the performance of our proposed scheme as a function of the convolutional kernel size W, comparing against the case where the network is trained to blend the original, not warped images. While the original images boast better quality because they are not affected by the occlusions induced by view warping, the baseline between the views poses a challenge for the convolutional network in blending the views. For this experiment, we test the trained network only on the Kendo sequence (the network is trained on the other six sequences in Tab. 4.1 due to the time required to explore all the possible configurations. Tab. 4.2 shows that as the size of the filters increases, the performance of both methods improves. However, the unwarped case performs significantly worse for the same kernel size, showing the benefits of preliminarily warping the views before blending.

We also observed that as the filters size increases, the network is more prone to overfit the training data. It can be easily explained by looking at how the number of learnable parameters increases with the size of the filter. In the rest of this section, we will experiment with 9×9 filters as they have shown a reasonable trade-off between complexity in terms of learnable parameters and performance.

kernel size	**7x7**	**9x9**	**13x13**	**17x17**
CNN-VB	37.71	38.21	38.68	38.98
CNN-VB-unwarped	21.26	21.75	24.29	24.75
Parameters	6400	10496	21760	37120

Table 4.2 Synthesized view PSNR for proposed method and proposed method on original views, Kendo sequence.

4.3.3 Synthesized View Quality

The goal of view synthesis is to create high-quality virtual views. Therefore, we evaluated the generated images through both non-perceptual (PSNR) and perceptual (SSIM) quality metrics. As mentioned, we set $W = 9$ for our filters and performed the cross-validation over the the 6 sequences in Tab. 4.1. The results are reported in Tab. 4.3, where we show the PSNR and the SSIM of the synthesized view T^V concerning the ground truth T^C. As a first reference scheme, we consider the MPEG reference view synthesis software VVS. As a second reference, we consider the convolutional architecture for temporal super-resolution [12].

Our proposed method outperforms VVS for all sequences in terms of SSIM and all but one sequence in PSNR. For example, Fig. 4.4 shows a detail of the Kendo sequence, where our method outperforms VVS by 0.7 dB. The figure shows a significant improvement in image detail around the warrior mask.

Concerning PoznanStreet, VVS outperforms our method by almost 0.8 dB: Fig. 4.5 shows that errors in the blended image are localized around foreground objects. We point out that this sequence has a different baseline (55cm) than the other sequences (between 20 and 27.5 cm) used for training, and we hypothesize two explanations for our results. First, the more significant baseline of this sequence may require convolutional filters more extensive than the 9x9 filters we used. To verify this hypothesis, we retrained our network with larger convolutional kernels of size 13x13. With this setup, the PSNR of our proposed method improved from 36.17 to 36.9 dB, nearly equalling VVS (36.96 dB), albeit at the price of doubling the number of learnable parameters (see Tab. 4.2). Second, the blending network may not have learned to generalize over baseline distances that differ from those of the training images. Due to the limited availability of depth maps for the training sequences, we were unable to train our network on multiple baselines. This observation highlights the importance of training the network to generalize across different camera baselines.

Finally, reference scheme [12] is consistently outperformed by both VVS and the proposed method: we recall that this scheme was designed for temporal super-resolution; this is not meant to deal with the wide baseline distance of our multiview sequences.

	PSNR			SSIM	
Sequence	**[12]**	**VVS**	**CNN-VB**	**VVS**	**CNN-VB**
Balloons	30.39	**37.07**	36.92	0.922	**0.965**
Kendo	26.88	38.21	**38.98**	0.972	**0.976**
Newspaper	27.95	34.53	**35.08**	0.930	**0.950**
PoznanStreet	32.20	**36.96**	36.17	0.922	**0.932**
PoznanHall	34.08	37.26	**37.43**	0.921	**0.932**
Painter-1	29.80	37.86	**37.88**	0.948	**0.956**
Painter-5	32.17	38.04	**38.12**	0.945	**0.953**
Painter-9	30.53	36.36	**36.44**	0.930	**0.941**
Lovebirds	25.716	34.56	**34.7**	0.909	**0.933**

Table 4.3 Quality of the synthesized view for the proposed and reference methods VVS and depth-aware video frame interpolation [12].

Fig. 4.4 Detail of the synthesized view for the Kendo sequence. Left: ground truth; center: VVS; right: proposed method.

Fig. 4.5 Detail of the synthesized view for the PoznanStreet sequence. Left: ground truth; center: VVS; right: proposed method.

The CNN-VB study proves to us that we need larger filter sizes to handle such displacement with a relatively large focal length and a significant disparity between the actual view and the desired view. However, as shown previously, this solution requires doubling the number of parameters and complexity. One idea is to make our architecture deeper by increasing the number of convolutional layers while keeping the parameters low. In the following section, we detail the upgraded version of CNN-VB.

4.3.4 CNN-based view blending upgraded (CNN-VB+)

CNN-VB+ as illustrated in Fig. 4.6 is an improved version of CNN-VB that we developed later on. Since CNN-VB is initially a very primitive architecture that validates preliminary warping, the views to use a smaller filter, we assume that this architecture can improve in two aspects.

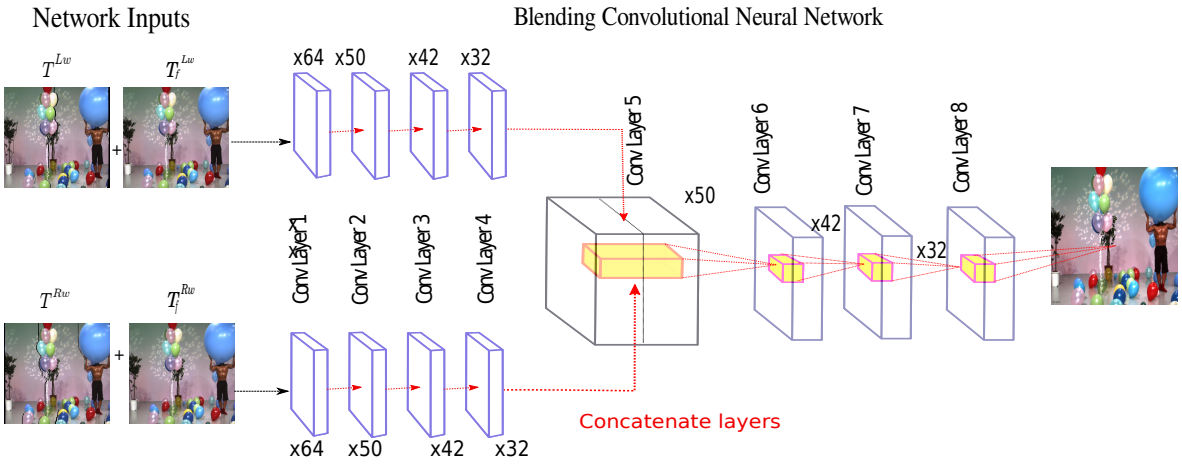


Fig. 4.6 The CNN-VB+ architecture as an improved version of CNN-VB

First, after warping and disoccluding the views, we compute two reliability maps that signal which pixels result from a disocclusion. The two maps are then provided in input to the ConvNet in charge of blending, which exploits the reliability maps as hints to improve the quality of the synthesized view. Second, we double the depth (8 vs.4 layers) of the ConvNet in charge of blending while keeping the parameters count constant. Each pair of warped texture and reliability maps is provided in input to a separate branch of four convolutional layers with 64, 50, 42, and 32 filters, respectively, all sized 5×5 rather than 9×9. The last four convolutional layers include 50, 42, 32, and 3 filters respectively sized 3×3 in place of 5×5. It can be shown that each feature produced in output by these four convolutional layers enjoys the same receptive field as in the case of the CNN-VB architecture. Therefore, this scheme enjoys a deeper convolutional pipeline to learn potentially better representations of the input views, leaving untouched complexity and receptive field size. Finally, CNN-VB+ takes in input and outputs RGB rather than grayscale images.

4.3.4.1 Results and discussion

Concerning CNN-VB and CNN-VB+, the latter improves over the former in all sequences, with gains in excess of 0.5 dB for *PoznanStreet* and *Lovebirds*. Similarly, CNN-VB+ outperforms MPEG VVS almost for all sequences, with a 0.8 dB gain for *Kendo*. We attribute such gains in part to the more profound convolutional architecture, introducing the reliability maps. Concerning SSIM, CNN-VB+ outperforms CNN-VB on average, albeit in some cases, CNN-VB scores better. Anyhow, a visual inspection of the synthesized view (Fig. 4.7) shows that even CNN-VB+ produces artifacts in the synthesized views, showing the intrinsic limits of this pooling-less architecture in view synthesis.

Sequence	PSNR [dB]		
	VVS	CNN-VB	**CNN-VB+**
Balloons	37.07	36.92	**37.18**
Kendo	38.21	38.98	**39.08**
Newspaper	34.53	35.08	**35.35**
PoznanStreet	**36.96**	36.17	36.81
PoznanHall	37.26	37.43	**37.50**
Painter-1	37.86	37.88	**38.01**
Painter-5	38.04	38.12	**38.18**
Painter-9	36.36	36.44	**36.51**
Lovebirds	34.56	34.70	**35.34**
Average $\pm$ Std	36.76 $\pm$1.38	36.85 $\pm$1.41	37.11$\pm$1.26

Table 4.4 Quality of the synthesized view for the proposed and reference methods in terms of PSNR

4.4 Conclusion

In this work, we explored the idea of preliminarily warping the reference views to the target position before a convolutional neural network eventually takes care of blending the warped views to the target position. Our experiments showed that our approach allows us to deal with more extensive baseline sequences with convolutional filters of reduced size and thus complexity, comparing favorably concerning state-of-the-art methods for view synthesis. Our research also showed the importance to train the convolutional network so that it can learn to generalize across images with different baseline distances. Later on, we improved our architecture by adding more convolutional layers while keeping the learnable parameters low and thus the complexity. We also add reliability maps that go along with the warped textures as inputs to the network. However, the network showed reduced performance on more

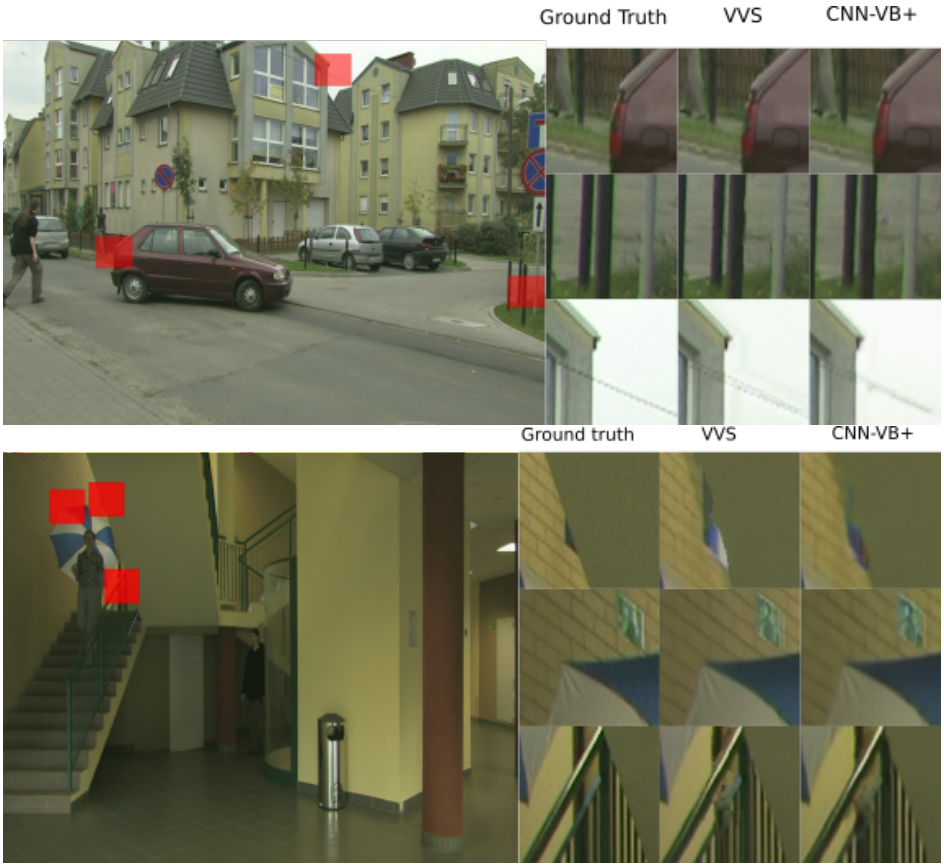


Fig. 4.7 Details from *PoznanStreet* (top) and *PoznanHall* (bottom) sequences. First column is ground truth, second column is VVS, and the last column is the proposed CNN-VB+.

complex cases, mainly due to the limited generalization capacity of such a simple architecture. Exploring CNN-VB and CNN-VB+ as simple architectures led us to understand the importance of more complex architectures that better understand input views and ensure a better generalization. In the following chapter, we detail a new approach for ConvNet-based Blender architecture.

Algorithm 1: Hole inpainting algorithm exemplified for view T^{Lw}

Input: (T^{Lw}, T^{Rw})
Output: Filled textures (T_f^{Lw}, T_f^{Rw})

$ListLeft \leftarrow$ the list of occluded pixels in I^{Lw};
$ListRight \leftarrow$ the list of occluded pixels in I^{Rw};
for (r,c) in $ListLeft$ **do**
  if if (r,c) is in ListLeft but not in ListRight
  then
    copy right to left, and remove from ListLeft
  else if (r,c) is in ListRight but not in ListLeft **then**
    copy left to right and remove from ListRight;
end
for (r,c) in $ListRight$ **do**
  if if (r,c) is in ListRight but not in ListLeft **then**
    copy left to right, and remove from ListRight
  else if (r,c) is in ListLeft but not in ListRight **then**
    copy left to right and remove from ListLeft;
end
for (r,c) in $ListLeft$ **do**
  if neighbor pixels are valid pixels not in ListLeft
  then
    $T_f^{Lw}(r,c) =$ Median[valid neighbor pixels in T_f^{Lw}]
  else
    continue
  end
end
for (r,c) in $ListRight$ **do**
  if neighbor pixels are valid pixels not in ListRight
  then
    $T_f^{Rw}(r,c) =$ Median[valid neighbor pixels in T_f^{Rw}]
  else
    continue
  end
end

Chapter 5

Hybrid Dual Stream Blender For Wide Baseline View Synthesis

This chapter proposes a residual encoder-decoder for image blending with a Siamese encoder to further keep the parameters count low. We also contribute a hole inpainting algorithm to fill the disocclusions in the warped views. After describing the context and the challenges of this method, we organize the rest of this chapter as follows:
Section 5.2 describes the proposed view synthesis method. Section 5.3 experimentally evaluates the performance of our method, including an ablation study. Finally, Section 5.4 discusses the learned lessons and discusses future research.

5.1 Context and challenges

In our previous work Chap. 4, we explored the idea of preliminary warping the reference views to the target position, then blending the warped views with the aid of a ConvNets. We showed that a plausible novel target view could be obtained with a simple architecture, which outperforms traditional algorithmic methods (blending and inpainting) in most cases. However, the network showed reduced performance on more complex cases, mainly due to the limited generalization capacity of such a simple architecture. The present work builds upon our previous research in Chap. 4 retaining the ideas of a hybrid algorithmic-learning scheme where reference views are preliminarily warped to the target position and using an inpainting method built around a median filter to handle occlusions. However, this work improves our previous approach under several aspects:

- We blend the warped views using a residual encoder-decoder architecture inspired by recent advances in image-to-image translation [83, 103]. Namely, we reformulate our problem as an image-to-image translation task, aiming to translate warped input real views into the target view.

- We provide the ConvNets in charge of blending the warped views, both the warped and the inpainted views, to allow the network to pick whatever is the best source for resolving each occluded pixel.
- Unlike image-to-image translation problems, we must deal with multiple reference views, which entails one encoder for each view. To keep low the parameter count, we introduce a *flip-convolve-flip* scheme that allows sharing parameters between encoders operating on symmetric inputs.
- We experiment using binary masks as an alternative to the inpainted views inputs to reduce the network complexity.

We experimentally evaluate our proposed approach over multiple wide baseline multiview video sequences comparing with purely algorithmic and purely learnable state-of-the-art approaches. Our experiments show that our architecture outperforms competitors in image quality for wide baseline sequences while striking a favorable balance between complexity and performance.

Our state-of-the-art analysis suggests that none of the existing methods is free from drawbacks in the form of artifacts in the synthesized image. With algorithmic-based approaches such as VVS, view blending and hole inpainting yield artifacts in the synthesized view. With learning-based methods, problems in generalization and consistency can be traced back to the end-to-end view warping and blending process. While blending warped reference views allows handling large baselines with small kernels, it does not solve the problem with the artifacts in the synthesized view. In our previous work [104], we took inspiration from convolutional architectures for image super-resolution [105]. Conversely, in this work, we take a different approach inspired by recent advances in Image-To-Image (I2I) translation [83,103]. I2I translation consists of mapping one image to another and tackles problems such as image colorization, super-resolution, and, to some extent, also view synthesis. Most I2I approaches rely on encoder-decoder architectures where the input image is first projected on a latent feature space by a convolutional encoder. Usually, the encoder relies on pooling layers or multiple-strided convolutional layers to reduce the spatial resolution of the feature maps. Next, such features may pass through a bottleneck layer that projects them over a feature space of (usually) lower dimensionality. Then, these features are projected back to the original pixel domain by a transposed convolutions decoder. The decoder usually employs transposed convolutions (or fractionally-strided convolutions) to recover the original resolution of the translated image. The transposed convolution is used to conduct optimal up-sampling, it also has learnable parameters. A further key challenge for our application is how to prevent occlusions in the warped images from generating artifacts in the synthesised view, a problem that in I2I architectures is usually not present. In the next section, we propose a convolutional architecture for view synthesis that takes inspiration from I2I methods

yet tackles challenges unique to view synthesis problems.

5.2 Hybrid dual stream blender (HDSB)

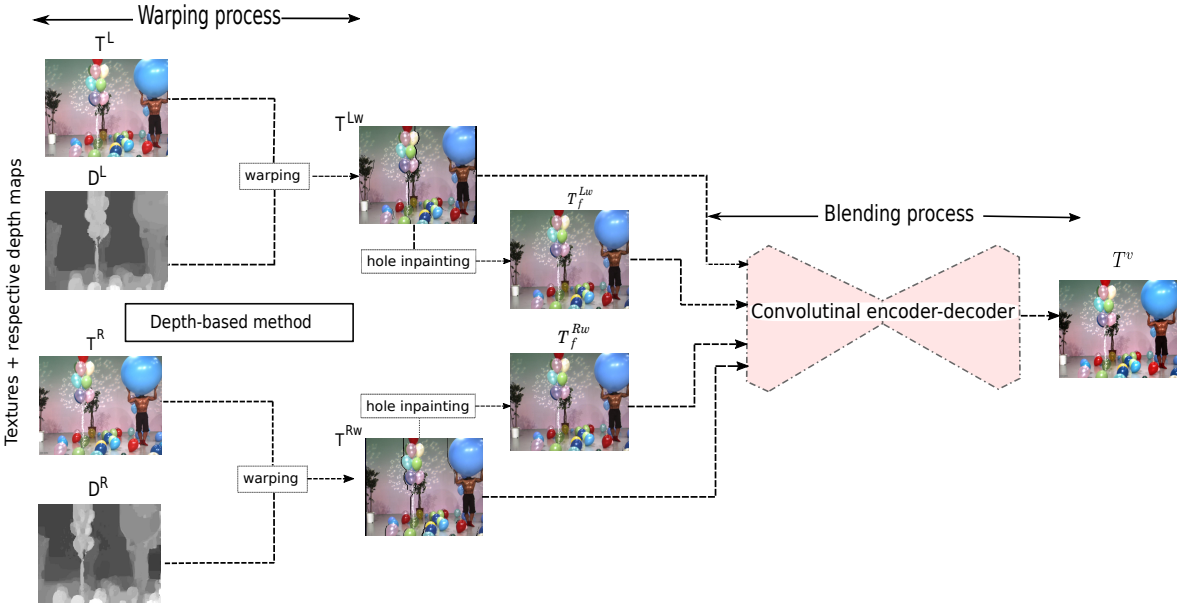


Fig. 5.1 The proposed hybrid pipeline for wide-baseline view synthesis: reference views are first warped to the target position, disocclusions are inpainted and eventually blended by a ConvNets.

Fig. 5.1 illustrates the proposed wide-baseline view synthesis pipeline, where given two left and right input reference views I^L and I^R are composed by textures T^L and T^R, and depths D^L and D^R respectively. We aim at synthesizing texture T^v of the target view that lies in the middle between the reference views. Towards this end, we propose a pipeline for view synthesis composed of a warping step followed by a hole filling step and a final blending step. First, the reference views are warped to the target position, producing warped references aligned with the target view to synthesize. Warping allows us to blend the warped images using a ConvNets with small kernels later on. Second, the hole inpainting consists in generating a filled texture (T_f^{Lw} and T_f^{Rw}) for each warped reference texture (T^{Lw} and T^{Rw}). Third and last, a ConvNets blends the left and right warped references (with filled textures T_f^{Lw} and T_f^{Rw}) to the target view T^v. The details of each step are detailed in the following.

5.2.1 Warping the reference views to the target position

The warping process consists in warping textures (T^{Lw}, T^{Rw}) to the target view position (T^v) with the aid of the depth maps. Namely, this step takes as input the two left and right

reference views T^L and T^R and produces as outputs two warped reference views T^{Lw} and T^{Rw} to the same intermediate novel target view position T^v. The textures are back-projected to the target position as follows. Let pixels t_r and t_v be the projections of a same real world point denoted by X, and with coordinates $(u_r, v_r, 1)$, $(u_v, v_v, 1)$ respectively. Lets us consider K_r, K_v and R_r, R_v the respectively 3x3 intrinsic camera parameters and the 3x3 rotation matrix for each camera. Then, t_v can then be expressed as

$$t_v = K_v R_v (K_r R_r)^{-1} (z t_r + K_r R_r C_r) - K_v R_v C_v, \tag{5.1}$$

where z is the depth value. In particular, the two pairs of texture and depth $I^L = \{T^L, D^L\}$ and $I^R = \{T^R, D^R\}$ are up-sampled to half-pixel or quarter-pixel accuracy, in which the warping and interpolation steps are carried out. In practical implementations of the above method, one has to decide the value to assign to pixels that are occluded in the reference views but visible in the target (disoccluded pixels). In practice, those pixels are often arbitrarily assigned a zero value, which entails a few drawbacks. A zero-valued pixel is ambiguous as it could represent either a non-occluded dark pixel or a disocclusion. While the ConvNets in charge of blending the warped views may learn this, this would make the learning problem more challenging. One possible solution is to provide an occlusion map for each warped texture as input to the network, allowing a hypothesis on each pixel claiming whether yes or not it should be given a value, An even better solution (at least, according to our experimental results) is to provide a different version of the warped image where the disoccluded pixels are filled as much as possible with relevant information. We take this latter approach, and we detail it in the following section.

5.2.2 Hole inpainting

The hole inpainting process and algorithm is the same used in previous work Chap. 4 Sec. 4.2.2. Therefore, due to the warping process of the reference views to the target position, some visible areas in the target view are invisible in the reference view, and thus, they appear as black holes in the warped views where each disoccluded pixel at a given position (r, c), $T^{Lw}(r, c)$ is set to zero.

We cope with occlusions using a two-steps hole filling process. As a first step, we copy non-occluded pixels in T^{Lw} to fill the black pixels in T^{Rw}, and vice versa. However, pixel copy is not sufficient to resolve all occluded pixels, thus as a second step we resolve the residual holes by applying the algorithm described in Alg. 1. An example of the filled textures (T_f^{Lw} or T_f^{Rw}) is represented in Fig. 5.2.

The resulting filled textures (T_f^{Lw}, T_f^{Rw}) go along within the warped textures (T^{Lw}, T^{Rw}) as inputs to the ConvNets responsible for the blending step described in the next section. Finally, an approximation of the proposed method consists in using binary masks, in place

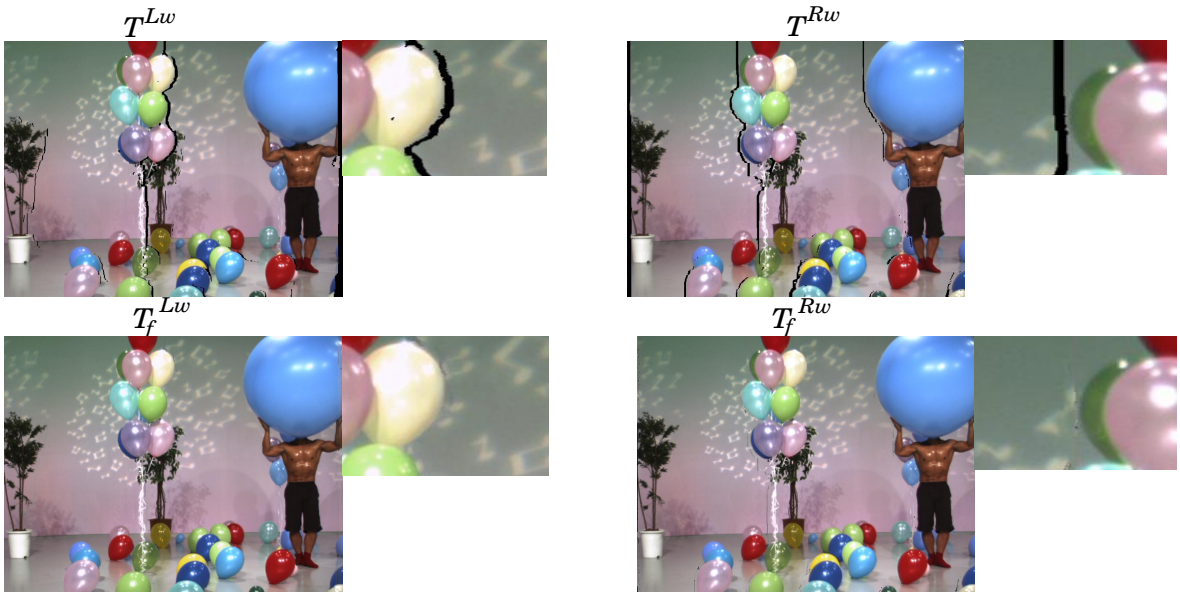


Fig. 5.2 The two left and right warped input textures T^{Lw} and T^{Rw} and their corresponding filled textures T_f^{Lw} and T_f^{Rw} with details *Baloons* sequence.

of the filled textures, increasing speed and trading off visual quality. Eventually, since generating filled warped textures is time and memory-consuming, we propose a lower complexity alternative based on occlusion maps. First, we generate a binary mask for each reference warped texture by labeling occluded pixels as 1, 0 otherwise. Fig 5.3 illustrates how occluded pixels in the reference image correspond to the visual black holes valued as 0. Therefore, as a low-complexity alternative, we propose to use the two reference warped textures and their corresponding binary masks (M_b^L, M_b^R) as inputs to our architecture, instead of the filled textures. In Sec. 5.3.3.2 we will use such method as a benchmark method to evaluate our proposed hole inpainting algorithm.

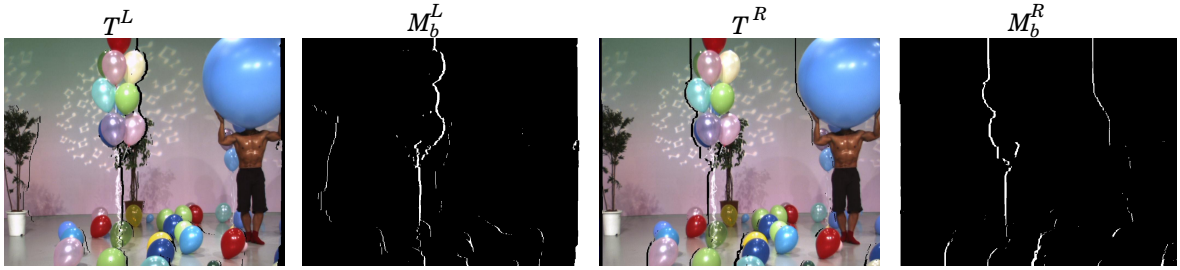


Fig. 5.3 The two left and right warped input textures and their corresponding binary masks *Baloons* sequence.

5.2.3 ConvNet-based blending

As a third and final step, the warped textures (T^{Lw}, T^{Rw}) and the relative filled counterparts (T_f^{Lw}, T_f^{Rw}) are blended into a novel viewpoint using a ConvNets. We describe here the architecture of the ConvNets and the relative training procedure.

5.2.3.1 ConvNet architecture

Recently, image-to-image translation architectures [83, 106] have shown that it is possible to map an image from a first visual domain to another image from a different domain. Indeed, we consider that the image-to-image problem is refined to view blending, so it may be reasonable to take inspiration from these architectures. However, there are some remarkable differences between image-to-image translation and the view blending problem considered here. First, we need to deal with a total of four inputs (the two warped reference textures and the relative filled textures) rather than with a single input image. Second, each pair of inputs (T^{Lw}, T_f^{Lw}) and (T^{Rw}, T_f^{Rw}) (cf. Fig 5.1), is characterized by a specific type of artifacts to be recovered. As one reference view is warped from the left side and the other from the right side, they do not share the same disocclusion problems. Therefore, our network should learn how to exploit such inputs together to obtain one synthesized view using and adapted architecture originally conceived for image-to-image translation task. Thus, we propose a specific encoder-decoder architecture that combines its four inputs and generates one output.

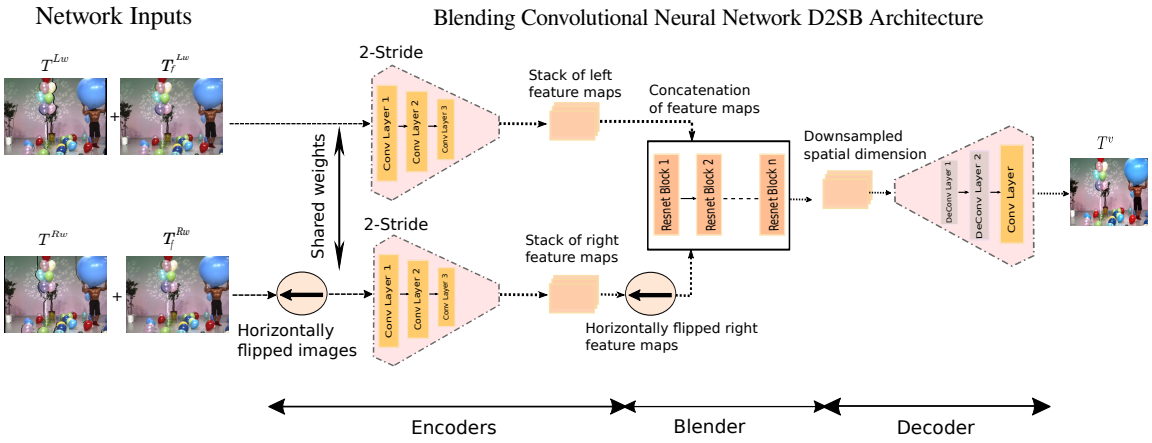


Fig. 5.4 Hybrid Dual Stream Blender Architecture sharing parameters in the encoder stage.

Our architecture is illustrated in Fig. 5.4 and is composed of three parts, the two *encoders*, the *blender*, and the *decoder*.

Encoders

Our ConvNets includes a pair of identical *encoders*, one for the left and one for the right view. We assume that all textures are 3-channels color images, e.g, in RGB or YUV format.

The left encoder (the top encoder in Fig. 5.4) takes in input the left view composed of textures (T^{Lw}, T_f^{Lw}); the right encoder(the bottom encoder in Fig. 5.4) takes in input the right view made of textures (T_f^{Rw}, T^{Rw}). The role of each encoder is to project the views to a spatially-subsampled latent feature space. To this end, each encoder includes three convolutional layers with 64, 128 and 256 filters respectively of size 7x7 for the first layer and 3x3 for the last two, all followed by ReLu activations. Filters in the convolutional layers have 2-units stride, so that the feature maps in output of each layer are half the size of the feature maps in input to the layer. Eventually, the feature maps output by the encoder are $\frac{1}{8} - th$ the size of the input textures. We found by extensive experimentation that such encoder topology achieves the best tradeoff between semantic depth and spatial resolution of the output feature maps. Next, we propose to reduce the number of learnable parameters by sharing weights among encoders [107]. However, just sharing the same weights among the two encoders would be suboptimal (as we experimentally verify in Sec. 5.3.3.4) since the left and right views do not share identical disocclusion artifacts. Indeed, occluded pixels in the left warped views will occur on the right side of objects, whereas in the right warped views occlusions will occur on the left side. That is, occlusion artifacts will show on the two opposite side of the objects in left and right views. Therefore, we propose a *flip-convolve-flip* approach that allows sharing parameters among encoders. First, we horizontally flip (mirror) the right view so that occluded pixels show on the right side of objects, as in the left view and as illustrated in Fig. 5.5. Then, the feature maps generated from the right encoder are horizontally flipped a second time. This produces feature maps that are semantically similar to those generated by the left encoder and can be easily merged by the bottleneck block later on while sharing parameters among encoders. While an extension of this scheme to the case of vertically arranged cameras by applying a vertical mirroring rather than horizontal may be theoretically envisaged, its discussion is out of the scope of this work which deals with arrays of 1-D horizontally arranged cameras. In Fig. 5.4, the mirroring operations are denoted by circled arrows.

Left warped texture | Right warped texture | Mirrored right warped texture

Fig. 5.5 Occlusions in the left view appear to the right of objects, occlusions in the right view to the left. When the right view is flipped, occlusions appear on the right of objects, as for the left view (*Newspaper* sequence).

Blender

The blender includes 6 residual blocks with 512 filters each and blends the feature maps extracted from the left and right views into a set of feature maps that holds a suitable representation of the desired target view. Our experiments showed that residual blocks are better suited than convolutional layers for the task of blending the downsampled feature maps into the target view. The blender block is our original answer to the problem of blending features in a latent, spatially subsampled, space originated from two different views of the same scene. By comparison, other image-to-image translation architectures deal with monoscopic images only, so they do not need to address this additional problem. In CNN-VB and CNN-VB+ 4, features extracted from the input views did not undergo any spatial downsampling and they were only concatenated to synthesize the target view. We will experimentally show by ablation study the advantages of blending the features in a spatially subsampled feature space.

