Multi-view acquisition and rendering of animated scenes

Beatrix-Emőke Fülöp-Balogh

N°d'ordre NNT :
2021LYSE1308

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

Ecole Doctorale N° accréditation 512
L'École Doctorale InfoMaths

Spécialité de doctorat : Informatique

Soutenue publiquement le 13/12/2021, par :
Fülöp-Balogh Beatrix-Emőke

Acquisition multi-vues et rendu de scènes animées

Devant le jury composé de :

BOYER, Edmond, Directeur de Recherche INRIA Grenoble Président

BOUSSEAU, Adrien Directeur de Recherche INRIA Sophia Antipolis Rapporteur
LOSCOS, Céline Professeure des Universités Université Reims Champagne Ardenne Rapporteure

CHAINE, Raphaëlle Professeure des Universités Université Lyon 1 Examinatrice

DIGNE, Julie Chargée de Recherche CNRS Lyon Directrice de thèse
BONNEEL, Nicolas Chargée de Recherche CNRS Lyon Co-directeur de thèse

Université Claude Bernard – LYON 1

Président de l'Université	M. Frédéric FLEURY
Président du Conseil Académique	M. Hamda BEN HADID
Vice-Président du Conseil d'Administration	M. Didier REVEL
Vice-Président du Conseil des Etudes et de la Vie Universitaire	M. Philippe CHEVALLIER
Vice-Président de la Commission de Recherche	M. Petru MIRONESCU
Directeur Général des Services	M. Pierre ROLLAND

COMPOSANTES SANTE

Département de Formation et Centre de Recherche en Biologie Humaine	Directrice : Mme Anne-Marie SCHOTT
Faculté d'Odontologie	Doyenne : Mme Dominique SEUX
Faculté de Médecine et Maïeutique Lyon Sud - Charles Mérieux	Doyenne : Mme Carole BURILLON
Faculté de Médecine Lyon-Est	Doyen : M. Gilles RODE
Institut des Sciences et Techniques de la Réadaptation (ISTR)	Directeur : M. Xavier PERROT
Institut des Sciences Pharmaceutiques et Biologiques (ISBP)	Directrice : Mme Christine VINCIGUERRA

COMPOSANTES & DEPARTEMENTS DE SCIENCES & TECHNOLOGIE

Département Génie Electrique et des Procédés (GEP)	Directrice : Mme Rosaria FERRIGNO
Département Informatique	Directeur : M. Behzad SHARIAT
Département Mécanique	Directeur M. Marc BUFFAT
Ecole Supérieure de Chimie, Physique, Electronique (CPE Lyon)	Directeur : Gérard PIGNAULT
Institut de Science Financière et d'Assurances (ISFA)	Directeur : M. Nicolas LEBOISNE
Institut National du Professorat et de l'Education	Administrateur Provisoire : M. Pierre CHAREYRON
Institut Universitaire de Technologie de Lyon 1	Directeur : M. Christophe VITON
Observatoire de Lyon	Directrice : Mme Isabelle DANIEL
Polytechnique Lyon	Directeur : Emmanuel PERRIN
UFR Biosciences	Administratrice provisoire : Mme Kathrin GIESELER
UFR des Sciences et Techniques des Activités Physiques et Sportives (STAPS)	Directeur : M. Yannick VANPOULLE
UFR Faculté des Sciences	Directeur : M. Bruno ANDRIOLETTI

Abstract

Recent technological breakthroughs have led to an abundance of consumer friendly video recording devices. Nowadays new smart phone models, for instance, are equipped not only with multiple cameras, but also depth sensors. This means that any event can easily be captured by several different devices and technologies at the same time, and it raises questions about how one can process the data in order to render a meaningful 3D scene. Most current solutions focus on static scenes only, LiDar scanners produce extremely accurate depth maps, and multi-view stereo algorithms can reconstruct a scene in 3D based on a handful of images. However, these ideas are not directly applicable in case of dynamic scenes. Depth sensors trade accuracy for speed, or vice versa, and color image based methods suffer from temporal inconsistencies or are too computationally demanding.

In this thesis we aim to provide consumer friendly solutions to fuse multiple, possibly heterogeneous, technologies to reconstruct and render 3D dynamic scenes.

Firstly, we introduce an algorithm that corrects distortions produced by small motions in time-of-flight acquisitions and outputs a corrected animated sequence. We do so by combining a slow but high-resolution time-of-flight LiDAR system and a fast but low-resolution consumer depth sensor. We cast the problem as a curve-to-volume registration, by seeing the LiDAR point cloud as a curve in the 4-dimensional spacetime and the captured low-resolution depth video as a 4-dimensional spacetime volume. We then advect the details of the high-resolution point cloud to the depth video using its optical flow.

Second, we tackle the case of the reconstruction and rendering of dynamic scenes captured by multiple RGB cameras. In casual settings, the two problems are hard to merge: structure from motion (SfM) produces spatio-temporally unstable and sparse point clouds, while the rendering algorithms that rely on the reconstruction need to produce temporally consistent videos. To ease the challenge, we consider the two steps together. First, for SfM, we recover stable camera poses, then we defer the requirement for temporally-consistent points across the scene and reconstruct only a sparse point cloud per timestep that is noisy in space-time. Second, for rendering, we present a variational diffusion formulation on depths and colors that lets us

robustly cope with the noise by enforcing spatio-temporal consistency via per-pixel reprojection weights derived from the input views.

Overall, our work contributes to the understanding of the acquisition and rendering of casually captured dynamic scenes.

Resumé

Les récentes percées technologiques ont conduit à une abondance d'appareils d'enregistrement vidéo conviviaux. De nos jours, les nouveaux modèles de smartphones, par exemple, sont équipés non seulement de plusieurs caméras, mais également de capteurs de profondeur. Cela signifie que tout événement peut facilement être capturé par plusieurs appareils et technologies différents en même temps, et cela soulève des questions sur la façon dont on peut traiter les données afin de restituer une scène 3D significative. La plupart des solutions actuelles se concentrent uniquement sur les scènes statiques, les scanners LiDaR produisent des cartes de profondeur extrêmement précises et les algorithmes stéréo multi-vues peuvent reconstruire une scène en 3D à partir d'une poignée d'images. Cependant, ces idées ne sont pas directement applicables en cas de scènes dynamiques. Les capteurs de profondeur échangent la précision contre la vitesse, ou vice versa, et les méthodes basées sur des images couleur souffrent d'incohérences temporelles ou sont trop exigeantes en termes de calcul.

Dans cette thèse, nous visons à fournir des solutions conviviales pour fusionner des technologies multiples, éventuellement hétérogènes, pour reconstruire et rendre des scènes dynamiques 3D.

Premièrement, nous introduisons un algorithme qui corrige les distorsions produites par de petits mouvements dans les acquisitions de temps de vol et produit une séquence animée corrigée. Pour ce faire, nous combinons un système LiDAR à temps de vol lent mais haute résolution et un capteur de profondeur consommateur rapide mais basse résolution. Nous avons présenté le problème comme un recalage courbe-volume, en voyant le nuage de points LiDAR comme une courbe dans l'espace-temps à 4 dimensions et la vidéo de profondeur à basse résolution capturée comme un volume d'espace-temps à 4 dimensions. Nous convoyons ensuite les détails du nuage de points haute résolution à la vidéo de profondeur en utilisant son flux optique.

Deuxièmement, nous abordons le cas de la reconstruction et du rendu de scènes dynamiques capturées par plusieurs caméras RVB. Dans des contextes occasionnels, les deux problèmes sont difficiles à fusionner : la structure à partir du mouvement (SfM) produit des nuages de points spatio-temporellement instables et parcimonieux,

tandis que les algorithmes de rendu qui reposent sur la reconstruction doivent produire des vidéos temporellement cohérentes. Pour relever le défi, nous considérons les deux étapes conjointement. Tout d'abord, pour SfM, nous récupérons des poses de caméra stables, puis nous différons l'exigence de points cohérents dans le temps sur la scène et ne reconstruisons qu'un nuage de points épars par pas de temps qui est bruité dans l'espace-temps. Deuxièmement, pour le rendu, nous présentons une formulation de diffusion variationnelle sur les profondeurs et les couleurs qui nous permet de faire face de manière robuste au bruit en appliquant une cohérence spatio-temporelle via des poids de reprojection par pixel dérivés des vues d'entrée.

Dans l'ensemble, nous montrons que notre travail a contribué à la compréhension de l'acquisition et du rendu de scènes dynamiques capturées simplement.

Acknowledgement

To be added.

Contents

1 Introduction **1**

1.1 Problem Statement 2

1.1.1 Camera Pose Estimation 2

1.1.2 Leveraging Motion Information 3

1.2 Contributions 3

1.2.1 Correcting Motion Distortions in Time-of-Flight Imaging . . . 3

1.2.2 Dynamic Scene Novel View Synthesis via Deferred Spatio-temporal Consistency 4

1.3 Potential Impact 5

1.4 Overview 6

2 Correcting Motion Distortions in Time-of-Flight Imaging **9**

2.1 Related Work 10

2.2 Overview 12

2.3 Scene Flow estimation 13

2.4 Registration in the 4D space 13

2.4.1 Problem formulation 14

2.4.2 Coarse initialization 14

2.4.3 Fine registration 15

2.5 Detail Transfer 17

2.6 Results 17

2.6.1 Simulated Data 18

2.6.2 Real Data 18

2.6.3 Limitations 21

2.7 Discussion 23

3 Dynamic Scene Novel View Synthesis via Deferred Spatio-temporal Consistency **25**

3.1 Related Work 27

3.1.1 Static scene IBR 27

3.1.2 Dynamic scene VBR 28

3.2 Method 29
3.2.1 Camera pose estimation and 3D scene points. 29
3.2.2 Novel depth and novel view rendering 31
3.2.3 Temporal consistency 35
3.2.4 Implementation details 36
3.3 Experiments and Results 36
3.3.1 Dataset Sequences 36
3.3.2 Ablation study 38
3.3.3 Novel depth and view comparisons 38
3.3.4 Challenging Sequence—Drone 42
3.3.5 Computational Resources 42
3.4 Limitations 47
3.5 Summary 47

4 Conclusions **49**
4.0.1 Machine Learning Methods 50
4.1 Future Work 51
4.1.1 Next Steps 52

Bibliography **53**

1 Introduction

Capturing and reconstructing scenes in 3 dimensions accurately is a well-studied subject in the computer graphics community with commercially available low-cost solutions for almost every possible scenario.

Artists can reconstruct real life objects and even whole environments in 3D using a couple of photographs instead of spending countless hours on recreating them virtually, thanks to image based reconstruction and rendering (IBR) techniques. Laser technologies such as LiDaR let us acquire depth measurements at sub-millimeter accuracy. They can be used to faithfully capture the shape of single objects such as historical artifacts and at the same time to reconstruct whole cities. These and similar methods also make immersive virtual tours, as if we were physically there, in museums and galleries possible and aid augmented reality tools in sensing the users' environment.

However, what all these approaches have in common is that they only work if the captured scene is static. A LiDaR scanner takes several seconds to do a single acquisition one point at a time, so any slight motion will cause distortions in the final point cloud. On the other hand, image based techniques rely on correspondences to triangulate 3D points which are error prone if the scene or an object changed between the capturing of two images. Even if we try to overcome this issue by capturing synchronized videos, thus reducing the problem to a sequence of unrelated IBR instances, the results still would not be satisfactory. Indeed, most approaches produce results that are not stable across consecutive frames causing a flickering effect in the output video.

How can one solve the 3D acquisition of dynamic scenes challenge? If we're interested in capturing the depth of a scene, depending on the application, one could resort to using commercial depth sensors like Microsoft Kinect or Creative Senz3D. These systems are able to capture the entire field of view of the sensor at an interactive frame rate (30-60 fps), but at a limited resolution and the resulting point clouds often lack details and are noisy. They are most useful for applications where accuracy and precision is not of paramount interest. Their most widespread usage is for interactive game consoles where a rough depth estimation, possibly coupled with the color video sequence, is enough to track the player.

For accurate dynamic scene reconstruction no satisfactory simple solution exists yet, so one must resort to much more complex and expensive solutions. These usually involve several types of sensors both for capturing depth and color images, as well as (structured) light sources to aid the correspondence search across the images. Not only that these setups require heavy calibration, and therefore cannot be used outside of a laboratory setting, they were also designed for highly specific tasks such as capturing facial animation and don't generalize well for other types of scenes.

1.1 Problem Statement

The goal of this thesis is to propose solutions for 3D *dynamic* scene acquisition and rendering in *casual settings* using widely available and user friendly technologies only. Moreover, we aim for our approach to be applicable for a wide variety of scenes.

Thus, the set of problems we need to tackle can be loosely divided in the following two categories.

1.1.1 Camera Pose Estimation

In the case of static scene reconstruction, fusing information captured from several locations in space is a well understood problem with widely agreed upon solutions. For technologies that capture the depth of the scene directly, the go-to technique is to register the resulting point clouds by minimizing some distance function between them [72]. On the other hand, camera pose estimation from color images in the wild is a more involved process which is based on matching and triangulating corresponding points across several images.

Both of these classes of techniques rely heavily on the scene being static and attempts to directly apply them to dynamic environments is doomed because the assumptions they were built on are invalidated by the captured motion. Matched correspondences from color images from different time instances do not necessarily translate to the same 3D location anymore and high resolution point clouds suffer from motion distortion.