Decoder

The decoder finally synthesizes the virtual view at the target position exploiting the downsampled feature maps produced by the blender. The decoder includes 3 transposed convolutional layers with 256, 128 and 3 filters per layer of size 3x3 for the first two layers and 7x7 for the last layer. The decoder upsamples the low resolution feature maps produced by the blender component to a higher resolution. The first two layers are followed by ReLu activation function, while the output layer is followed by a Tanh. We also stack a batch normalization layer after each convolutional layer as in ResNet blocks, as our experiments showed it speeds up the training process. Overall, our network produces in output a three-channels view where each view is expected to have approximately zero-mean and the pixel intensity is bounded in the $[-1,+1]$ interval by the output layer nonlinearity.

5.2.3.2 Training procedure

The network is trained in a fully supervised way on quintuplets of patches $(\widetilde{t}^{lw},\widetilde{t}^{lw}_f,t^c,\widetilde{t}^{rw}_f,\widetilde{t}^{rw})$ extracted respectively from textures (T^{Lw}, T^{Lw}_f, T^C, T^{Rw}, T^{Rw}_f), where and T^C is the ground-truth image to synthesize.

The training process for the proposed scheme may be hindered by the limited availability of suitable data. In facts, multiview plus depth video sequences are not easily produced or available, and most of our sequences are taken from the MPEG test material. However, we avoided the use of computer-generated (CG) video sequences. Typically, the characteristics of CG data are relevantly different from natural content. In CG data we have perfect depth-maps but also, depending on the rendering techniques, one can typically achieve a somewhat limited complexity of textures, noise levels are much lower than natural videos, and

some phenomena are more difficult and more computationally intensive to be rendered (*e.g.*, non-Lambertian surfaces, sub-surface scattering, *etc*.). In short, relying only on computer-generated data would not improve the training process so much if the methods are then to be used on natural data.

We randomly apply a vertical flip on each quintuplet as a form of data augmentation, to increase the diversity in our training samples. Our experiments showed that, due to the limited availability of suitable video contents, such augmentation method is fundamental to avoid overfitting on the training data. In order to keep the geometrical relationship between left, center and right patches, only a vertical flip is applied in this case. Before being provided in input to the network, patches are normalized so that the average per-channel pixel intensity has zero mean and unitary deviation.

Concerning the loss function to minimize at training time, we alternatively experiment with two options. The first function measures the distortion in the pixel space over a ground truth, while the latter aims instead at assessing image reconstruction quality as *perceived* by the user.

Pixel-based reconstruction loss.

For each pair of reference patches $(\widetilde{t}^{lw},\widetilde{t}^{rw})$ provided in input, the network is trained to minimize the quadratic error between the network output t^v and the ground truth t^c. That is, at training time we minimize the loss function

$$L(w,t^v,t^c) = \frac{1}{n}\sum_{i=1}^{n}(t_i^v - t_i^c)^2, \tag{5.2}$$

where t_i^v and t_i^c are the i-th pixel of t^v and t^c, respectively. We train our network by back-propagating the gradient of the above error function and the network parameters are updated using the Adam algorithm [102].

Perceptual reconstruction loss.

Alternatively, we experiment with a perceptual loss function [108] to observe the *perceived* quality of the synthesized view. Perceptual loss relies on a feature extractor usually trained for image classification to compare two images based on their high-level features representations. Style transfer [83] experiments show that the perceptual loss training may achieves visually more pleasant images than per-pixel loss functions. Johnson *et al.* [109] first proposed the use of perceptual loss of image transformation tasks using a VGG16 trained on ImageNet [108]. The feature reconstruction loss is defined as

$$l_{feat}^{\phi}(\widetilde{y},y) = \frac{1}{CHW}\left\|\phi(\widetilde{y}) - \phi(y)\right\|^2, \tag{5.3}$$

where l_{feat}^{ϕ} is the feature reconstruction loss from one layer of loss network ϕ of the content image $\widetilde{y}$ and content representation of the output image y. C is the number of filter in the input image, H and W are the height and the width of the input image. $\phi(\widetilde{y})$ and $\phi(y)$ are the feature representation of the content of the target image and the feature representation of the output of the target image respectively. Our image transformation network is thus trained using stochastic gradient descent to get weights that minimize the total loss, which is a weighted product of the feature reconstruction loss. The training procedure ends after the perceptual loss function measured over a validation set distinct from the training set stop decreasing.

5.3 Experiments and Results

In this section, we quantitatively and qualitatively evaluate our method HDSB in a comparative way and we perform ablation studies to validate each of our design choices.

5.3.1 Experimental setup

Fig. 5.6 Example of large holes (in green) in the *Technicolor painter* sequence with a baseline $= 21cm$.

We experiment with well-known multi-view sequences commonly used in MPEG experiments as defined in the MPEG CTCs and detailed in Tab. 4.1 (views and depth maps are in uncompressed YUV format). Such sequences account for a wide range of content types with natural or artificial light, simple or complex objects motion, and different resolution. All sequences are captured with a linear 1D camera array, i.e. cameras axes are parallel, non-convergent. The inter-cameras distance is up to 55cm (e.g. *poznanstreet*) cf. Tab. 4.1 with a moderately long focal length (23mm on average) which makes the angle of view narrower and the overlap in the field of view smaller. Therefore, we notice large holes in the warped views even in our smallest baseline sequences, like the *"painter"* sequence, as illustrated in Fig. 5.6. The *Painter* sequence is captured with a 2D camera array, so we extract three linear,

non-overlapping, camera setups of this scene for a total of 9 sequences. From each sequence we extract 3 neighbor views from the first 100 frames: the left and right views are used as references (T^L, T^R) and the central view is used as target view (*i.e* ground-truth) T^C. For each sequence, we preliminary warp T^L and T^R to obtain T^{Lw} and T^{Rw} and then we generate the corresponding filled textures T_f^{Lw} and T_f^{Rw} using the methods described in the previous Sec. 4.2.1. Then, from each sequence we randomly extract 10k quintuplets of co-located patches $(\widetilde{t}^{lw}, \widetilde{t}_f^{lw}, t^c, \widetilde{t}_f^{rw}, \widetilde{t}^{rw})$. Each sequence is alternatively reserved for testing for a total of 10k testing patches. The other 8 sequences are used for training and validation purposes, for a total of 70k training patches and 10k validation patches. Such approach guarantees that the test sequence is always left out from the training set, i.e. there is no cross-contamination between train and test sets. All patches used for training are 64×64 as our experiments revealed it allows a favorable tradeoff between patch size and number of non-overlapping patches that can be extracted from the available training video sequences. Patches have fifty-fifty chance to be vertically flipped forming data augmentation. Our experiments reveal that a reasonable trade-off between performance and convergence time can be achieved using batches of 128 patches, and a learning rate of 0.0001, leading to the convergence of our learning algorithm after 100 epochs. In all our experiments, including losses experiments, we use the same hyper-parameters showing the best results. Concerning the parameters optimization algorithm, we rely on Adam, with weight decay $= 0$ and betas $= (0.9, 0.999)$. Our method is implemented in Pytorch and all the experiments are performed on a server with an NVIDIA RTX2080GPU. Finally, in all our experiments we measure the quality of the synthesized view both using the PSNR and the SSIM metrics computed over the Y (luma) channel.

5.3.2 Comparison with prior works

In this subsection we compare our proposed view synthesis method with a number of references and provide a quantitative and qualitative analysis of the synthesized view quality.

5.3.2.1 Reference schemes

Many of the state-of-the-art methods listed in Chap. 3 are learnable end-to-end architectures in principle comparable with ours. However, while some require in input more than two reference views, others deal only with small baseline views [110], [4]. The pipeline proposed in [26] is comparable to our, however it relies on four warped reference textures and a global mesh of the scene as input to their blending/inpainting network and the source code to reproduce their method is not available. On the other hand, the video frame interpolation method [12] when applied to our setup did yield very weak performances, we hypothesize because designed for different purposes purposes. For the above reasons, we compare with

the following reference schemes:

CNN-VB is the method described in 4, It shares with the approach proposed here the preliminary algorithmic warping followed by a learnable blending. This architecture notably includes no pooling layers nor other feature downsampling methods and, albeit very primitive, validated the idea of preliminary warping the views to use smaller filters. For these reasons, we keep it as a very baseline reference.

CNN-VB+ It is an improved version of CNN-VB as illustrated in Fig. 4.6 described in Chap. 4. This scheme improves over CNN-VB in two aspects: 1) the *reliability maps* and 2) the number of the convolutional layers). Therefore, this scheme enjoys a deeper convolutional pipeline to learn potentially better representations of the input views, leaving untouched complexity and receptive field size. Finally, CNN-VB+ takes in input and outputs RGB rather than grayscale images, so it is to compare with our proposed HDSB.

VVS, the MPEG-I view synthesizer reference software described in Sec. 3.1.3. This method is not learning-based and is completely based on an algorithmic approach. VVS and HDSB share the basic warping scheme, so VVS is a proper reference to assess the benefits of a learning-based blending stage over an algorithmic based approach.

Synsin, [4] is a purely learnable end-to-end view synthesis method. Synsin allows for synthesizing novel target views of a scene given a single image only, using generative adversarial networks (GAN) techniques and a new differentiable point cloud renderer. In our experiments, it is refined on our training set following the same procedure described for the other methods to allow for a fair comparison.

5.3.2.2 Results and discussion

In Tab. 5.1 and 5.2, we compare our proposed method HDSB with the references above. It is clear that Synsin shows weak performances in all sequences, in terms of visual quality and objective quality (cf. Tab. 5.1,Tab. 5.2 and Fig. 5.7). The results are linked to the use of only one reference image unwarped as input to the network, to generate a high-resolution novel view located at a long distance from the reference view. A possible explanation to this result could be that, this method is effective on small baseline cases and lower image resolution. Concerning CNN-VB and CNN-VB+, the latter improves over the former in all sequences, with gains in excess of 0.5 dB for *PoznanStreet* and *Lovebirds*. Similarly, CNN-VB+ outperforms MPEG VVS almost for all sequences, with a 0.8 dB gain for *Kendo*. We attribute such gains in part to the deeper convolutional architecture, in part to the introduction of the reliability maps. Concerning SSIM, CNN-VB+ outperforms CNN-VB on the average, albeit in some cases CNN-VB scores better. Anyway, a visual inspection of the synthesized view (Fig. 5.7) shows that even CNN-VB+ produces artifacts in the synthesized views, showing the intrinsic limits of this pooling-less architecture in view synthesis. In the following, we compare HDSB mainly against CNN-VB+, which is the best reference so far.

Sequence	PSNR [dB]				
	Synsin [4]	VVS [10]	CNN-VB [104]	CNN-VB+	**proposed HDSB**
Balloons	20.84	37.07	36.92	37.18	**37.59**
Kendo	21.87	38.21	38.98	39.08	**39.19**
Newspaper	19.28	34.53	35.08	35.35	**35.65**
PoznanStreet	18.16	36.96	36.17	36.81	**37.44**
PoznanHall	18.65	37.26	37.43	37.50	**38.30**
Painter-1	20.45	37.86	37.88	38.01	**38.21**
Painter-5	20.32	38.04	38.12	38.18	**38.87**
Painter-9	20.17	36.36	36.44	36.51	**38.01**
Lovebirds	19.89	34.56	34.70	35.34	**35.54**
Average $\pm$ Std	19.9589 $\pm$1.13	36.76 $\pm$1.38	36.85 $\pm$1.41	37.11$\pm$1.26	**37.64**$\pm$1.28

Table 5.1 Quality of the synthesized view for the proposed and reference methods in terms of PSNR

Tab. 5.1 shows that HDSB significantly outperforms all references by a significant margin (over 0.5 dB on the average). HDSB scores a top gain of 0.8 dB over CNN-VB+ for the *PoznanHall* sequence and improves by 0.6 dB for *PoznanStreet*. SSIM results show similar trends, with HDSB consistently outperforming every reference. We hypothesize that such gains are mainly due to the downsampling and upsampling of the feature maps performed by the encoder and the decoder respectively. In addition, we hypothesize that residual blocks in the bottleneck may have also a role in such gain. Concerning the visual assessment, Fig. 5.7 illustrates how HDSB improves over the three references. For *PoznanStreet*, for example, HDSB preserves the hanging lines on the image background, that are otherwise lost by the reference methods. Similarly, the shape of the black pole in the foreground looks much more like the ground truth as synthesized by HDSB. For *PoznanHall*, our method remarkably approaches the desired outcome in the reconstruction of the stairs and the handrail, or the clarity of the exit green plate on the wall.

Concerning computational complexity, inference and training times are as follows. The inference time of CNN-VB+ network is around few seconds ($2-3$ sec) per frame, and around 8 hours for the network to converge in the training process with the Mean Squared Error (MSE) loss over 100 epochs. The inference time of our network HDSB is on average 4.32s/frame, and the required training time the network to converge is 12 hours using MSE loss on 100 epochs. For both methods, training time drops by 3 times using the perceptual loss. Finally, Synsin inference time lies between 2 and 10 seconds per frame depending on image resolution, while training required 3 days to converge over 350 epochs. About the purely algorithmic VVS, the time to synthesize one frame is around 100 sec, where 83% of this time is due to inpainting/blending.

Sequence	SSIM				
	Synsin [4]	VVS [10]	CNN-VB [104]	CNN-VB+	**proposed HDSB**
Balloons	0.752	0.922	0.965	0.964	**0.978**
Kendo	0.826	0.972	0.976	0.981	**0.992**
Newspaper	0.875	0.930	0.950	0.951	**0.967**
PoznanStreet	0.728	0.922	0.932	0.956	**0.968**
PoznanHall	0.745	0.921	0.932	0.961	**0.966**
Painter-1	0.841	0.948	0.956	0.955	**0.970**
Painter-5	0.829	0.945	0.953	0.948	**0.961**
Painter-9	0.813	0.930	0.941	0.937	**0.952**
Lovebirds	0.789	0.909	0.933	0.951	**0.973**
Average $\pm$ Std	0.7998$\pm$0.049	0.933$\pm$0.02	0.949$\pm$1.16	0.956$\pm$0.12	**0.969**$\pm$0.01

Table 5.2 Quality of the synthesized view for the proposed and reference methods in terms of SSIM

5.3.3 Ablation studies

In this subsection, we alternatively ablate one element from our HDSB architecture and we assess the effect on the synthesized view quality.

5.3.3.1 Encoder-decoder architecture

As a first ablation study, we explore the advantages of the encoder-decoder architecture with feature map downsampling implemented by HDSB. Namely, we modify HDSB avoiding to downsample the feature maps by reducing the stride of the filters in the convolutional layers to 1 pixel (we refer to this architecture as "Wo/EnDe"). In other words, Wo/EnDe is such that all the feature maps produced by the hidden convolutional layers have the same size as the input and output images. This architecture is composed of three convolutional layers, operating independently on the left and the right views with 64 filters each and kernel size 7×7, 3×3, and 3×3 respectively. The bottleneck is composed of 8 residual blocks, and the last convolutional layer is composed of 3 filters with 7×7 kernel size. The only difference with respect to HDSB is the removal of the down-sampling. Indeed we preserve the same number of convolutional layers, the residual blocks, and the number of the learnable parameters to study the impact of an encoder-decoder architecture on the training/testing processes.

We train and test this architecture from scratch according to the same procedure used for HDSB. However, due to the lack of downsampling, this architecture has a larger memory footprint which forced us to reduce the batch size from 128 to 80 samples. Preliminary experiments show that such architectures slows down the training process and increases the convergence time.

In Tab. 5.3, the experiments show how adopting an architecture without an encoder-

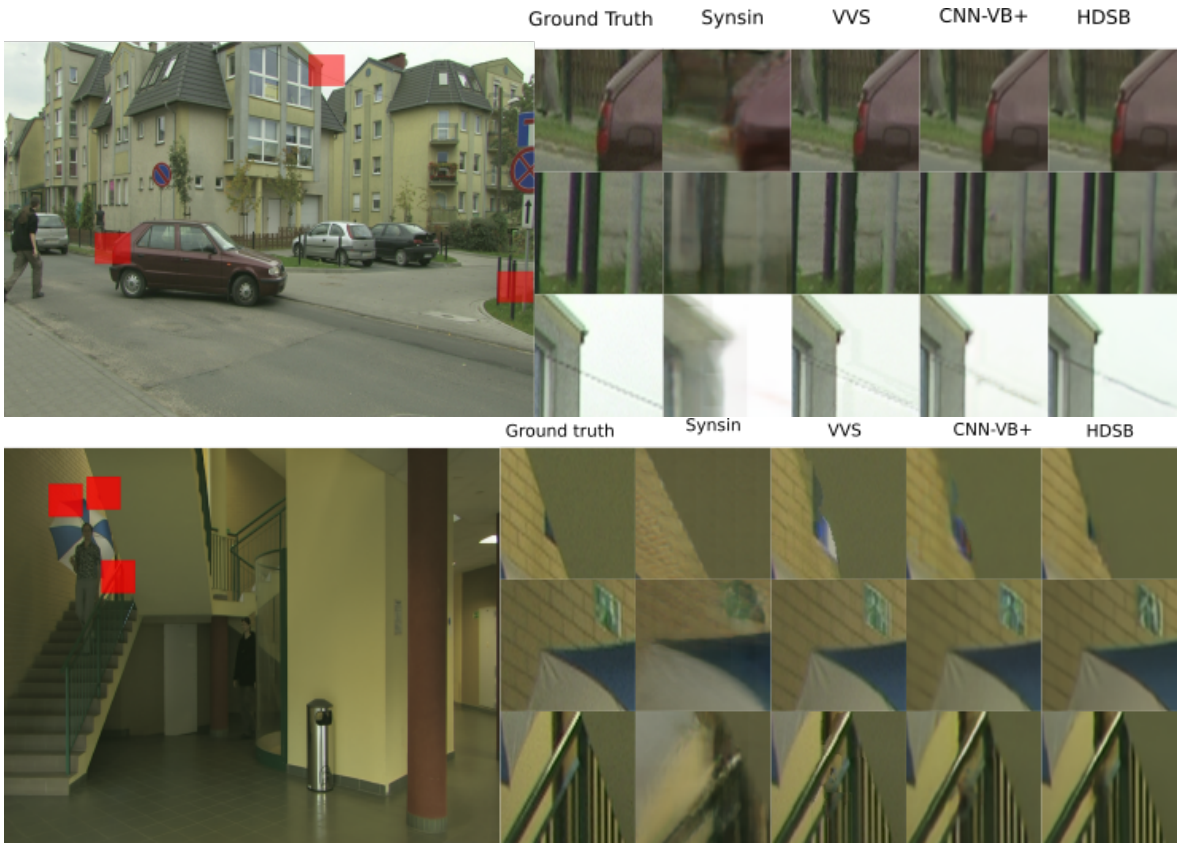


Fig. 5.7 Details from *PoznanStreet* (top) and *PoznanHall* (bottom) sequences. First column is ground truth, second column is Synsin, third column is VVS, fourth column is CNN-VB+, and the last column is the proposed HDSB.

decoder reduces in average the PSNR and the SSIM by almost 0.5 dB and 0.023 respectively on our test sequences. We illustrated the results in Fig. 5.8 for *Balloons* and *Newspaper* respectively. We notice that the architecture without encoder-decoder (refered to in Tab. 5.3 as Wo/EnDe) do not generalize well, and tends to overfit on the training data.

Indeed, HDSB our method focuses during training on fewer number of activation points to reduce redundancy in feature maps. It also yields the network output to be more tolerant for small translational changes in input images, which means that an encoder-decoder architecture can tolerate equivariances in input images produced due to the warping process. Indeed the two reference views are warped to the same target position, however they do not originally share the same lighting and angles conditions.

5.3.3.2 Hole inpainting experiments

We also elaborate on the effect of using different additional inputs to our network rather than the filled textures with the warped images. The filled textures T_f^{Lw} and T_f^{Rw}, were introduced in our proposed approach to strengthen the inpainting task in the network, and thus by better

Fig. 5.8 Ablation of the feature map downsampling: *Balloons* (top) and *Newspaper* (bottom). First column is the ground truth, second column is the architecture without encoder-decoder, and last column is our proposed architecture.

filling the occluded pixels in the output texture.

We notice that using filled textures yields to better PSNR performances than using binary masks on all our test sequences in table 5.4. Whereas, for the SSIM we did not notice any distinction. However, the improvements achieved in terms of PSNR are not easily visible with the naked eye, and thus the visual quality difference is indistinguishable between the two methods on our test datasets. Notably, we significantly raise the computational speed and we lower the memory consumption by using binary masks. Therefore, for the sake of simplicity, we use the binary masks as inputs to our network in the following experiments.

5.3.3.3 Loss functions experiments

In this section we consider the effect of changing the loss function used to train our network. All the results presented previously in this paper come from our neural network trained using a per-pixel loss function, the MSE. We re-train the proposed architecture on the same

Sequence	PSNR[dB]		SSIM	
	Wo/EnDe	**HDSB**	Wo/EnDe	**HDSB**
Balloons	36.90	**37.59**	0.943	**0.972**
Kendo	38.71	**39.19**	0.966	**0.985**
Newspaper	35.25	**35.65**	0.942	**0.962**
PoznanStreet	36.76	**37.44**	0.947	**0.965**
PoznanHall	37.30	**38.30**	0.960	**0.965**
Painter-1	38.04	**38.21**	0.958	**0.964**
Painter-5	38.39	**38.87**	0.944	**0.958**
Painter-9	37.58	**38.01**	0.933	**0.949**
Lovebirds	35.11	**35.54**	0.896	**0.969**
Average $\pm$ Std	37.12$\pm$1.27	**37.56**$\pm$1.35	0.943$\pm$0.02	**0.966**$\pm$0.01

Table 5.3 Quality of the synthesized view with our encoder-decoder architecture HDSB and without encoder-decoder architecture

Sequence	PSNR[dB]	
	HDSB	Wo binary mask
Balloons	37.24	**37.59**
Kendo	39.12	**39.19**
Newspaper	35.53	**35.65**
PoznanStreet	37.22	**37.44**
PoznanHall	38.25	**38.30**
Painter-1	38.09	**38.21**
Painter-5	38.85	**38.87**
Painter-9	37.95	**38.01**
Lovebirds	35.41	**35.54**
Average $\pm$ std	37.56$\pm$1.35	**37.63**$\pm$1.36

Table 5.4 Quality of the synthesized view using the filled textures and the binary masks.

Sequence	SSIM for HDSB		PSNR for HDSB	
	w/per-pixel loss	w/perceptual loss	w/per-pixel loss	w/perceptual loss
Balloons	0.972	**0.980**	**37.24**	35.84
Kendo	0.985	**0.989**	**39.12**	37.16
Newspaper	0.962	**0.968**	**35.53**	33.35
PoznanStreet	0.965	**0.966**	**37.22**	34.27
PoznanHall	0.965	**0.968**	**38.25**	36.41
Painter-1	0.964	**0.965**	**38.09**	37.03
Painter-5	0.958	**0.962**	**38.85**	37.18
Painter-9	0.949	**0.957**	**37.95**	36.22
Lovebirds	0.969	**0.971**	**35.41**	32.94
Average $\pm$ std	0.966$\pm$0.01	**0.969**$\pm$0.01	**37.56**$\pm$1.35	35.6$\pm$1.65

Table 5.5 Quality of the synthesized view using two different loss functions during the training process.

training datasets, but we use instead the perceptual loss function, detailed in Sec. 5.2.3.2, essential for the training convergence. The experiment shows that high-quality visual images are generated when the perceptual loss is minimized. As well as it increases the SSIM on all the sequences, cf. Tab. 5.5, over 0.003 in average. With the perceptual loss function the computation of the loss between the output and the desired image is based on the image content and style rather than on the individual pixel values, and thus we expect that the PSNR will decrease. However, we notice that when the network is optimized towards another metric the PSNR decreases of 2 dB as shown in Tab. 5.5. Finally, in terms of complexity, the training with perceptual loss is three times faster than with MSE. In all our experiments in this work, we used the MSE loss function to evaluate and compare our results using both perceptual SSIM and non perceptual PSNR quality metrics. In the end, we observed that perceptual loss leads to visually pleasant synthesized images, while somehow reducing the training time. On the other hand, MSE is more appropriate when objectively benchmarking methods in PSNR terms.

5.3.3.4 Effect of the encoder architecture

Unlike image-to-image mapping monoscopic architectures that feature just one encoder, our neural architecture features one one encoder for the left view and one for the right view. Such design choice is motivated by the observation that disocclusion artifacts in the two warped views lie on opposite side of the objects. For the same reason, we proposed the *flip-convolve-flip* approach to be able to share parameters among encoders. We now experiment with two different encoder typologies as follows.

First, we consider a HDSB variant where we drop the two *convolve-flip-convolve* encoders with shared parameters in favor of a single encoder. This scheme leaves inaltered the number of learnable parameters in the network, however the encoder takes in input both left

Sequence	PSNR			SSIM		
	HDSB	**HDSB-1E**	**HDSB-2E**	HDSB	**HDSB-1E**	**HDSB-2E**
Balloons	**37.24**	35.92	37.03	**0.978**	0.967	0.972
Kendo	39.12	**39.30**	39.20	**0.992**	0.990	0.985
Newspaper	**35.53**	35.24	35.46	**0.967**	0.965	0.962
PoznanStreet	**37.22**	36.3	37.18	**0.968**	0.958	0.965
PoznanHall	38.25	**38.32**	38.31	**0.966**	0.964	0.965
Painter-1	38.09	38.01	**38.18**	**0.970**	0.963	0.964
Painter-5	**38.85**	38.25	38.81	**0.961**	0.960	0.958
Painter-9	**37.95**	37.55	37.67	**0.952**	0.941	0.949
Lovebirds	**35.41**	34.51	34.92	**0.973**	0.962	0.969
Average	**37.52**	37.04	37.42	**0.970**	0.963	0.966

Table 5.6 Effect of the encoder architecture: HDSB-1E includes just one shared encoder for both views, HDSB-2E includes separated encoders for each view.

and right warped views and relative masks. That is, the encoder now faces the challenge of dealing with occlusions potentially on both sides of the objects. This scheme is referred to as *HDSB-1E* in the following.

Second, we consider another HDSB variant where the encoders do not share parameters, i.e. each encoder indipendently processes the left or the right view. This scheme has the potential to deliver better performance as the network includes more learnable parametrs and each encoder learns specialized feature for each view. Obviously, in this case the right view is not mirrored anymore. This scheme is referred to as *HDSB-2E* in the following. We train and test both architectures as for HDSB, and we show the results of these experiments in Tab. 5.6.

Concerning HDSB-1E, the quality of the synthesized view is usually lower than HDSB. For example, for *Balloons* HDSB-1E scores 35.92 dB against 37.59 dB of HDSB, i.e. HDSB scores almost 0.5 dB higher. While for some sequences (e.g.: Kendo, PoznanHall) HDSB-1E scores marginally better, on the average HDSB scores almost 0.5 dB higher on the average. In terms of SSIM, HDSB-1E always score worse than HDSB. We recall that HDSB and HDSB-1E count the same number of parameters, nevertheless the inspection of the training curves shows that the loss function of HDSB-1E flutters more. We attribute such results to the complexity of the single encoder to deal with occlusions on both sides of the objects.

Concerning HDSB-2E, it outperforms HDSB-1E almost for any sequence yet it outperforms HDSB only for *Painter-1*. We recall that HDSB-2E counts twice as many parameters in the encoder as HDSB-1E. The analysis of the training curves show that HDSB-2E is more likely to overfit to the training data in reason of the higher parameters count. We hypothesize that if significantly more training sequences are available, HDSB-2E may have an edge over HDSB.

5.3.4 Approach's limitations

In this work, we have a limited amount dataset because of the non-availability of its open-source. Furthermore, this dataset shares more or less the same characteristics (such as cameras configuration rigs, focal length, object distances from the camera). Therefore, we were interested in testing our method on completely different scene contents and configurations than those used in the training dataset. In these following examples, we do not dispose of the ground truth of the synthesized camera; thus we visually note its quality.

The first example was using a convergent camera configuration rig to understand why HDSB shows its drawbacks since there are no similar cases in the training dataset. As a result, our network could not predict well a plausible performance. *PoznanFencing* is a scene proposed by the University of Poznan as a contribution [111] to the *MPEG-I Video* working group. The camera rig of *PoznanFencing* is convergent and includes 10 cameras, and an illustration of the views and depth maps of the different cameras is available in Appendix A.6.

Fig. 5.9 shows the results of HDSB on *PoznanFecing comparing to VVS*. We visually notice that VVS did not reconstruct well the novel view, but either does HDSB. The performance of HDSB is mainly related to the incapacity of the network to generalize with not enough variety (in particular convergent rigs) in the training dataset, which is a challenging task to be made.

The second example was applying HDSB on a parallel camera configuration rig but with a vast disparity and small cameras angle of view (*PandmoniumRig1* scene). Thus, the cameras share a little information about the scene, and the displacement from one reference point of view to another is significant (80*cm*). *PandemoniumRig1*is a scene produced for b<>com project. The camera rig of PandemoniumRig1 is linear and consists of 5 stereo camera pairs. Since there are no similar cases in our training dataset, HDSB has difficulties reconstructing high-quality novel synthesized views.

Fig. 5.9 illustrates the results of HDSB on *PandmoniumRig1* scene comparing to VVS, where once again both failed. An illustration of the views and depth maps of the different cameras is available in Appendix A.4.

5.4 Conclusion

We presented a hybrid approach to wide baseline view synthesis where the warping is algorithmic while the blending is learnable and inspired by image-to-image convolutional architectures. Extensive experiments on real multi-view video sequences show better performance than pure-algorithmic approaches while avoiding the complexity of purely learning-based approaches and taught us some lessons. First, an encoder-decoder architecture improves over our previous super-resolution-based method by projecting the input features over a spa-

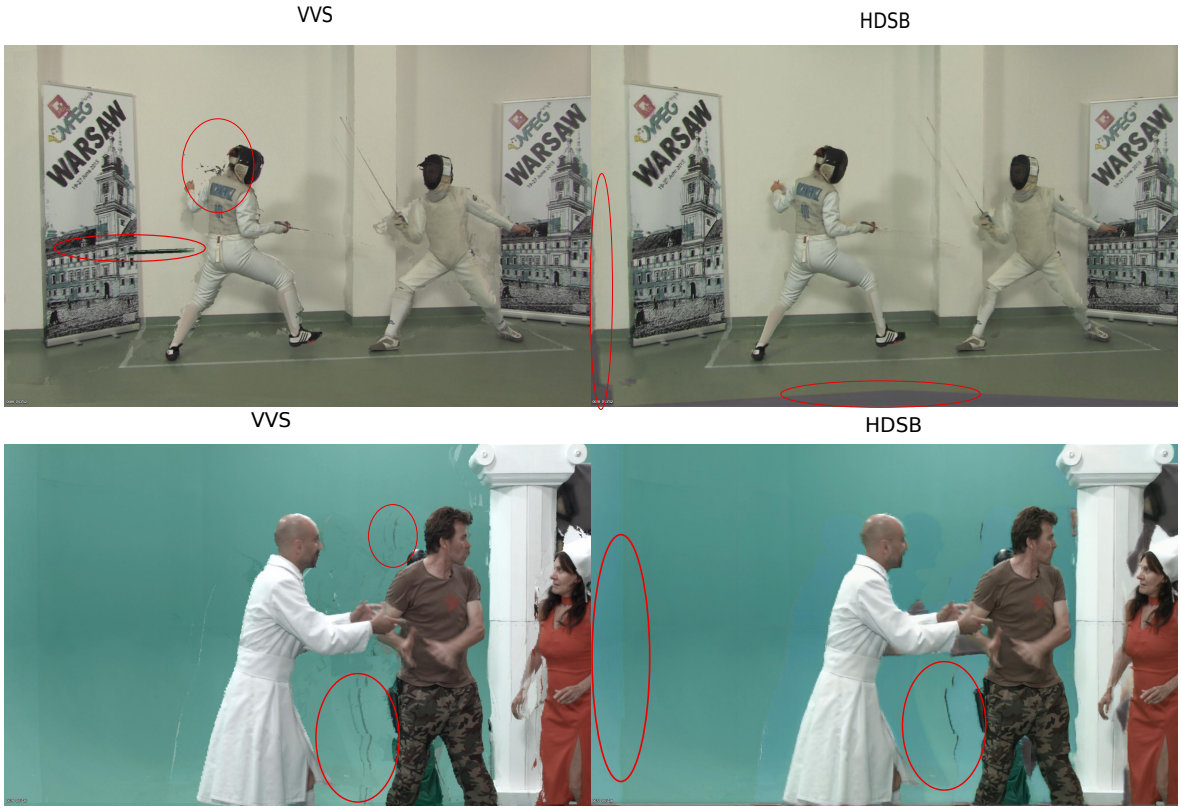


Fig. 5.9 Limitation of HDSB and VVS on: *PoznanFencing* (top) and *Pandemoniumrig1* (bottom). The red circles point out the main area affected by the drawbacks of the methods.

tially subsampled latent feature space. Second, the encoder complexity can be reduced by resorting to a smart *flip-convolve-flip* approach that allows us to share parameters among encoders reducing the network complexity. Third, providing additional filled textures to the blender helps to prevent disocclusions-induced artifacts better than binary masks. Finally, experiments with perceptual loss show visually pleasant images, yet at the expense of a drop in objective visual quality.

Our current research aims to exploit the method's drawbacks and limitations to improve the method efficiency and generalization on as many different scene contents as possible. That is the reason that conducted the research work that is studied and detailed in the following chapter.

Chapter 6

A Multi-View Stereoscopic Video Database With Green Screen (MTF & Pandemonium) For Video Transition Quality-of-Experience Assessment

In this chapter, we introduce a multi-view stereoscopic video database with a green screen, called MTF, for the usages in computer vision applications, particularly for free navigation, free-viewpoint television, and video transition QoE assessment.

In the following sections, we describe the context and the challenges in Sec.6.1. In Sec. 6.2 we review state-of-the-art datasets used for free navigation. In Sec. 6.3 we detail the process of the creation of our dataset, and we dedicated one section 6.4 to the description of the content of the dataset. The dataset characteristics are listed in section 6.5. Finally, we conclude in section 6.6.

6.1 Introduction and context

A film transition is a technique used in post-production of film/video editing by which a series of frames or different points of view are combined. In this chapter we talk about the transition from one point of view to another. In the past decades we were used to watch all kinds of films on a simple 2D display such as television, tablet, cinema, or mobile phone, where all the transitions have become naturally acceptable by the viewer's eye. Few years ago, the Head-Mounted Display (HMD) started to take a big place in the rising technologies, as it gives the user the opportunity to experience the immersive environments. In this case, the viewer's eyes are much closer to the screen than any other simple tablet screen. In both cases, the viewers often need to change their point of view to have a better understanding of

the storytelling. To make this transition happen a visual effect is required, known as transition. Often when filmmakers want to join two shots together, they use a basic cut where the first image is instantly replaced by the next. However, there are many other advanced types of transition that can reconstruct the geometry of the scene in the video and provide a virtual video-based-rendering transition to seamlessly join two videos. Likewise, in free navigation systems, users can interactively control their point of view while watching a video. Such applications use a relatively small number of real cameras in the scene, and intermediate views are generated using DIBR algorithms. This process provides realistic transitions between real cameras of the scene. For a given visualization system the transition type is definitely an impact factor on the user's QoE. For example, since human brain may not adapt to the transition in the HMD yet, viewers may feel a sort of disorientation or dizziness. But different transition type may cause different level of discomfort.

Tompkin *et al.* [112] made a study comparing different types of transition in a video to determine users' preference. For their study they captured their own videos. It is a collection of real outdoor touristic places, where the dynamic video transition helps in having a different view of the touristic site. The dataset used in the free navigation in [52] includes vegetation, vehicles and other complex geometry presented in everyday urban scenes. MPEG-I [113] released a series of multi-view test videos for free-viewpoint television such as the MPEG contributions of Poznan University [111, 114, 115](e.g. *'poznanHall', 'poznanStreet'*) and Nayoga university [116] (e.g. *'Balloons', 'Kendo'*). They focused on real dynamic, complex and textured scenes with moving actors, sometimes with shows and audience. In these sequences, a transition is made to allow users to randomly change their point of view in the scene with small camera displacements (5 to 30 cm). Other navigation dataset is for autonomous driving research [117], or for a virtual house or place visiting [118, 119].

We know that existing open access video sequences are hardly representative of actual DIBR method use-cases. Due to the lack of major points of interest in a scene, observers' attention may only be drawn by the persistent artifacts and imperfections all over the rendered image, which may result in severe judgements.

In our first shooting, we filmed a scene theater called Pandemonium, where we placed the cameras so the viewer would need to change his point of view to follow the actors and keep going with the scene story.

In our second shooting 'MTF', we try to draw observers' attention by one or two major points of interest of the storytelling, so observer's attention would be restricted to one particular area of the scene. Moreover, our contents are designed to prompt the users to change their point of view in the scene at a given time to help understand the story events. In total we created three different scenarios that motivate viewers to change their point of view while watching a scene:

- *Moving point of interest*: when the point of interest of the scene is not visible anymore

in the current point of view and become visible in another point of view of the scene.

- *Changing point of interest*: when the user have two points of interest from two different points of view and he wants to switch between them.

- *Out of curiosity*: when changing the point of view in a scene is not necessary for the understanding of the story but useful for curious viewers to see more details of the scene.

Unlike other state-of-the-art datasets, our sequences were recorded in a studio instead of on-locations. Shooting all our videos against a green screen offers the possibility to virtually change scene locations and to add colors, textures or virtual subjects to the background, based on the experiment objectives when the MTF is used. Another new characteristic of our dataset is the wide distance between real cameras of the scene, which remains a challenging constraint for DIBR methods. The MTF (Mystery in Thorigné Fouillard) is publicly available on : https://drive.google.com/drive/folders/1MYiD7WssSh6X2y-cf8MALNOMMish4N5j?usp=sharing

6.2 Related work

We briefly introduce existing multi-view video datasets in this section. This section is divided into two categories: (1) state-of-the-art synthetic scenes, and (2) state-of-the-art real scenes.

6.2.1 Synthetic Scenes

With the rising of machine learning, a large dataset is often required for massive training. Generating synthetic content is much easier in terms of volume. Replica's high level of realism rendering dataset [120] has a challenging goal to enable machine learning research that relies on many computer vision tasks. The challenging task is to make machine learning systems trained on their dataset to be directly transferred to real-world image and video data. Synthia [121] is also proposed for semantic segmentation in urban scenarios, and it contains realistic synthetic frame sequences captured by a driving car in a virtual world.

6.2.2 Real scenes

To push forward the performance of visual recognition, several real scene datasets have been developed, e.g., Middlebury [122], and optical flow [123]. KITTI [117] is a dataset originally proposed for stereo, optical flow visual odometry, and 3D object detection tasks. It contains frame sequences captured by a car traveling through urban city scenes with camera

poses. Matterport3D [124] is a large dataset of building-scale scenes used keypoint matching semantic segmentation and region classification. MPEG-I [113] they created a series of datasets aiming for free-viewpoint tasks. They are widely used for view synthesis QoE assessment, and they are the most similar to our dataset [111, 114–116].

Unlike our dataset, we do not find a QoE-relevant real scene dataset with a green screen and a wide baseline for video transition evaluation in all these databases.

6.3 Dataset creation

In a film studio with green screen, we set up the scenery, which is about a rest-bar with all the essential equipment, such as chairs/tables and a bar counter. The main characters are three volunteer actors, and some other volunteers played the role of the random clients in the rest-bar. Due to our tight budget, we only select the necessary equipment required for a film-making.

First, we develop and design a capturing system that allows to parameterize and synchronize real cameras in a scene. In the following, we details the three sub-systems of this capturing system, which are :

- The monitoring system
- The control system
- The camera mounting system that responds to the filming constraints.

6.3.1 General architecture

The objective of the capturing system is to be able to control a set of cameras via a monitoring software. This system is made of three parts:

- A hardware control system based on electronic components
- A software control system to drive the components and to monitor the images of the cameras
- A physical system for fixing the cameras responding to the cameras positioning constraints.

6.3.1.1 Operating principle

The system implemented must meet the specific needs of our project:

- Cameras are arranged in pairs and must be positioned in a rectilinear way.

- The distance between two cameras of each pair is 11 cm.
- The distance between each two pairs of cameras is 80 cm.
- The cameras must be all synchronized at image capturing level.
- All cameras have the same settings (focus, aperture and sensitivity and the start/stop trigger control)

The Fig. 6.1 shows the blocks forming the whole solution in a simplified version. The following parts detail each block.

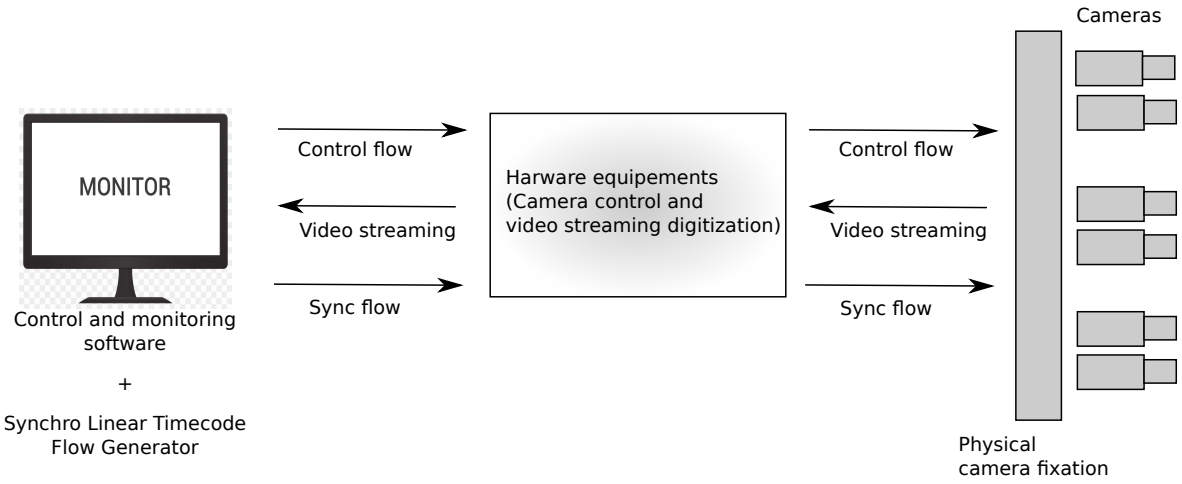


Fig. 6.1 The general architecture scheme

.

6.3.1.2 Exchanged streams

To control and synchronize all the cameras, it is necessary to implement several types of streams:

- Image synchronization recording stream: this stream of type Genlock flow (REF IN [1]) sends a set of peaks to each camera indication to the camera when each image should be recorded. Therefore, all cameras are recorded at the same time.
- Synchronization stream between the cameras: In addition to the REF IN stream, it is necessary to provide the cameras a time-code audio stream (Linear TimeCode (LTC) [2]) to add a timestamp to each image in its metadata. This stream allows the synchronization of all the cameras between them. However, it is possible that some cameras

[1]The REF IN clock is a standard tri-level signal that synchronizes all the camera clocks so that the image recordings happen at the same time.

[2]The LTC allows to indicate a universal clock to a set of off-network equipment, here cameras

start recording one frame earlier than the rest. Thanks to the time-code it is possible re-synchronize all the cameras in the post-production phase.

- Control stream: We chose the S-Bus [3] protocol to send the control commands to the cameras. The S-Bus messages implemented in the selected cameras (Blackmagic), allow us to adjust the cameras (aperture, sensitivity, framerate encoding, focus) and at the same time to control the recording.

All those logical streams (S-Bus, REF IN and LTC) go through and electronic box called the Master Card designed especially for this project. The streams are uni-directional toward the cameras however, the cameras do not notify with a receipt confirmation. In addition to these logical streams, and electronic stream was set up to empower the cameras and to avoid any possible problems related to the battery autonomy. To reduce the number of the cable installation, the logical and electronic streams flow through the same Video Graphics Array (VGA) cable (DB15 connector). Finally, the monitoring video stream goes through High-Definition Multimedia Interface (HDMI) cables connected to a hub whose function is to convert the video flow into an IP flow that is more easily usable (cf. 6.2.

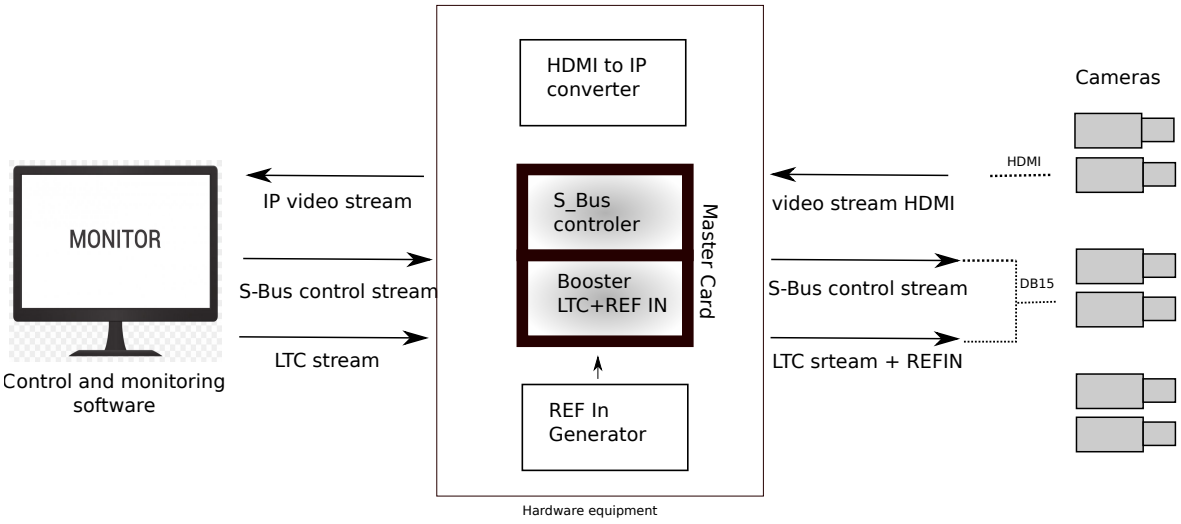


Fig. 6.2 The general architecture of the monitoring control system

.

6.3.1.3 Hardware and Software elements

The hardware part is formed of :

[3]Historically the S-bus is propriety protocol used in model making. It allows to send control commands. Blackmagic Micro Cinema Camera (BMMCC) use this protocol for remote control

- **BMMCC**, which offers the particularity of providing a DB15 connector (The DB15 connector is an analog socket, with 15 pins, from the D-Subminiatures (D-Sub) connector family) through which, all the control flows, as well as the power supply.
- A **camera support system** that meets the project requirements.
- A **Master card** driven by the control PC. This card is the core of the control system, it receives a set of streams (Ref In, timecode, PC command) and redistributes them to the HUB cards. The master card is powered by the control PC via the usb cable.
- **HUB cards** that multiply the signals from the Master card to the cameras. Each HUB card allows the connection of up to eight cameras. The HUB cards are connected to each other in a sequence. The HUB cards also supply the cameras with power.
- A **control PC** that drives the Master card via a monitoring software. The control PC also transmits the timecode signal via its audio output (3.5mm jack).
- A **multi-channel HDMI-IP converter** that receives video streams on the camera's HDMI port and outputs a video stream over IP. Each input stream generates an output stream on an Ethernet port.
- A **switch** that allows the connection of all the IP video streams with the control PC. Each camera emits a stream which is transmitted through an Ethernet cable once the HDMI-IP conversion is done.
- A **Genlock generator** which emits the sync signal (REF IN). This generated is connected to the Master.
- A **power supply units** to supply each HUB card with $\pm 5V$ and the eight cameras with $15V$.
- **Extension cards** that connect between the camera and the VGA cable. These cards are necessary because of the presence of short-circuited signals in the VGA cable. The extension board allows the VGA cable to be connected to the camera by redirecting the power signal.

In terms of cabling, only two cables reach the camera:

- A **VGA-type** cable with DB15 connector for the REF IN, S-BUS and power supply streams. This cable also has 3.5mm jack connector to transmit the LTC stream.
- A **HDMI** type cable to get the monitoring video streams.

The software part is composed of:

- A **Monitoring-Control** software that is located in the control PC, it is designed to display a set of nine monitoring windows. The streams that feed these windows come from the HDMI-IP converter. In addition to this monitoring function, the software sends control commands to the master box for the control of the cameras.
- A **Control Software** that is located in the master box. Its function is to convert the commands received from the control PC into intelligible S-BUS commands for the cameras. It also addresses the cameras via the HUB cards.

Fig. 6.3 illustrates the detailed hardware and software elements.

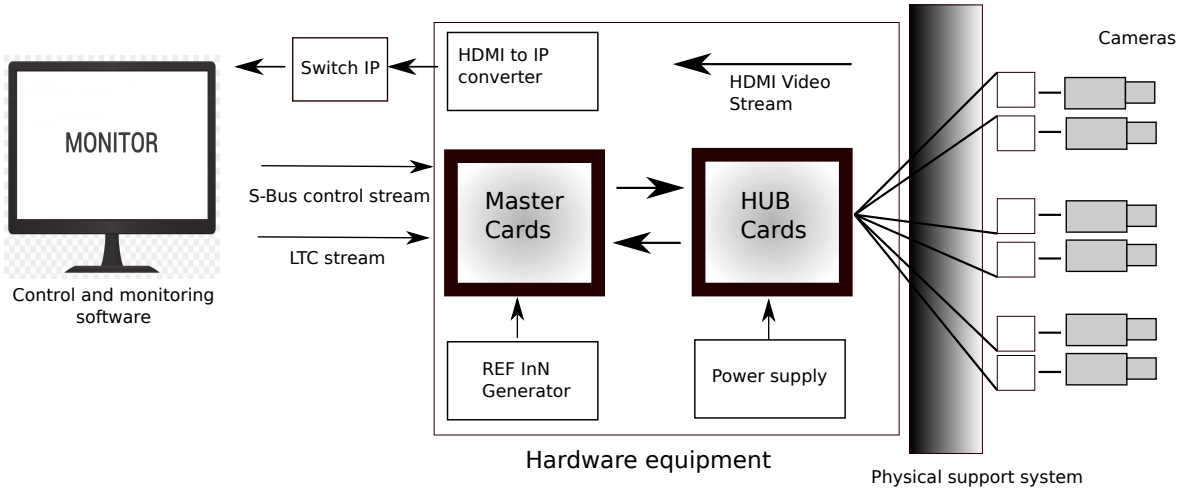


Fig. 6.3 The detailed hardware and software elements.

6.3.2 The monitoring system

The monitoring system consists of two parts:

- a software that displays the monitoring screens, and
- a hardware part that feeds the software with video streams.

6.3.2.1 The monitoring software

The monitoring software is developed in C++/QT framework, and uses the VLC-QT library which allows to benefit from the functionalities of the VLC video player inside a program developed via the QT framework. The software offers the possibility to open a display up to nine video streams in full HD (1080p) over IP simultaneously (cf. 6.4. To monitor more than nine feeds, a pagination system allows you to add other pages to add nine more monitoring screens each time. The monitoring software allows each screen to be displayed in "zoom"

mode to better visualize particular stream. In addition to the monitoring functionalities, the software offers a set of controls for the piloting of the cameras. These functionalities are all accessible via the menu located in the lower part of the interface.

Fig. 6.4 Main screen of the developed monitoring system.

6.3.2.2 Hardware monitoring system

As previously mentioned, each camera has an HDMI output dedicated to monitoring. These streams are digitized via an HDMI-PI converter. The monitoring task is to check if the cameras are well positioned (focusing on the elements of interest in the scene) and well adjusted (the focus is correct). The HDMI-IP converter has a standalone conversion unit that accepts and HDMI stream as input, and generates an video stream on IP as output. The use of converter allows to increase the capacity by adding additional processing units if needed.

6.3.3 The control system

6.3.3.1 Control software

To control the cameras at the recording level and to be able to modify their settings, the monitoring software includes a set of buttons that activate the transmission of the corresponding commands to the master box.

REC start/stop: The recording command works as a trigger. The same command activates and deactivates the recording. There is no feedback on whether or not the camera is

taking the command into account.

Aperture/Iris: For the aperture value we multiply it by 10 to deal with integers.

Focus: The focus control allows the fine tuning.

ISO: The ISO command works as a trigger. There is no feedback from the camera, the current value is unknown.

White Balance: The white balance works as a trigger. There is no feedback from the camera, the current value is not known.

Framerate = $60fps$

Codec = ProRes HQ.

6.3.3.2 Hardware control equipment

The control system is based on the use of:

- A PC under Windows to run the monitoring-control software but also to power the master box via a USB plug.
- A master box which hosts an 'Arduino nano' which runs the software for the master box also it sends the Reflex signals to the HUb cards. The master box also sends the REF IN and the time-code signals to the HUB cards.
- HUB cards whose mission is to re-transmit the commands from the master box to the cameras. These cards are also responsible for transmitting the other streams (power, REF IN, signal, time-code) to the cameras.
- VGA cables with audio plug that connect the master box, the HUB cards and the cameras. The choice of these cables was guided by the fact that the Black-magic cameras have a DB15 connector that accepts as input all the streams necessary for the project.

6.3.4 The camera mounting system

Fig. 6.5 shows the specific mounting system we developed to place the cameras at the same height level. As shown in Fig. 6.6 the system consists of individual camera supports and aluminum rectangular rulers, that are rigid enough to handle the cameras' weight. Hinges are then used to connect the camera supports with the ruler, using vertical adjustable stands. Finally, we mount stands to keep steady the aluminum rulers. All these pieces except for the stands, are created with a 3D printers, then modeled and duplicated for mass production.

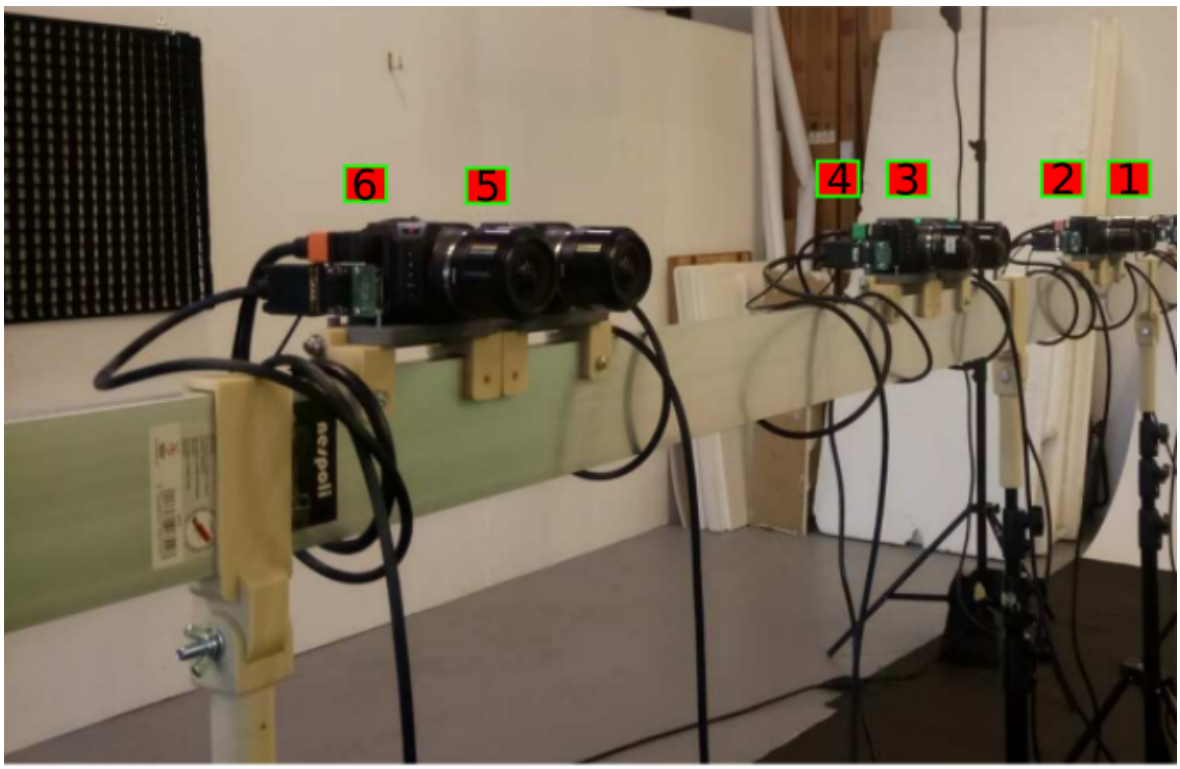


Fig. 6.5 Cameras mounting system, with cameras numbered from 1 to 6.

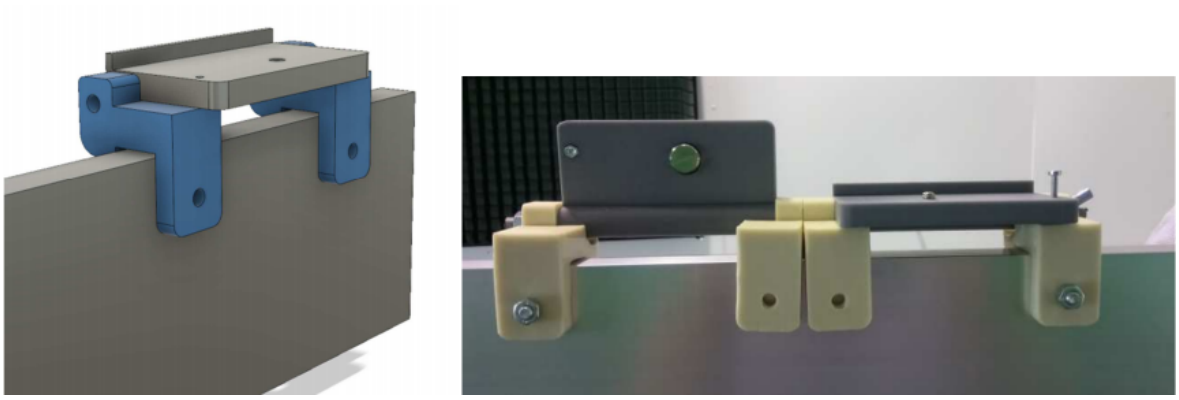


Fig. 6.6 Individual adjustable camera supports and rectangle aluminum ruler connected together using hinges.

6.4 Dataset content

6.4.1 Pandemonium

Pandemonium is a famous theater play played all over the world in different languages. As the capital of Hell is known in the epic poem, Pandemonium combines the Greek prefix pan-, meaning "all," with the Late Latin daemonium, meaning "evil spirit."

Fig. 6.7 Pandemonium- behind the scene.

Nowadays, Pandemonium is defined as a place with chaos, noise, and confusion. We called over a group of professional actors that are used to perform this play. We filmed them from 10 different points of view, by placing the cameras at different positions, where some of them show the backstage scenes as shown in Fig. 6.7. Fig. 6.8 illustrates an example of the different points of view that the viewer can navigate through to follow the play storyline.

6.4.2 MTF

The storytelling that we present comprises three scenes, each of that has a video length of 1 minute.

The story is about a young man called Charles, who has recently moved to the city for his studies and has rented a room at his new landlady's house. Charles is from a very well-known noble family in his hometown. At that time a special symbol was designed exclusively for this family as a sign of honor and respect. This symbol defines each member of this family throughout history. Born and raised in his family's hometown, Charles moved to a nearby town for study purposes when he was eighteen. Shortly after he left, he luckily met his new landlady, and they seem to get along with each other. One night the landlady decided to invite Charles to have a drink in a restaurant to get to know each other better.

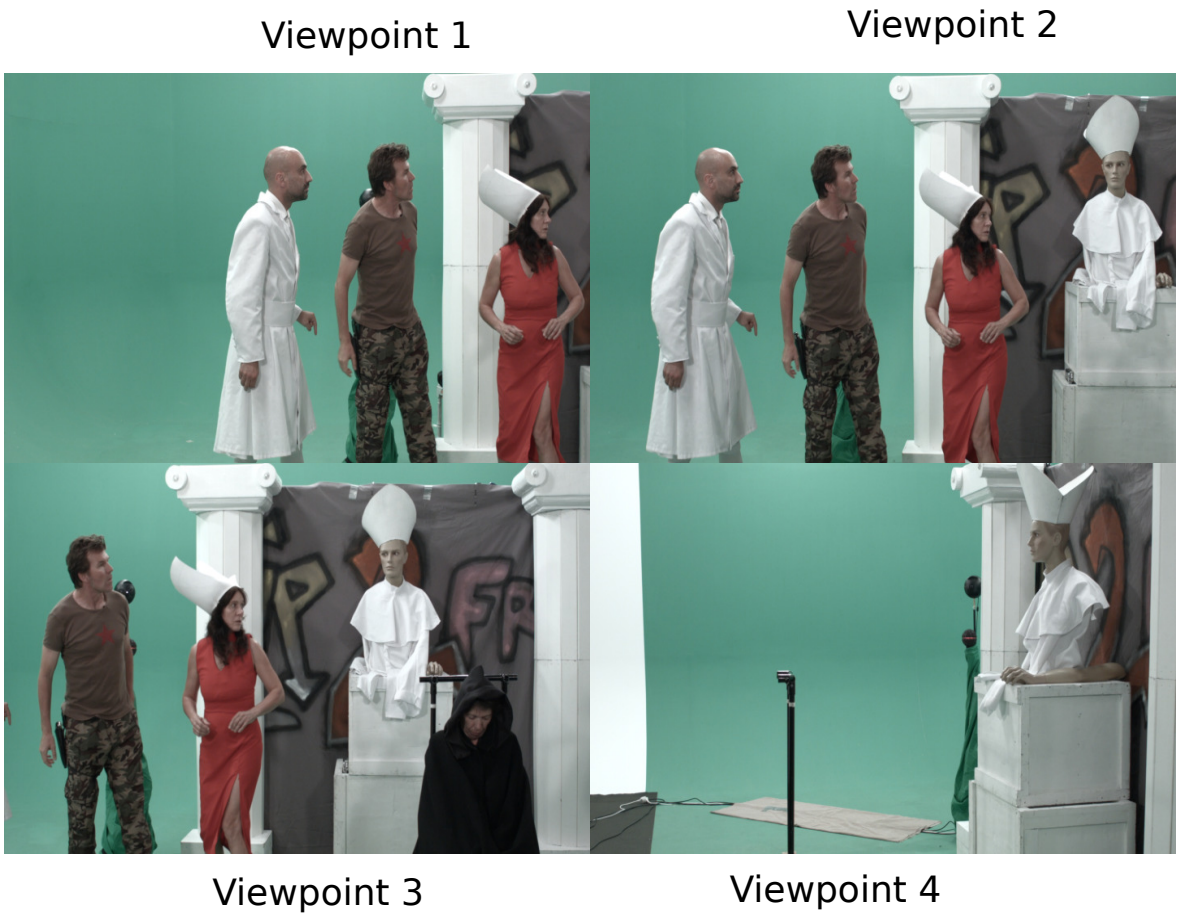


Fig. 6.8 Pandemonium- different viewpoints of the scene.

6.4.3 Scene 1: "key-chain"

The landlady was at the bar, standing next to the door waiting for Charles who was a few minutes late, coming back from university. When Charles arrived, he was feeling a little uncomfortable and nervous about being late for their first drink out, so he apologized to her and dropped by accident his key-chain. Afterwards, the landlady had already spotted an available table for two, so they decided to get it. Suddenly, while walking towards that table, and after being very close to the table, Charles noticed that he lost his keys. He panicked and started looking for them while the landlady tried to calm him down by giving him her hand. Shortly after, the landlady found the keys on the ground next to the entrance door. Charles felt relieved, whereas the landlady was admiring his beautiful key-chain and told him so. Charles proudly explained to her that it is a symbol of his family, and only him and his siblings own this unique key-chain.

6.4.4 Scene 2: "The gossip"

Now that things got back on track, they went back to their places and enjoyed their time, ordered something to drink and started gossiping about anything and everything. The landlady couldn't help herself but ask more details about the story behind the key-chain. With a lot of pleasure Charles opened up to her and told her his story, which helped them to bond with each other.

6.4.5 Scene 3: "The barman"

The third scene is about the mysterious barman. Few hours later, they decided to finally leave so they stood up and went towards the checkout. Finding no one to whom they could pay the bill, The landlady waved at the barman who was a little busy preparing drinks for other clients. He replied by just shaking his head as a sign that he is 'coming'. The landlady noticed directly a tattoo on his arm which looked exactly the same as Charles's key-chain. Curiously, Charles had just told her that it was a symbol of his family only! Who is this guy? Charles was extremely shocked and puzzled and kept staring at him wondering about the truth behind what he just saw!

6.5 Dataset characteristics

The particular features that we identify in our dataset are: the use of a green screen, the wide baseline camera configuration and finally, the creation of points of interest in the scenes to draw the viewer's attention and motivate him to change his point of view. The dataset also comes with additional utilities to facilitate its usage, which are described later in this paper.

6.5.1 Green Screen

Our three scenes are recorded in a green screen studio as shown in Fig. 6.9, which enables the background matting process. This process consists in separating the foreground from the image background thus, it will allow to replace the background with a wide variety of applications, in particular background related to bars/restaurants. To facilitate the foreground/background separation, we avoided the use of green color in the scene. Background matting is widely used in video production, graphics, studio filming, photoshop, and it is modeled with the following formula

$$I = (1-\alpha)\cdot B + \alpha \cdot F \tag{6.1}$$

Where every pixel in a given image I is a combination of foreground F and background B, and α controls the image opacity, a key component to separate the foreground image.

Fig. 6.9 The three pairs of cameras in the green screen studio

6.5.2 Wide baseline camera setup

DIBR methods use real images to generate new virtual images. However these techniques still suffer from disocclusions when there is a large displacement between the real and the desired point of view. Some visible area in the real view become invisible in the synthesized one, and missing information appears as black holes which need to be filled. Our dataset with its small number of real cameras and wide baseline setup, can be a useful experimental material for wide baseline DIBR method test bed.

6.5.3 Points of interest and stimulated transitions

In every scene we created a scenario that motivates the user to move from the current point of view to another. In the following we present examples of transitions for each scene.

6.5.3.1 Moving point of interest (scene 1)

We spot two interesting points of view as illustrated in Fig. 6.10, the first view (Camera N°5) shoots the entrance of the restaurant next to the entrance door, and the second one (Camera N°1) covers the scene around the table. Therefore, we placed a pair of cameras in front of

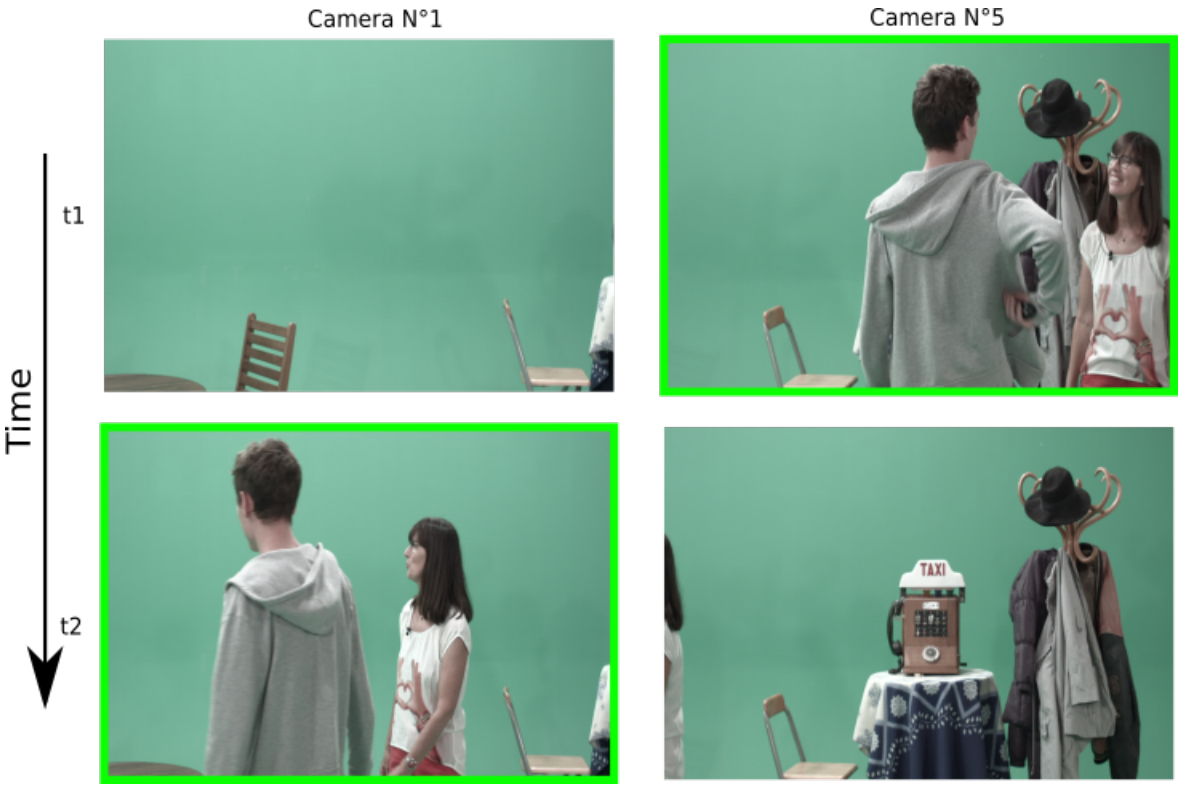


Fig. 6.10 *Scene 1: Moving point of interest.* At time $t1$ the user is watching the view of Camera N°5. At time $t2$, the user move to the view of Camera N°1.

the entrance and another pair in front of the table. The user is going to move between these two locations according to the moving points of interest (Charles and the landlady).