1.1.2 Leveraging Motion Information

Even though moving objects raise several challenges for the acquisition process, they also provide additional information about the scene that must not be overlooked and instead should be leveraged for a more accurate reconstruction. The challenge is therefore not only to overcome the difficulties introduced by the motion, but also to use the additional information provided by it to improve the results.

A straightforward assumption that holds true for most environments is that the scene does not change considerably between two consecutive time instances captured at an interactive frame rate. This can be leveraged to initialize the reconstruction or rendering at a time instance based on neighboring frames which could both result in efficiency gains and regularize the final solution.

1.2 Contributions

In this thesis we introduce two new methods for two different set ups that enable the high quality reconstruction and rendering of dynamic scenes requiring no laborious calibration or expensive equipment.

1.2.1 Correcting Motion Distortions in Time-of-Flight Imaging

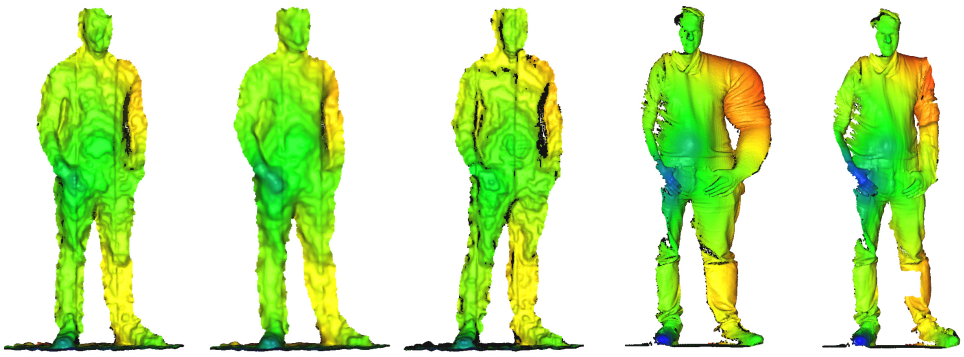


Fig. 1.1: A consumer depth sensor (e.g. Kinect) acquires dense low-resolution scans at a high rate while a LiDaR scanner acquires sparse high resolution scans containing time distorsions. We recover details by undistorting the LiDaR scan.

Time-of-flight imaging produces extremely accurate depth measurements, but it requires several seconds to capture the whole environment making it impractical

for the acquisition of dynamic scenes. Conversely, infrared depth sensors, such as Kinect, capture a low resolution noisy depth maps of its entire field of view at an interactive frame rate. We provide the insight that using both technologies at the same time, one can undistort and partly recover a more accurate geometry in the presence of moderate motions (Fig. 1.1).

We present a method that registers an accurate LiDaR point cloud captured at a low temporal frame rate to a coarse spatiotemporal depth-sensor point cloud captured at interactive frame rate, without any preliminary device cross-calibration. We formulate this problem as a spacetime curve-to-volume rigid correspondence problem efficiently solved using a Hough transform and Iterative Closest Point algorithm. Our intuition is that, due to the capture time, a LiDAR point cloud can be seen as a curve in a 4-dimensional spacetime, in which points are each identified by a single (x, y, z, t) value, while a real-time depth video camera produces (x, y, z) dense slices for individual time instances at a high frame-rate, resulting in a sliced 4-dimensional spacetime volume. We then use these correspondences to transfer details from the LiDaR to the Kinect point cloud, leading to a detailed animated model.

We validate our method on synthetic data, and demonstrate it by recovering high-resolution dynamic geometries under moderate motion.

This was published at the ACM Siggraph Conference on Motion, Interaction and Games in 2018.

1.2.2 Dynamic Scene Novel View Synthesis via Deferred Spatio-temporal Consistency

Image-based rendering (IBR) creates novel views of a scene by combining existing pictures taken from different camera positions. This technique is useful for generating new images, because the captured imagery can 'fill in' for complex geometries and materials even when we can only reconstruct partial structure, such as sparse depth points, rather than a full scene geometry.

While IBR quality and efficiency has improved over the past two decades of research, casually captured dynamic scenes are difficult to handle as both the cameras and the scene objects move. They complicate camera pose and depth estimation, causing ghosting or bleeding artifacts across the rendered novel views around the moving objects and in the background.

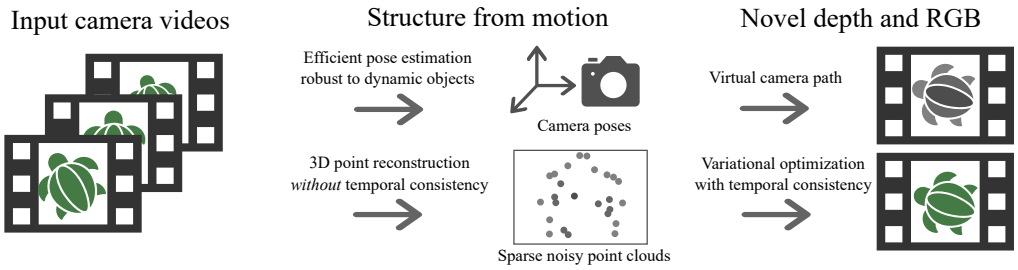


Fig. 1.2: Given a small set of video sequences of a performance, our method computes camera poses and sparse points, then optimizes those points into a novel video sequence following a user-defined camera path. Our space-time SfM intentionally does not compute temporal consistency for points on dynamic objects and instead defers spatio-temporal consistency in both depth and RGB reconstructions to the novel view synthesis stage via our variational formulation.

We propose a solution to address both these challenges. First, we use a coarse-to-fine structure from motion (SfM) algorithm to estimate camera poses in spite of moving objects and without any explicit dynamic object segmentation. We only place additional temporal constraints on camera parameters, and not the 3D scene points themselves. Then, using the sparse and noisy depth points reconstructed by the SfM, we employ a diffusion process to render the final videos. We achieve spatio-temporal consistency by weighing the influence of each depth and color value based on its agreement with the reprojected input views, as well as previously rendered views. An overview of our method is shown in Fig. 1.2.

We demonstrate this approach on real-world sequences with a small numbers of unstructured cameras capturing video clips over a moderate baseline, with tripod-mounted and hand held capture. We perform baseline comparisons to recent proposed approaches. Further, we test both the camera pose computation and the rendering process using a synthetic scene with known ground truth values, in an ablation study.

This was submitted to the *Computer and Graphics* journal and published as a preprint on *arXiv*.

1.3 Potential Impact

Casual cinematography In recent years one could observe a significant surge in affordable and consumer friendly technologies, such as mobile phones equipped with stereo cameras and structured light sources, geared towards capturing the world in 3D. While these sensors are crucial in the development of applications that

let users easily manipulate their own photos and videos, there is a strong need for software solutions that are capable of handling complex scenes.

Our methods use depth and/or color data from uncalibrated sensors and aim to reduce the difficulty of handling dynamic scenes in a casual setting. They could ease the adaptation of high quality visual effects, which highly benefit from reliable depth information, outside of big budget projects such as Hollywood movies or video games.

Moreover, the techniques we developed to capture and process dynamic scenes could aid in the adaptation of methods that previously were available for static scenes only to dynamic scenes.

Virtual Reality At the same time with the progress of affordable sensors, interdependently, devices capable of displaying and interacting with 3D scenes have also been developed and perfected. The most straightforward example of such device are the virtual reality setups that have applications ranging from gaming, to remote learning, to virtual tours etc.

Yet, creating content efficiently for them in a casual setting still poses numerous challenges. Capturing real world dynamic scenes using multiple hand-held cameras and then rendering them from virtual view points would enable users to watch their own videos in an immersive manner.

Augmented Reality In the context of augmented reality, representing the environment of the user in 3D is a powerful tool which allows for more sophisticated blending of the real and virtual scene. The assumption that this environment is static doesn't necessarily hold in most use cases. Our method could be used to generate depth map videos of the surroundings of the user that can be used to virtually interact with the scene.

1.4 Overview

- Chapter 2 presents a novel method to capture high resolution point clouds of animated scenes by coupling a time-of-flight LiDaR system with a consumer depth sensor.

- In chapter 3 we introduce a new multi-view novel view synthesis approach for dynamic scenes.
- Chapter 4 concludes the results of this thesis as well as elaborates on possible future research directions.

2 Correcting Motion Distortions in Time-of-Flight Imaging

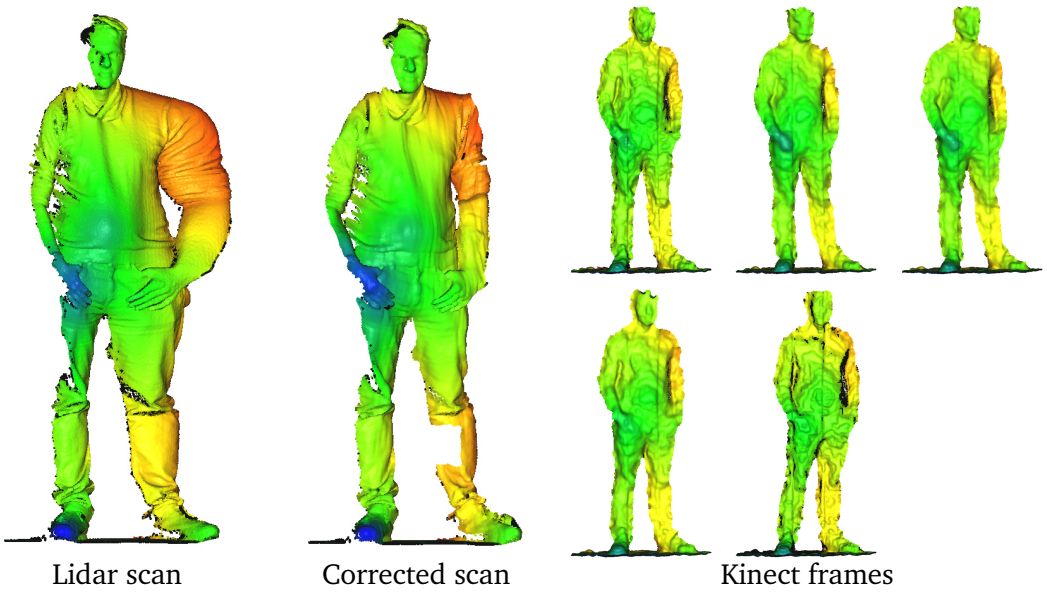


Fig. 2.1: A consumer depth sensor acquires dense low-resolution scans at a high rate (right) while a LiDaR scanner acquires sparse high resolution scans containing time distorsions (left). We recover details by undistorting the LiDaR scan (bottom right).

Capturing accurate 3D geometries is a powerful way for artists to design sceneries, for historians to reconstruct old monuments, for real-estate agents to communicate their products or for navigation systems to provide context. It has become a widespread need, and, when it comes to static environments, is now mostly sucessfully performed using laser technologies such as LiDaR, that capture environments at sub-millimeter accuracy. When it comes to slightly moving, let alone fully animated scenes, this technology breaks. In fact, capturing a single frame can take tens of seconds, which makes any motion problematic. Even small motions manifest as distortions (Fig. 2.1) altering the reconstructed point cloud. For dynamic scenes, one often resorts to much less accurate systems such as infrared sensors (Microsoft Kinect or Creative Senz3D), or structure-from-motion using multiple video cameras. These systems allow for capturing rough depth of an entire field of view at 30-60 frames per

second, albeit at low resolution, with an accuracy of centimeters and numerous outliers. These setups gained popularity with interactive console 3D games for which neither precision nor accuracy is crucial. For accurate dynamic scene reconstruction, no satisfactory solutions exists and one often resorts to high-resolution templates deformed to match rough motions. This is particularly the case for facial animation, but does not generalize well to different geometries.

We provide the insight that using both a LiDaR scanner and a consumer depth camera at the same time, one can undistort and partly recover a more accurate geometry in the presence of moderate motions. We design a method that registers an accurate LiDaR point cloud captured at a low temporal framerate to a coarse spatio-temporal depth-sensor point cloud captured at interactive framerate, without any preliminary device cross-calibration. We formulate this problem as a spacetime curve-to-volume rigid correspondence problem efficiently solved using a Hough transform and Iterative Closest Point algorithm. Our intuition is that, due to the capture time, a LiDaR point cloud can be seen as a curve in a 4-dimensional spacetime, in which points are each identified by a single (x, y, z, t) value, while a real-time depth video camera produces (x, y, z) dense slices for individual timestamps at a high frame-rate, resulting in a sliced 4-dimensional spacetime volume. We then transfer details from the LiDaR to the Kinect[1] point cloud and advect them across frames, leading to a detailed animated model. We validate our method on synthetic data, and demonstrate it by recovering high-resolution dynamic geometries under moderate motion.

2.1 Related Work

Acquisition of high resolution dynamic shapes has been tackled using stereo and active light projection systems [45], such as fringe projection and time shifting [82]. In the special case of facial motion capture, Zhang et al. [81] propose to use synchronized video cameras and structured light projector and fit a highly detailed template to the resulting geometry to get a high resolution facial animation. Bradley et al. [12] avoids templates by using a high-resolution multi-camera setup to reconstruct detailed facial geometry. Weise et al. [75] use a consumer depth sensor to animate a face template. A combination of active light and stereo was also proposed for capturing scenes in real time with motion compensation [74]. More generally, high spacetime resolution capture can be performed using multiview

[1] While we employ the term Kinect due to its popularity, in practice we use a similar technology, the ASUS Xtion sensor.

stereo techniques [80, 23]. Yet the resolution is often limited, depending on the number of views and size of captured objects. Sensors also need synchronization and often, heavy calibration, which can make them difficult to use in practice.