At 45" of the video: actors disappear from the field of view (Camera N°5) and start to appear in another view (Camera N°1). The viewer will thus, naturally move from Camera N°5 to Camera N°1.

At 1'05" of the video: actors shows up again in the Camera N°5 and disappear from the Camera N°1. The viewer will naturally, move from Camera N°1 to Camera N°5.

6.5.3.2 Changing the point of interest (scene 2)

Fig. 6.11 illustrates two people talking around a table. Here, we have two views, each of which points to one character, Charles (Camera N°1) or the landlady (Camera N°4). We placed a pair of cameras in front of Charles and another pair in front of the landlady. In this case, the user switches between the two points of view when he changes his point of interest, e.g the person who is talking may draw more attention.

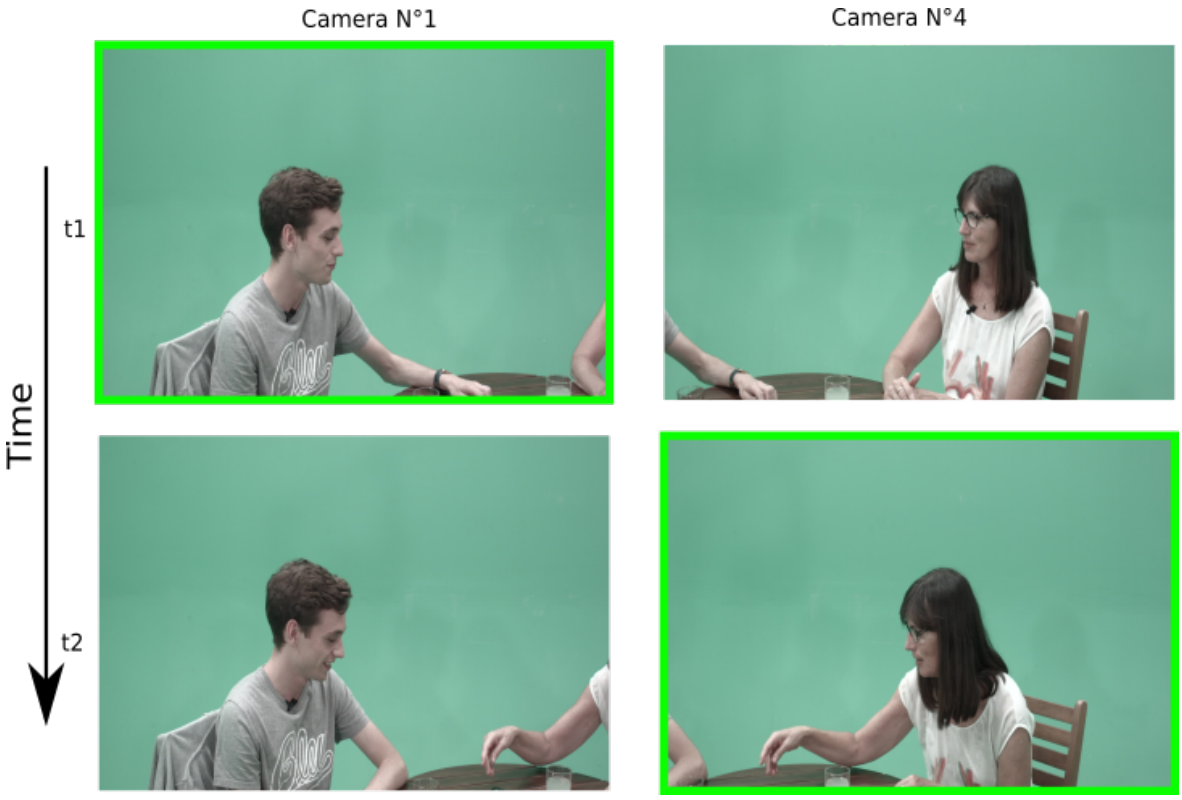


Fig. 6.11 *Scene 2: Changing point of interest.* At time t1 the user is watching the view of Camera N°1. At time t2, the user move to the view of Camera N°4.

6.5.3.3 Out of curiosity (scene 3)

There are two different points of view, the first one (Camera N°5) captures Charles and the landlady waiting to pay their order, and the second one (Camera N°1) shows the barman preparing drinks. We placed two pairs of parallel cameras in front of each point of view as shown in Fig.6.12.

At 47" of the video, the viewer in Camera N°5 sees that the landlady waves at the barman and notices the tattoo. The viewer moves from Camera N°5 to Camera N°1, to see the barman's tattoo out of curiosity.

At 53" of the video, the viewer can go back to the Camera N°5, to continue watching the events of the main characters. The viewer moves from Camera N°1 to Camera N°5.

Note that, in the three scenes, we have an intermediate point of view between the left and right views.

6.5.4 Dataset utilities

Our dataset is provided with essential utilities to facilitate its usage.

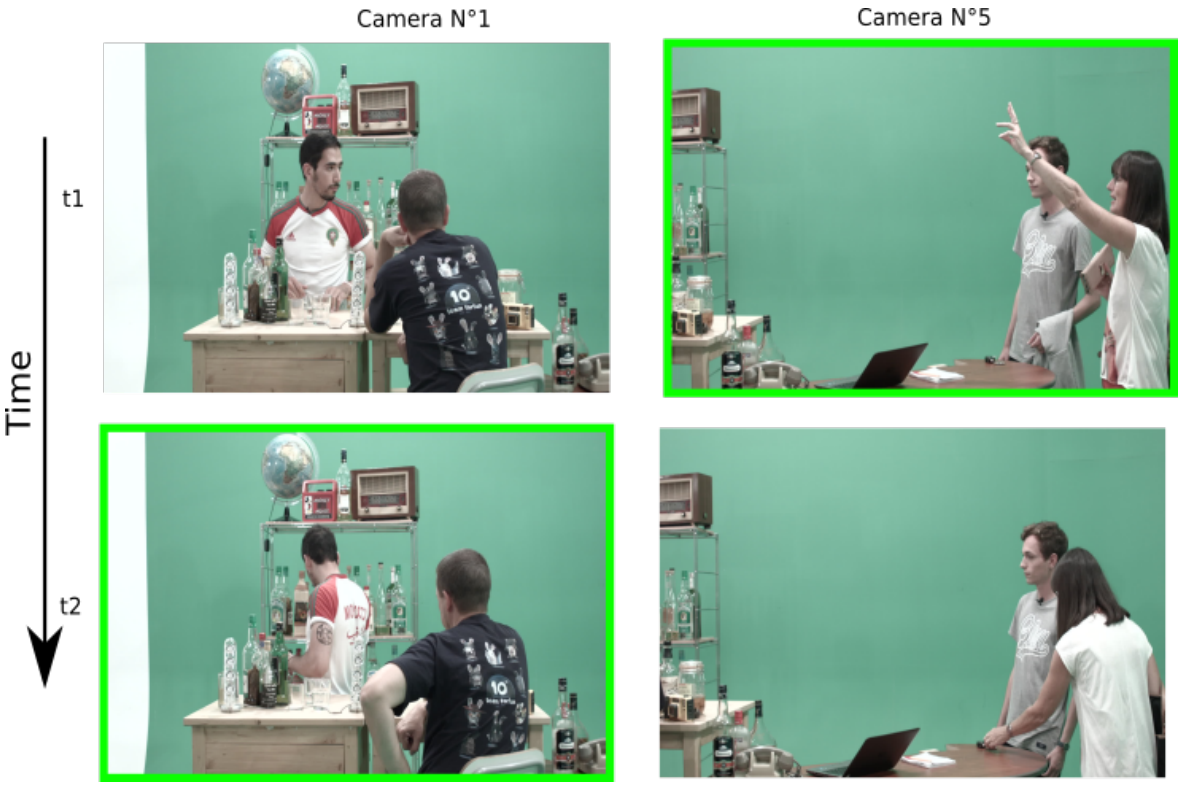


Fig. 6.12 *Scene 3: Out of curiosity*. At time t1 the user is watching the view of Camera N°5. At time t2, the user move to the view of Camera N°5.

6.5.4.1 ColorChecker

As a pre-filming step our cameras captured a ColorChecker to be used for color calibration. In particular, it helps for color correction between two points of view that may have a different color cast, due to lightning coloration difference. To achieve this we recommend the DAVINCI RESOLVE color correction software [125].

6.5.4.2 Calibration checkerboard

Likewise, our cameras captured a checkerboard for camera calibration (intrinsic and extrinsic camera parameters estimation). In particular, it helps in cameras pose estimation. To realize this we recommend several softwares: Alicevision meshroom software [126] , the photogrammetry software Colmap [127] and the Capturing Reality Software [128].

6.5.4.3 Scene complexity

Fig. 6.13 shows the Spatial Information (SI) and Temporal Information (TI) indexes [129] , computed on the luminance component of each sequence (six sequences of our dataset in red, and MPEG test sequences in blue [113–116]). The MPEG sequences are the mostly used in

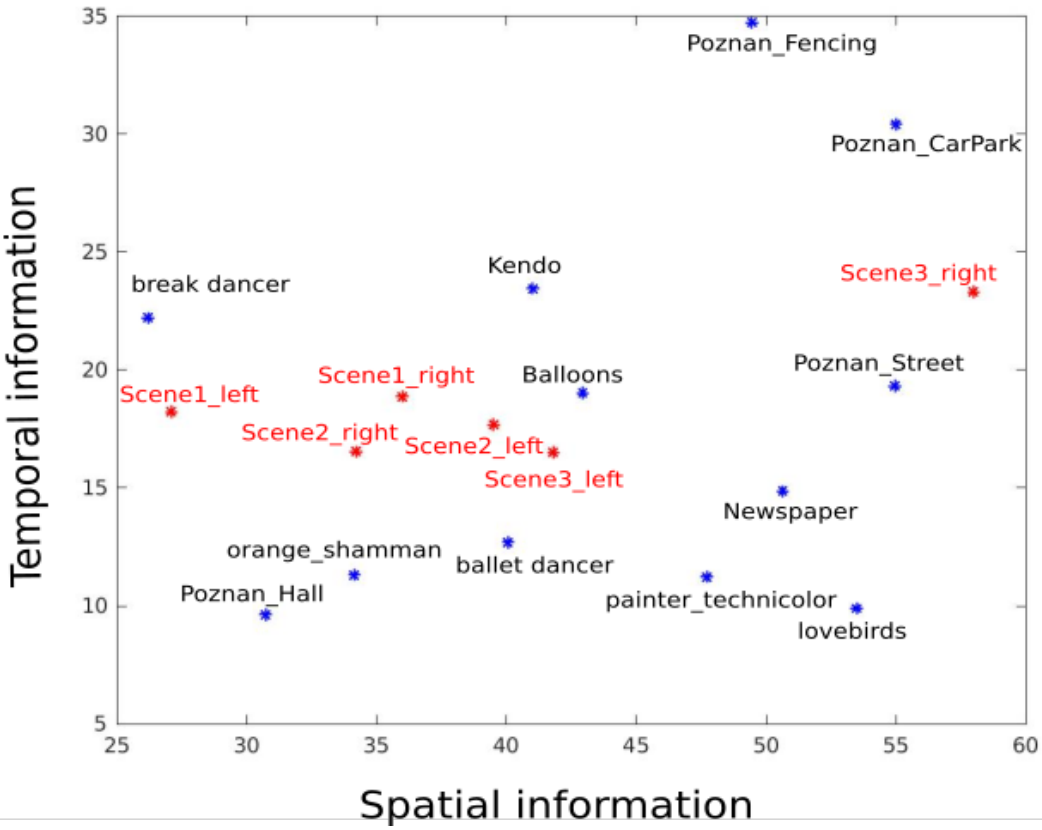


Fig. 6.13 Spatial Information (SI) and Temporal Information (TI) indexes of our scenes (red) and MPEG test sequences (blue)

QoE tests, in particular for view synthesis quality assessment. From Fig. 6.13, we can notice that our sequences can also be used complementary to MPEG sequences, for some usages.

6.5.4.4 Audio files

In our dataset storytelling, the audio is a key component for the understanding of the transition motivation. Therefore, we provide the audio file for each point of view (a pair of cameras) of the scene. The videos are syncing with their respective audio files and can be played using any video/audio player, e.g. VLC media player using *open multiple files* option. In addition to the standard microphones, we used Higher Order Ambisonics (HOA) which is a flexible approach for representing and rendering 3D sound fields. The format used in our scenes, is 4th order HOA microphones (25 channels).

6.6 Conclusion

In this chapter, we present a new dataset with a green screen production for video transition QoE assessment. Our dataset has a storytelling, and each transition between two points of view has an interest for the user. As well as it has similar scene complexity as state-of-the-art dataset and can be considered as complementary sequences with additional features. Due to our limited budget and time we only have 2 sequences. However, the green screen will allow us to change the background of the scenes and multiply the dataset. Finally, our work may attract new collaborations, that allow us to extend our production.

Chapter 7

MUSE: A Multi-view Synthesis Enhancer

In this chapter, we introduce the MUSE (MUlti-view Synthesis Enhancer) method, which is an evolution of the HDSB method and is also based on a hybrid algorithmic-learning-based scheme. We evaluate the visual quality of this new approach by using subjective tests to compare the visual rendering quality with the studied view synthesis methods: HDSB and VVS. Two tests are performed for this purpose. The first one aims to determine whether MUSE is better than the two others or not. The second one's objective is to learn whether the level of quality obtained depends on the content of the scene or the configuration of the filming rings. Finally, a third test is performed to determine whether users prefer different transition effects. For this purpose, the test focuses on comparing a transition performed with a view synthesis method with the most common transition effects.

7.1 Context and challenges

The last chapter presented a hybrid approach where reference views were algorithmically warped to the target position and then blended via a ConvNet. We proposed a residual encoder-decoder for image blending with a Siamese encoder to keep the parameters low. We also contributed a hole inpainting algorithm to fill the disocclusions in warped views. However, this method did not guarantee a network generalization due to the lack of variety in the limited available training database. The present work builds upon our previous research retaining the idea of a hybrid algorithmic-learning scheme where reference views are preliminarily warped to the target position using an inpainting method built around a mean value to handle occlusions. However, this work improves our previous approach under several aspects:

- The warped and inpainted left and right reference views are preliminary merged by a sum weighted by a factor α, to increase the contribution of the closest reference view.

- The masks of the disocclusion areas are preliminary merged by the boolean binary operator "*and*", to generate one mask that contains areas invisible by either the two reference views.
- A concatenation of the generated mask and the merged references form the only one input data considered by the network, and a single encoder can thus be used.
- The network does not blend the views but corrects the image artifacts resulting from the pre-merged image, and improves the quality of the final image.
- The learning process of the network is called "intra-content" and therefore, it is different from the inter-content learning process used for HDSB. Only the triplet images (left warped image, right warped image and the reference view) are considered for a given content.

The merging of the warped and inpainted textures coming from the lateral reference views ensures, in particular, a blending of the non Lambertian areas of the image (e.g. reflections, surfaces changing appearance according to the viewpoint).

7.2 Pre-synthesis process

The pre-synthsis process, in the same way as in HDSB method, uses the two reference views to generate the view to be synthesized. The steps leading to the generation of the warped and inpainted left and right reference texture images T_f^{Lw} T_f^{Rw}, and the generation of the binary masks (M^L, M^R) corresponding to the disocclusion area in (T^{Lw} ,T^{Rw}) respectively, are the same as those described for the HDSB method (add reference section). The view T' is pre-synthesized by merging the warped ans inpainted texture images T_f^{Lw} T_f^{Rw} by a sum weighted by a factor α:

$$T' = \alpha \cdot (T_f^{Lw}) + (\alpha - 1) \cdot T_f^{Rw}, \tag{7.1}$$

We use a factor for α that corresponds to the normalized distance between the virtual view and the two reference views and allows to increase the contribution of the reference view that is closest to the virtual one. For example, if the virtual view is very close to the left view, the factor α will be close to 1 and the contribution of the left view will be dominant. The fusion of the binary masks M^{Lb}, M^{Rb} is performed by the boolean operator *and*, and the produced common disocclusion map contains then only the areas that are not observed by either the left or the right view. The tensor B is the result of the concatenation of T' which is a 3-channels color image (in RGB or YUV format) and the mask M' of the disoccluded areas. B is the input to the neural network whose role is to improve the quality of the pre-synthesized view T^v.

7.3 ConvNet-based View enhancer

The ConvNet view enhancer corresponds to enhancing the pre-synthesized image T' quality, which is the last phase of the process. We briefly describe here the architecture and the relative training procedure.

7.3.1 ConvNet architecture

The architecture of the enhancer neural network is similar to the architecture used by the HDSB method. However, there are some remarkable differences between HDSB architecture and the view-enhancing problem considered here. First, we only need to deal with a single input (the concatenation of the merged reference views with the mask, B rather than four inputs in HDSB architecture, thus only one encoder can be used instead of two. Second, the pre-synthesized image is characterized by particular types of artifacts other than those generated due to disocclusion problems, but instead due to algorithmic blending/inpainting imperfections, such as ghosting problems (as shown in 7.1).

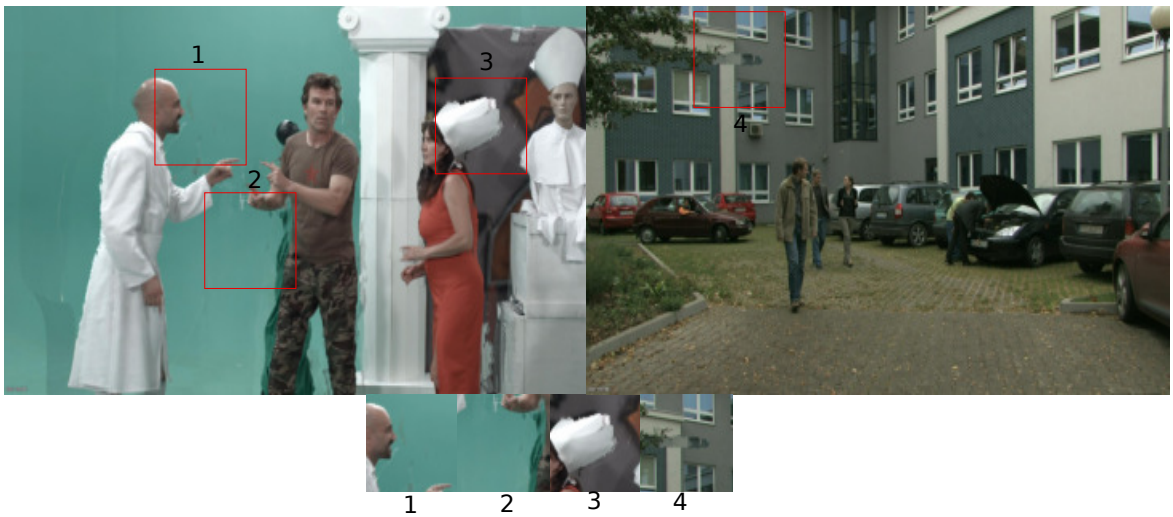


Fig. 7.1 Illustrations of algorithmic blending/inpainting imperfections in the pre-synthesized image.

Therefore, our network should learn how to enhance the quality of the input blended image by reducing artifacts and ghosting effects and refining the image details using an adapted architecture. Thus, we propose a specific encoder-decoder architecture that uses a single blended image as input and generates one output image. Fig.7.2 illustrates our architecture made of three parts, the *encoder*, the *enhancer* and the *decoder*.

Encoder. Our ConvNet, includes one encoder that takes in input the view made of the pre-synthesized view T' and the mask M'. Similarly to HDSB encoders, the role of MUSE

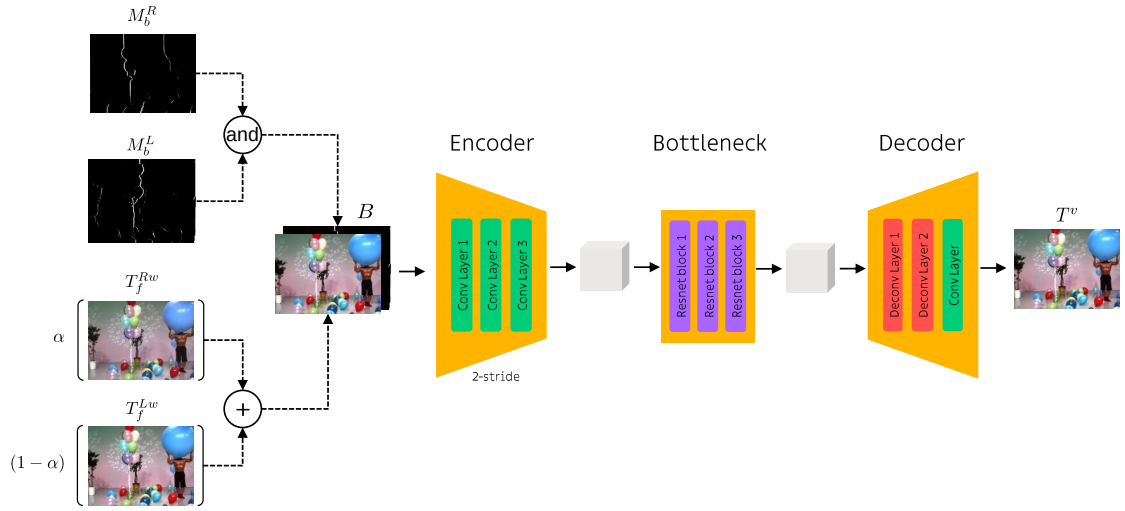


Fig. 7.2 Illustration of the MUSE architecture. MUSE is based on two main stages: the pre-synthesis of the view T' and its correction or enhancement T^v. The parameter α corresponds to the normalized distance between the virtual view and the two left and right reference views that frame the target view and regulate the reference views' mixing within the pre-synthesized view.

encoder is to project the view to a spatially-subsampled latent feature space. To this end, the encoder includes three convolutional layers with 64, 128 and 256 filters respectively of size 7x7 for the first layer and 3x3 for the last two, all followed by ReLu activations. Filters in the convolutional layers have 2-units stride, so that the feature maps in output of each layer are half sized of the feature maps in input to the layer. This encoder typology is inspired by HDSB architecture as it achieves the best tradeoff between semantic depth and spatial resolution of the output feature maps.

Enhancer. The enhancer of our ConvNet is similar to the blender in HDSB architecture, and it aims to correct the input image imperfections and artifacts. The enhancer includes 6 residual blocks with 512 filters each. These layers combine the feature maps extracted from the pre-synthesized image and based on these features it makes the decision about how to transform them into a set of feature maps that holds a suitable representation of the desired target view.

Decoder. Similarly to the HDSB decoder, MUSE decoder synthesizes the virtual view at the target position exploiting the downsampled feature maps produced by the enhancer. This decoder also includes 3 transposed convolutional layers with 256, 128 and 3 filters per layer of size 3x3 for the first two layers and 7x7 for the last layer. The decoder upsamples the low resolution feature maps produced by the enhancer component to a higher resolution. The first two layers are followed by ReLU activation function, while the output layer is followed by a

hyperbolic tangent. We also stack a batch normalization layer after each convolutional layer as in ResNet blocks, as our experiments showed it speeds up the training process. Overall, MUSE network just as HDSB network, produces in output a three-channels view where each channel is expected to have approximately zero-mean and the pixel intensity is bounded in the $[-1,+1]$ interval by the output layer nonlinearity.

7.3.2 Dataset and protocol

We train and test our network with multi-view sequences commonly used in MPEG experiments as defined in the MPEG CTCs (views and depth maps are in uncompressed YUV format). We also add our sequences (real and synthetic) to that list of sequences, producing views and depth maps in the same format as MPEG sequences. Tab. 7.1 summarizes the characteristics of the different sequences.

Sequence	PandemoniumRig1	PoznanStreet	PoznanFencing	PoznanCarpark	OrangeShaman	TechnicolorPainter	Adventure	VikingVillage
Baseline [m]	0.79	0.14	0.22	0.14	0.2	0.072	0.8	0.8
hfov [deg]	36	58	58	58	90	46	66	92
Focal [pix]	2900	1700	1700	1700	960	2400	1500	940
depth n5 [m]	3.7	4.8	0.63	4.6	1.3	2.4	3.1	7.7
depth n95 [m]	7.5	44	0.8	28	4.6	4.2	6.6	85
Disparity min [pix]	310	5.4	470	8.5	42	42	180	8.8
Disparity max [pix]	630	49	600	52	150	73	390	97
Delta disparity [pix]	320	44	130	44	110	31	210	88
Number of images	600	250	250	250	300	300	510	510
Length [s]	10	10	10	10	10	10	10	10
Frame rate [image/s]	60	25	25	25	30	30	50	50
Type	natural	natural	natural	natural	synthetic	hybrid	synthetic	synthetic
Number of cameras	10	9	10	9	5	4	22	22
Resolution [pix]	1920 x 1080	1920 x 1088	1920 x 1080	1920 x 1088	1920 x 1080	2048 x 1088	1920 x 1080	1920 x 1080

Table 7.1 Summary of the characteristics of each scene and their associated capturing system

We used 8 contents or multi-view video scenes, where 3 contents are synthetic (Adventure, OrangeShaman, and Viking Village), and the other 5 are from real video capturing (PandemoniumRig1, PoznanCarpark, PoznanFencing, PoznanStreet, and TechnicolorPainter). Fig. 7.3 illustrates the 8 different contents created and selected.

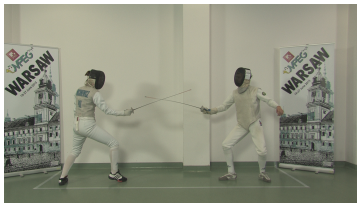
(a) Adventure

(b) OrangeShaman

(c) VikingVillage

(d) PandemoniumRig1

(e) PoznanCarpark



(f) PoznanFencing

(g) PoznanStreet

(h) TechnicolorPainter

Fig. 7.3 Illustration of the selected contents.

Adventure is a scene generated for MUSE experiments using the Unity3D platform (cf. [130]) from a free 4D scene model. The camera rig of the Adventure scene is linear and has 11 stereo camera pairs.

An illustration of the views and depth maps of the different cameras is available in Appendix A.1.

OrangeShaman is an excerpt from the original scene proposed by Orange as a contribution [131] to the MPEG Immersive Video working group. Only the cameras in the third row of the original camera array were selected.

An illustration of the views and the depth maps of the different cameras is available in the Appendix A.2.

VikingVillage is a scene generated for the purposes of MUSE experiments, using the Unity3D platform, from a free 4D scene model.

An illustration of the views and depth maps of the different cameras is available in Appendix A.3.

PandemoniumRig1 is a scene produced for the purposes of b<>com project. The capturing equipment used is identical to that described in Sec. 7.3.2. The camera rig of PandemoniumRig1 is linear and consists of 5 stereo camera pairs.

Intrinsic and extrinsic camera calibrations were performed using the Meshroom[1] method proposed by AliceVision. The depth maps were estimated from the results of calibrated and dense stereo pairs using Flownet2 [132].

An illustration of the views and depth maps of the different cameras is available in Appendix A.4.

PoznanCarpark is a scene excerpt from the original scene proposed from the original scene proposed by the University of Poznan as a contribution [133] to the *MPEG-I Video* working group. The camera rig of PoznanCarpark is linear and consists of 9 cameras.

An illustration of the views and depth maps of the different cameras is available in Appendix A.5.

PoznanFencing is a scene excerpt from the original scene proposed by the University of Poznan as a contribution [111] to the *MPEG Immersive Video* working group. The camera rig of PoznanFencing is convergent and includes 10 cameras.

An illustration of the views and depth maps of the different cameras is available in Appendix A.6.

PoznanStreet is a scene excerpt from the original scene proposed by the University of Poznan as a contribution [133] to the *MPEG-I Video* working group. The camera rig of Poznanstreet is linear and consists of 9 cameras.

An illustration of the views and depth maps of the different cameras is available in Appendix A.7.

TechnicolorPainter is an extract of the original scene proposed by Technicolor as a contribution [134] to the *MPEG-I Video* working group. Only the images from the cameras in the second row of the original $4x4$ camera array were selected.

An illustration of the views and depth maps of the different cameras is available in Appendix A.8.

7.3.3 Training procedure

The network is trained in a fully supervised way on a triplets of patches $(\widetilde{t'}, \widetilde{m}, t^c)$ extracted respectively from the mask M' and the textures (T', T^C), where T^C is the ground-truth image to synthesize. However, the training process follows the intra-content training strategy which is one of the very successful strategy used in [23]. Thus, the network is only trained on the triplet images (the merged warped left and right reference images, the mask and the ground-truth image) of the given content. Previously, in the HDSB network, we faced network generalization problems due to the limited production and availability of multiview plus depth sequences used for the training process. To deal with this problem, one idea is to perform one complete network training per sequence by using each time the available real

[1] cf.https://alicevision.org/meshroom

views of this sequence (the real left and right views to be warped to a real intermediate view that corresponds to the one to be synthesized). We assume that this method will allow the network to retain the sequence content features by over-learning on the available views. Therefore, such a network will produce a better image quality of a virtual point of view in the same sequence.

For each sequence, we extract three neighbor views from the first 750 frames: the left and right views are used as references (T^L, T^R), and the central view is used as target view (*i.e.* ground-truth T^C). For each triplet, we preliminary warp, inpaint, and blend T^L and T^R to obtain T', and we generate the corresponding binary mask M' using the methods described earlier in Sec.5.2.2. From each sequence we extract 75k triplets of co-located patches $(\widetilde{t'}, \widetilde{m}, t^c)$, where 7500 patches are reserved for testing. The other patches are used for training and validation purposes, for a total of 52k training patches and 15k validation patches. Similar to the HDSB method, for the training, we used patches of 64x64, batches of 128 patches, and a learning rate of 0.0001, leading to the convergence of our learning algorithm after 100 epochs. Concerning the parameters optimization algorithm, we use Adam with weight decay $= 0$ and betas = (0.9, 0.999). In all our training experiments, we use the same hyper-parameters showing the best results. This method is also implemented in Pytorch, and all the experiments are performed on a server with an NVIDIA RTX2080GPU.

Concerning the loss function to minimize training time, we use the same MSE loss described in the previous chapter 5.

7.4 Experiments and results

In this section, we qualitatively evaluate MUSE. We perform a series of subjective tests aiming to evaluate the visual quality of our method in the case of an actual view synthesis application, which is a *video transition*. A video transition is a technique used in post-production video-editing in order to join two shots together. In this experiment, we consider the transition from one point of view to another in the same scene. Therefore, we use our method to create a virtual transition between two cameras in the same scene to change the user's point of view.

To achieve this, we create several intermediate views that are played one after another, such as a "path" between the starting and ending points. Tab. 7.1 shows the scene properties for each sequence to create the transitions, i.e., the number of real cameras in the scene, the baseline... It also shows the number of synthesized images in every scene, necessary to transition between two real cameras, with its duration. Fig. 7.5 shows the geometry of the path where it is a straight line in all the sequences except for *PoznanFencing* sequence it is a curve. We can rely on the methods described in Sec. 2.2.2 and 2.3 for the generation of virtual views in a parallel cameras setup (straight-line transition) or convergent cameras

setup (curved transition).

We evaluate the quality of such transition applied on all the contents enumerated in the Sec. 7.3.2. We thus, compare the quality of transitions generated by MUSE to transitions created with other view synthesis methods.

Fig. 7.4 illustrates the final targeted user experience. The user (1) visualizes, with the help of a 6-DoF player installed on a device such as a PC, a tablet, or a virtual reality headset, the video sequence associated with a real camera (2). He/she can navigate from one real point of view to another real point of view. Between two real points of view, the images associated with a virtual camera (3) are generated by an off-line synthesizer such as MUSE and played back by the player (1).

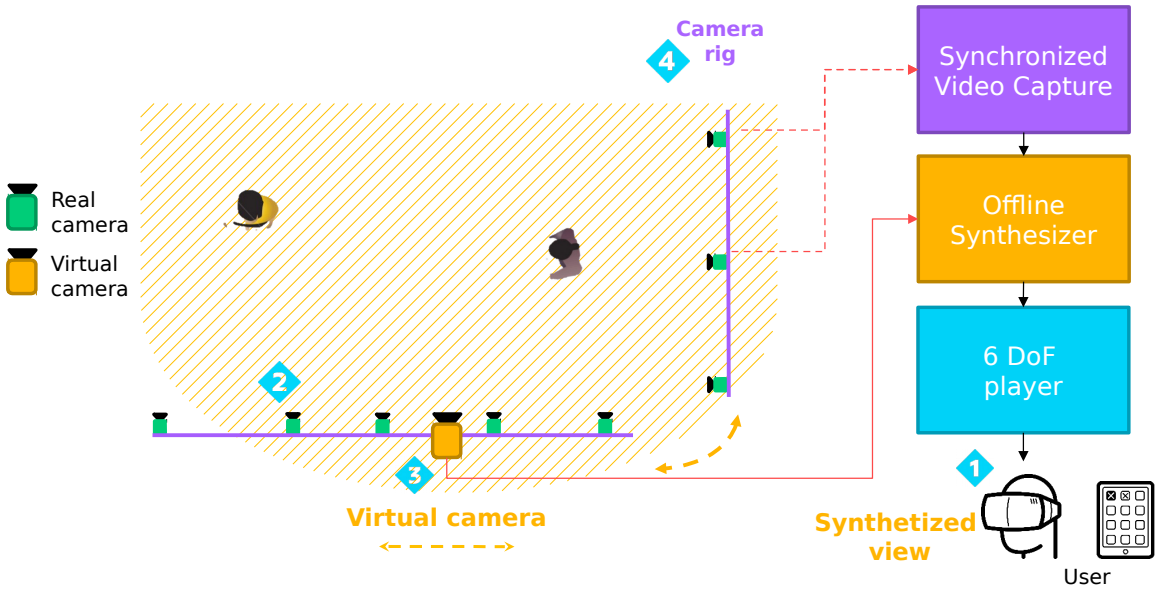


Fig. 7.4 From multi-view capture to 6DoF rendering

To generate the intermediate views corresponding to the displacement of the virtual camera between two real viewpoints, the positions of the virtual cameras are computed. The positions of the virtual cameras were linearly interpolated to guarantee the constant speed of movement of the virtual viewpoint. The virtual camera orientations were determined by interpolating the quaternions describing the real camera poses using the Spherical Linear intERPolation (SLERP) method [135]. Fig. 7.5 illustrates the camera rig configurations associated with each scene (in green), and the virtual camera positions considered for intermediate view synthesis (yellow).