Static point set super-resolution has been tackled by Kil et al. [44] where several nearby scans are registered and merged together to obtain a high resolution point cloud. More recently, Hamdi-Cherif et al. [37] nonlocally merge self similar patches of a LiDaR scan to improve its resolution. In a quite different setting, Haefner et al. [36] proposed to perform single frame super-resolution from a kinect scan by using shape from shading to solve this ill-posed problem.

Texture synthesis and transfer. High resolution and detailed animations synthesis is a hot topic in computer generated animation research. Rohmer et al. [65] generate detailed wrinkles on an animated mesh to make it look more realistic, Bertiken et al. [9] propose a way to transfer details from similar areas of one shape to another, using metric learning.

Enhancing Videos with stills. Our method share similarities with the problem of enhancing a low-quality video with high resolution stills. The main difference is that no motion-induced geometric distorsions appear in still photography whereas rolling-shutter-like distorsions are accounted for in our LiDaR point cloud. The video enhancement problem bas been tackled by considering a *spacetime* volume of (x, y, t) pixel coordinates. In that space, a video is a subvolume, while a still photography is a plane. By aligning videos and stills in that volume [16], Shechtman et al. [68] merge the information from these two sources and increase spacetime resolution. Liu et al. [49] improve the spacetime resolution using a sparse decomposition on a pre-learned dictionary. When the scene is static, Bhat et al. [11] use structure-from-motion to reconstruct a 3D proxy from the video. Then for each frame, the best still photograph is selected and used to improve the video using image-based rendering and Markov Random Fields. Similarly Gupta et al. [35] enhance a video by selecting pixels from neighboring high-resolution stills using a graph-cut formulation. Ancuti et al. [2] proposed a Maximum a Posteriori-based modeling of this problem that is also limited to static scenes.

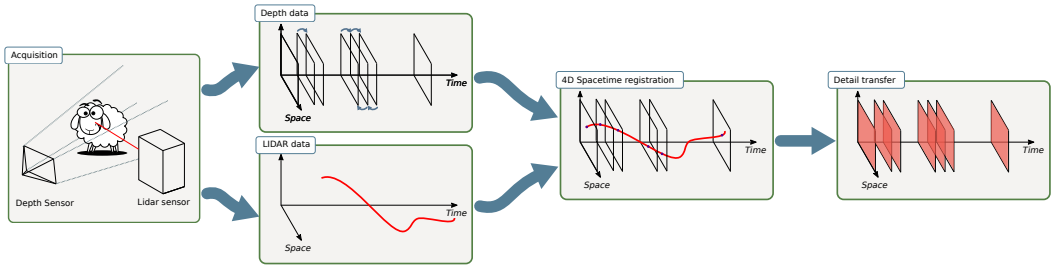


Fig. 2.2: Overview of our high resolution dynamic point set acquisition and processing algorithm

2.2 Overview

As input, our method takes two 3D point clouds of the same dynamic scene, under small to moderate motion: a set of low-resolution point clouds obtained at 30fps from a structured-light infrared depth sensor (such as Kinect) during the motion, and a single accurate but distorted time-of-flight laser LiDaR point cloud taken during the same period of time. While the former provides a low-resolution point set at regular time intervals, the latter provides a highly accurate point set but at a single time stamp t for each point. We will refer to the structured-light frames as LR frames, and similarly, to the time-of-flight data as HR data. In practice, both setups capture depth values of the scene with respect to the device. Due to motion in the captured scene, the HR point cloud appears distorted (Figure 2.1) but each point is precisely captured. The LR frames do not suffer from such time-distortion but exhibit a poor quality: spatially inaccurate and quantized depth values with large noise at a low 0.3 mega-pixel resolution. Our core idea is thus to un-distort the HR data that is accurate in space based on the motion captured by the LR data that is accurate in time. The process if summarized in Figure 2.2.

Our goal is to resample HR points in time to obtain a high-resolution point set for each time frame. This is achieved through three steps. First, we estimate a motion field between the depth sensor LR frames using an off-the-shelf RGB-D optical flow technique. Second, we observe that, up to missing data and noise, the HR point cloud can be exactly registered to the LR data via a *rigid* transform. In the 4D spacetime continuum, we see the LR frames as a set of 3D spatial "slices" taken at regular time intervals, while the HR point cloud is seen as a time-parameterized curve in the 4D volume as each captured 3D point corresponds to a unique time stamp (Figure 2.2). This registration step hence amounts to finding a curve pattern within a 4D volume. We robustly perform this step using a global Iterative Closest Point algorithm initialized via an adaptation of a coarse generalized Hough transform.

Finally, we use the motion field to advect details from the registered HR data across depth sensor frames.

2.3 Scene Flow estimation

To be able to transfer details from the time-of-flight HR acquisition to the LR depth video frames, it is necessary to track 3D points of the LR depth frames through time. A solution to this problem would be to work solely on the depth information and use an algorithm to register Dynamic Point Sets without computing point to point correspondences [54]. However, in our case, the depth sensor provides color information which, if properly registered to the depth frames, gives valuable information. The flow obtained from RGBD images is called *scene flow*. It is a variant of the more common optical flow between color images (e.g. [29]), with the additional challenge that the flow should also account for depth information, and not only the color. In recent years, several approaches have been proposed to solve this problem (e.g. [42, 40, 63]). We use the approach of Quiroga et al. [63] that extends variational optical flow estimation from color image sequences to RGBD videos. This method favors a piecewise smooth scene flow by modeling motions as twists and introducing a total variation regularization.

The computed scene flow provides a way to track a point across all frames until the end of the sequence or until it becomes occluded. It will later be used to advect details from the HR dataset.

2.4 Registration in the 4D space

In the 4-dimensional spacetime volume, the LR data provides a set of regularly spaced 3D hyperplanes while HR data provides a single curve parameterized by the time t (see Section 2.2). We adopt a two-step procedure to align the curve to the 3D hyperplanes. We see this operation as the problem of searching for a pattern within a point cloud. We initialize this search using a coarse, discretized, Hough transform, which we fine-tune in a second step using an ICP. Because of efficiency concerns, we first register the data in the 3D space only, disregarding the time component completely, where we interpret the two point clouds as their projections along the time axis to the 3D space. Then, we perform both the coarse and fine registration

steps again, this time on 4-dimensional point sets. This section describes these steps in more details.

2.4.1 Problem formulation

If both the LR and HR sensors are located at the same place, share the exact same field of view and the capture starts at the same time t_0, then all captured points are completely aligned in spacetime and share the same spacetime coordinate frame. However, this is never the case as both cameras capture different portions of the scene and are hard to synchronize. In addition, the raw HR data do not directly include a time-stamp for each captured 3D point, instead we roughly estimate the time-stamp using the total acquisition time and the scanline pattern of the acquisition. The first operation we perform is thus a registration procedure that brings both datasets to the same space. This amounts to estimating a rotation and translation in space, together with some translation and possible scale in time, to match one dataset to the other. Moreover, due to the inaccuracy of the LR frames, adjusting a scale in space is also necessary for a good alignment. The whole registration thus corresponds to the search for a single global 3D spatial rotation $R = (\theta_x, \theta_y, \theta_z)$, a 4D spacetime translation $T = (T_s, t_t)$ and a 4D spacetime scale $s = (s_s, s_t)$.

We parameterize R by three Euler angles, the translation T_s by three coordinates, and t_t, s_s and s_t are three scalars. This registration procedure amounts to estimating 9 parameters. We will denote the entire 9-d transformation $\mathbf{T}$. Denoting $\mathcal{H}$ the HR point set (resp. $\mathcal{L}$ the low-resolution structured light data) in 4D, we formulate the registration problem as the minimization:

$$\min_{\mathbf{T}} \sum_{p} \|\mathbf{T}p - q\|^2,$$

where $p \in \mathcal{H}$ and $q \in \mathcal{L}$ is the closest LR point to p.

2.4.2 Coarse initialization

Rough estimates of these parameters can be obtained using a voting scheme akin to the generalized Hough transform traditionally used for detecting shapes in images, by discretizing a well-chosen parameter space, and computing the scores of each set of parameters. Historically, the Hough transform was first introduced to detect straight lines on images by discretizing line parameters and scoring them [28]. It

was later extended to more general shapes by discretizing a space of shape template transforms [5].

In our case, we detect our HR curve on the LR volume by discretizing the 9-dimensional space of transformation parameters described in Sec. 2.4.1. However, given the sheer amount of data and the curse of dimensionality affecting our 9-dimensional space, some adaptations are needed.

First, it's easy to see that by placing both the consumer depth sensor and the LiDaR system in an upright position facing the scene θ_x and θ_z become negligibly small for the purpose of the coarse registration step, so they can be safely omitted. Then, scaling in space is only included to make up for the inaccuracy of the LR data and thus it should be close enough to 1 not to alter the results of the Hough Transform applied at such a low resolution. Considering these observations the 9-dimensional parameter space $\mathbb{P}$ is first safely reduced to a 6-dimensional space $\mathbb{P}'$.

As mentioned previously, the registration procedure is divided into an initial phase computing only the transformation $\mathbf{T}_s$ in space and a final one computing the translation t_t and scale s_t in time. In the context of the Hough Transform this means that $\mathbb{P}'$ can be further divided into two separate parameter spaces $\mathbb{P}_s(\theta_y, t_x, t_y, t_z)$ and $\mathbb{P}_t(t_t, s_t)$ which are discretized at a given resolution.

For each set of parameters of the form $\mathbf{T}_s = (\theta_y, t_x, t_y, t_z)$ and $\mathbf{T}_t = (t_t, s_t)$, corresponding to a bin in $\mathbb{P}_s$ and $\mathbb{P}_t$ respectively, we compute a score which represents how well the transformed point cloud $\mathbf{T}_s\mathcal{H}$ in space and $\mathbf{T}_t\mathcal{H}$ in 4D spacetime are registered to $\mathcal{L}$ and seek to maximize it. When working on $\mathbb{P}_s$, the LR data is converted, at a given rough resolution $\tilde{\mathcal{L}}$, to a voxel grid $\mathcal{V}_{\tilde{\mathcal{L}}}$ with each voxel $\mathbf{v}$ storing the number of points lying in it. Then the HR data is transformed using $\mathbf{T}_s$ and discretized into a grid $\mathcal{V}_{\widetilde{\mathbf{T}_s\mathcal{H}}}$ using the same resolution as $\tilde{\mathcal{L}}$. The score of $\mathbf{T}_s$ is computed as $\sum_{\mathbf{v}} min(\mathcal{V}_{\tilde{\mathcal{L}}}(\mathbf{v}), \mathcal{V}_{\widetilde{\mathbf{T}_s\mathcal{H}}}(\mathbf{v}))$. The final solution is found as the transformation $\mathbf{T}_s$ with the highest score.

In the case of time registration, 4-dimensional voxels would not allow a high enough resolution neither in the parameter space $\mathbb{P}_t$ nor for the voxel box. Our alternative solution is to search for a transformation $\mathbf{T}_t$ that minimizes the point-wise distance between $\mathbf{T}_t\mathcal{H}$ and $\mathcal{L}$.

2.4.3 Fine registration

The solution of the coarse registration step is only known up to the precision of the parameter space discretization. This is insufficient as we aim at transferring

millimeter-scale details. Hence, in a second step, we refine the rough estimate both in the 3D case first and then in the 4D spacetime. As the coarse solution is assumed to be close to the optimal solution, we can now resort to a local optimization, namely an ICP [10], to refine $\mathbf{T}$. Let $\mathcal{H}' = \mathbf{T}\mathcal{H}$, the new transformation $\mathbf{T}'$ is found by iterating the classical two steps: 1) assign to each point $p_i \in \mathcal{H}'$ its closest point $q_i \in \mathcal{L}$ (if no point is found closer to a given threshold then the point is simply omitted). 2) Find the transform $\mathbf{T}'$ minimizing:

$$\sum_i \|\mathbf{T}' p_i - q_i\|^2,$$

which is solved using Kabsch algorithm for the translation in 4D and rotation in space. The scale in time is solved by computing the standard deviation of the time stamps of the matched points σ_p and σ_q of $\mathcal{H}'$ and $\mathcal{L}$ and deriving the scale as $s_s = \frac{\sigma_q}{\sigma_p}$. Conversely, the scale in space is computed by averaging the ratios of the 3D Euclidean distances between HR point pairs and their LR counterparts as follows:

$$s_s = \frac{1}{|\mathcal{H}'|^2} \sum_{i,j} \frac{\|p_i - p_j\|}{\|q_i - q_j\|}.$$

At the end of the iterations the HR points p_i with no sufficiently close LR counterpart q_i are considered occluded from the point of view of the LR sensor and are discarded.

The result of the spacetime registration can be seen in Figure 2.3: for different time values t, we display the LR and HR points that lie in a small temporal neighborhood around t, showing that our spacetime registration matches well the datasets despite the difference in resolution.

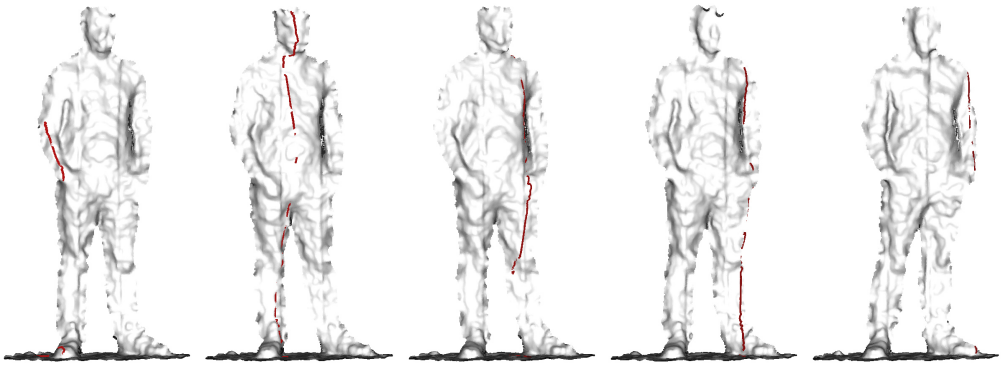


Fig. 2.3: Registration result. For 5 different time values t, we select LR and HR points that lie in a small temporal neighborhood after the spacetime registration. The LR points are oriented and displayed in grayscale values while the HR points are shown in red. The LiDAR scanline acquisition process results in vertical lines.

2.5 Detail Transfer

Now that both the high and low resolution point clouds share the same spacetime coordinate frame, the last step of our algorithm advects the HR point cloud to enrich all frames of the LR point cloud. To do so, we rely on our estimate of the scene flow between consecutive LR frames (see Section 2.3), and advect HR points accordingly.

Let us consider a point $p \in \mathcal{H}$ and F^t its closest LR frame in time. A search in the 3D space finds $Q_p^t \subset F^t$ as the set of its nearest neighbors in frame t. The motion estimated at each point $q_i^t \in Q_p^t$ is then interpolated in space and time to bring p to the exact timestamp t of F^t. Next p is advanced by one frame in time based on the scene flow of the points Q_p^t. This process is performed iteratively until no sufficiently close nearest neighbor can be found which means p is occluded.

2.6 Results

We validate our approach using two datasets. First, using synthetic data, we make sure our method allows to transfer details with sufficient accuracy, and evaluate any reconstruction error. Second, we showcase our method on real data and show it to be of sufficient accuracy to be used to undistort small to moderate motions.

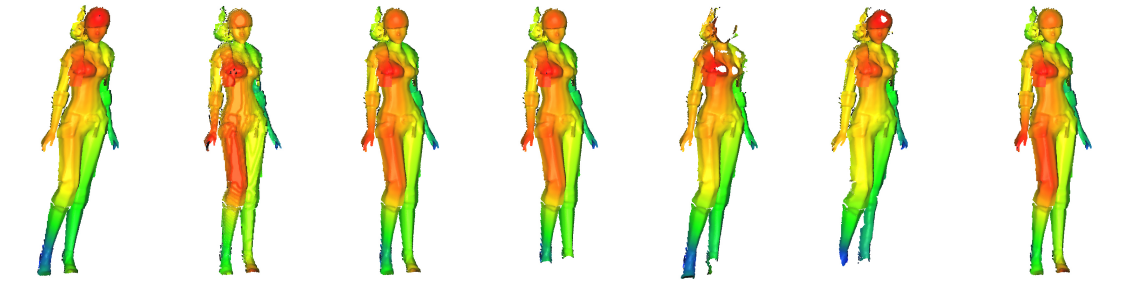


Fig. 2.4: Synthetic dataset along with reconstructed frames for validating various steps of our method as described in Table 2.1

2.6.1 Simulated Data

Our simulated data consists of a single character undergoing a rigid transformation $\mathbf{T} = (T_s, R_s)$, raytraced from different viewpoints using parameters similar to LR and HR devices. This synthetic data allows us to synthesize a distortion-free dynamic point set, serving as a groundtruth, and compare our result to it.

To simulate the noise introduced by the commercial depth sensor we generated depth-bins increasing in size proportionally to the distance from the camera mimicking the quantization errors introduced by the sensor and matched the computed depth values to the depth-bins. However we do not simulate any depth sensor calibration error.

Table 2.1 shows the accuracy of the different steps of our method. By using the known values of the parameters of the 4D registration and the precomputed motion field we assess separately the noise introduced only by the detail transfer. On the contrary, by using no prior knowledge of the scene the average distances indicate the accumulated error introduced throughout the steps of our method.

Figure 2.4 shows the reconstructed frames along with the ground truth of a dataset.

2.6.2 Real Data

We now turn to the more difficult case of real data. Our experimental setup is the following. A LiDaR and an ASUS Xtion sensor acquire the same animated scene yielding respectively a high resolution depth data (corresponding to $\mathcal{H}$) and a low-resolution dense depth video (corresponding to $\mathcal{L}$). Our system does not require any manual calibration: both acquisition systems are only roughly synchronized and

Tab. 2.1: Average point-wise distances between reconstructed and generated ground truth point clouds using either known (GT) registration parameters and motion field or computing them using our method (E). All errors are below 2% of the shape height (2.5 units)

Used methods	Distance
GT registration and GT motion field (1)	0.01271
E registration and GT motion field (2)	0.01463
GT registration and E motion field (3)	0.04368
E registration and E motion field (4)	0.04479

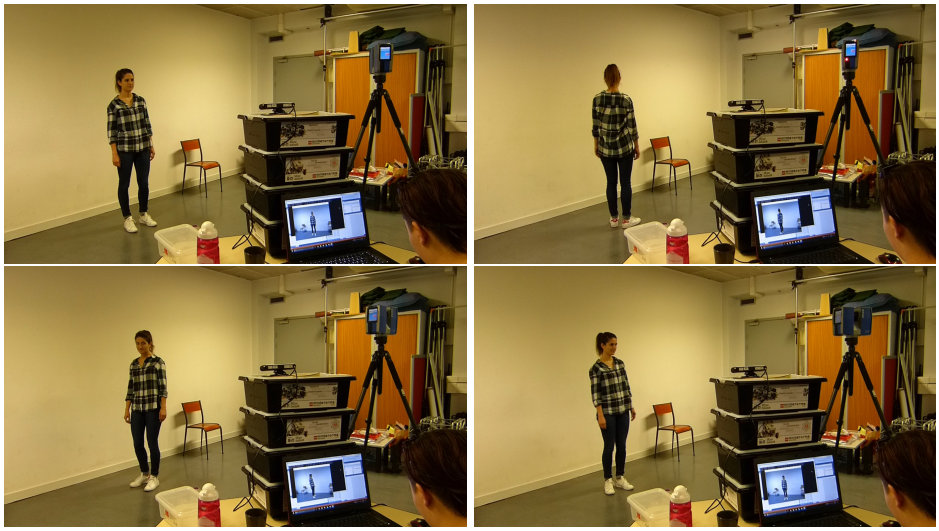


Fig. 2.5: Our experimental setup: a Kinect and a LiDaR acquire the same scene from different viewpoints. No calibration is required.

we only need them to start roughly at the same time (in practice, the Asus depth sensor capture often starts several seconds before the LiDaR as the LiDaR performs an automatic self-calibration procedure at the start of each capture). A picture of the acquisition system is shown on Figure 2.5.

The Xtion sensor captures 30 depth and color frames per second with a resolution of 640×480, the capture is performed using the OpenNi 2 library [59], removing distortion and yielding the final 4D point cloud in millimeters and seconds. In practice, we observed an accuracy of roughly $\pm 5cm$ at $1.5m$, which corresponds to our capture distance. To compensate for noisy depth values in the LR sequence, we filter depth values using a bilateral filter with standard deviations in space and values : $\sigma_s = 1.16$ and $\sigma_v = 64$.

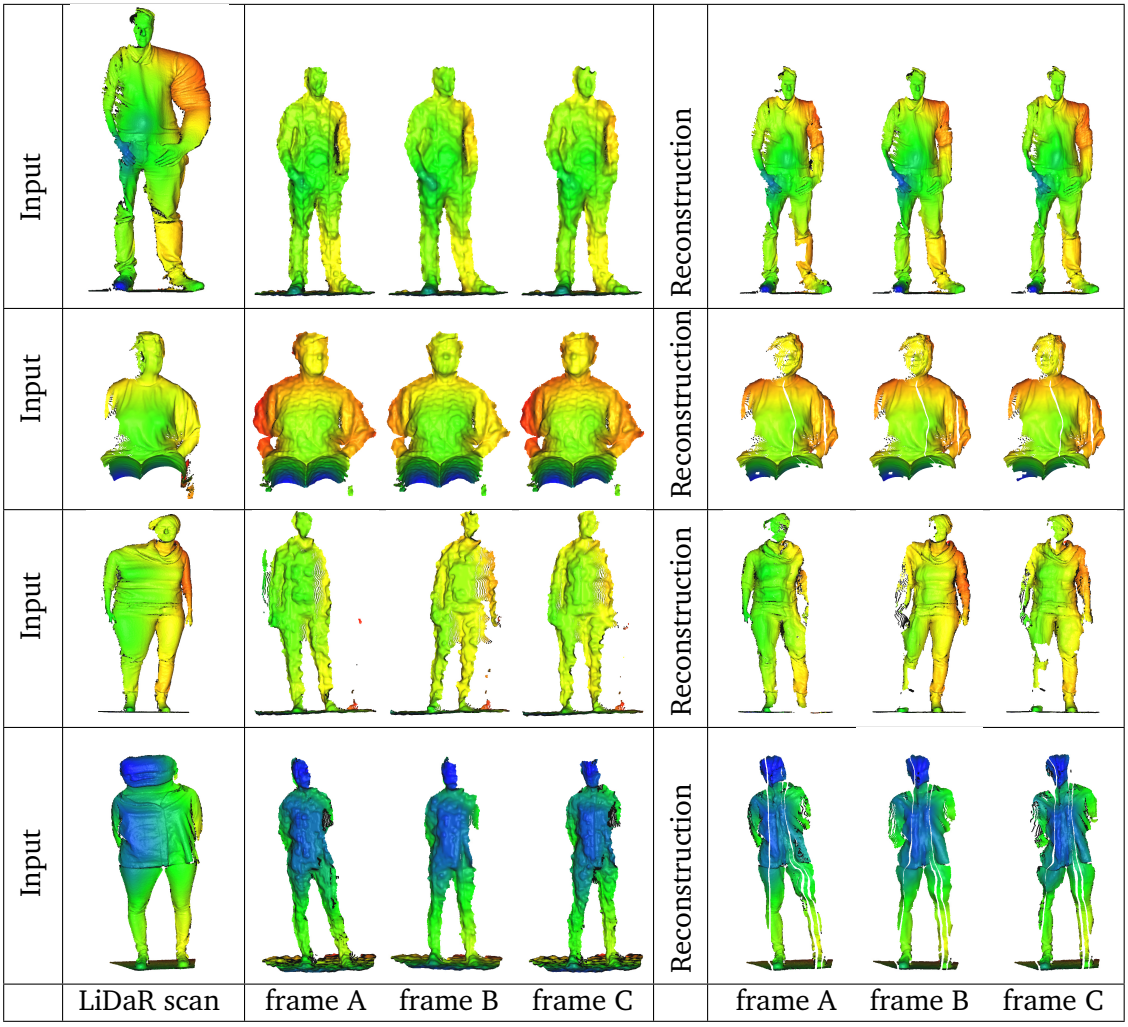


Fig. 2.6: Undistorting LiDaR scans (left) from consumer depth camera sequences (center). Our result (right) show higher spatial accuracy than the consumer depth camera while respecting the global motion. Video results can be seen in supplemental materials.

The LiDaR scanner is a FARO Laser Scanner Focus^S X 330. It was set to capture one point every 50mm at a $2m$ distance. A laser ray is emitted and reflected by a mirror that directs the beam towards the scene and controls the angle of the ray. By rotating around its axis, this mirror allows for a complete rotation of the laser ray, however it only measures the time of flight on a given angular range (set here to $150°$). Furthermore the device itself rotates around the vertical axis, as illustrated in Fig. 2.7. A full scan at this resolution takes around $14s$. To prevent large static objects giving too much weight to the spatial registration compared to temporal variables, we cropped out the walls and ground of the HR scan.

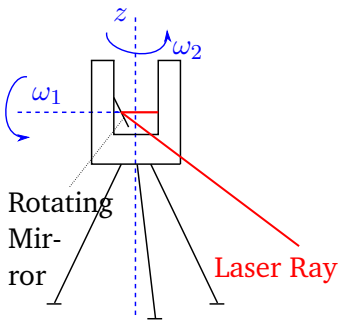


Fig. 2.7: Functional schematic of the LiDaR Scanner.

We organized two acquisition sessions in total. Our first trial highlighted several possible points of improvement. First, our original implementation to capture the depth videos was slow and caused significant drops in the frame rate. This hindered our efforts both to register the point clouds, especially in the case of the fine registration which relies heavily on the distances between corresponding points in the spacetime, and to track the motion across the scene. We improved the speed of the acquisition by optimizing the data streaming process, although random small delays of 2-3 frames at a time still persisted.

Second, diverging from our original plan to solely use the depth videos, we decided to leverage the color information captured by the Xtion device, to compute a much more robust scene flow. In order to do so, we needed to account for the small displacement between the two sensors on the device. We used the OpenNI 2 library at capture time to register the two streams.

Figure 2.6 shows the results of our method applied on several real datasets. By comparing the reconstructed frames to the LiDaR scan, one can see the high frequency details transferred across the frames. The accuracy of the overall motion of our reconstructed sequence can be assessed by observing the corresponding LR frames and the accompanying video.