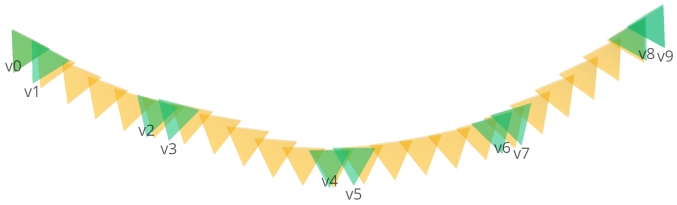
(a) Adventure

(b) OrangeShaman

(c) VikingVillage

(d) PandemoniumRig1

(e) PoznanCarpark



(f) PoznanFencing

(g) PoznanStreet

(h) TechnicolorPainter

Fig. 7.5 Illustration of the camera rig configurations of the different scenes (in green: reference cameras; in yellow: example of virtual cameras considered for the synthesis of the intermediate views).

7.4.1 Experimental setup

7.4.1.1 Test-bed

The hardware architecture implemented for our subjective tests is described in Fig. 7.6. The PC manages the test sequencing, with the stimuli presentation to be evaluated. The display of the voting interface follows after. In order to reproduce video sequences of different frame rates (25Hz, 30Hz, 50Hz, and 60Hz depending on the content), a screen with V-SYNC/G-SYNC capabilities is used to guarantee a real-time display without visual artifacts. Therefore, the constituted test-bed optimizes the display of uncompressed video sequences (yuv/1080p/8bits format), recorded on SSD and played with "mpv."s player (an open-source media player software based on ffmpeg, cf. [136]).

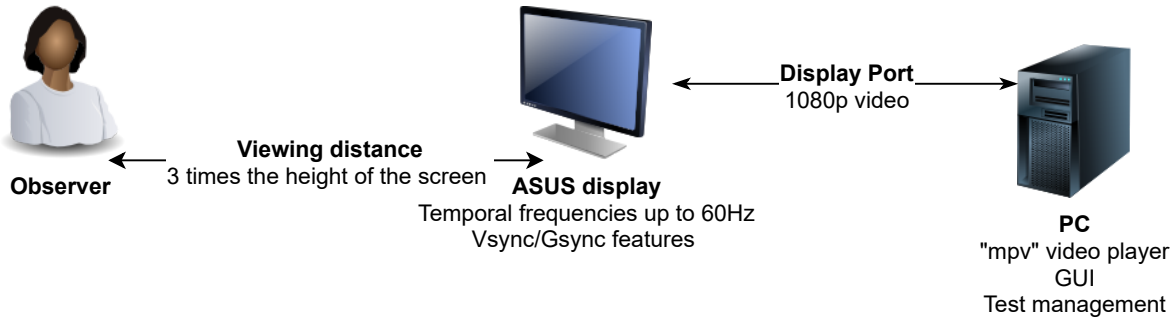


Fig. 7.6 Hardware architecture implemented for the subjective tests.

Operating with the display settings can affect the results of the subjective tests. Therefore, it is essential to ensure that the display selected for the testing meets the requirements of the subjective testing standards, particularly those defined at the International Telecommunication Union (ITU) [137]. The display used for our subjective tests is a 24-inch LED PC monitor (ASUS VG248QE). The decision of the screen selection and the screen settings was based on the following tests:

- measurement of the time responses oscillating between 30Hz and 60Hz,
- photo-metric and colorimetric calibration (SDR, BT.709) [138]
- black level adjustment (BT.814 test pattern standardized to ITU-R [137]) and
- Validation of the maximum spatial frequencies displayed on the screen (Society for information display (SID) frequency test pattern [139]).

Temporal test patterns alternating white and black images at 30Hz and 60Hz were presented to validate the excellent display of video content on the display screen. The oscillating time responses were calculated from the oscillograms of the temporal luminance variations

displayed on the screen (Fig. 7.7 for 30Hz and Fig. 7.8 for 60Hz), The computation measures the time responses between 10 and 90 % of the response at one step (black to white for the increasing time and white to black for the decreasing time).

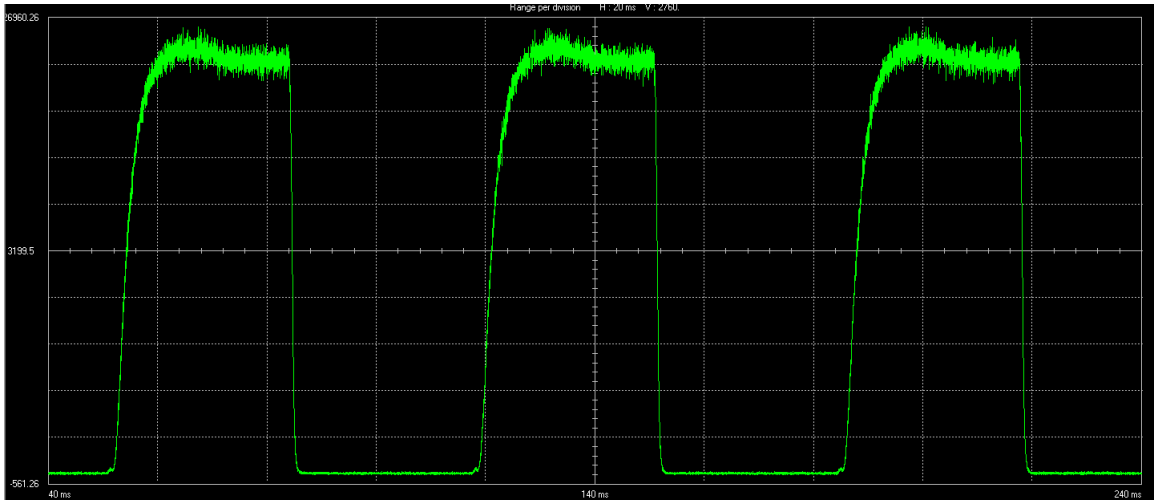


Fig. 7.7 Temporal response of a test pattern with alternating black and white images at 30Hz.

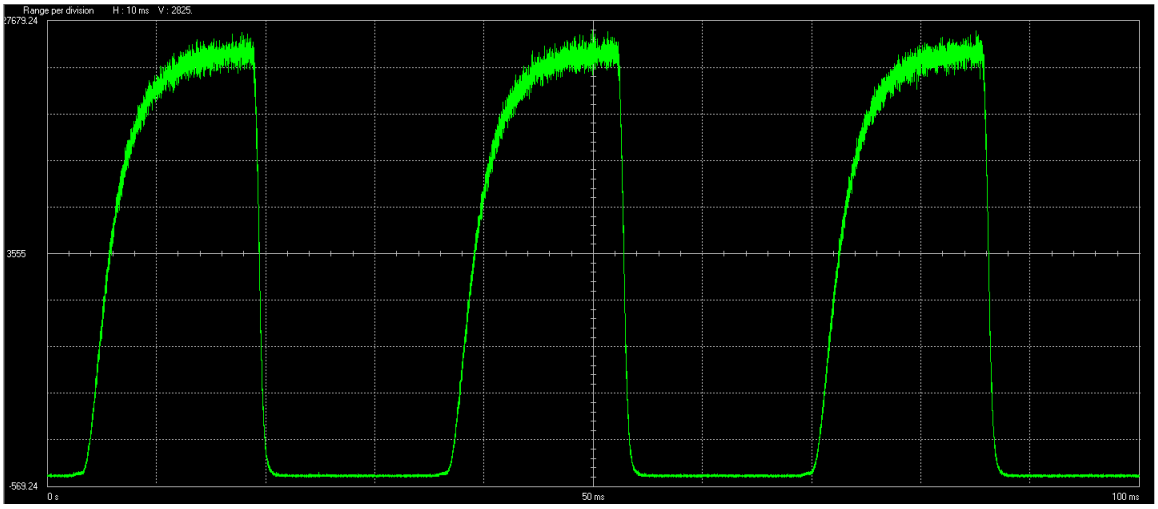


Fig. 7.8 Temporal response of a test pattern with alternating black and white images at 60Hz.

The time responses measured with the photometer "Optiscope" and the software "DisplaySpec" of the company ELDIM are respectively 5.29 ms/1.43 ms (60Hz) and 5.1 5ms/1.65 ms (30Hz) for the rise and fall times (Fig. 7.9 and Fig. 7.10).

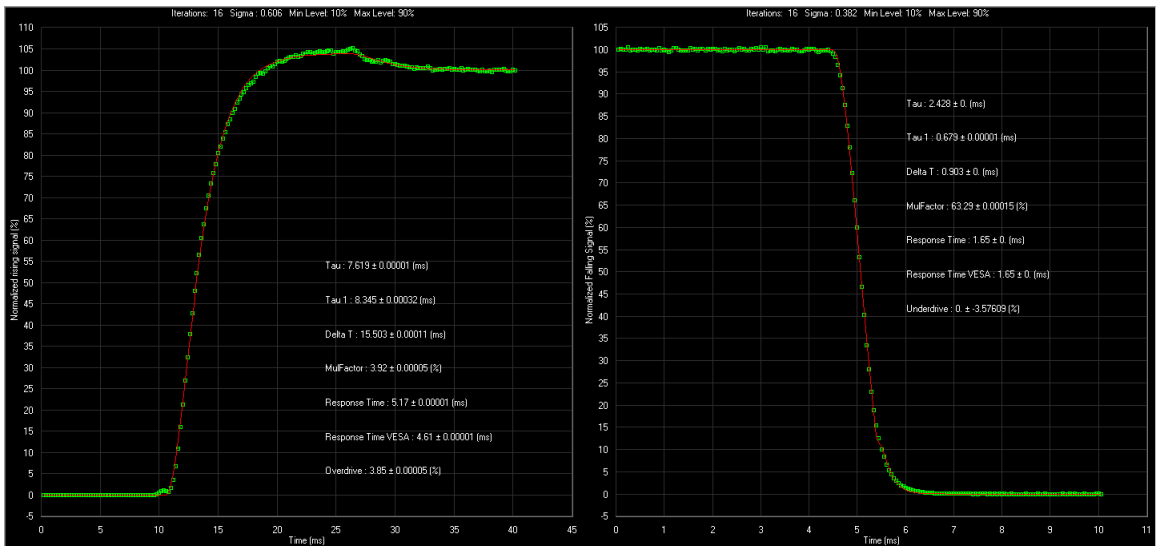


Fig. 7.9 Response time, up and down at 30Hz

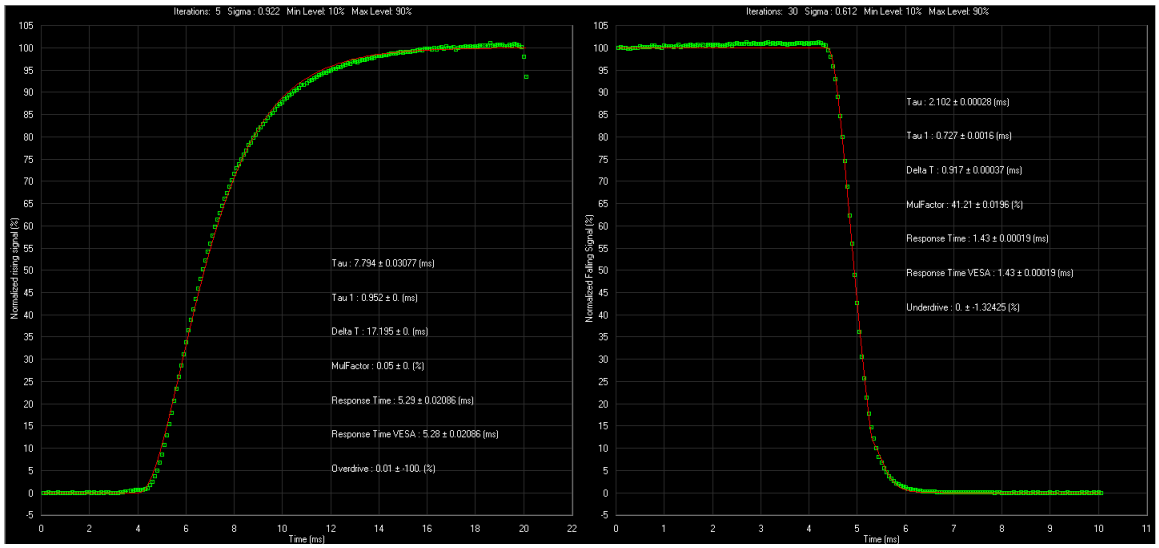


Fig. 7.10 Response time, up and down at 60Hz

The photometric and colorimetric calibration of the screen was performed with “i1-profiler” software and an “x-rite” sensor [140]. Finally, two test patterns were used to adjust the black level display (SDR test pattern according to recommendation ITU-R BT.814-4 [137]) and the SID test pattern [139] to check the correct display of the maximum spatial frequencies for HD image definition (Fig. 7.11).

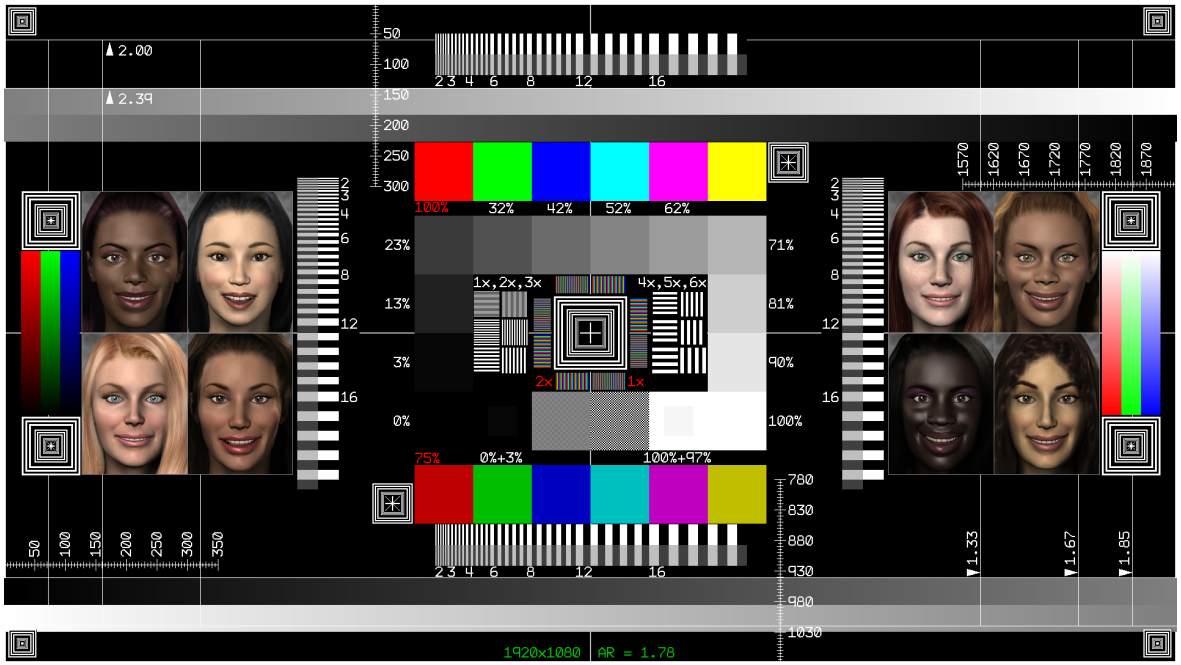


Fig. 7.11 Visual test pattern used to validate the display of HD images (1080p).

All the tests and settings carried out indicate a level of performance of the ASUS display chosen is in line with the one required for the subjective tests.

7.4.1.2 Display conditions

All our subjective tests were performed in a dedicated room with the characteristics described in Tab. 7.2. This viewing environment was defined according to the recommendations of the ITU-R BT.500-14 [137]. It guarantees the visualization comfort of the participants and ensures optimal visibility of the details and the reproducibility of the results compared to similar studies. Fig. 7.12 illustrates the test environment.

display screen	definition: 1920x1080 framerate: 25Hz, 30Hz, 50Hz et 60Hz peak brightness: 120 cd/m2
viewing distance	3 times the height of the screen
background color	grey
background brightness	15% of the peak brightness of the screen, i.e 18 cd/m2

Table 7.2 Display conditions.

Fig. 7.12 Participant during a subjective test.

7.4.1.3 Testers

27 non-expert observers between the age of 23 and 53 participated in these tests. All of them had a vision in accordance with the criteria defined in the ITU recommendations, namely:

- visual acuity of at least $10/10$ (Snellen's E test) and
- non-deficient color vision (Ishihara's chromatic tests).

The following part is dedicated to the description of the first conducted subjective test to evaluate and compare the visual quality of MUSE in a practical use-case.

7.4.2 Test 1: Evaluation of the perceived quality of the view synthesis methods

The main objective of this subjective test is to compare the visual quality of the different view synthesis methods studied in this work: VVS [21], HDSB and MUSE. These tests should al-

low us to determine if the MUSE method outperforms the other two methods. Consequently, a second objective is to establish whether the level of the restored quality depends mainly on the content of the scenes or the configuration of the capturing rigs.

7.4.2.1 Assessment methodology

The assessment method used is the ACR method described in ITU-T recommendation P.913 (cf. [141]). The choice of this method was imposed by several constraints related to the available scenes. There is no ground truth for the natural contents, i.e., there is no real camera performing the path simulated by the studied view synthesis methods. The absence of the ground truth led to selecting a test method without a reference sequence (real camera path). These subjective test methods are not accurate in discriminating between view synthesis methods, here HDSB, VVS, and MUSE. A Test 2 was defined to directly compare the synthesis methods and assess their perceived quality difference on a 7-point comparison scale to overcome this drawback.

The process of a subjective test according to the principles of the ACR method is as follows:

Each video sequence to be evaluated is shown for 10 seconds on the display screen (Fig. 7.14). Before the video sequence, we show for 2 seconds a grey screen indicating the number of the sequence to be evaluated. At the end of the video sequence, the observer must give his opinion on the perceived visual quality by means of a rating scale comprising the following labels: *poor*, *mediocre*, *satisfying*, *good* or *excellent* (Fig. 7.15). To this end, the user can move a cursor over the category corresponding to the perceived quality level. Once the vote is validated, the following video sequence is presented on the screen, preceded by a 2 seconds grey screen indicating the sequence number, followed by the voting interface. Fig. shows an example of transitions created by the three different methods (HDSB, VVS and MUSE) on *PoznanFencing* sequence.

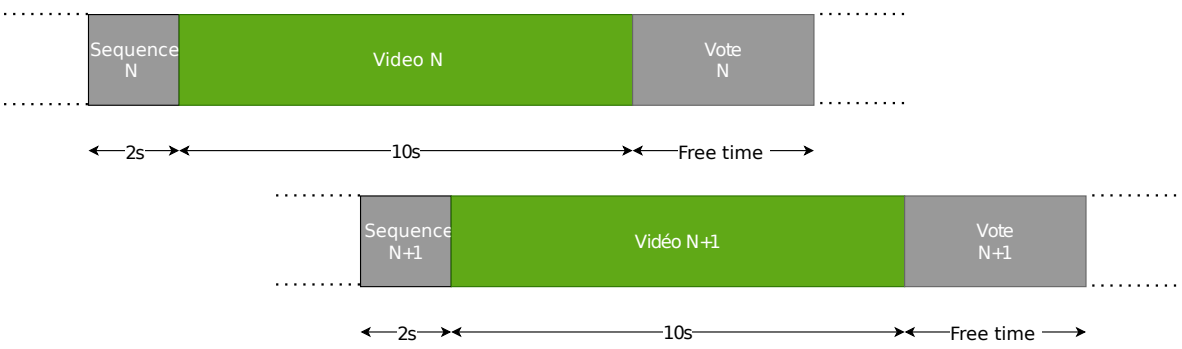


Fig. 7.14 Sequencing specific to the ACR method

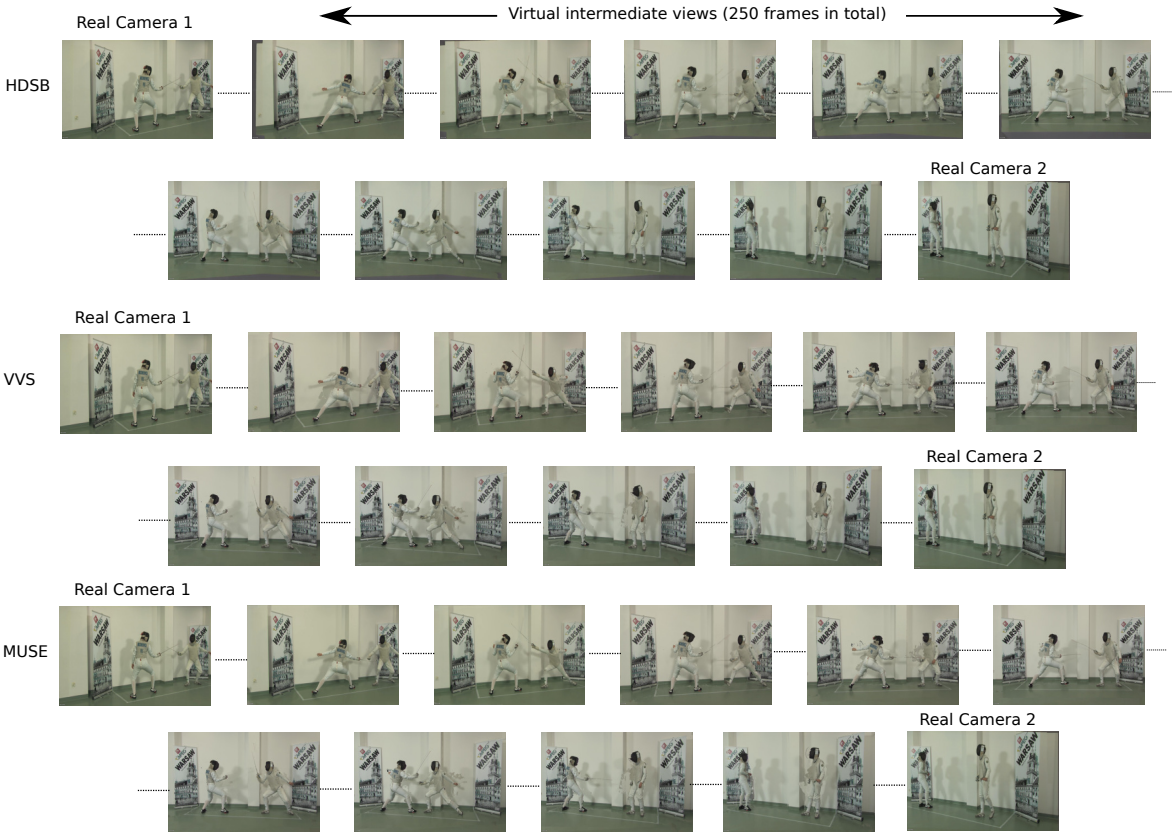


Fig. 7.13 Illustration of video transitions created by the three view synthesis methods (HDSB, VVS and MUSE) on *PoznanFencing* video sequence

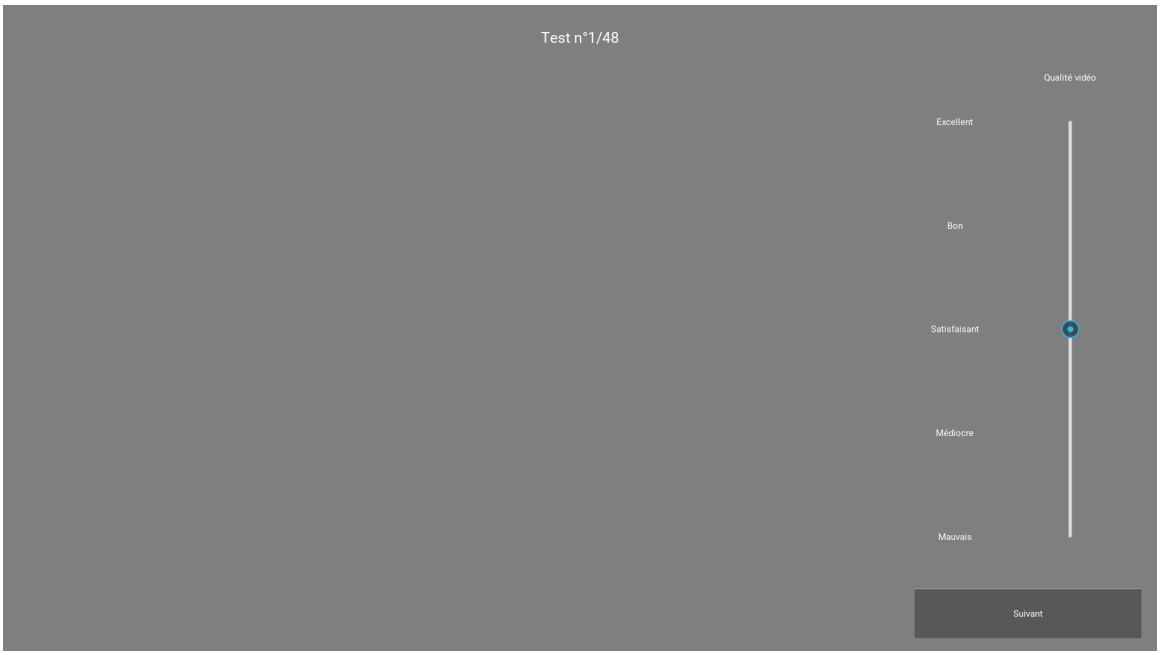


Fig. 7.15 User interface for the evaluation of video sequences with the ACR method.

Scene	MOS
PandemoniumRig1	1.4
PoznanFencing	1.7
PoznanCarpark	2.3
PoznanStreet	3.2
TechnicolorPainter	3.9
Adventure	2.8
OrangeShaman	3.2
VikingVillage	3.2

Table 7.3 MOS or average quality score obtained for each scene.

The 24 test transitions used are those extracted from the 8 different sequences described in Sec. 7.3.2, and each of of these transitions is available in 3 versions (VVS, HDSB, and MUSE) For each observer, the order of presentation of the sequences is different, and each sequence is repeated once as specified in the ITU-T recommendation P.913 (cf. [141]). Before starting the test, the protocol and the objectives of the test are presented to each participant in a test instruction sheet (see Appendix B.1).

7.4.2.2 Results and discussion

This section details and analyzes the results in terms of MOS.

MOS vs Scene

Tab. 7.3 and Fig. 7.16 illustrate the MOS obtained by averaging the inter-observer and inter-method scores (HDSB, VVS and MUSE) per scene.

We first observe that the results are strongly related to the scenes. The quality of PandemoniumRig1 is perceived as *bad*, PoznanFencing and PoznanCarpark as *mediocre*, Adventure, PoznanStreet, OrangeShaman and VikingVillage as *satisfying*, and Technicolor as *good*. In yellow on the figure, the results obtained with the synthetic scenes are globally better than those obtained with the natural scenes. The reliance of the results on the scenes is confirmed by the results of ANalysis Of VAriance (ANOVA) shown in Tab. 7.5.

An ANOVA test is a method to see if an experiment's results are meaningful. In other words, it helps to figure out if we need to reject the null hypothesis or accept the alternate hypothesis.

The *p-values* shown in the column ($PR(>F)$ are below the 5% level of significance and show that the distribution of the votes by content are significantly different.

The difference in the perceived quality between the scenes can be explained by visual artifacts that change from one scene to another. Concerning the natural scenes, the most

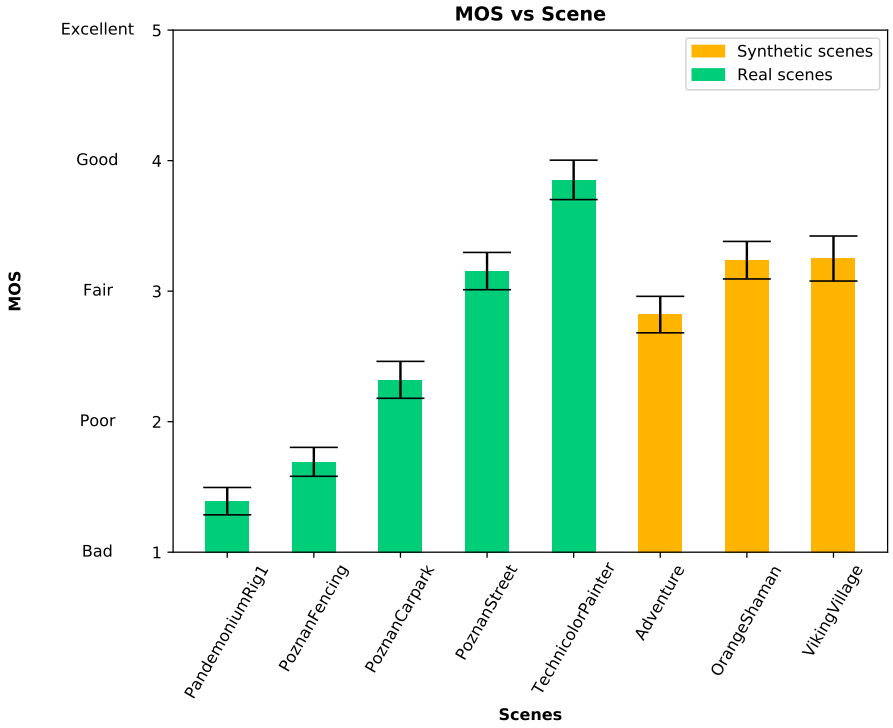


Fig. 7.16 MOS or average quality scores and 95% confidence intervals obtained for each scene.

damaging artifacts are related to an imperfect estimation of the scene's geometry or, in the case of DIBR methods, to a problem of the estimation of the depth maps associated with each reference view. These artifacts are even more important when the gap between the reference view and the virtual view is more important. Tab. 7.1 lists the disparity (cf. Sec. 2.3) amplitudes associated to each scene.

We observe that the amplitudes [2] of disparity are more critical in the PandemoniumRig1 (320 pixels) and PoznanFencing (130 pixels) scenes with the lowest perceived qualities. TechnicolorPainter is the highest-rated and has the lowest range of disparity (31 pixels).

Concerning the synthetic scenes, the depth maps are ideal and the disparity amplitude can not explain the difference in the perceived quality of the scenes sequences.

MOS vs Methods

Tab. 7.4 and Fig. 7.17 illustrate the MOS obtained by computing the average of the inter-observer and inter-scene scores for each methods (HDSB, VVS and MUSE).

First, the result shows that the HDSB, VVS, and MUSE methods differ in the rendered quality. The results obtained by MUSE are better than those obtained by the VVS and HDSB methods. For MUSE, the quality performance level is globally considered as satisfying.

[2]The disparity amplitude is the difference between the maximum and minimum disparity.

Method	MOS
HDSB	2.4
MUSE	3.1
VVS	2.7

Table 7.4 MOS or average quality scores obtained for each view synthesis method.

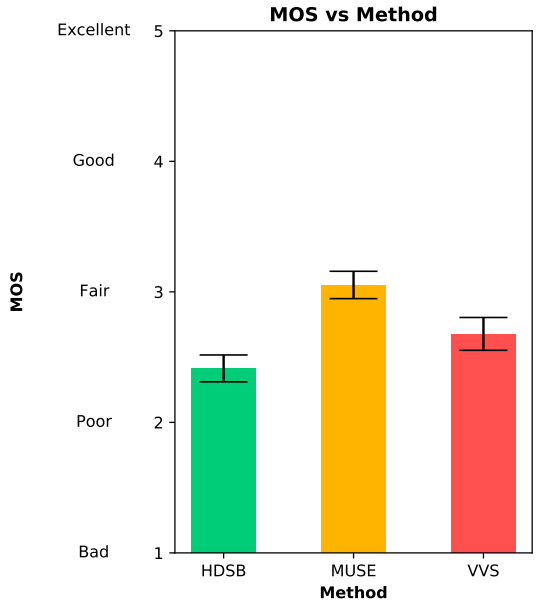


Fig. 7.17 MOS or mean quality scores and 95% confidence intervals obtained for each view synthesis method.

The ANOVA, shown in Tab. 7.5, highlights significant differences between the studied methods. The values of the *p-values* for the factors C(*methods*), specified in the columns ($PR(>F)$), are way below the level of 5% of significance.

MOS vs (Scenes , Methods)

We just noticed that the MUSE method performs better overall than the HDSB method, but it is necessary to analyze the results for each content in detail. Tab. 7.6 and Fig. 7.18 illustrate the MOS obtained by calculating the average of inter-observer scores for each scene and each method (HDSB, VVS and MUSE).

We observe that MUSE obtains the highest quality rating for all scenes except for the VikingVillage scene, for which the VVS method performs favorably. However, to assess the significance of the differences between MOS, we need to conduct a more profound analysis. First, the observer rejection procedure proposed in Appendix A of Recommendation [141]

index	sum_{sq}	df	F	PR(>F)
C(method)	43	2	67	$4.6 \cdot 10^{-27}$
C(scene)	390	7	170	$2.7 \cdot 10^{-14}$
C(method):C(scene)	39	14	8.7	$2.6 \cdot 10{-}17$
Residual	190	600	-	-

Table 7.5 Results of the ANOVA for all the methods (HDSB, VVS and MUSE).

was performed. The results from a single observer were discarded from the MOS calculations. From the original 27 observers, 26 voters are considered. Each observer's bias voting was estimated, and the votes of each user were corrected by this bias as recommended in section 12.4 of the recommendation [141]. Finally, Student's tests were performed, and the results of the *p-values* are shown in Tab. 7.7.