2.6.3 Limitations

Our method has some limitations. First it can only handle small motions. Large motion over a long time will generate too much occlusions in the LiDAR data, and the spacetime registration and point tracking will fail, creating artefacts illustrated in Figure 2.8. Missing regions can appear due to points being occluded or unobserved in the HR scan during the motion or abnormal time delays between consecutive LR

Fig. 2.8: Failure case: large motion are not handled well by our method. Left: Lidar Scan. Depth sensor frames (top row) and corresponding corrected frames (bottom row).

frames that cause whole slices of the HR data not corresponding to any LR frame to be lost during the ICP. This produces vertical stripes of missing points in the results (see the last row of figure 2.6). In this case, merging the LR and HR data would allow for filling in holes.

Calibration The consumer depth sensor further suffers from heavy distortions. This issue has been identified and investigated by Clarkson et al. [20] and Herrera et al. [15], who both propose calibration procedures. We tested the approach proposed by Herrera et al. [15], which jointly optimizes for color and depth camera calibration parameters using a standard checkers board setup akin to the one in Fig. 2.9. We found that it did not substantially improve our depth videos and much of the distortion remained unchanged. Seeing these results, and our desire to remain calibration-free, we did not investigate this issue any further. However, our reconstructions are of limited accuracy, exhibiting spatially low-frequency artifacts, that can be attributed to these distortions.

Aside from a calibration procedure, our registration process and hence the final reconstruction could be improved by swapping out the ICP algorithm for a non-rigid registration step. We hope even better results could be achieved in the future using this approach.

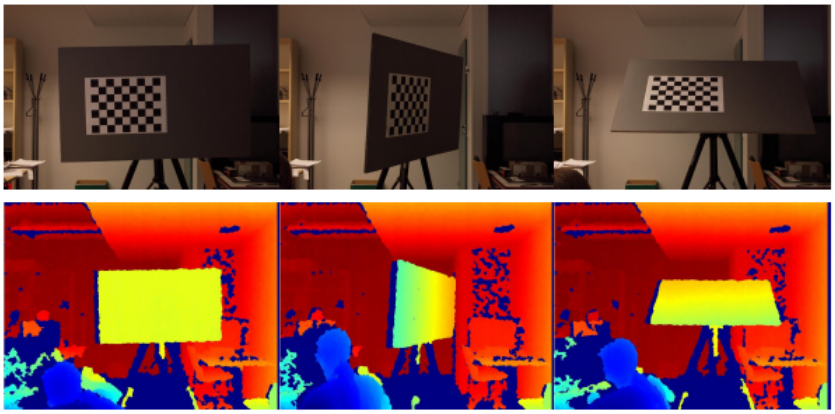


Fig. 2.9: Sample color (top row) and disparity (bottom row) images used to calibrate consumer depth sensors. Image taken from Herrera et al. [15].

2.7 Discussion

We introduced a way to capture animated scenes and produce high resolution point sets by combining a consumer depth sensor and a high precision Time-of-Flight scanner. We showed that by formulating the problem in spacetime we were able to register the datasets and advect details across the frames to undistort moderate motion sequences.

3 Dynamic Scene Novel View Synthesis via Deferred Spatio-temporal Consistency

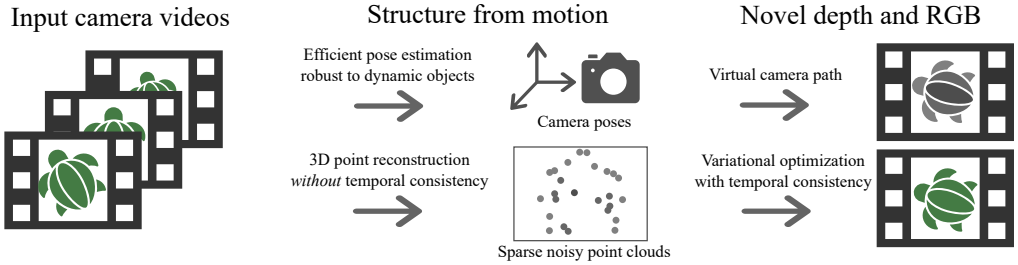


Fig. 3.1: Given a small set of video sequences of a performance, our method computes camera poses and sparse points, then optimizes those points into a novel video sequence following a user-defined camera path. Our space-time SfM intentionally does not compute temporal consistency for points on dynamic objects and instead defers spatio-temporal consistency in both depth and RGB reconstructions to the novel view synthesis stage via our variational formulation.

In this chapter we tackle the problem of novel-view synthesis (NVS), which creates a new view of a scene by combining existing images captured from different viewpoints. Much progress in NVS has been made over the past two decades to tackle its two core problems: 1) how to build a proxy scene geometry to aid in rendering, such as constructing simplified sparse depth points or a piecewise planar mesh via structure from motion (SfM), and 2) how to interpolate or extrapolate an image via the reprojected proxy given the existing captured imagery. NVS increases in difficulty across many axes: as the cameras become farther apart (wide baseline), as their number decreases (few camera), as they become handheld (casual capture), as the scene itself contains motion (dynamic scene), as the scene phenomena become more visually complex (geometry, materials, and motion), and as the time given to generate the result decreases (compute cost).

We consider dynamic scenes captured by a small number of cameras (5–12) over baselines of around 60°, as might occur with a crowd of people capturing an event (Figure 3.1). Within this scenario, we include sequences with casual handheld cameras. This is a relatively rare and challenging setting because both the cameras

and the scene objects move simultaneously, and because sequences with only a small number of casual cameras make robustness hard to obtain. This complicates camera pose estimation and depth estimation in SfM and, if the proxy geometry is not perfect, causes ghosting, bleeding, and flickering artifacts across views and time during NVS in both moving objects and the background. Thus, one key component of any algorithm is a way to enforce spatio-temporal consistency in both the SfM and the NVS to reduce these artifacts.

We propose to address these challenges by *deferring* the difficult problem of reconstructing dynamic objects in time via SfM, and instead using a NVS approach to enforce temporal consistency. To ease the task of reconstructing dynamic scenes via SfM, many approaches first segment out moving objects or feature points and process the static background and the dynamic foreground separately [73, 57, 58]. Instead, we first recover camera poses for all views without any explicit dynamic object segmentation. Then, we recover scene points on both static and dynamic objects without temporal consistency and performing per-frame SfM across views only. This is easier to solve, but leads to significantly noisy reconstructions temporally.

Next, we turn our sparse (and noisy) reconstructed point clouds into novel views. This is commonly completed by densifying points into a depth map [41] for each view in a consistent way, and using the depths to reproject and merge input RGB views into a virtual view. We present a formulation which only densifies a depth map in the virtual camera's view, rather than for all input views, which leads to a more efficient solve. For this, we take a coarse-to-fine variational approach and solve a diffusion-based formulation. Importantly, this formulation lets us enforce robust temporal consistency in the output depth to overcome the initial noisy reconstructions from the SfM. To determine our final RGB values, we also solve for the output color within the coarse-to-fine variational formulation.

We perform comparisons to recently-proposed approaches in point densification and view interpolation, using both optimization and learning-based approaches. Further, we show results on a synthetic dataset in an ablation study. In a nutshell, we show that considering SfM and NVS together allows us to ease the difficult temporally-consistent reconstruction problem and instead cope with it at the rendering stage. Overall, our work takes another step forward in improving digital content creation for scenes captured by multiple video cameras.

3.1 Related Work

Rendering a novel viewpoint of a real-world scene captured with photographs is a problem that has received much attention over the past 30 years [79].

3.1.1 Static scene IBR

Image-based rendering (IBR) has initially attempted to render static scenes either from set of images or videos. This can be achieved either via warping input views using optical flow [18], using coarse geometric proxies [33] or via deep learning approaches [30]. In complex environments, IBR techniques often need some 3D proxy reconstruction. For example, the Lumigraph [33, 14] uses planar or coarse geometric proxies; Shade et al. [67] used multiple planar sprites; and Debevec et al. [25] employed photogrammetric reconstructions of buildings. Others have used 3D meshes from multi-view stereo reconstructions [69, 39]. For instance, Chaurasia et al. [17] proposed a depth-based synthesis using planar superpixel patches [1]. Matzen et al. [51] used two spherical cameras to synthesize an omni-directional stereo panorama. Recently, Riegler and Koltun [64] synthesized new views via neural textures atop a Delaunay reconstruction of sparse points obtained from video of static scenes. Beyond surface geometry, NeRF [53] performs an expensive optimization to create a volumetric function that is then rendered to synthesize new views.

Solving problems in the gradient domain can help too; for instance, to achieve smoother interpolations [46] or to densify sparse scene points. Holynski and Kopf spatio-temporally propagate sparse depth samples in a single view by solving a Poisson problem [41]. This method relies on camera motion to detect depth edges, which limits it to static scenes. Inspired by gradient domain approaches, we formulate a variational approach that jointly enforces depth smoothness and consistency, color smoothness and consistency, as well as temporal consistency. Our approach additionally works with multiple potentially-dynamic cameras, and introduces a view-consistency term to ensure geometric consistency between views.

Deep learning can also be employed for static scene IBR. This includes plane sweep volumes [31] and multi-plane images to interpolate between two static narrow-baseline views [83] or between multiple views at once [52, 30], appearance flows to generate novel views from a single image of isolated objects [84], and light-field view interpolation [43]. Hedman et al. [38] use a geometric proxy and learn blending weights between view reprojections using a CNN. To improve the quality around

depth discontinuities, Choi et al. [19] use a 3D uncertainty volume as a proxy and neural network-based patch refinement. Srinivasa et al. [71] train a CNN to predict a light field from a single image for small-baseline view synthesis. Similarly, Song et al. [70] synthesize new views from a single image of a static scene using deep learning.

While these techniques were not designed for videos and so neither explicitly maintain temporal consistency nor are constrained by speed, we nevertheless compare our approach to relevant methods for static scenes taken frame by frame.

3.1.2 Dynamic scene VBR

For dynamic scenes, please see dos Anjos et al. [3] for an exhaustive survey on video-based rendering (VBR) techniques. The need for controlled capture setting is shared by many methods. Zitnick et al [85] use a specific system of 8 cameras combined with segmentation based stereo to extract the geometry. Similarly Wilburn et al. [76] use an array of 100 tightly-packed cameras. Broxton et al. [13] describe a custom camera array of 46 synchronized cameras mounted on a dome used to capture 6DoF wide-baseline light field videos. Guo et al. [34] relight video with a set up of 331 light sources and 90 cameras, while Collet et al. [21] require 106 cameras. In a less constrained way, Pozo et al. [62] create a 16-camera rig to reconstruct 360 panoramic videos and synthesize new views. Penner and Zhang [61] use a soft volumetric representation for narrow baseline IBR to enforce smooth reconstructions. This method can handle motion, but has trouble handling unstructured data and works best from camera arrays. Our method also works with handheld cameras.

Casually-captured videos have also been considered. Ballan et al. [4] allow for quick transitions between handheld video sequences. Their method segments a single dynamic foreground subject approximated by a planar proxy, and creates a 3D reconstructed static background. To cope with dynamic background objects reprojecting incorrectly, the method blurs background transitions between captured viewpoints. Our method assumes no segmentation nor planarity assumptions for dynamic objects. Lipski et al. [48] use dense correspondence fields to interpolate views between videos. They disambiguate matches in difficult cases by manually drawing correspondence lines on image pairs to use as priors in their matching algorithm. Mustafa et al. [57, 58] reconstruct isolated moving objects after segmenting them out from the initial video. These methods focus on specific object meshes, and so do not provide re-rendering of an entire scene from a novel viewpoint.

Recently, Luo et al. [50] introduced a consistency term by fine tuning a neural network to improve the estimated depth per point. This works for a single camera with no or limited dynamic motion. Bansal et al. [6] use foreground and background extraction together with a self-supervised CNN based composition operator, and Yoon et al. [78] use deep learning to extrapolate new views from a single monocular video camera; we compare our approach to this method.

Outside of NVS, other video reconstruction tasks raise consistency questions. Vo et al. [73] used a spatio-temporal bundle adjustment technique and human motion priors to reconstruct actor performances by temporally aligning videos at sub-frame precision. Bao et al. [7] using deep learning for consistent video super resolution. Finally, Davis et al. [24] recovered depth in dynamic scenes by unifying structured light and laser scanning into a space-time stereo framework.

3.2 Method

Our algorithm takes as input a set of casually-captured synchronized videos. We also provide the focal lengths for a pair of cameras (required by OpenMVG [56]), while the remaining focal lengths are estimated automatically by our algorithm. Our method proceeds in two steps (Figure 3.1):

1. **Camera pose estimation and 3D scene points.** We perform a three-step structure from motion reconstruction to provide both the set of camera poses and a set of sparse 3D points for each time step (Section 3.2.1).

2. **Novel depth and novel view rendering.** We densify the sparse points into a depth map and render a new virtual camera frame by optimizing a coarse-to-fine variational formulation while enforcing spatio-temporal consistency (Section 3.2.2).

3.2.1 Camera pose estimation and 3D scene points.

Let us consider a set of S synchronized video views of a dynamic scene, each composed of T frames. We call $I = \{I_{s,t} | s = 1, ..., S; t = 1, ..., T\}$ the set of all frames indexed by s (camera index) and t (time step). At each frame, via SfM, we recover the camera parameters $\mathbf{C}_{s,t}$ consisting of the intrinsic matrix and extrinsic rotation and translation matrices, and a set of sparse 3D points for each time step. First, we efficiently recover a set of camera poses for all frames. In contrast to

other methods [4, 57, 58], we estimate poses without an explicit dynamic object segmentation step. Second, we recover 3D points by solving a per-timestep SfM problem without a complex temporal reconstruction. We solve each SfM problem with an *a contrario* algorithm [55]. This automatically adapts thresholds to the input data instead of using global thresholds, which is more flexible to different inputs.

Efficient camera pose estimation A straightforward approach for accurate SfM is to solve a problem across all frames simultaneously, but this can be expensive and memory prohibitive. A second approach might consider solving only between consecutive time steps, but this is known to produce camera position drift [22]. Instead, we take a coarse-to-fine approach.

We begin by computing SfM across keyframes at every κ time steps of each video. We detect and match SIFT keypoints within this subset and then simultaneously solve for all camera poses and 3D points. Then, we refine our estimate with a second SfM that only matches keypoints between successive frames of the same camera view, with previously-estimated camera poses held fixed. This considers every frame of every video, but we only match $I_{s,t}$ to $I_{s,t+1}$, and not to $I_{s+1,t}$ or $I_{s+1,t+1}$. To recover smooth camera paths per view, we add two additional penalty terms to the bundle adjustment:

$$w(t-t')\ \|C_{s,t} - C_{s,t'}\|^2, \quad t-3 \leq t' \leq t+3 \tag{3.1}$$

and

$$w(t-t')\ \|A_{s,t} - A_{s,t'}\|^2, \quad t-3 \leq t' \leq t+3, \tag{3.2}$$

where $w(t-t')$ is a Gaussian weight function, $C_{s,t}$ is the center of each camera pose, and $A_{s,t}$ is the angle-axis representation of the rotation matrix $R_{s,t}$. This second SfM reduces computation time over all-pairs matching while still reducing drift by constraining the frame-to-frame pose estimates by the keyframe pose estimates. For hyperparameters, smaller κ will increase processing time, while larger κ may make it more difficult to match fast camera motion. We found $\kappa = 20$ to be a good compromise in our test sequences.

3D scene points To recover 3D points across the scene, we solve a keypoint reconstruction problem that is independent per time step. Taking as fixed the recovered camera poses for each video frame, we match 2D keypoints between frames with the same timestamp, then reconstruct a set of sparse 3D points per time step. This

is our key to handling dynamic scenes: as 2D keypoints are not matched in time, moving objects are correctly recovered in space even if their motion makes matching over time difficult. However, this knowingly produces temporal inconsistencies; we will recover from these errors in novel view synthesis where it is easier to enforce consistency (Sec. 3.2.2).

Post processing Finally, we increase the density of our point matches using PatchMatch [8], as proposed in the OpenMVS[1] and COLMAP[2] [66] frameworks. This process splats points to each view and assigns colors to the 3-D point cloud.

3.2.2 Novel depth and novel view rendering

Our SfM recovers a set of camera poses and an RGB 3D point cloud per time step. However, at this stage of our algorithm, projecting these points to a novel view still leaves large regions of empty space. To synthesize more realistic views, we diffuse these points in depth and RGB in the new view in image space while enforcing spatio-temporal constraints.

Notation We will often warp the content of a frame I_* into the domain of the novel view I_t: this reprojection is computed using the extrinsic and intrinsic parameters of both reprojected frames and virtual camera, as well as the depth map D with values d associated to each pixel. We will denote it $I_*^{proj}(x) = I_*(\mathbf{C}_*\mathbf{C}_t^{-1}(x_t, d_t))$, where $\mathbf{C}^{-1}(x, d)$ is the image plane to world coordinate system transformation of the pixel location x given its depth value d. We also denote by $\hat{\cdot}$ a sparse map. The sparse depth map obtained by projecting the sparse point cloud into frame t of the new virtual camera path is then $\hat{D}_t$ and its corresponding sparse color image $\hat{I}_t$.

Algorithm progression We wish to warp a frame $I_{s,t}$ to the novel view I_t to be blended into a final novel view. For this, we need both the estimated camera poses and the dense depth maps D_t, which are yet to be computed. But, to properly constrain the diffusion of the sparse depth values $\hat{D}_t$, recovered in Sec. 3.2.1, we need RGB information from the virtual camera's point of view. Thus, we jointly solve for the depth maps D_t and color images I_t by minimizing the energy functional:

$$E = E_D + E_I. \tag{3.3}$$

[1] https://github.com/cdcseacave/openMVS
[2] https://colmap.github.io/

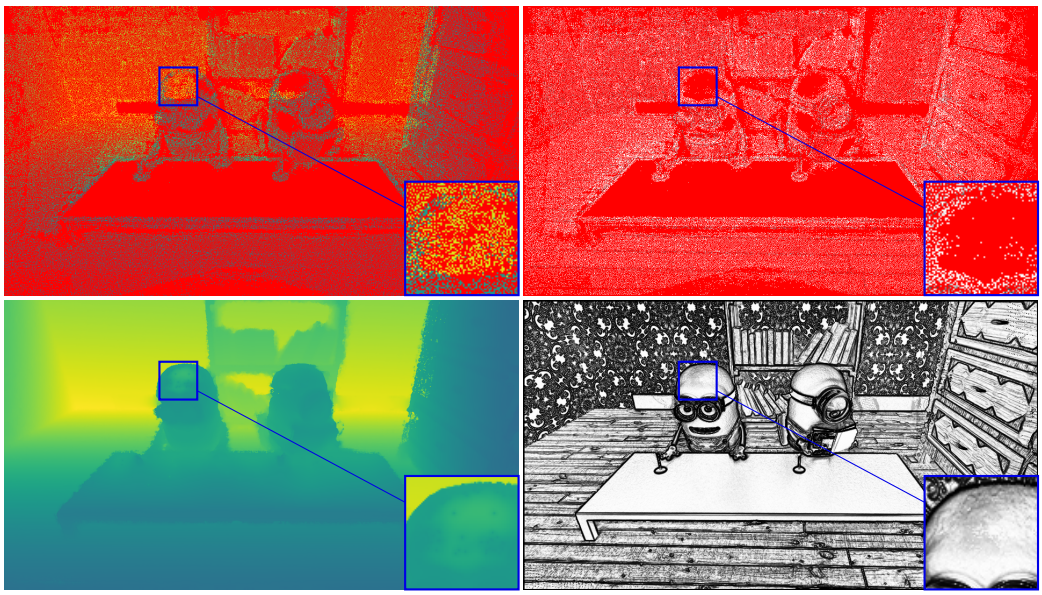


Fig. 3.2: *Top row:* Sparse reconstructed 3D points (left) and their weights $w_{\hat{D}}$ (right) projected into the virtual view. Red indicates areas of empty space; depth map is bright green in far depth regions. *Bottom row:* Points diffused into a full depth map D (left) according to the weight map w_D (right). Note how the how the color edges are correctly identified via Eq. 3.5, and how the occluded points from behind the head of the character on the left are given no weight by Eq. 3.6 (top right) and so do not corrupt the depth.

The functional relates terms constraining the depth map (E_D) to terms constraining the color image (E_I) by weights that guide the diffusion process. We solve E iteratively: we first solve for the depth map D_t while fixing the color values I_t, and then conversely we fix the depth values and solve for color. This avoids having to solve a nonlinear system of equations, and lets us use slightly-improved depth values to warp the input frames at each step. This betters the estimate of the rendered RGB image, which in turn constrains the diffusion of the depth.

Depth diffusion We project the sparse point cloud into the novel view, creating the sparse depth map $\hat{D}_t$ as an initialization. Then, we densify it by minimizing the following energy:

$$\begin{aligned} E_D &= \int_{x\in\Omega} w_D(x,t)\|\nabla D_t(x)\|^2 \mathrm{d}x \\ &+\lambda_{PC}\int_{x\in\Omega} w_{\hat{D}}(x,t)\ \|D_t(x)-\hat{D}_t(x)\|^2 \mathrm{d}x. \end{aligned} \tag{3.4}$$

The first integral is a smoothness term controlled by weight w_D. We wish diffusion to decrease around color edges to produce sharp results. We also wish diffusion of

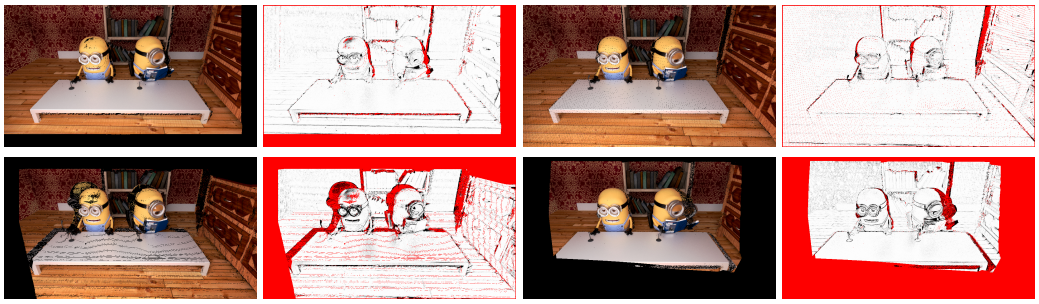


Fig. 3.3: Four closest input images $I_{s,t}$ projected onto the virtual camera's view point alongside their corresponding weight maps w_P (Eq. 3.8).

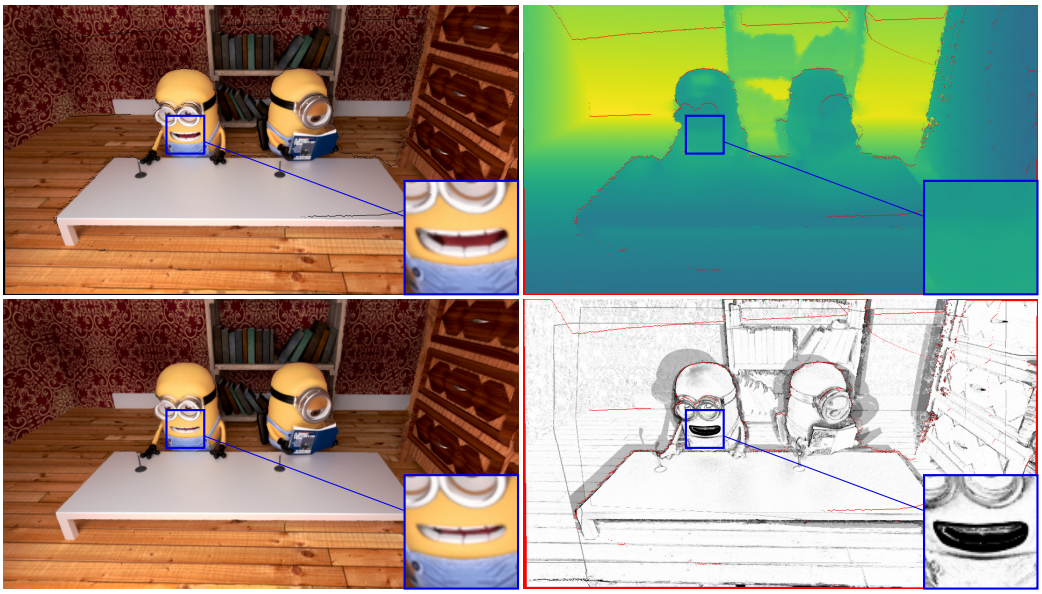


Fig. 3.4: *Top row:* Color image I_{t-1} and depth map D_{t-1} of a previous time step. *Bottom left:* I_{t-1} reprojected to the camera view by D_{t-1} of the current time step. *Bottom right:* Weight map w_T (Eq. 3.11) modulates consistency, notably around the moving mouth of the character on the left.

depth values to increase when the colors from reprojected input views are similar. As such, we define w_D as:

$$w_D(x,t) = \frac{1}{\|\nabla I_t\|^2 \sum_{s=1}^{n} \sigma_{vis}^s(x,t)} \sum_{s=1}^{n} w_P^s(x,t), \tag{3.5}$$

where $1/\|\nabla I_t\|^2$ modulates depth diffusion around color edges, and $1/\sum_{s=1}^{n} \sigma_{vis}^s(x,t)$ is a normalization factor that accounts for each pixel's visibility in the novel view. As both the visibility term σ_{vis} and the projection weight w_P^s pertain more to the color diffusion process, we will defined them later on in Eq. 3.8.

The second integral reduces the weight of sparse 3D points that are occluded from the point of view of the virtual camera or are erroneously reconstructed. For this, we relax the constraint of D_t where it exactly matches the projected sparse point cloud:

$$w_{\hat{D}}(x,t) = \exp\left(-\frac{\|\hat{I}_t(x) - I_t(x)\|^2}{2\sigma^2}\right). \tag{3.6}$$

In Figure 3.2, we show example weight maps $w_{\hat{D}}$ and w_D that govern the depth diffusion process, as defined in Eqs. 3.5 and 3.6.

There are three parameters in this diffusion process: σ controls the soft occlusion tolerance, and we set $\sigma = 0.075$ in all our experiments; the sparse point cloud attachment weight λ_{PC}, which we set in the range $\lambda_{PC} = 0.25$–2; and the temporal consistency term set in the range $\lambda_T = 0.01$–0.1.