These results show that:

- MUSE vs VVS: MUSE performs significantly better for Adventure, PandemoniumRig1, PoznanCarpark, PoznanFencing and TechnicolorPainter contents. The MOS differences for OrangeShaman and PoznanStreet contents are not significant. VVS performs favorably for VikingVillage sequence.
- MUSE vs HDSB: MUSE has better performance for all contents. The difference in MOS observed for the PoznanFencing scene is not significant.
- HDSB vs VVS: the difference in MOS for the Adventure, PandemoniumRig1, PoznanCarpark, PoznanStreet and TechnicolorPainter scenes are not significant. While for the OrangeShaman and VikingVillage scenes, VVS performs favorably and significantly, HDSB performs well for PoznanFencing.

Conclusions

The main objective of this subjective test is to compare the visual rendering quality of the studied view synthesis methods (VVS, HDSB, and MUSE) to determine if we improve in MUSE over the other two methods. The analysis of the results of this first subjective test shows that:

- MUSE visual rendering quality is better than the VVS visual rendering quality for Adventure, PandemoniumRig1, PoznanCarpark, PoznanFencing, and TechnicolorPainter contents. The rendering quality of both methods is equivalent fro OrangeShaman and PoznanStreet contents. VVS performs favorably for the VIkingVillage sequence. For this scene, we could observe that prominent, annoying, and time-varying artifacts are predominant in the areas where occlusion alternate with disocclusion zones. We can

Method	Scene	MOS
HDSB	Adventure	2.6
	OrangeShaman	2.8
	PandemoniumRig1	1.1
	PoznanCarpark	2.1
	PoznanFencing	1.8
	PoznanStreet	2.8
	TechnicolorPainter	3.6
	VikingVillage	2.4
MUSE	Adventure	3.2
	OrangeShaman	3.5
	PandemoniumRig1	1.8
	PoznanCarpark	2.7
	PoznanFencing	2.2
	PoznanStreet	3.4
	TechnicolorPainter	4.2
	VikingVillage	3.4
VVS	Adventure	2.6
	OrangeShaman	3.4
	PandemoniumRig1	1.2
	PoznanCarpark	2.1
	PoznanFencing	1.1
	PoznanStreet	3.3
	TechnicolorPainter	3.8
	VikingVillage	3.9

Table 7.6 MOS or average quality scores obtained for each scene and each view synthesis method.

reasonably hypothesize that the temporal smoothing provided by VVS would bring again compared to the pure spatial processing performed by MUSE. Thus, the MUSE method brings a global gain compared to the VVS method. The taking into account of the temporal neighborhood represents an improvement perspective of the MUSE method.

- The quality of the visual rendering of MUSE is better than that of the visual rendering of HDSB for all the contents except for the PoznanFencing scene for which it is equivalent, so the improvement brought by the new approach compared to the first one is obvious.

- HDSB gives equivalent results to VVS for the Adventure, PandemoniumRig1, Poz-

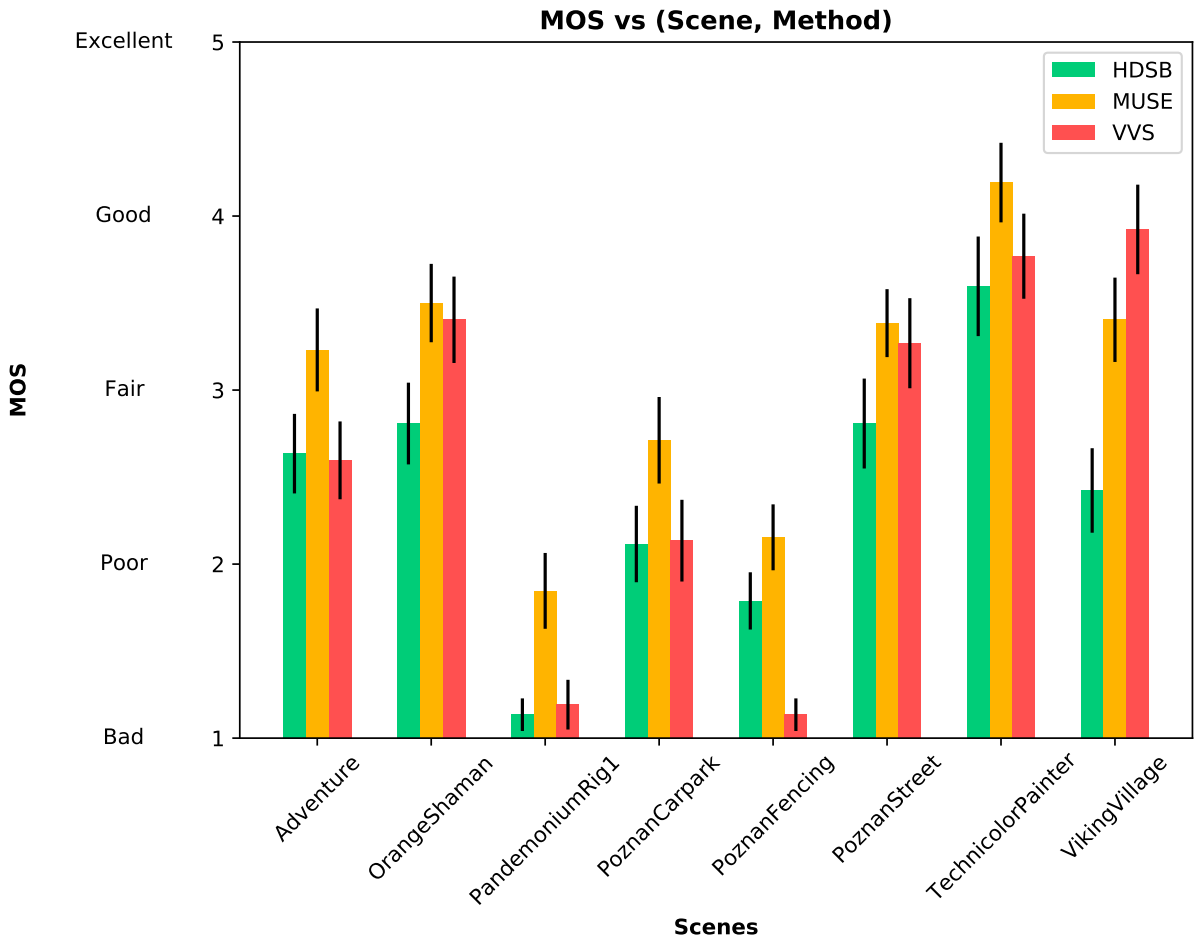


Fig. 7.18 MOS or average quality scores and 95% confidence intervals obtained for each scene and each view synthesis method.

nanCarpark, PoznanStreet, and TechnicolorPainter scenes. For OrangeShaman and VikingVillage scenes, VVS performs favorably and significantly. HDSB performs favorably and significantly compared to VVS for the PoznanFencing scene.

As a corollary, a second objective was to determine if the level of rendered quality depends on the scenes' content or the configuration of the shooting rigs.

The analysis of the test results shows that the quality of the view synthesis is strongly dependent on the content. The visible artifacts are related to errors in the depth maps, which are more critical when the disparity amplitude is more significant. As well as the camera focal lengths, the spacing, and the object relative distances in the scene, affect the quality of the synthesized views. The synthetic contents for which ideal depth maps have been used escape this rule. A more detailed study of the link between the configuration of the rig and the quality of the synthesized views would allow us to confirm these hypotheses and is added to the manuscript perspectives.

Scene	MUSEvsVVS	MUSEvsHDSB	HDSBvsVVS
Adventure	0.00023	0.00018	0.81
OrangeShaman	0.6	0.00012	0.002
PandemoniumRig1	$4.6 \cdot 10^{-05}$	$9.2 \cdot 10^{-06}$	0.61
PoznanCarpark	0.00035	0.00024	0.9
PoznanFencing	$1.7 \cdot 10^{-08}$	0.041	$3 \cdot 10^{-05}$
PoznanStreet	0.45	$4.8 \cdot 10^{-05}$	0.0087
TechnicolorPainter	0.0043	0.00069	0.29
VikingVillage	0.0018	$4.8 \cdot 10^{-07}$	$8.2 \cdot 10^{-12}$

Table 7.7 Results of the Student test associated with the average quality scores obtained.

7.4.3 Test 2: Quantification of the differences in the quality of the image synthesis methods.

The objective of the second test is an extension of the previous test. Here, the objective is to quantify the difference in visual quality between the different methods of view synthesis (HDSB, VVS, and MUSE). This test is more accurate than the previous one in estimating the differences in rendering between the synthesis methods from a subjective point of view. Therefore, for each scene, it is possible to determine if MUSE offers a better visual rendering than the two other solutions and quantify this difference on an appropriate perceptual scale. However, this method does not determine the level of perceived visual quality for each measurement point (a view synthesis method associated with a scene), as was the first subjective test case. The two subjective tests are, thus, complementary.

7.4.3.1 Assessment methodology

The test method used proposes a test protocol similar to that of the DSCQS (Double Stimulus Continuous Quality Scale) method described in the Recommendation ITU-RBT.500-14 [137]. It requires prior scheduling of the sequences to be tested because the vies synthesis methods will be compared in pairs (video A and video B). Therefore, for a given scene, three pairs will be compared: (MUSE vs VVS), (MUSE, HDSB), and (VVS, HDSB). In the subjective test, each pair is judge twice by each participant, and the order of each pair is reversed. For each comparison, the order is as follows (see Fig. 7.19):

- First display of the pair “Video A/ Video B”

    1. display of a grey screen showing the letter A (duration of 1 sec.)
    2. display the content of the video A (duration 10 sec.)

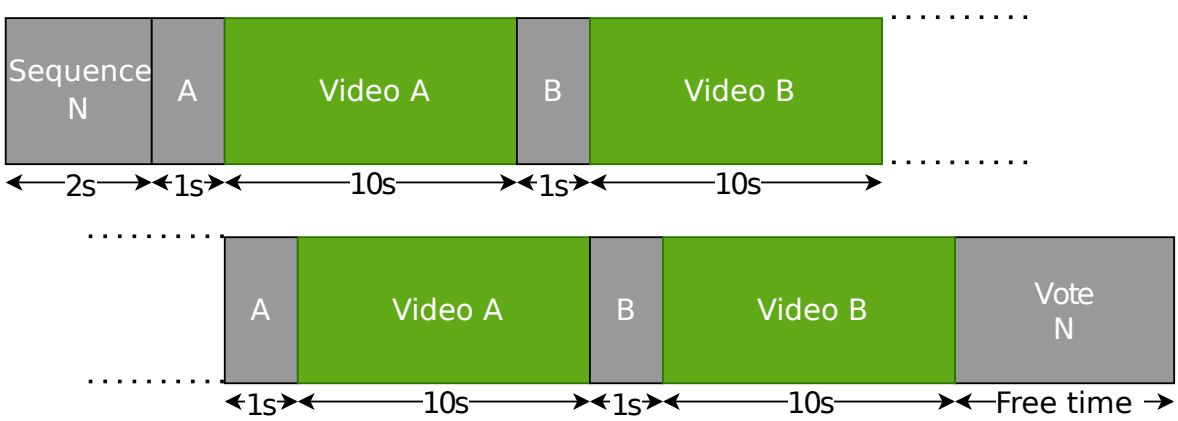


Fig. 7.19 Sequencing specific to the comparison method

3. display of a grey screen showing the letter B (duration 1 sec.)
4. display the content of the video B (duration 10 sec.)

- Second display of the pair "Video A/ Video B"

1. display of a grey screen showing the letter A (duration of 1 sec.)
2. display the content of the video A (duration 10 sec.)
3. display of a grey screen showing the letter B (duration 1 sec.)
4. display the content of the video B (duration 10 sec.)

- display of the voting interface integrating the comparison scale

Due to the double showing of each pair, participants are then asked to report their opinion on the perceived difference quality between each video via a 7-point comparison scale (cf. Fig. 7.20). Each participant is thus, asked to judge whether the quality of video B is "Much worse," "Less good," "Slightly worse," "Equivalent," "Slightly better," "Better," or "Much better" compared to the quality of video A. The choice is then made by moving a cursor to the appropriate category on the comparison scale. Once the vote is validated, the next pair is presented to the participant.

Before starting the test, the protocol and the objectives of the test are presented to each participant in a test instruction sheet (see Appendix B.2). From the collected data, we calculate the mean and confidence interval of each comparison. We assign a score to each category (from -3 for the "Much worse" difference to $+3$ for the "Much better" difference). Comparing the differences between the view synthesis methods can thus, be made on a scene-by-scene basis or across all scenes.

Fig. 7.20 User interface for comparing visual differences between view synthesis methods

7.4.3.2 Results and discussion

In the following sections, we analyze the results in terms of MOS or average quality scores compared between each pair of methods.

MOS vs Methods

Tab. 7.8 and Fig. 7.21 illustrate the inter-observer and inter-scene MOS for each pair of the compared methods.

Compared methods	MOS
HDSB vs VVS	-0.335
MUSE vs HDSB	1.157
MUSE vs VVS	0.780

Table 7.8 MOS for each pair of compared methods .

We observe that:

- the quality results of HDSB are relatively equivalent to those of VVS
- the quality results of MUSE are slightly better than those of HDSB
- the quality results of MUSE are slightly better than those of VVS

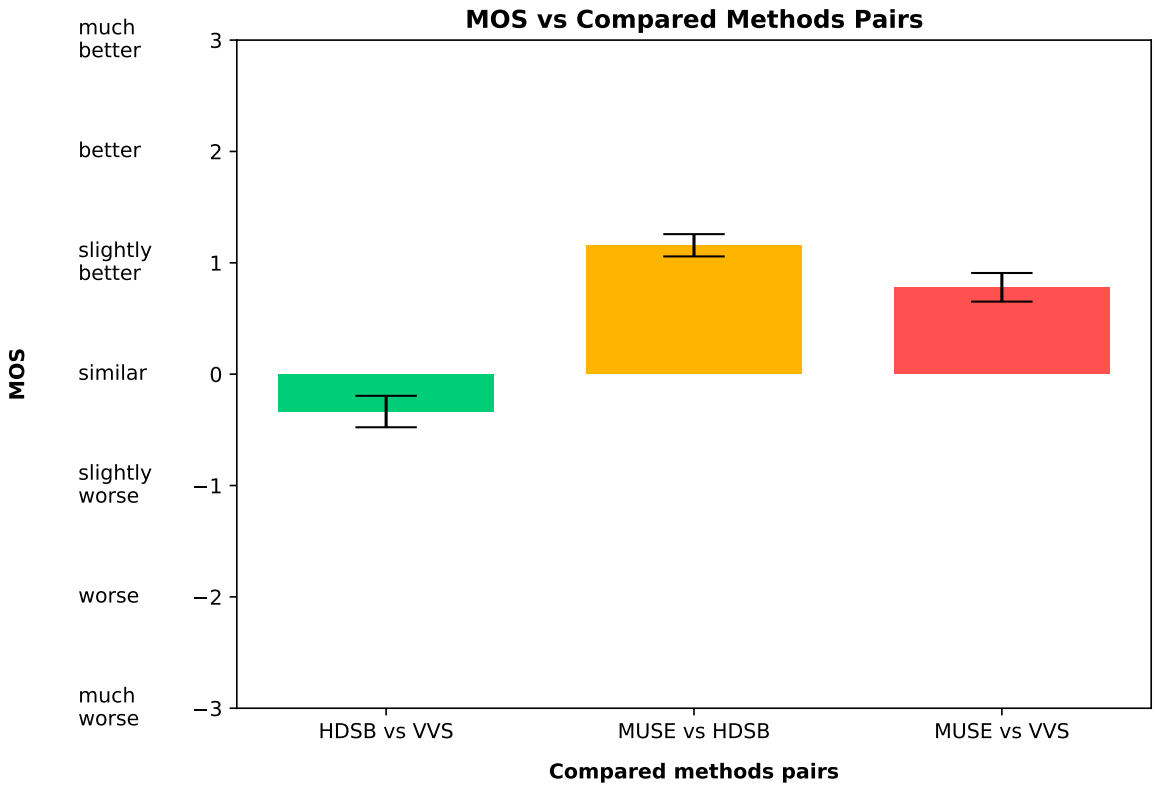


Fig. 7.21 MOS and 95% confidence intervals for each pair of methods compared.

MOS vs (Scenes, Methods)

Tab. 7.9 and Fig. 7.22 illustrate the inter-observer MOS for each scene and pair of compared methods

We observe that:

- while HDSB performs favorably compared to VVS for Poznanfencing scene, it produces equal or worse results than VVS.
- MUSE is a real improvement compared to HDSB. If for the PoznanStreet and TechnicolorPainter contents, the results are very slightly better, they are slightly better or even better for the other contents.
- MUSE performs favorably compared to VVS for all contents except the VikingVillage scene. The contributions are most significant for the real scenes PandemoniumRig1 and PoznanFencing. The results for the OrangeShaman and PoznanStreet sequences are relatively comparable.

Compared methods	Scene	MOS
HDSB vs VVS	Adventure	-0.74
	OrangeShaman	-0.98
	PandemoniumRig1	-0.67
	PoznanCarpark	0.28
	PoznanFencing	1.60
	PoznanStreet	-0.54
	TechnicolorPainter	0.26
	VikingVillage	-1.90
MUSE vs HDSB	Adventure	1.50
	OrangeShaman	1.20
	PandemoniumRig1	2.00
	PoznanCarpark	0.96
	PoznanFencing	1.30
	PoznanStreet	0.48
	TechnicolorPainter	0.44
	VikingVillage	1.40
MUSE vs VVS	Adventure	0.87
	OrangeShaman	-0.04
	PandemoniumRig1	1.90
	PoznanCarpark	1.10
	PoznanFencing	2.30
	PoznanStreet	0.20
	TechnicolorPainter	0.78
	VikingVillage	-0.83

Table 7.9 Inter-user MOS for each scene and pair of compared methods.

Conclusions

This test allowed us to quantify the difference in visual quality between the different view synthesis algorithms (HDSB, VVS and MUSE). The analysis of the results corroborates the conclusions of the first test and shows that:

- MUSE performs favorably compared to VVS for all the contents except for the VikingVillage scene The contributions are most significant for the real scenes PandemoniumRig1 and PoznanFencing. The results for the OrangeShaman and PoznanStreet sequences are relatively comparable

- MUSE is a real improvement over HDSB. The results for PoznanStreet and TechnicolorPainter are very slightly better, while the results for the other contents are slightly

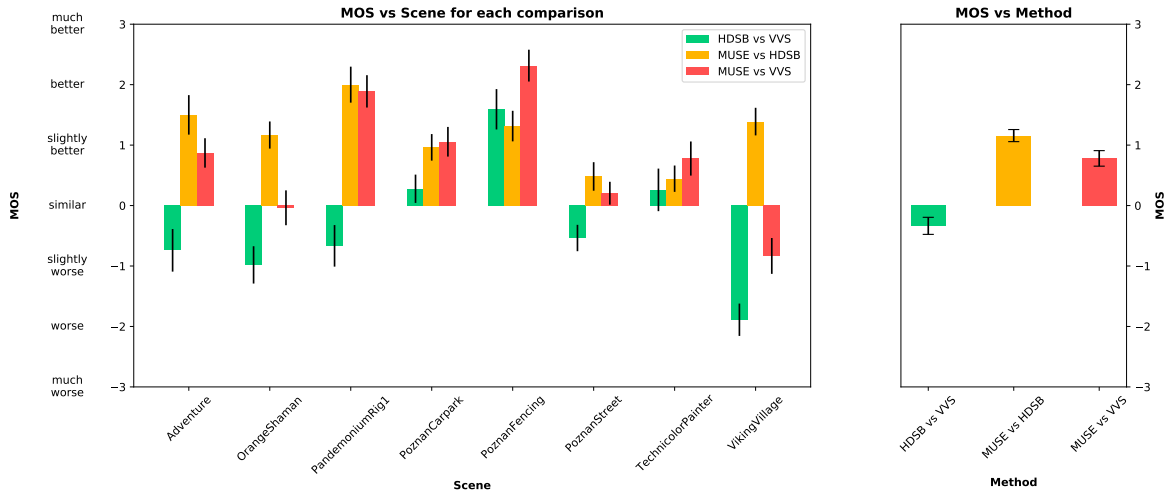


Fig. 7.22 MOS and 95% inter-user confidence intervals for each scene and pair of compared methods

better or even better.

- HDSB produces equivalent results, slightly worse or worse than those produced by VVS, except for the PoznanFencing scene, for which the visual quality of the HDSB method is judged better.

Therefore, MUSE allows a visual quality improvement over the state-of-the-art view synthesis methods proposed by VVS. This improvement is judged slightly better on all the contents.

These two tests allowed us to know what is the best view synthesis method in terms of visual quality in an actual use case, which is the video transition between two points of view in the same scene. As a result, we also discovered the relation between the type of the content (type of textures, quality of the depth maps, baseline between the real and the virtual viewpoint...) and the view synthesis method performance. However, the question then arises as to whether these view-synthesis-based transitions are preferred over the classical transition methods when realizing a transition between two real points of view in the same scene. Indeed, in addition to the view synthesis, different “transition” modes can be chosen to move from the point of view A to another B, such as Cut, Blend (or crossfade), Morphing, etc... To chose the preferred transition mode, it is primordial to answer the following questions:

- What is the preferred mode of transition between two points of view of the same scene?
- Does the quality of the view synthesis affect the preferred mode of transition?

In order to provide answers to these questions, a third subjective test was performed. For this test, a ranking by order of preference of different transition modes was performed by a panel of 26 participants for different selected scenes.

In the following section, we describe in detail these subjective tests, and we discuss their main results.

7.4.4 Test 3: Evaluation of the preference of different types of inter-view transitions

A *video transition* is a technique used in live or post-production to change the point of view in a given scene. It is usual in television when the director switches from one camera to another, such as during a soccer game, and in film production when the dialogue is framed in a field/counter-field mode.

There are several techniques to realize the transition from one point of view to another in the same scene. Therefore, a transition can either be instantaneous or taking place over a period of time associated with more advanced transition effects. The most common effects:

- The “Cut”, where the connection between the two points of view of the same scene is made instantaneously, to the nearest frame. This type of transition is the most popular.

- The “Blend” (cross-fade), where the transition effect consists in progressively fading the first point of view to make the second one appear progressively by an adequate mix of the two views. This transition can be of variable duration depending on the required artistic effect.

- The “Blur”, where the switch from the first point of view to the other is achieved by a progressive and then digressive blur.

- The “Morphing”, it consists in deforming the image of the first point of view in order to generate the image of the second point of view by an Ad-hoc geometrical deformation. Morphing is usually used in film industry.

Several factors can affect the choice of a transition effect and the perception of its rendering from a subjective point of view. These factors include artistic purpose, duration, and dissimilarity of the two points of view.

This subjective test aims to determine whether there is a user preference for different types of transition effects. To this end, the study focuses on comparing a transition performed with a view synthesis method (here MUSE) with the most common effects mentioned above. The preference will be evaluated for different scenes whose characteristics are detailed in Sec. 7.3.2. These are the same scenes used in the previous two subjective tests.

7.4.4.1 Preparing of the test sequences

From each of the 8 selected scenes, we created five sequences of 10 seconds duration from the recorded videos of the two cameras in the scene. Concerning the transitions with temporal effect, the duration of the effect is set to two seconds (cf. Fig. 7.23).

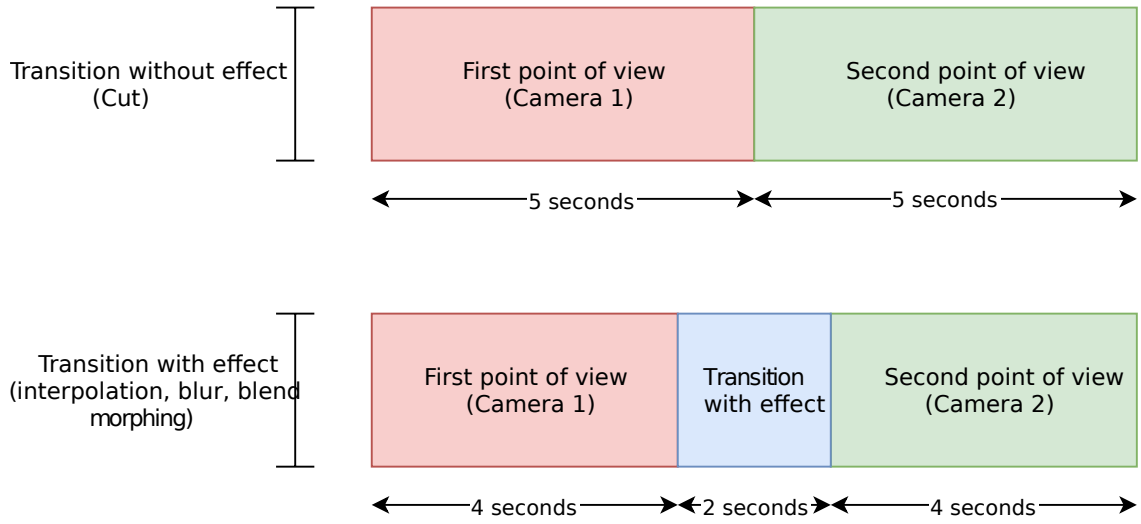


Fig. 7.23 Transition effects evaluated for each scene in the subjective test

The post-production software "DaVinci Resolve" [125] from Blackmagic Design was used to generate the different transitions, with our without effects. The two-camera viewpoints integration in the timeline allowed the creation of the sequences with the effects proposed by the software. Concerning the view-synthesis-based transition, the MUSE sequences used in tests 1 and 2 were reused. In this specific case, the editing was made as follows: 4 seconds for the first point of view, 2 seconds for the sequence with the view synthesis (the virtual transition created by generating 200 intermediate virtual frames), and finally 4 seconds for the second point view. The format of the sequences used for the two points of view was the same as for the classic transitions integrated into the software. Therefore, for each scene, 5 types of transition were created: View synthesis (MUSE), Cut, Blend, Blur, and Morphing. Fig. 7.24 illustrates an example of the 5 types of transitions created on the *Adventure* sequence.

7.4.4.2 Assessment methodology

For a given scene, the chosen test method consists of ranking by order of preference the different proposed transitions. To this end, a specific graphical interface was developed (cf. Fig. 7.25). The user must then visualize video sequences identified by letters going from A to E. By clicking on a letter, he/she starts the playback of a video sequence associated with a type of transition, chosen among the five possible ones. After finishing the video, the

observer must place the letter in front of an integer indicating an order of preference (1,2,3,4 or 5) using the “up” and “down” buttons, 1 being the preferred ranking. Each sequence associated with a letter can be played several times, and the ranking can be reviewed as many times as necessary until the final choice is made. Once the ranking is validated, the tester is asked to rank in order of preference the five sequences associated with a new scene. The transitions related to the buttons A to E are randomly assigned by scene and by participants.

Fig. 7.25 Voting interface used for preference testing of transitions between views

Before starting the test, the protocol and the objectives of the test are presented to each participant in a test instruction sheet (see Appendix B.3). At the end of the test, each type of transition (View synthesis, Cut, Blend, Blur, and Morphing) is associated with an order of preference (number from 1 to 5) by scene and by the observer.

From the generated data, a statistical analysis of the results is performed. The objective is to determine if there is a pronounced user preference for a particular transition effect type among the five studied. After computing the average ranking of the transition effects, a Friedman test is performed to determine if the transitions have significantly different scores. If so, a Conover’s posthoc test is done to compare the preferences between the two transitions.

7.4.4.3 Results and discussion

In the next sections, we analyze the results in terms of the average ranking of preference votes for the different types of transition.

Average ranking vs. transition type

Tab. 7.10 and Fig.7.26 illustrate the results obtained in terms of average inter-observer and inter-scene ranking by transition type.

Transition type	Averaged ranking
Blur	4.2
Cut	2.5
Fading/blending	2.7
Morphing	3.9
View Synthesis/Muse	1.6

Table 7.10 Average ranking by transition type.

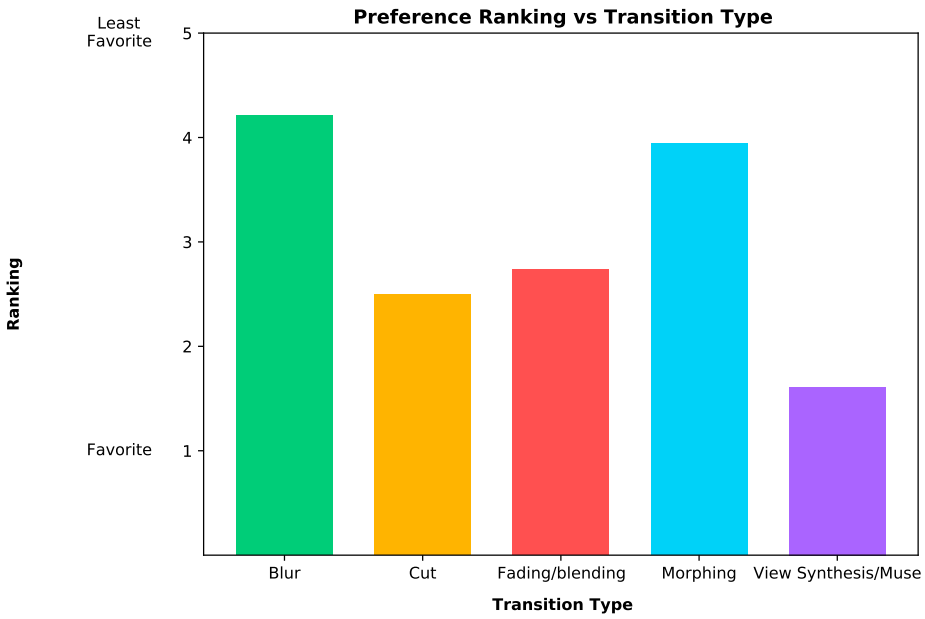


Fig. 7.26 Average ranking by transition type.

We observe that view synthesis is the preferred type of transition, ahead of simple transitions with, in order of preference: Cut, Blend, Blur and Morph.

Average ranking vs (Scenes, Transition type)

Tab. 7.11 and Fig. 7.27 illustrate the results of the average inter-observer rankings by transition type and by scene.

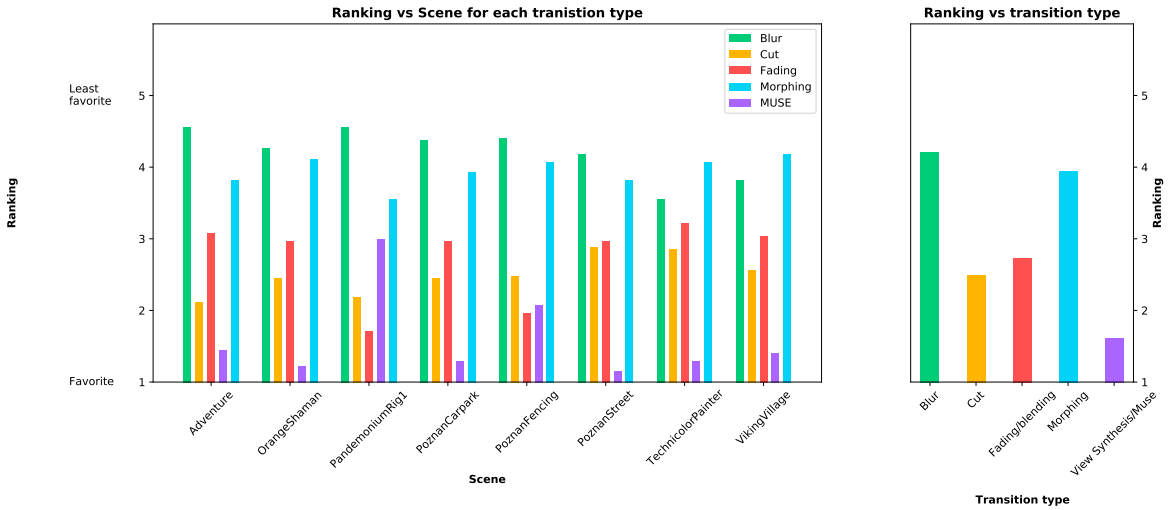


Fig. 7.27 Average inter-observer rankings by transition type and scene

.

Tab. 7.12 shows the results of the Friedman test by scene. We observe that the inter-transition differences are significant for all scenes (p-$value$<0.05). Conover's tests must thus, be performed to evaluate the significance of the differences between the transition effects by pair and by scene.

Scene	p_{value}
PoznanFencing	$9.1 \cdot 10^{-12}$
VikingVillage	$1.5 \cdot 10^{-10}$
PoznanCarpark	$3.7 \cdot 10^{-13}$
OrangeShaman	$6.2 \cdot 10^{-14}$
PandemoniumRig1	$3.6 \cdot 10^{-11}$
TechnicolorPainter	$9.8 \cdot 10^{-10}$
PoznanStreet	$3.6 \cdot 10^{-12}$
Adventure	$5.9 \cdot 10^{-14}$

Table 7.12 Results of the Friedman test by scene.

Tab. 7.13 shows the results of Conover's tests of parity comparison of the transitions for each scene. They reveal the importance in the classification.

We observe that:

- For the Adventure scene, the MUSE method is the preferred transition mode. However the value of the Conover tests obtained for the pair of transitions (MUSE,CUT) (cf.

Tab 7.13 (a)) is 0.27. In this case, the difference in the ranking between the Synthesis and the Cut is not thus significant (i.e *p-value*>0.05). The two types of transition share thus the first place in the ranking.

- For the OrangeShaman scene, the MUSE method is the preferred transition mode. The *p-value* of the Conover test obtained for the pair of methods (MUSE, Cut) (cf. Tab. 7.13 (b)) is 0.023. The difference in the ranking seems significant (i.e *p-value*<0.05). The Cut and the Blend occupy thus, the second place in the ranking behind the MUSE method.

- For the PandemoniumRig1 scene, the Blend transition mode is preferred over the MUSE method. The *p-value* of the Conover test obtained for the pair of methods (Blend, Cut) (cf. Tab. 7.13 (c)) is 0.041. The difference does not seem significant (i.e *p-value*>0.05) and the Cut shares the first place in the ranking. For this scene, the MUSE method occupies the second place in the ranking.

- For the PoznanCarpark scene, MUSE is the preferred transition mode. The *p-value* of the Conover test obtained for the pair of methods (MUSE,Cut) (cf. Tab. 7.13 (d)) is 0.037. he difference in the ranking seems significant (i.e *p-value*<0.05), and the Cut takes thus, the second place in the ranking.

- For the PoznanFencing scene, the Blend method is the preferred transition mode. The *p-value* of the Conover test obtained for the pair of methods (Blend, MUSE) (cf. Tab. 7.13 (e)) is 1. The difference does not seem significant (i.e *p-value*>0.05) and MUSE shares thus the first place in the ranking. The first place is also shared with the Cut method (*p-value*>0.94).

- For the PoznanStreet scene, MUSE is the preferred transition mode. The *p-value* of the Conover test obtaned for the pair of methods (MUSE, Cut) (cf. Tab. 7.13 (f)) is 0.00079. The difference ranking is therefore very significant (i.e *p-value*< 0.05) and the Cut is thus, ranked second.

- For the TechnicolorPainter scene, MUSE is the preferred transition mode. The *p-value* of the Conover test obtained for the pair of methods (MUSE, Cut) (cf. Tab. 7.13 (g)) is 0.036. The difference in ranking is therefore significant (i.e *p-value*< 0.05) and the Cut is thus, ranked second.

- For the VikingVillage scene, MUSE is the preferred transition mode. The *p-value* of the Conover test obtained for the pair of methods (MUSE, Cut) (cf. Tab. 7.13 (h)) is 0.047. The difference in ranking is therefore significant (i.e *p-value*< 0.05) and the Cut is thus, ranked second.

It is interesting to note that the visual quality rendering of MUSE could affect the preference test results. Test 1 showed that the PandemoniumRig1 scene performs the worst in the view synthesis quality assessment by being rated as "mediocre." Considering the results of MUSE preference ranking in the other scenes, it is likely that the quality of the rendered view synthesis method has a negative impact on the order of preference classification for the Pandemoniumrig1 scene.

Therefore, MUSE takes first place in ranking the preferred transitions for all the scenes (of which five times it was solo), except for PandemoniumRig1, for which only the Cut transition mode beats it. In comparison, the methods Cut and Blend only come out on top twice each. These two methods seem as an alternative when the quality of the view synthesis is judged to be mediocre. The Morphing and the Blur methods are not to be preferred in the context of this study.

Conclusions

The goal of this subjective test was to determine whether there is a preference for different types of transition effects. For this purpose, the study aimed to compare a transition performed with a view synthesis method MUSE with the most common transition effects mentioned above (Cut, Blend, Blur, or Morphing).

The results of this study showed that:

- MUSE takes the first place in the ranking of the preferred transitions modes in all the scene except for PandemoniumRig1 which holds the second place. This shows that even if the visual results present imperfections, they are preferred over the simple transitions such as the Cut and the Blend.
- Cut and Blend are considered as alternatives when the quality of the view synthesis is judged to be mediocre. The Morphing and the Blur are not preferred in our study.

7.5 Conclusion and discussion

In this Chapter, we proposed a new view synthesis method MUSE that improves over our previous work (HDSB) and the state-of-the art methods such as VVS. The purpose of our method is to generate virtual intermediate views using real views of the same scene.

We qualitatively evaluate our method by creating two subjective tests (test 1 and 2) that compare MUSE performances with two other view synthesis methods of the state-of-the-art (HDSB and VVS). These tests were performed on an actual view synthesis use-case which is a video transition between two points of view in the same scene. Our current research regarding the view synthesis technique aims to exploit the temporal information to improve

the hole filling procedure and impose temporal consistency among neighboring frames as an additional constraint at training and inference times.

Concerning the first test, the three methods MUSE, HDSB and VVS were evaluated for different selected scenes. The main objective of this test is the visual rendering quality of the view synthesis methods to determine MUSE if improves over the two other methods. As a consequence, a second objective is to determine if the level of quality rendered depends on the scene content or on the configuration of the filming rigs. The analysis of the results shows that:

- MUSE brings a global gain compared to the state-of-the-art method VVS.
- One perspective to improve MUSE performance is to exploit the temporal information to improve the hole filling procedure and impose temporal consistency among neighboring frames as additional constraint training and inference times.
- The first two subjective tests studied in this work showed that we improve in MUSE over our previous work HDSB.
- The configuration of the rigs (focal lengths of the cameras, baselines and relative distance of the objects in the scene) has an impact on the quality of the synthesized views. A more deep study of the relation between the rig configuration and the quality of the synthesize view, is added to our perspective in this work. Such a study will give us a better understanding on how to design a camera rig that meet a specific quality requirements.

Concerning the second test, MUSE, HDSB, and VVS were compared, two by two, in terms of visual quality. This test quantifies the difference in visual quality between the different view synthesis algorithms (MUSE, HDSB, and VVS). The analysis of the results confirms the previous test conclusions showing that: MUSE improves the visual quality of the state-of-the-art view synthesis method proposed by VVS and HDSB. In general terms, MUSE is judged slightly better in all contents.

These two tests led us to a third test that determines whether or not, in such use-case, the view-synthesis-based transition is, anyhow, the transition mode preferred by the viewer. Therefore, we compare this transition mode to other primary and widespread transition modes such as the Cut, Blend, Blur, and Morphing. The results show that:

- MUSE takes first place in the ranking of the preferred transition as it is considered the most natural transition in all the scenes except for PandemoniumRig1; it is the second-best transition. This test shows that while visual results of the view synthesis have imperfections, this transition mode is preferred over the simple ones like the Cut or the Blend.

- However, the Cut and Blend transitions seem to be alternatives when the view synthesis quality is degraded. The Morphing and the Blur are the least preferred transition methods in our context.

Fig. 7.24 Transition effects generated for one sequence *Adventure*

Transition type	Scene	MOS
Blur	Adventure	4.6
	OrangeShaman	4.3
	PandemoniumRig1	4.6
	PoznanCarpark	4.4
	PoznanFencing	4.4
	PoznanStreet	4.2
	TechnicolorPainter	3.6
	VikingVillage	3.8
Cut	Adventure	2.1
	OrangeShaman	2.4
	PandemoniumRig1	2.2
	PoznanCarpark	2.4
	PoznanFencing	2.5
	PoznanStreet	2.9
	TechnicolorPainter	2.9
	VikingVillage	2.6
Fading/blending	Adventure	3.1
	OrangeShaman	3
	PandemoniumRig1	1.7
	PoznanCarpark	3
	PoznanFencing	2
	PoznanStreet	3
	TechnicolorPainter	3.2
	VikingVillage	3
Morphing	Adventure	3.8
	OrangeShaman	4.1
	PandemoniumRig1	3.6
	PoznanCarpark	3.9
	PoznanFencing	4.1
	PoznanStreet	3.8
	TechnicolorPainter	4.1
	VikingVillage	4.2
View Synthesis/Muse	Adventure	1.4
	OrangeShaman	1.2
	PandemoniumRig1	3
	PoznanCarpark	1.3
	PoznanFencing	2.1
	PoznanStreet	1.1
	TechnicolorPainter	1.3
	VikingVillage	1.4

Table 7.11 Average inter-observer rankings by transition type and scene

(a) Adventure

	Blur	Cut	Fading/blending	Morphing	View Synthesis/Muse
Blur	1	$1.3 \cdot 10^{-06}$	0.0045	0.27	$1.1 \cdot 10^{-09}$
Cut	$1.3 \cdot 10^{-06}$	1	0.11	0.0011	0.27
Fading/blending	0.0045	0.11	1	0.27	0.0017
Morphing	0.27	0.0011	0.27	1	$2.5 \cdot 10^{-06}$
View Synthesis/Muse	$1.1 \cdot 10^{-09}$	0.27	0.0017	$2.5 \cdot 10^{-06}$	1

(b) OrangeShaman

	Blur	Cut	Fading/blending	Morphing	View Synthesis/Muse
Blur	1	0.00048	0.017	0.73	$2.5 \cdot 10^{-09}$
Cut	0.00048	1	0.47	0.0013	0.023
Fading/blending	0.017	0.47	1	0.028	0.00079
Morphing	0.73	0.0013	0.028	1	$1.2 \cdot 10^{-08}$
View Synthesis/Muse	$2.5 \cdot 10^{-09}$	0.023	0.00079	$1.2 \cdot 10^{-08}$	1

(c) PandemoniumRig1

	Blur	Cut	Fading/blending	Morphing	View Synthesis/Muse
Blur	1	$2.9 \cdot 10^{-06}$	$2 \cdot 10^{-08}$	0.092	0.0036
Cut	$2.9 \cdot 10^{-06}$	1	0.41	0.012	0.19
Fading/blending	$2 \cdot 10^{-08}$	0.41	1	0.00035	0.017
Morphing	0.092	0.012	0.00035	1	0.41
View Synthesis/Muse	0.0036	0.19	0.017	0.41	1

(d) PoznanCarpark

	Blur	Cut	Fading/blending	Morphing	View Synthesis/Muse
Blur	1	0.00018	0.0079	0.47	$1.7 \cdot 10^{-09}$
Cut	0.00018	1	0.47	0.0054	0.037
Fading/blending	0.0079	0.47	1	0.086	0.0015
Morphing	0.47	0.0054	0.086	1	$1.9 \cdot 10^{-07}$
View Synthesis/Muse	$1.7 \cdot 10^{-09}$	0.037	0.0015	$1.9 \cdot 10^{-07}$	1

(e) PoznanFencing

	Blur	Cut	Fading/blending	Morphing	View Synthesis/Muse
Blur	1	0.00013	$1.5 \cdot 10^{-06}$	1	$4.1 \cdot 10^{-06}$
Cut	0.00013	1	0.94	0.0019	1
Fading/blending	$1.5 \cdot 10^{-06}$	0.94	1	$3.2 \cdot 10^{-05}$	1
Morphing	1	0.0019	$3.2 \cdot 10^{-05}$	1	$8 \cdot 10^{-05}$
View Synthesis/Muse	$4.1 \cdot 10^{-06}$	1	1	$8 \cdot 10^{-05}$	1

(f) PoznanStreet

	Blur	Cut	Fading/blending	Morphing	View Synthesis/Muse
Blur	1	0.021	0.029	0.79	$2.5 \cdot 10^{-09}$
Cut	0.021	1	0.86	0.14	0.00079
Fading/blending	0.029	0.86	1	0.16	0.00048
Morphing	0.79	0.14	0.16	1	1.3e-07
View Synthesis/Muse	$2.5 \cdot 10^{-09}$	0.00079	0.00048	$1.3 \cdot 10^{-07}$	1

(g) TechnicolorPainter

	Blur	Cut	Fading/blending	Morphing	View Synthesis/Muse
Blur	1	0.43	0.79	0.7	$8.6 \cdot 10^{-06}$
Cut	0.43	1	0.79	0.035	0.0036
Fading/blending	0.79	0.79	1	0.26	0.00018
Morphing	0.7	0.035	0.26	1	$4.4 \cdot 10^{-08}$
View Synthesis/Muse	$8.6 \cdot 10^{-06}$	0.0036	0.00018	$4.4 \cdot 10^{-08}$	1

(h) VikingVillage

	Blur	Cut	Fading/blending	Morphing	View Synthesis/Muse
Blur	1	0.027	0.23	0.54	$2 \cdot 10^{-06}$
Cut	0.027	1	0.54	0.0023	0.047
Fading/blending	0.23	0.54	1	0.047	0.0023
Morphing	0.54	0.0023	0.047	1	$4.4 \cdot 10^{-08}$
View Synthesis/Muse	$2 \cdot 10^{-06}$	0.047	0.0023	$4.4 \cdot 10^{-08}$	1

Table 7.13 Results of the Conover pairwise comparison tests of transitions for each scene. Value values below 5%, highlighted in green, indicate the significance of the differences in ranking

Chapter 8

Conclusion

Due to the numerous improvements of the view synthesis methods to generate a free-artifacts novel view, dealing with occlusions for wide baseline camera setup remains a challenging task in the computer vision community. Indeed, the introduction of technologies based on deep learning allowed the view synthesis field to see the light again after the constant failures of the traditional algorithmic approaches. However, the computational demand of the state-of-the-art learning-based view synthesis methods is thus further increased, especially for scalable use-cases.

The objective of this thesis is thus, on the one hand, to investigate a lightweight both algorithmic and learning based view synthesis approach that is not time and memory-consuming. Nevertheless, at the same time, it can generate a plausible novel viewpoint for a wide baseline. On the other hand, this thesis aimed as well to put the user in the center of the investigations, that led us to a series of subjective tests allowing the user to evaluate the different proposed view synthesis method on real cases, and to make a preference on a video transition type. Consequently, we were encouraged to create a more adaptable database which is a multi-view stereoscopic dataset for video transition quality of experience.

8.1 Achieved Work

The contribution of this thesis is organized into two parts. The first one focused on developing a new view synthesis pipeline to build up a novel synthesized view that deals with the disocclusion problems for wide baseline use-cases.

First, a low-complexity view synthesis scheme has been proposed based on preliminary warping the reference views to the target position and blending them using a neural network architecture. Our approach deals with wide baseline sequences with convolutional filters of reduced size and thus complexity since we noticed that large kernels increase the network complexity at deployment time and make the network prone to overfitting during training. We can sum up the approach with three steps:

- We use an algorithmic method to warp the two reference views to the target view position.
- We develop a simple inpainting algorithm to fill in the discovery holes due to the warping process.
- We use a convolutional neural network of four convolutional layers without downsampling the feature maps.

This contribution mainly shows us the benefits of preliminarily warping the views before blending with a ConvNets by showing how the unwarped case performs significantly worse for the same kernel size and needs larger filter sizes for better results. Also, as the filter size increases, the network is more prone to overfit the training data, looking at how the number of learnable parameters increases with the size of the filters. We experimentally show that our method performs favorably against both traditional and convolutional synthesis methods while retaining lower complexity concerning the latter. However, this work also shows the importance of training the convolutional network to generalize images with different baseline distances. This work was presented in the International Conference on 3D Immersion (IC3D) in 2019 [104].

Second, to deal with network model generalization, we propose a residual encoder-decoder for image blending with a Siamese encoder to keep the parameters count low. Therefore, while the warping is algorithmic, the blending is still learnable, inspired by image-to-image convolutional architectures. We also contribute a hole inpainting algorithm to fill the disocclusions in the warped views.

The difference from our previous work relies on the blender ConvNets where we project input features over a spatially subsampled latent feature space. Second, we reduce the encoder complexity by resorting to a smart *flip-convolve-flip* approach that lets us share parameters among encoders decreasing the network complexity. The idea is to reduce the number of learnable parameters by sharing weights among the two encoders [107] of the two left and right reference warped inputs. However, just sharing the same weights among the two encoders would be suboptimal since the left and right views do not share identical disocclusion artifacts. The *flip-convolve-flip* approach allows sharing parameters among encoders and thus lower the network complexity. Third, providing additional filled textures to the blender helps to prevent disocclusions-induced artifacts better than binary masks. Our view synthesis experiments on real multiview sequences show better objective image quality than state-of-the-art methods due to fewer artifacts in the synthesized images. However, in this approach, we identify the importance of having various enormous datasets to generalize the network better and exploiting the temporal information for a temporal consistency between frames used to create a virtual video transition. We presented this work at Signal Processing: Image Communication Conference in 2021 [142].

The second part of our contributions in this thesis focused on the user's quality of experience for the video transition based on the view synthesis.

First, we assume that when a viewer watches a video for which the content is not attractive, he would harshly judge the image's quality instead of focusing on the point of interest in the image. Therefore, we propose a novel concept of creating a dataset for quality of experience video assessment where the storytelling draws the user's attention in the scene. We film two different scenarios; the first is a theater play called Pandemonium. We placed the cameras in the scene, so the viewer must change the camera using a transition to follow up with the storyline and keep sight of the characters. The second one is a short movie created on behalf of the video transition quality of experience assessment. Each transition has an interset to be done while watching the video. We thus, film three different scenes of the movie with three different motivations for a transition. Thus, the users will be focused on the exciting part of the scene and will maybe not pay attention to the tiniest artifacts all over the image. We film the dataset in a green screen studio, which gives the possibility of a diversity of contents. However, it requires a significant working time to explore the dataset and post-processing it for computer vision purposes, from frames extraction to depth estimation and synthesized views creation. The reason why only Pandemonium content was used in our work, however, the movie dataset was presented at the Quality of Multimedia experience international conference (Qomex) in 2021 [143] and received the best student paper award.

Last, for better network generalization, we propose a novel view synthesis approach called MUSE, which is algorithmic for the warping and the blending and only learnable-based for the inpainting/enhancement of the blended image. This approach's particularity is that training is done with the intra-content strategy, such as in [144]. Since it is hard to collect many synthesized views of various contents with ground-truth images, applying this strategy for the training stage seems beneficial. Then, to reduce the network inputs and help better perform and generalize, we pre-blend the two warped reference views using an α weighting factor that represent the distance from reference views to the target view. Therefore the input image is reduced to only one image, and the training is applied on every sequence each time.

As part of the user experience contributions part, we evaluate this method using subjective tests by comparing our method to our previous work HDSB and state-of-the-art VVS. To achieve this, we create video transitions using eight different contents, including Pandemonium, with the three methods mentioned previously. Then, we develop two subjective tests that aim to evaluate the quality of the video transition. The first one is to rate the general quality and the second one to compare the quality of videos between every two methods. The experiments show that our view enhancer architecture MUSE allows the best performance in all sequences and overcomes the state-of-the-art methods, especially for wide baseline sequences. Following the same prospect to explore the user experience, we perform the third test. The latter aims to identify the user preference for a video transition type; we thus offer five different types of transition (cut, blur, blend, morphing, and View synthesis). For the

view synthesis, we use our so far best performer, MUSE. Our results show that the users' preference mainly happens on the cut transition, where the view synthesis transition led to visible and annoying artifacts in the final rendered transition. This work has been presented in an ongoing revue.

8.2 Limitations

We have two significant limitations of the proposed method, MUSE. First, in a video transition generated by MUSE, we lack a temporal consistency between the reconstructed frames due to the processing of frames one by one without building any relation between them. We notice this problem in the image areas receiving uncoherent treatment of hole inpainting between two successive temporal frames. The second limitation is the necessary training for each sequence, which requires having several real points of view placed in the scene to use as ground truths for the training phase. Also, this strategy is time and memory consuming which does not make it beneficial for real-time applications. Instead, the HDSB method is designed for more generalized use-cases where the network only needs to be trained one time. However, in this case, our main limitation is the limited availability of the training dataset, which does not help the network perform on sequences more complex than the one in the training dataset. Also, the warping process of data as input to the network for the training and testing is time and memory-consuming. Moreover, the time required for operating with the dataset we created is more significant than the dataset in the state-of-the-art, which was a limitation that did not permit using it in the series of the performed subjective tests. Also, we could not explore the possibility of changing the background after filming our two sequences in a green screen studio.

8.3 Prospects and Future Works

The contributions presented in this dissertation opens opportunities for future works on view synthesis and quality of experience assessments for video transition. This section presents the proposed research directions and prospects.

Concerning the view synthesis methods, we could improve the HDSB and MUSE methods' performance. First, we give the network the two warped images and their correspondents' inpainted images as inputs. However, we assume that the depth information of the warped images could be beneficial for a better prediction of the desired output image. Thus, we could warp the depth maps of each reference view to the target position and apply an inpainting algorithm to fill the holes that we assume will be better performing on texture-less images. This idea will help the network better learn how to fill the missing information in the image or enhance the quality of a pre-synthesized image. Second, we could improve

the performance of the networks by adding the temporal information, which means give the networks as inputs in addition to the textures at the current timestamp, the previous, and the following frames of the video. This idea will help to coherently fill the holes of the dis-occluded areas in a warped image. However, also it imposes a temporal consistency while creating a video transition using several virtual frames.

Concerning the dataset we created for more meaningful subjective tests for video transition quality of experience assessment, we could make it more useful by developing a subjective test for video transition using headphones to let the sound guide the video. We could experiment with the importance of storytelling that draws the participants' attention. In addition, it would be an interesting experiment to see the effect of their votes on the same methods we are evaluating. For neural network training purposes, this dataset could be used several times by changing the background of the green screen studio, which will provide an enormous variety of different helpful content to increase the training dataset.

Furthermore, we could improve the proposed application, allowing the user to transition between two real cameras, stop at an intermediate virtual camera, and watch the scene at any desired position. This idea could be applied once we obtain a better-rendered image using view synthesis methods, especially for a wide baseline camera setup.

References

[1] N. K. Kalantari, T.-C. Wang, and R. Ramamoorthi, "Learning-based view synthesis for light field cameras," *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)*, vol. 35, no. 6, 2016.

[2] S. Niklaus and F. Liu, "Context-aware synthesis for video frame interpolation," *CoRR*, vol. abs/1803.10967, 2018.

[3] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," *CoRR*, vol. abs/1702.02463, 2017.

[4] 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.

[5] Intel, "Intel horsepower makes it happen," 2019.

[6] J. Damiani, "Black mirror:bandersnatch could become netflix's secret marketing weapon," 2019. `https://www.theverge.com/2019/1/2/18165182/black-mirror-bandersnatch-netflix-interactive-strategy-marketing`.

[7] H. Shum and S. B. Kang, "Review of image-based rendering techniques," in *Visual Communications and Image Processing*, 2000.

[8] S. M. Seitz and C. R. Dyer, "View morphing," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, (New York, NY, USA), p. 21–30, Association for Computing Machinery, 1996.

[9] MPEG, "Report on experimental framework for 3d video coding"," October 2010.

[10] I. organization of standardisation, "Coding of moving pictures and audio,isoiec jtc1sc29wg11 mpegw18076, macau,sar cn," 10 2018.

[11] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 37, 2018.

[12] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang, "Depth-aware video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3703–3712, 2019.

[13] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. Duvall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec, "Immersive light field video with a layered mesh representation," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 86–1, 2020.

[14] M. Domanski, A. Dziembowski, T. Grajek, A. Grzelka, K. Klimaszewski, D. Mieloch, R. Ratajczak, O. Stankiewicz, J. Siast, J. Stankowski, and K. Wegner, "Free-viewpoint television demonstration for sports events," *ISO/IEC JTC1/SC29/WG11, Doc. MPEG M 41994 (2018)*, 2018.

[15] J. Xu, J. Zheng, Y. Xu, R. Tang, and S. Gao, "Layout-guided novel view synthesis from a single indoor panorama," 2021.

[16] B. Pesquet-Popescu, M. Cagnazzo, and F. Dufaux, "Motion Estimation - a Video Coding Viewpoint," in *Academic Press Library in Signal Processing Volume 5: Image and Video Compression and Multimedia*, Elsevier, 2013.

[17] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *2007 IEEE International Conference on Image Processing*, vol. 1, pp. I – 201–I – 204, 2007.

[18] G. Petrazzuoli, M. Cagnazzo, F. Dufaux, and B. Pesquet-Popescu, "Using distributed source coding and depth image based rendering to improve interactive multiview video access," in *IEEE International Conference on Image Processing*, vol. 1, (Bruxelles, Belgium), pp. 605–608, 2011.

[19] F. Dufaux, B. Pesquet-Popescu, and M. Cagnazzo, *Emerging technologies for 3D video: creation, coding, transmission and rendering.* John Wiley & Sons, 2013.

[20] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 670–679, 2017.

[21] J. Jung and P. Boissonade, "VVS: Versatile View Synthesizer for 6-DoF Immersive Video." working paper or preprint, Apr. 2020.

[22] Z. Wang and J. Zhou, "A novel approach for depth image based rendering, based on non-linear transformation of depth values," in *2011 International Conference on Image Analysis and Signal Processing*, pp. 138–142, 2011.

[23] E. Penner and L. Zhang, "Soft 3d reconstruction for view synthesis," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–11, 2017.

[24] 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.

[25] 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.

[26] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep blending for free-viewpoint image-based rendering," *ACM Transactions on Graphics (SIGGRAPH Asia Conference Proceedings)*, vol. 37, November 2018.

[27] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, pp. 417–424, 01 2000.

[28] C. Guillemot and O. Le Meur, "Image inpainting : Overview and recent advances," *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 127–144, 2014.

[29] I. Daribo and B. Pesquet-Popescu, "Depth-aided image inpainting for novel view synthesis," in *2010 IEEE International Workshop on Multimedia Signal Processing*, pp. 167–170, 2010.

[30] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," 2018.

[31] G. Luo, Y. Zhu, Z. Weng, and Z. Li, "A disocclusion inpainting framework for depth-based view synthesis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[32] 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.

[33] E. Chang, H. T. Kim, and B. Yoo, "Virtual reality sickness: A review of causes and measurements," *International Journal of Human–Computer Interaction*, vol. 36, no. 17, pp. 1658–1682, 2020.

[34] M. Recenti, C. Ricciardi, R. Aubonnet, I. Picone, D. Jacob, H. A. R. Svansson, S. Agnarsdottir, G. H. Karlsson, V. Baeringsdottir, H. Petersen, and P. Gargiulo, "Toward predicting motion sickness using virtual reality and a moving platform assessing brain, muscles, and heart signals," *Frontiers in Bioengineering and Biotechnology*, vol. 9, p. 132, 2021.

[35] W. Wu, A. Arefin, R. Rivas, R. Sheppard, and Z. Yang, "Quality of experience in distributed interactive multimedia environments: Toward a theoretical framework," *MM'09 - Proceedings of the 2009 ACM Multimedia Conference, with Co-located Workshops and Symposiums*, pp. 481–490, 01 2009.

[36] S. Schmidt, S. Möller, and S. Zadtootaghaj, "A comparison of interactive and passive quality assessment for gaming research," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–6, 2018.

[37] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[38] R. Krishnamurthy, P. Moulin, and J. Woods, "Optical flow techniques applied to video coding," in *Proceedings., International Conference on Image Processing*, vol. 1, pp. 570–573 vol.1, 1995.

[39] Z.-w. Liu, P. An, S.-x. Liu, and Z.-y. Zhang, "Arbitrary view generation based on dibr," in *2007 International Symposium on Intelligent Signal Processing and Communication Systems*, pp. 168–171, IEEE, 2007.

[40] C.-M. Cheng, S.-J. Lin, S.-H. Lai, and J.-C. Yang, "Improved novel view synthesis from depth image with large baseline," in *2008 19th International Conference on Pattern Recognition*, pp. 1–4, 2008.

[41] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, (New York, NY, USA), p. 31–42, Association for Computing Machinery, 1996.

[42] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, (New York, NY, USA), p. 425–432, Association for Computing Machinery, 2001.

[43] J. Shade, S. Gortler, L.-w. He, and R. Szeliski, "Layered depth images," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, (New York, NY, USA), pp. 231–242, ACM, 1998.

[44] P. Ndjiki-Nya, M. Koppel, D. Doshkov, H. Lakshman, P. Merkle, K. Muller, and T. Wiegand, "Depth image-based rendering with advanced texture synthesis for 3-d video," *IEEE Transactions on Multimedia*, vol. 13, no. 3, pp. 453–465, 2011.

[45] A. I. Purica, E. G. Mora, B. Pesquet-Popescu, M. Cagnazzo, and B. Ionescu, "Multiview plus depth video coding with temporal prediction view synthesis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 2, pp. 360–374, 2016.

[46] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1200–1212, 2004.

[47] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," 2016.

[48] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, "High-resolution image inpainting using multi-scale neural patch synthesis," 2017.

[49] MPEG, "High efficiency video coding"," April 2013.

[50] M. Tanimoto, T. Fujii, K. Suzuki, N. Fukushima, and Y. Mori, "Reference softwares for depth estimation and view synthesis," *ISO/IEC JTC1/SC29/WG11 MPEG*, vol. 20081, p. M15377, 2008.

[51] MPEG, "Oding of moving pictures and audio"," October 2018.

[52] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis, "Depth synthesis and local warps for plausible image-based navigation," *ACM Trans. Graph.*, vol. 32, July 2013.

[53] J. Lin, W. Wang, J. Yao, T. Guo, E. Chen, and Q. F. Yan, "Fast multi-view image rendering method based on reverse search for matching," *Optik*, vol. 180, pp. 953 – 961, 2019.

[54] S. Prakash, T. Leimkühler, S. Rodriguez, and G. Drettakis, "Hybrid image-based rendering for free-view synthesis," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 4, May 2021.

[55] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow, "Scalable inside-out image-based rendering," *ACM Trans. Graph.*, vol. 35, Nov. 2016.

[56] J. Xiao, M. M. Hannuksela, T. Tillo, M. Gabbouj, C. Zhu, and Y. Zhao, "Scalable bit allocation between texture and depth views for 3-d video streaming over heterogeneous networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 1, pp. 139–152, 2015.

[57] C. Fehn, "Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3dtv," in *Stereoscopic Displays and Virtual Reality Systems XI*, vol. 5291, pp. 93–104, International Society for Optics and Photonics, 2004.

[58] M. Gotfryd, K. Wegner, and M. Domanski, "View synthesis software and assessment of its performance," *ISO/IEC JTC1/SC29/WG11, MPEG*, p. M15672, 2008.

[59] L. Zhu, Y. Zhang, M. Yu, G. Jiang, and S. Kwong, "View-spatial–temporal post-refinement for view synthesis in 3d video systems," *Signal Processing: Image Communication*, vol. 28, no. 10, pp. 1342 – 1357, 2013.

[60] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE Transactions on Image Processing*, vol. 10, no. 8, pp. 1200–1211, 2001.

[61] Levin, Zomet, and Weiss, "Learning how to inpaint from global image statistics," in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 305–312 vol.1, 2003.

[62] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, "Simultaneous structure and texture image inpainting," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2, pp. II–707, 2003.

[63] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 28, Aug. 2009.

[64] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, "Image Melding: Combining inconsistent images using patch-based synthesis," *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2012)*, vol. 31, no. 4, pp. 82:1–82:10, 2012.

[65] J. Hays and A. A. Efros, "Scene completion using millions of photographs," *ACM Transactions on Graphics (SIGGRAPH 2007)*, vol. 26, no. 3, 2007.

[66] X. Bai, C. Yan, H. Yang, L. Bai, J. Zhou, and E. Hancock, "Adaptive hash retrieval with kernel based similarity," *Pattern Recognition*, vol. 75, 03 2017.

[67] B. Xiao, E. R. Hancock, and R. C. Wilson, "Graph characteristics from the heat kernel trace," *Pattern Recognition*, vol. 42, no. 11, pp. 2589–2606, 2009.

[68] L. Zhou, X. Bai, X. Liu, J. Zhou, and E. Hancock, "Learning binary code for fast nearest subspace search," *Pattern Recognit.*, vol. 98, 2020.

[69] J. Thatte and B. Girod, "A statistical model for disocclusions in depth-based novel view synthesis," in *2019 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, IEEE, 2019.

[70] S. Li, C. Zhu, and M.-T. Sun, "Hole filling with multiple reference views in dibr view synthesis," *IEEE Transactions on Multimedia*, vol. 20, no. 8, pp. 1948–1959, 2018.

[71] P.-J. Lee *et al.*, "Adaptive edge-oriented depth image smoothing approach for depth image based rendering," in *2010 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–5, IEEE, 2010.

[72] C. Yao, T. Tillo, Y. Zhao, J. Xiao, H. Bai, and C. Lin, "Depth map driven hole filling algorithm exploiting temporal correlation information," *IEEE Transactions on Broadcasting*, vol. 60, no. 2, pp. 394–404, 2014.

[73] X. Bi, B. Yang, J. Zeng, T. Wei, and Y. Xiang, "A novel holes filling method based on layered depth map and patch sparsity for complex-scene images," *Microelectronics Journal*, vol. 114, p. 105140, 2021.

[74] S. Satapathy and R. R. Sahay, "Robust depth map inpainting using superpixels and non-local gauss–markov random field prior," *Signal Processing: Image Communication*, vol. 98, p. 116378, 2021.

[75] Y.-W. Lee, D.-M. Chang, C.-C. Chen, and C. Yang, "Hole filling in image conversion using weighted local gradients," *Sensors and Materials*, vol. 33, p. 1577, 2021.

[76] P. Gao, T. Zhu, and M. Paul, "Disocclusion filling for depth-based view synthesis with adaptive utilization of temporal correlations," *Journal of Visual Communication and Image Representation*, vol. 78, p. 103148, 2021.

[77] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[78] R. Lopez, J. Regier, M. I. Jordan, and N. Yosef, "Information constraints on auto-encoding variational bayes," 2018.

[79] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," July 2017.

[80] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2016.

[81] Y. Zeng, J. Fu, H. Chao, and B. Guo, "Learning pyramid-context encoder network for high-quality image inpainting," 2019.

[82] Y. Zeng, Z. Lin, J. Yang, J. Zhang, E. Shechtman, and H. Lu, "High-resolution image inpainting with iterative confidence feedback and guided upsampling," 2020.

[83] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[84] K. Regmi and A. Borji, "Cross-view image synthesis using geometry-guided conditional gans," *Computer Vision and Image Understanding*, vol. 187, p. 102788, 2019.

[85] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang, "Deepmvs: Learning multi-view stereopsis," 2018.

[86] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," 2012.

[87] 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.

[88] T. Volker, G. Boisson, and B. Chupeau, "Learning light field synthesis with multiplane images: Scene encoding as a recurrent segmentation task," in *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 633–637, 2020.

[89] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision*, pp. 405–421, Springer, 2020.

[90] S. Wizadwongsa, P. Phongthawee, J. Yenphraphai, and S. Suwajanakorn, "Nex: Real-time view synthesis with neural basis expansion," *arXiv preprint arXiv:2103.05606*, 2021.

[91] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelnerf: Neural radiance fields from one or few images," 2021.

[92] T. Habtegebrial, V. Jampani, O. Gallo, and D. Stricker, "Generative view synthesis: From single-view semantics to novel-view images," *arXiv preprint arXiv:2008.09106*, 2020.

[93] G. Riegler and V. Koltun, "Free view synthesis," *CoRR*, vol. abs/2008.05511, 2020.

[94] R. Rombach, P. Esser, and B. Ommer, "Geometry-free view synthesis: Transformers and no 3d priors," *arXiv preprint arXiv:2104.07652*, 2021.

[95] F. Dufaux, B. Pesquet-Popescu, and M. Cagnazzo, eds., *Emerging Technologies for 3D Video: Creation, Coding, Transmission and Rendering*. John Wiley & Sons, Ltd, 2013.

[96] Lu Wang, Ju Liu, Jiande Sun, Yannan Ren, Wei Liu, and Yuling Gao, "Virtual view synthesis without preprocessing depth image for depth image based rendering," in *2011 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, May 2011.