Color diffusion Given depth map D_t, we initialize the RGB image to a projection of the color in the input point cloud. Then, we densify it by minimizing the following diffusion energy:

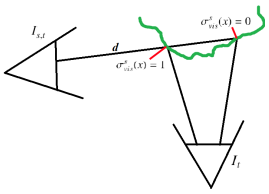
$$\begin{aligned} E_I = &\int_{x\in\Omega} \|\nabla I_t\|^2 \\ &+ \sum_{s=1}^{n} \int_{x\in\Omega} \lambda_P w_P^s(x,t) \|I_t(x) - I_{s,t}^{proj}(x)\|^2 \mathrm{d}x \\ &+ \sum_{s=1}^{n} \int_{x\in\Omega} \lambda_G w_P^s(x,t) \|\nabla I_t(x) - \nabla I_{s,t}^{proj}(x)\|^2 \mathrm{d}x \end{aligned} \tag{3.7}$$

The first integral encourages smooth gradients over the intensity of the novel view, which aids blending of the projected input images especially along their borders. The second integral constrains the RGB intensities and I_t to be close to the intensities of

$I^{proj}_{s,t}$, and the third integral constrains the RGB gradients similarly. They are both modulated by the weight

$$w^s_P(x,t) = \sigma^s_{vis}(x,t) \exp\left(-\frac{\|I^{proj}_{s,t}(x) - I_t(x)\|^2}{2\sigma^2}\right), \tag{3.8}$$

which measures the agreement of each warped input frame with the novel view. Figure 3.3 shows a set of warped input frames along with their weight maps w^s_P.



w_P incorporates visibility term $\sigma^s_{vis}(x,t)$ that is 1 for a given pixel x of the novel view I_t only if, out of every pixel that is projected to the same pixel location in an input image $I_{s,t}$, x has the smallest depth value d in the input image's coordinate frame.

The color diffusion relies on two new parameters: λ_P and λ_G balance the weight over the data and the gradient equality constraints. We set them both in the range 5–20. σ and λ_T serve the same function and values as in the depth map diffusion.

3.2.3 Temporal consistency

We enforce temporal consistency within novel views by additional terms in E_D and E_I. With slight abuse of notation:

$$E_D = \cdots + \lambda_T \int_{x\in\Omega} w_T(x,t)\ \|D_t(x) - D^{proj}_{t-1}(x)\|^2 \mathrm{d}x, \tag{3.9}$$

$$E_I = \cdots + \lambda_T \int_{x\in\Omega} w_T(x,t)\ \|I_t(x) - I^{proj}_{t-1}(x)\|^2 \mathrm{d}x. \tag{3.10}$$

These terms constrain depth D_t to remain similar to the warped previous depth d_{t-1}, and for color I_t similarly. This constraint is relaxed by a weight

$$w_T(x,t) = \frac{1}{n}\sum_{s=1}^{n} \exp\left(-\frac{\|I^{proj}_{t-1}(x) - I^{proj}_{s,t}(x)\|^2}{2\sigma^2}\right) \tag{3.11}$$

for pixels for which an agreement in color was not reached. This is expected in regions containing motion because the depth values of frame $t-1$ may be invalid,

as is the case around the mouth of the character on the left in Figure 3.4. w_T allows the computation of depth and color values of these pixels to rely more freely on the other terms of the functional, like the data term of the depth or the color of the warped input images.

3.2.4 Implementation details

To avoid using input frames that are far away from the novel camera's view, we rank each input camera based on its distance from the novel camera according to the following formula:

$$r_F(s) = \frac{1}{\|C_t - C_{s,t}\|^2} \exp\left(-\frac{\arccos\left((\mathrm{tr}(R_t R_{s,t}^T) - 1)/2\right)}{2\pi\sigma^2}\right) \tag{3.12}$$

This penalizes frames that are either far in center or in viewing direction from the novel view. Then, we use the first $n = 4$ ranked input frames to minimize the functional Eq. 3.3.

For efficiency, we also proceed in a multiscale fashion: we solve for depth and color at a coarse resolution, and then use these to initialize a finer resolution—our lowest level is 1/64 of the original frame size. Finally, we also proceed in a streaming manner: we reproject the previous frame's depth and color (denoted as D_{t-1}^{proj} and I_{t-1}^{proj}) into the current virtual camera pose for use within the temporal consistency constraint.

3.3 Experiments and Results

3.3.1 Dataset Sequences

Real-world existing dataset We exploit existing datasets used in the context of novel view synthesis, all of them captured using camera arrays:

- Jumping [78]: A group of four people jump (12 cameras).
- Skating [78]: A person rides a skateboard (12 cameras).
- Playground [78]: A person flies a dinosaur balloon (12 cameras).
- Umbrella [78]: A person opens and rotates an umbrella (12 cameras).

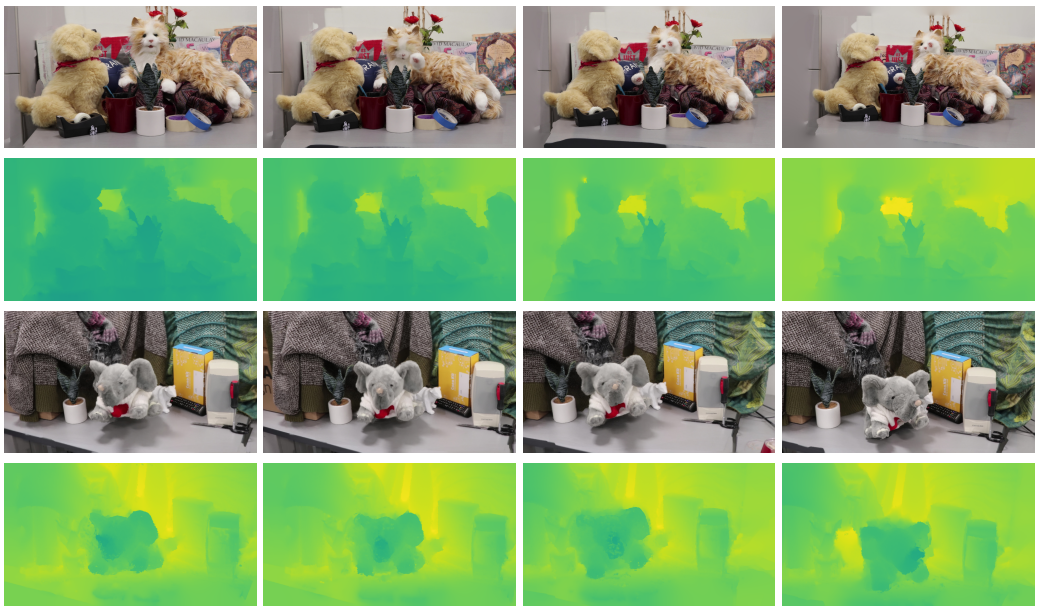


Fig. 3.5: Color and depth results for Cat&dog and Elephant-wiggle scenes.

- DynamicFace [78]: A person of making faces (12 cameras).
- Breakdancers [85]: A person break dancing in front of 4 people (8 cameras).

Custom dataset We test our algorithm on three 100-frame real world sequences that we acquired each with five cameras at 1920×1080 resolution. The cameras were hand held or set on tripods (Canon Rebel EOS T7i). We additionally generate a synthetic sequence using 11 input cameras to compare to ground truth RGB and depth estimation from a 12th camera. Our sequences are:

- Cat and dog: Two pet animatronics,
- Minions (synthetic): a rendering of two characters laughing behind a table,
- Elephant wiggle: A puppet hanging by a wire, and
- Drone: A drone hanging by a wire.

Figure 3.5 shows rendered frames from novel views and corresponding depth maps for the Cat and dog, and the Elephant wiggle sequences. While some artifacts remain in the depth video, the generation of the final novel view RGB rendered sequence is robust to these and has fewer artifacts. Note that the borders of the view partially appear blurry when there is insufficient field of view overlap between input videos.

Method	PE (mm)	OE (°)	Median	Mean RE (pix)
Naive SfM	0.045	2.181	0.080	0.148
Static objects only	0.044	2.149	0.079	0.141
No path smoothing	0.045	2.187	0.074	0.144
Our SfM	0.045	2.180	0.074	0.144
Ground Truth Poses	0	0	0.079	0.149

Tab. 3.1: Ablation Study. We compare estimated camera pose accuracy for naive SfM, naive SfM using a ground truth mask for the static parts of the scene, and an ablated version of our space time SfM without camera smoothing. While adding smoothing slightly has little effect on the positional error (PE), it reduces orientation error (OE). Our approach also minimizes the median reprojection error (RE) of the feature points. SfM minimizes reprojection errors by construction which explains why using ground truth camera poses increases reprojection errors, but results in perfect camera poses by construction.

3.3.2 Ablation study

We ablate our SfM method using the synthetic dataset with moving objects, where points are known to be either static or dynamic (Table 3.1). We compared the recovered pose over 30 timestamps and 11 cameras. First, we compare against a naive SfM approach that solves for all frames simultaneously without consideration of dynamic objects. Next, some methods rely on segmenting out moving objects to cope with dynamic scenes [57, 58]. To compare to this idea, we created a segmentation-based SfM baseline from the naive SfM by performing reconstruction only from points that are known to be static using perfect ground truth masks. While the segmentation slightly aids the recovery of camera positions, its positive effect is not clear on the 3D reconstruction, even though the dynamic object segmentation is a pixel accurate ground truth. Against both baselines, our method can make better use of dynamic points to more accurately recover scene points. Finally, we compare against the non-smoothed camera path version of our approach. While the rotation error decreases, the positional error slightly increases. Overall, we found smoothing to provide better final results.

3.3.3 Novel depth and view comparisons

We compare our method to four recent methods, including deep-learning-based methods requiring external training databases: Deep Blending [38], Local Light Field Fusion [52], Extreme View Synthesis [19], and MonoCam [78]. Furthermore, we use the Breakdancers scene to compare to the results provided by two older

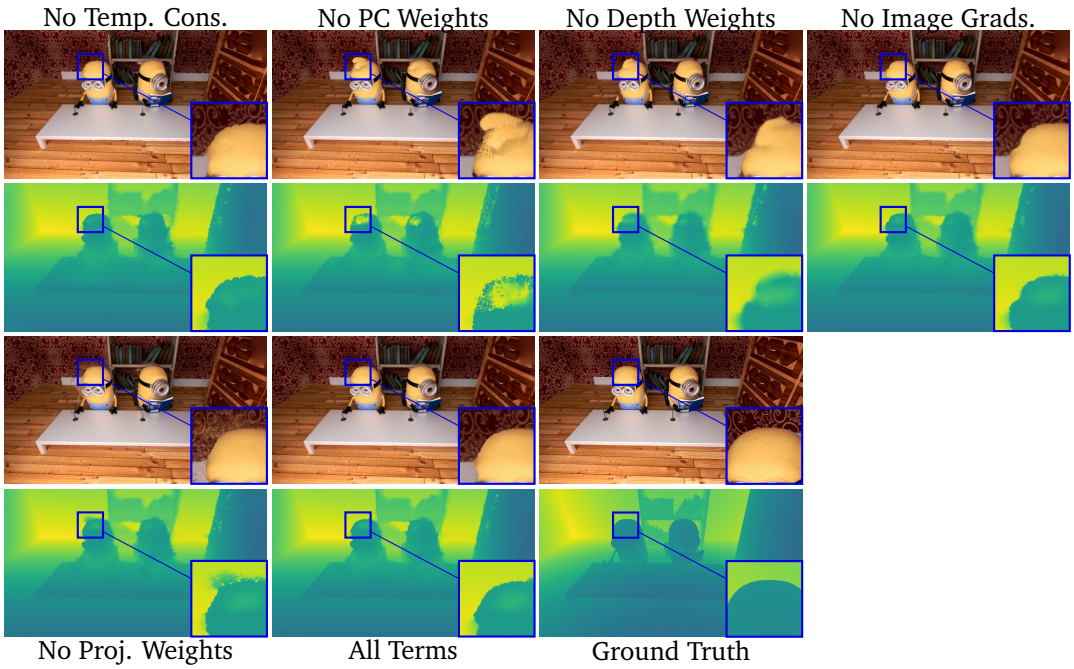


Fig. 3.6: Rendering Ablation Study. We show our rendered result without temporal consistency (1st column), without weights on the projected sparse 3D points (2nd column), without depth weights and inverse image gradients (3rd column), only without inverse image gradients (4th column), without weights on the projected input images (5th column), together with the full result (6th column) and the groundtruth (7th column), for both the depth (first row) and the color view (2nd row).

Method	Scenes	Min. Views	Training Time	Preprocess time per frame	Render time	Figure #
DB [38]	Static	4	37 h	8 h	real time	3.7
LLFF [52]	Static	6	?	10 min	real time	3.9
EVS [19]	Static	2	?	10 min	98 sec	3.8
MonoCam [78]	Dynamic	1	*Authors' results, no timing info*			3.10
VI [85]	Dynamic	8		*no info*	real time	3.11
VVC [48]	Dynamic	5		*partially manual*	real time	3.11
Ours	**Dynamic**	**4**		**2 min**	**6.8 sec**	3.7–11

Tab. 3.2: Comparisons regarding scene type, minimum number of input views, and speed. LLFF and EVS use pre-trained networks, so we did not re-train them.

methods [85, 48] that best match our intended setup. Each of these methods work with different numbers of input views and require different amounts of processing time. Some of these methods are only intended for static scenes, and so we would expect them to produce temporally inconsistent results. Table 3.2 summarizes these properties. In this paper, we extract frames to illustrate the comparisons; please see the accompanying video to better evaluate the differences.

Static—Deep Blending [38]. We compare our rendering method with Deep Blending(DB) [38] which learns optimized weights for blending 4 layers of mosaic images where the first layer is composed of the best fitting pixels, the second the second best etc. based on a heuristic. For the comparison, we first reconstructed each scene separately for each time step as described in their method. Afterwards, to be able to use the same camera path as for our results, we registered each time step to our full space-time reconstruction based on the camera positions. Finally, we used the pre-trained network provided by the authors to render each frame. Figure 3.7 shows DB not always being able to reconstruct marginal parts of the scenes, moreover our results appear sharper.