[97] M. Gotfryd, K. Wegner, and M. Domanski, "View synthesis software and assessment of its performance," 2008.

[98] C. Fehn, "Depth-image-based rendering (dibr), compression and transmission for a new approach on 3d-tv," *Proc. SPIE*, vol. 5291, 05 2004.

[99] P.-J. Lee and Effendi, "Adaptive edge-oriented depth image smoothing approach for depth image based rendering," *2010 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–5, 2010.

[100] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[101] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[102] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[103] J.-Y. Zhu, T. Park, P. Isola, Efros, and A. A., "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[104] N. Hobloss, A. Purica, A. Fiandrotti, M. Cagnazzo, R. Cozot, and W. Hamidouche, "A hybrid approach to wide baseline view synthesis with convolutional neural networks," in *2019 International Conference on 3D Immersion (IC3D)*, pp. 1–7, IEEE, 2019.

[105] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, "Video super-resolution with convolutional neural networks," *IEEE Transactions on Computational Imaging*, vol. 2, June 2016.

[106] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8798–8807, 2018.

[107] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, pp. 737–744, 1994.

[108] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[109] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*, pp. 694–711, Springer, 2016.

[110] X. Xu, Y.-C. Chen, and J. Jia, "View independent generative adversarial network for novel view synthesis," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7791–7800, 2019.

[111] M. Domanski, A. Dziembowski, A. Grzelka, D. Mieloch, O. Stankiewicz, and K. Wegner, "Multiview test video sequences for free navigation exploration obtained using pairs of cameras," 05 2016.

[112] J. Tompkin, M. H. Kim, K. I. Kim, J. Kautz, and C. Theobalt, "Preference and artifact analysis for video transitions of places," *ACM Trans. Appl. Percept.*, vol. 10, pp. 13:1–13:19, Aug. 2013.

[113] MPEG-Immersive, "Immersive video." `https://mpeg.chiariglione.org/tags/mpeg-i-visual`.

[114] M. Domanski, A. Dziembowski, A. Kuehn, M. Kurc, A. Luczak, D. Mieloch, J. Siast, O. Stankiewicz, and K. Wegner, "Poznan blocks - a multiview video test sequence and camera parameters for free viewpoint television," 01 2014.

[115] M. Domanski, A. Dziembowski, M. Kurc, A. Luczak, D. Mieloch, J. Siast, O. Stankiewicz, and K. Wegner, "Poznan university of technology test multiview video sequences acquired with circular camera arrangement – "poznan team" and "poznan blocks" sequences," 02 2015.

[116] MPEG, "Mpeg dataset from nayoga university." http://www.fujii.nuee.nagoya-u.ac.jp/multiview-data/.

[117] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[118] J. Flynn, M. Broxton, P. E. Debevec, M. DuVall, G. Fyffe, R. S. Overbeck, N. Snavely, and R. Tucker, "Deepview: View synthesis with learned gradient descent," *CoRR*, vol. abs/1906.07316, 2019.

[119] Google, "Realestate 10k dataset." `https://google.github.io/realestate10k/`, 2010 (accessed December 7, 2014).

[120] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The replica dataset: A digital replica of indoor spaces," 2019.

[121] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3234–3243, 2016.

[122] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, pp. 1573–1405, Apr. 2002.

[123] D. Scharstein and S. Baker, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, pp. 1573–1405, Mar. 2002.

[124] A. X. Chang, A. Dai, T. A. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from RGB-D data in indoor environments," *CoRR*, vol. abs/1709.06158, 2017.

[125] Blackmagic Design, "Davinci resolve 17."

[126] AliceVision, "Tha alicevision meshroom software." `https://alicevision.org/`.

[127] Colmap, "Colmap: The photogrammetry software." `https://colmap.github.io/`.

[128] ColmapSoftware, "Colmap: The capturing reality software." `www.capturingreality.com`.

[129] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li, "Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction," in *2019 AAAI Conference on Artificial Intelligence (AAAI'19)*, 2019.

[130] Unity, "Create and grow more with unity," 2021.

[131] J. Jung, B. Kroon, and J. Boyce, "Common test conditions for immersive video," *ISO/IEC JTC1/SC29/WG11, Doc. MPEG N*, vol. 18789, 2019.

[132] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2462–2470, 2017.

[133] M. Domanski, T. Grajek, K. Klimaszewski, M. Kurc, O. Stankiewicz, J. Stankowski, and K. Wegner, "Poznan multiview video test sequences and camera parameters," *ISO/IEC JTC1/SC29/WG11, Doc. MPEG M*, vol. 17050, 2009.

[134] D. Doyen, T. Langlois, B. Vandame, F. Babon, G. Boisson, N. Sabater, R. Gendrot, and A. Schubert, "Light field content from 16-camera rig," *ISO/IEC JTC1/SC29/WG11, Doc. MPEG M*, vol. 40010, 2017.

[135] K. Shoemake, "Animating rotation with quaternion curves," in *SIGGRAPH '85*, 1985.

[136] MPV, "a free, open source, and cross-platform media player," 2021.

[137] ITU-R, "Recommendation itu-r bt.500-14: Methodologies for the subjective assessment of the quality of television images," *International Telecommunications Union: Geneva, Switzerland*, 2019.

[138] E. global, *SDR and HDR Settings Guide*. Eizo global.

[139] S. community, "The society for information display," 2021.

[140] X. rite Pantone, "X-rite color management solutions, services and softwares," 2021.

[141] "Méthodes d'évaluation subjective de la qualité vidéo, de la qualité audio et de la qualité audiovisuelle des vidéos internet et de la télévision de qualité distribution quel que soit l'environnement," June 2021.

[142] N. Hobloss, L. Zhang, S. Lathuilière, M. Cagnazzo, and A. Fiandrotti, "Hybrid dual stream blender for wide baseline view synthesis," *Signal Processing: Image Communication*, 2021.

[143] N. Hobloss, L. Zhang, and M. Cagnazzo, "A multi-view stereoscopic video database with green screen (mtf) for video transition quality-of-experience assessment," in *QoMEX*, p. ???, June 2021.

[144] S.-Y. Su, F. Yu, M. Zollhoefer, and H. Rhodin, "A-nerf: Surface-free human 3d pose refinement via neural rendering," *arXiv preprint arXiv:2102.06199*, 2021.

Appendix A

Contents

A.1 Details on the different scenes of the dataset

A.1.1 Adventure

Scene	Adventure
Baseline [unit]	0.8
Horizontal field of view [deg]	66
Focal [pix]	1.5e+03
5th depth percentile [unit]	3.1
95th depth percentile [unit]	6.6
Min. Disparity [pix]	1.8e+02
Max. Disparity [pix]	3.9e+02
Disparity amplitude [pix]	2.1e+02
Number of frames	510
Transition Duration [s]	10
Framerate [frame/s]	50
Type	synthetic
Camera number	22
Resolution [pix]	1920 x 1080

Table A.1 Main characteristics of the Adventure scene.

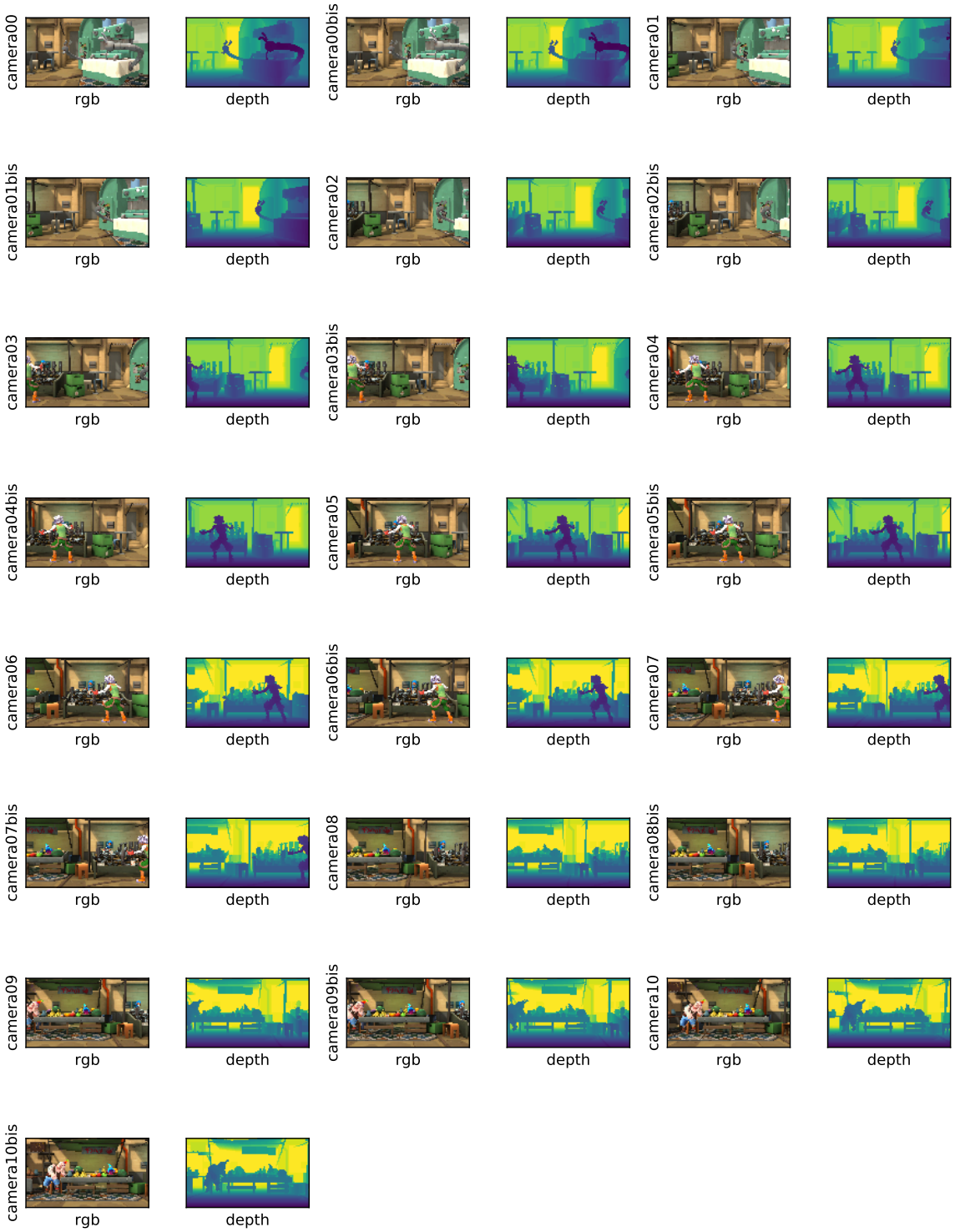


Fig. A.1 Illustration of the views of the different cameras and the associated depth maps for the sceneAdventure.

A.1.2 OrangeShaman

Scene	OrangeShaman
Baseline [unit]	0.2
Horizontal field of view [deg]	90
Focal [pix]	9.6e+02
5th depth percentile [unit]	1.3
95th depth percentile [unit]	4.6
Min. Disparity [pix]	42
Max. Disparity [pix]	1.5e+02
Disparity amplitude [pix]	1.1e+02
Number of frames	300
Transition Duration [s]	10
Framerate [frame/s]	30
Type	synthetic
Camera number	5
Resolution [pix]	1920 x 1080

Table A.2 Main characteristics of the OrangeShaman scene.

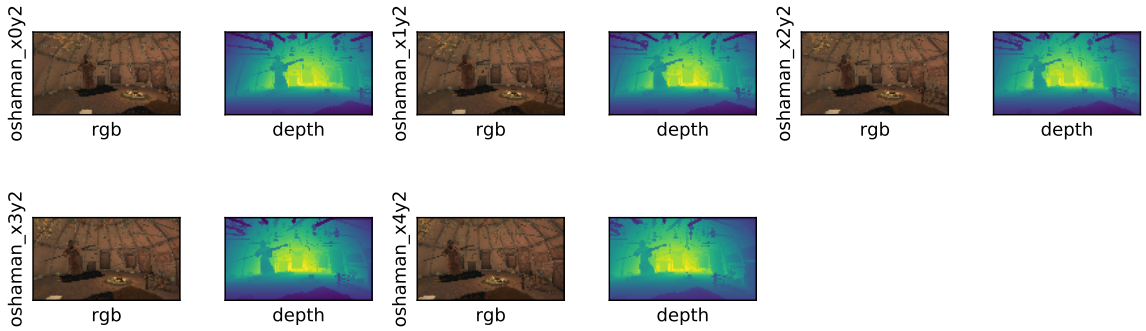


Fig. A.2 Illustration of the views of the different cameras and the associated depth maps for the sceneOrangeShaman

A.1.3 VikingVillage

Scene	VikingVillage
Baseline [unit]	0.8
Horizontal field of view [deg]	92
Focal [pix]	9.4e+02
5th depth percentile [unit]	7.7
95th depth percentile [unit]	85
Min. Disparity [pix]	8.8
Max. Disparity [pix]	97
Disparity amplitude [pix]	88
Number of frames	510
Transition Duration [s]	10
Framerate [frame/s]	50
Type	synthetic
Camera number	22
Resolution [pix]	1920 x 1080

Table A.3 Main characteristics of the VikingVillage scene.

Fig. A.3 Illustration of the views of the different cameras and the associated depth maps for the sceneVikingVillage

A.1.4 PandemoniumRig1

Scene	PandemoniumRig1
Baseline [unit]	0.43
Horizontal field of view [deg]	36
Focal [pix]	2.9e+03
5th depth percentile [unit]	2
95th depth percentile [unit]	4
Min. Disparity [pix]	3.1e+02
Max. Disparity [pix]	6.3e+02
Disparity amplitude [pix]	3.2e+02
Number of frames	600
Transition Duration [s]	10
Framerate [frame/s]	60
Type	natural
Camera number	10
Resolution [pix]	1920 x 1080

Table A.4 Main characteristics of the PandemoniumRig1 scene.

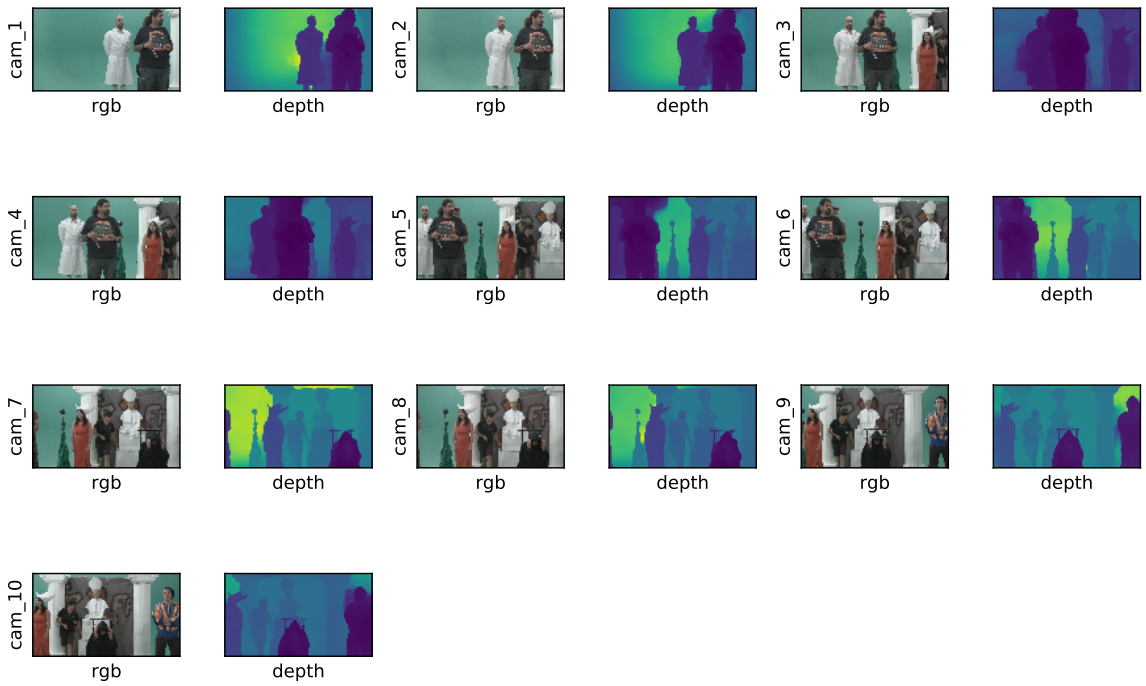


Fig. A.4 Illustration of the views of the different cameras and the associated depth maps for the scenePandemoniumRig1

A.1.5 PoznanCarpark

Scene	PoznanCarpark
Baseline [unit]	1.6
Horizontal field of view [deg]	58
Focal [pix]	1.7e+03
5th depth percentile [unit]	53
95th depth percentile [unit]	3.3e+02
Min. Disparity [pix]	8.5
Max. Disparity [pix]	52
Disparity amplitude [pix]	44
Number of frames	250
Transition Duration [s]	10
Framerate [frame/s]	25
Type	natural
Camera number	9
Resolution [pix]	1920 x 1088

Table A.5 Main characteristics of the PoznanCarpark scene.

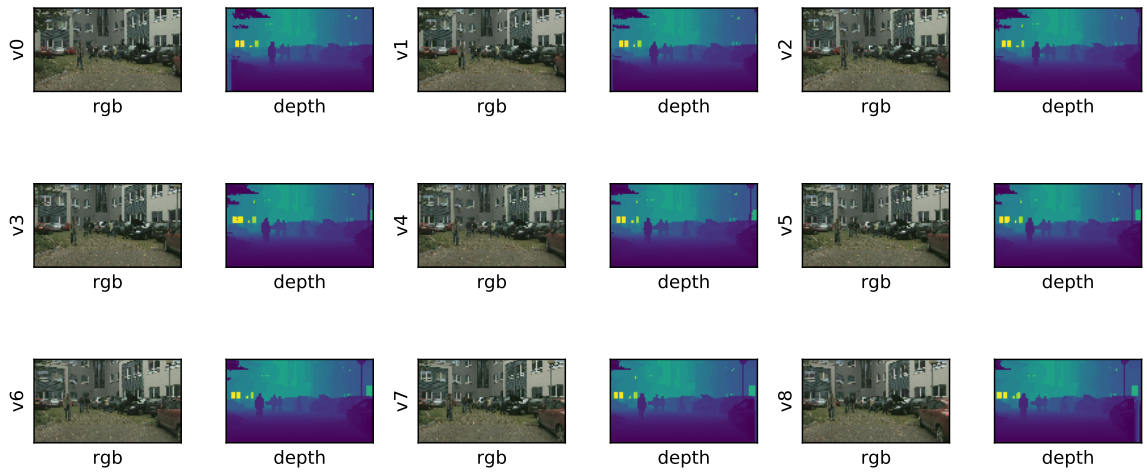


Fig. A.5 Illustration of the views of the different cameras and the associated depth maps for the scenePoznanCarpark

A.1.6 PoznanFencing

Scene	PoznanFencing
Baseline [unit]	1.6
Horizontal field of view [deg]	58
Focal [pix]	1.7e+03
5th depth percentile [unit]	4.6
95th depth percentile [unit]	5.8
Min. Disparity [pix]	4.7e+02
Max. Disparity [pix]	6e+02
Disparity amplitude [pix]	1.3e+02
Number of frames	250
Transition Duration [s]	10
Framerate [frame/s]	25
Type	natural
Camera number	10
Resolution [pix]	1920 x 1080

Table A.6 Main characteristics of the PoznanFencing scene.

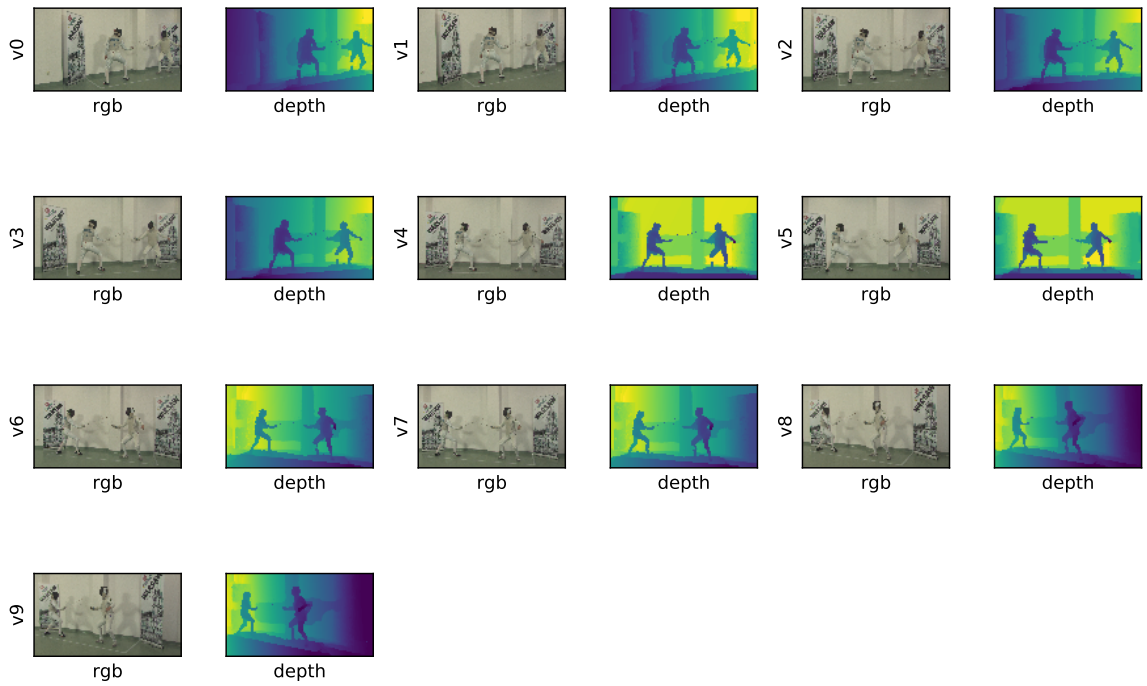


Fig. A.6 Illustration of the views of the different cameras and the associated depth maps for the scenePoznanFencing

A.1.7 PoznanStreet

Scene	PoznanStreet
Baseline [unit]	1.6
Horizontal field of view [deg]	58
Focal [pix]	1.7e+03
5th depth percentile [unit]	56
95th depth percentile [unit]	5.2e+02
Min. Disparity [pix]	5.4
Max. Disparity [pix]	49
Disparity amplitude [pix]	44
Number of frames	250
Transition Duration [s]	10
Framerate [frame/s]	25
Type	natural
Camera number	9
Resolution [pix]	1920 x 1088

Table A.7 Main characteristics of the PoznanStreet scene.

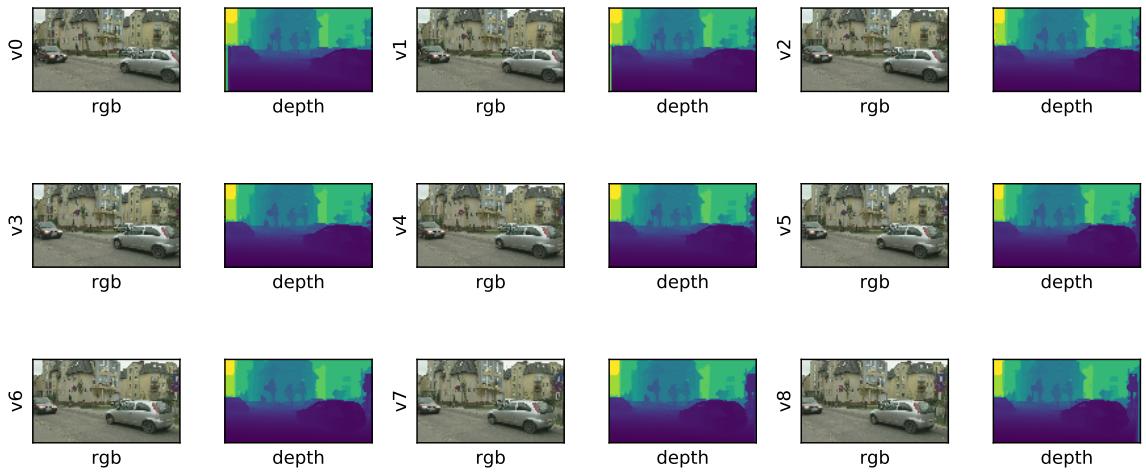


Fig. A.7 Illustration of the views of the different cameras and the associated depth maps for the scenePoznanStreet

A.1.8 TechnicolorPainter

Scene	TechnicolorPainter
Baseline [unit]	0.072
Horizontal field of view [deg]	46
Focal [pix]	2.4e+03
5th depth percentile [unit]	2.4
95th depth percentile [unit]	4.2
Min. Disparity [pix]	42
Max. Disparity [pix]	73
Disparity amplitude [pix]	31
Number of frames	300
Transition Duration [s]	10
Framerate [frame/s]	30
Type	hybrid
Camera number	4
Resolution [pix]	2048 x 1088

Table A.8 Main characteristics of the TechnicolorPainter scene.

Fig. A.8 Illustration of the views of the different cameras and the associated depth maps for the sceneTechnicolorPainter

Appendix B

Test instructions

B.1 Test instructions 1

Test instructions

Bienvenue à b<>com,
Vous allez participer à un test destiné à évaluer la qualité perçue de séquences vidéo (image seule – sans audio). Votre jugement doit prendre en considération votre **ressenti global de la qualité** vidéo sur la durée totale de la séquence. Pour compléter votre jugement, d'autres critères plus spécifiques peuvent être utilisés pour évaluer la qualité vidéo, comme le rendu des détails et des mouvements ou encore la perception de dégradations visuelles.
Huit contenus vidéo différents (**scènes**) ont été sélectionnés et seront soumis à votre jugement (Escrime, Peintre, Village Viking, etc.). Le thème des scènes ne doit pas être pris en compte dans la notation.
Les séquences vidéo présentées ont chacune une durée de 10 secondes.
Après la visualisation de chaque séquence, vous la noterez sur **une échelle de qualité**, par l'intermédiaire d'un curseur. Pour chaque évaluation, vous devez reporter votre jugement en utilisant un des labels suivants : Mauvais, Médiocre, Assez bon, Bon, Excellent. A l'issu du vote, il suffit d'appuyer sur le bouton « suivant » pour lancer la visualisation de la séquence vidéo suivante.
Le test est constitué d'un total de 48 séquences vidéo à noter.
Nous vous remercions de votre participation.

B.2 Test instructions 2

Consignes de test

Bienvenue à b<>com,
Vous allez participer à un test destiné à évaluer la qualité perçue de séquences vidéo (image seule – sans audio). Votre jugement doit prendre en considération votre **ressenti global de la qualité vidéo** sur la durée totale de la séquence. Pour compléter votre jugement, d'autres critères plus spécifiques peuvent être utilisés pour évaluer la qualité vidéo, comme le rendu des détails et des mouvements ou encore la perception de dégradations visuelles.
Huit contenus vidéo différents (**scènes**) ont été sélectionnés et seront soumis à votre jugement (Escrime, Peintre, Village Viking, etc.). Le thème des scènes ne doit pas être pris en compte dans la notation.
Pour chaque scène, les vidéos seront présentées par paires avec le séquencement suivant :

« vidéo A » « vidéo B »
« vidéo A » « vidéo B »
« vote »

Les vidéos A et B ont chacune une durée de 10 secondes.
Après avoir visualisé deux fois les vidéos A et B, vous pourrez noter la qualité vidéo perçue de la « vidéo B » par rapport à la « vidéo A ». Pour cela, vous utiliserez une **échelle de comparaison à 7 notes** comprenant les labels suivants : Largement moins bon, Moins bon, Légèrement moins bon, Equivalent, Légèrement meilleur, Meilleur, Largement meilleur. A l'issu de votre vote, il suffit d'appuyer sur le bouton « suivant » pour lancer la visualisation de la paire de séquences vidéo suivante.
Le test est constitué d'un total de 48 paires de séquences vidéo à noter. Une pause d'environ 10 minutes est prévue après la visualisation des 24 premières paires.
Nous vous remercions de votre participation.

B.3 Test instructions 3

Test instructions

Bienvenue à b<>com,
Vous allez participer à un test destiné à évaluer la préférence de différents types de transition dans des séquences vidéo (image seule – sans audio). Une transition est une façon de passer d'une caméra à une autre caméra d'une même scène. Il existe différentes façons d'effectuer ce changement de point de vue (passage d'une caméra à l'autre). Votre jugement consiste à indiquer votre **préférence pour les différentes transitions proposées**.
Huit contenus vidéo différents (scènes) ont été sélectionnés et seront soumis à votre jugement (Escrime, Peintre, Village Viking, etc.). Le thème des scènes ne doit pas être pris en compte dans la notation.
Pour chaque scène, 5 types de transition sont proposés. Les séquences vidéo présentées ont chacune une durée de 10 secondes.
Avant d'évaluer **votre préférence**, vous devrez sélectionner une séquence vidéo en cliquant sur le bouton associé (A, B, C, D, E) puis la visualiser en appuyant sur le bouton « Lecture ». L'objectif du test est de classer par ordre de préférence les transitions de 1 à 5. Pour cela, vous devez utiliser les touches « haut » et « bas » pour changer l'ordre selon votre préférence. La transition préférée sera classée 1 et la moins préférée 5. Les autres transitions seront notées 2, 3 et 4 en fonction de votre préférence.
Une fois le classement terminé, appuyer sur la touche « suivant » pour passer à la scène suivante.
Nous vous remercions de votre participation.

Appendix C

Publications

C.1 Scientific journals

[J1] N. HOBLOSS, L. Zhang, S. Lathuilière, M. Cagnazzo, and A. Fiandrotti
"Hybrid Dual Stream Blender ForWide Baseline View Synthesis",
published in the *Elsevier Signal Processing: Image Communication* journal, June 2021.

[J2] N. HOBLOSS, L. Zhang, M. Cagnazzo, J.Fournier and N. Ramin
"A Multi-view synthesis enhancer and video transition quality-of-Experience assessment",
in preparation.

C.2 International Conferences

[C1] N. HOBLOSS, A. Purica, A. Fiandrotti, M. Cagnazzo, R.Cozot and W. Hamidouche
"A hybrid approach to wide baseline view synthesis with convolutional neural network",
presented at the *2019 International Conference on 3D Immersion (IC3D), December 2019.*

[C2] N. HOBLOSS, L. Zhang and M. Cagnazzo,
"A Multi-View Stereoscopic Video Database With Green Screen (MTF) For Video Transition Quality-of-Experience Assessment ",
presented at the *International Conference on Quality of Multimedia Experience (QoMEX), and received the Best Student Paper award in June 2021.*

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre: Amélioration de l'expérience utilisateur en navigation libre via la synthèse d'image

Mot clés : Synthèse de vue, Navigation libre, expérience utilisateur, dataset, transition vidéo

Resumé : Dans l'acquisition de la vidéo multi-vue le centre d'attention peut être contrôlé par les téléspectateurs plutôt que par un réalisateur, ce qui implique que chaque téléspectateur peut observer un point de vue unique. Par conséquent, ceci exige de placer des caméras autour de la scène à capturer, ce qui pourrait être très coûteux. La génération de caméras virtuelles pour remplacer une partie des caméras réelles de la scène réduit le coût de la configuration de la vidéo multi-vues.

Cette thèse se concentre sur la génération de transitions vidéo virtuelles dans les scènes capturées par vidéo multi-vues pour se déplacer virtuellement d'un point de vue réel à un autre dans la même scène. Moins nous utilisons de caméras réelles, moins il y a de dépenses nécessaires dans la vidéo multi-vues ; cependant, plus la baseline est importante.

Les méthodes de synthèse de vue ont attiré notre attention, comme une approche de notre problème. Cependant, dans la littérature, ces méthodes souffrent toujours d'artefacts visuels dans l'image rendue finale en raison des occultations dans la nouvelle vue virtuelle cible.

Dans un premier temps, nous proposons une approche hybride de la synthèse de vues dans laquelle nous déformons d'abord les vues de référence en corrigeant les occultations. Nous fusionnons les vues pré-traitées via une architecture de convolution simple. Le warping des vues de référence réduit la distance entre les vues de référence, ainsi que la taille des filtres convolutionnels et donc de réduire la complexité du réseau. Ensuite, nous présentons une approche hybride, où nous fusionnons les vues pré-warpées via un encodeur-décodeur résiduel avec un encodeur siamois afin de maintenir le nombre des paramètres bas. Nous proposons également un algorithme d'inpainting des trous pour combler les désoccultations dans les vues warpées.

En plus, nous nous concentrons sur la qualité de l'expérience de l'utilisateur pour la transition vidéo et la base de données. D'abord, nous réalisons un dataset créatif pour la qualité d'expérience de la transition vidéo. Ensuite, nous proposons un optimiseur de synthèse de vues multiples algoritmic-learning-based. Le travail vise à évaluer subjectivement les approches de synthèse de vues proposées sur 8 différentes séquences vidéo en réalisant une série de tests subjectifs.

Title: Improving the user experience in free navigation via image synthesis

Keywords : View synthesis, free navigation, quality of experience, dataset, video transition

Abstract : In multi-view capture, the focus of attention can be controlled by the viewers rather than by a director, which implies that each viewer can observe a unique point of view. Therefore, this requires placing cameras around the scene to be captured, which could be very expensive. Generating virtual cameras to replace part of the real cameras in the scene reduces the cost of setting up multi-view video.

This thesis focuses on generating virtual video transitions in scenes captured by multi-view video to virtually move from one real viewpoint to another in the same scene. The fewer real cameras we use, the less expensive is required in the multi-view video; however, the larger the baseline is.

View synthesis methods have attracted our attention as an approach to our problem. However, in the literature, these methods still suffer from visual artifacts in the final rendered image due to occlusions in the new target virtual view.

As a first step, we propose a hybrid approach to view synthesis. We first warp the reference views by correcting the occlusions. We merge the pre-processed views via a simple convolution architecture. Warping the reference views reduces the distance between the reference views and the size of the convolutional filters and thus reduces the complexity of the network. Next, we present a hybrid approach. We merge the pre-warped views via a residual encoder-decoder with a Siamese encoder to keep the parameters low. We also propose a hole inpainting algorithm to fill in disocclusions in warped views.

In addition, we focus on the quality of user experience for the video transition and the database. First, we perform a creative dataset for the quality of experience of the video transition. Second, we propose an algorithmic-learning-based multiple view synthesis optimizer. The work aims to subjectively evaluate the proposed view synthesis approaches on 8 different video sequences by performing a series of subjective tests.