Static—Extreme View Synthesis [19]. Figure 3.8 shows a comparisons with Extreme View Synthesis (EVS) [19]. As input, EVS receives our SfM results. As expected, it exhibits flickering since this method is designed for static scenes and does not enforce temporal consistency. In addition, EVS cannot handle high resolution input because of its intense memory usage; we had to lower the resolution of the input video from 1920×1080 to 1280×720. For the same reason we also could not increase the depth resolution of its scene reconstruction step, which leads inaccurate depth maps and thus severe ghosting in the effected areas.

Ours DB [38]

Fig. 3.7: Comparison with Deep Blending [38] on the Cat-Dog and Elephant-wiggle sequences.

Static—Local Light Field Fusion [52]. Figure 3.9 shows a comparison with Local Light Field Fusion (LLFF) [52]. As input, LLFF receives our SfM results. Since our 3D reconstruction is left noisy by design, which is not expected by this method, we fixed the minimum and maximum depths to known correct values LLFF can use for its Multi-Plane Image computations. This reduces the flickering in their video, but it does not eliminate it completely. Our result also appears sharper and with less ghosting artifacts. Since LLFF requires at least 6 cameras to work, we could only compare on the 12-camera dataset sequences.

Dynamic—Monocam [78]. Figure 3.10 shows a comparison with Monocam [78]. Here we use the results given by the authors directly for the comparison. It is important to note that the sequences provided by the authors differ slightly from the ones used in the corresponding paper [78] and for which we have the results. For instance in the skating sequence, the skater is doing hand gestures in the provided input sequence contrary to the published processed result. This nevertheless allows qualitative comparisons. This figure shows that dynamic background objects like the plants in the umbrella sequence appear static if the virtual camera is static and are not consistent if the virtual camera is dynamic. Monocam results also exhibit temporal coherence artifacts. For instance, the reflections in the jumping and skating sequences jump back and forth based on which view was used to render them. Please see these in the accompanying video.

Dynamic—View Interpolation, Virtual Video Camera [85, 48]. Figure 3.11

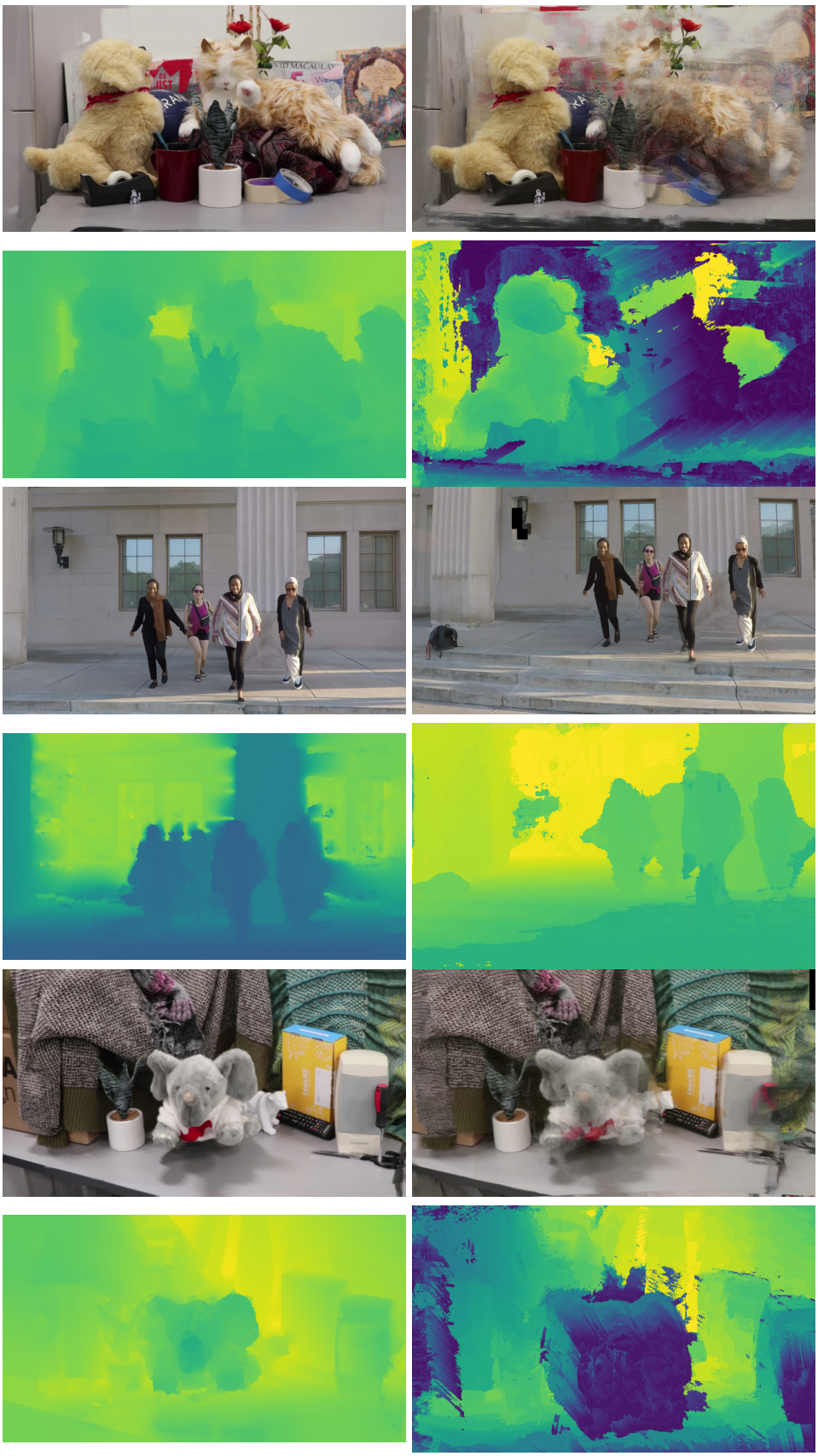
shows a comparison with View Interpolation (VI) [85] and Virtual Video Camera (VVC) [48] methods. We approximately reproduced the camera path of the video provided by the authors for the Breakdancers scene for this comparison. While VI requires a fixed and calibrated camera grid, our method can handle hand-held devices. VVC eliminates these restrictions, but relies on user input to correct correspondence matches.

3.3.4 Challenging Sequence—Drone

This sequence shows a quadrocopter drone (Figure 3.12). The drone has many thin features: the chassis, the fan blades, and exposed wires between battery and motors. Here, if our stereo reconstruction fails to find feature points on or nearby thin features at the correct depth, then our consistent propagation cannot provide the correct depth. As such, we see ghosting effects.

3.3.5 Computational Resources

We implemented our system in C++ on a Intel(R) Xeon(R) CPU ES-2630 v3 @2.4GHz computer. We used the OpenMVG library to compute Structure from Motion and sparse depth maps; and both OpenMVS and COLMAP to compute the PatchMatch-based sparse depth map post processing (Section 3.2.1). We parallelize the code using OpenMP and run on 32 cores; the rendering algorithm loads up to 2GBs of data per frame. As an example of wall-clock time, it took 2.2 hours to process the elephant-wiggle sequence (5 cameras, 100 frames per camera). The computation time breaks down to camera and sparse depth estimation (2 hours), and the rendering itself (6.8s per frame, 11.3 minutes for the whole video).



Ours EVS [19]

Fig. 3.8: Comparison with Extreme View Synthesis [19] on the Cat-Dog, Jumping and Elephant-wiggle sequences. Our method produces fewer artifacts than EVS.

Ours LLFF[52]

Fig. 3.9: Our method (left) in comparison with Local Light Field Fusion [52] (right) on the Jumping, Playground, Umbrella and Skating sequences.

Ours Monocam [78]

Fig. 3.10: Comparison with Monocam [78] on the Jumping, Playground, Umbrella and Skating sequences. Stronger temporal inconsistencies in results computed with Monocam can be seen in the accompanying video.

VI [85] VCC [48] Ours

Fig. 3.11: Comparison with [85] and [48] on the Breakdancers sequence.

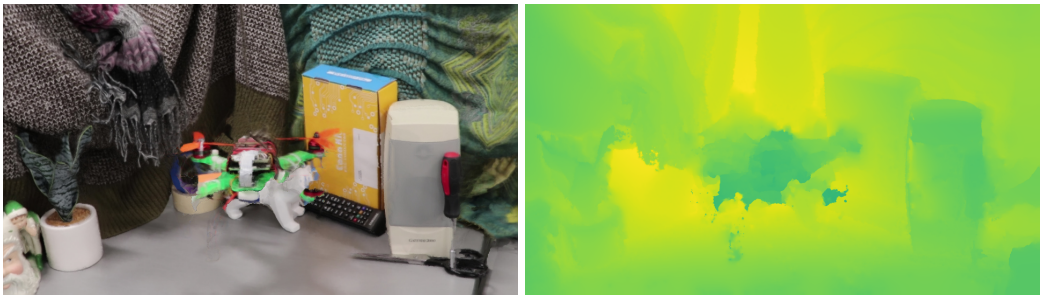


Fig. 3.12: *Limitations:* In the drone sequence, the scene has thin features which makes geometry reconstruction difficult. Here, our consistent propagation has trouble correcting for missing sparse feature points.

3.4 Limitations

Our method has several limitations. First, our choice of the OpenMVG library [56] for computing the SfM has the drawback that we must provide focal lengths for a pair of cameras to initiate the reconstruction process. Second, our method requires that the video sequences should have enough texture on the objects and in the background such that enough SIFT keypoints can be detected and matched. Another limitation lies in the amount of motion in the frame: conceptually, if SIFT keypoints are only detected on moving objects, then camera pose estimation will fail. In practice, we did not find this to be a problem. Furthermore, if the baseline is too wide, then not enough points will be obtained on moving objects and the depth propagation will fail. Finally, our optimization has parameters that can be tuned for each sequences; we provide reasonable initial values (Sec. 3.2.2), but tweaking can improve quality.

Finally, our current implementation is unoptimized C++ running on a CPU. Even if we optimize the implementation, one bottleneck is that keypoints from several images must be matched, and this is time consuming. If we consider SfM as an offline task to be performed once per scene, then the view rendering part currently takes 7 seconds per frame. Given the fixed grid, GPU-based diffusion optimizers are possible, which would produce a much more application-friendly render time.

3.5 Summary

We introduce a novel view synthesis method which can handle dynamic scenes. It is based around the key insight that reconstructing temporally-consistent 3D points on dynamic objects is hard, yet a structure-from-motion reconstruction method need not be temporally consistent if temporal consistency can be enforced in the rendering algorithm. We show that this can be accomplished by deferring consistency to a variational screen-space formulation, which makes it easy to robustly enforce spatio-temporal consistency via reprojection constraints weighted by confidences. While our setting has some restrictions, we show competitive results against existing baselines for video-based rendering without using any learning-based approaches. In the future, we hope to reduce constraints in camera motions and temporally with asynchronous videos.

4 Conclusions

In this thesis we tackled the problem of capturing and reconstructing dynamic scenes using only everyday equipment in casual settings and applying no prior calibration. We did so in order to make high quality 3D videos more readily available to the end users. With the rise of virtual and augmented reality platforms, it is crucial to provide easy solutions for the creation of 3D content. Hardware technologies able to capture 3D scenes are also becoming prolific. Several newer smart phones are equipped with stereo cameras and even depth sensors.

However, the processing of the dynamic sequences captured by these technologies still poses challenging problems on multiple fronts. Firstly, in the case of depth sensors, there's no widespread hardware technology yet that is able to accurately capture depth maps at high enough frequency to be reasonably applicable for dynamic scenes. Secondly, even though the capturing of RGB videos is elementary, their processing poses several questions that are yet to be satisfyingly solved. For instance, the sheer volume of data provided by videos renders most algorithms devised for single images unreasonable in face of dynamic scenes. Camera and scene motion also need to be specifically handled, and the solution needs to be consistent across all frames of the video.

The insight that we gained was that one can combine multiple technologies or devices to effectively overcome the issues caused by the scene motion. Specifically, we introduced two novel algorithms that provide solutions to the above mentioned problems.

We combined a consumer depth sensor and a LiDaR scanner (Chapter 2) to capture high resolution moderately dynamic sequences. By formulating the problem in spacetime, we were able to register the point clouds acquired by the heterogeneous setup and transfer the details of the high precision Time-of-Flight scan to the noisy depth video captured by the consumer depth sensor.

We also employed several hand-held and tripod-mounted cameras capture video sequences of dynamic scenes for novel view synthesis (Chapter 3). Our key insight was that one can avoid the difficult problem of generating temporally consistent

reconstructions of the scene by deferring the enforcement of consistency to render time.

We thoroughly tested our methods on self-captured real world datasets, as well as generated synthetic scenes to quantify our results.

4.0.1 Machine Learning Methods

Concurrently with out work, several machine learning based methods tackling the problem of dynamic 3D acquisition have been published. First came Bansal et al.[6] by employing unstructured multi-view capture to render novel view videos by employing a scene specific neural network to composite a dynamic foreground with a static background. Consequently, Yoon et al.[78] used monocular acquisition to render novel views by correcting depths computed for standalone frames using an incomplete multiview reconstruction with the help of a self-supervised network.

More recently, the advent of neural radiance fields[53] (NeRF) ensued a boom in dynamic scene 3D novel view synthesis. Using monocular capture, Li et al.[47] rerender dynamic scenes with the help of neural scene flow fields, while Xian et al.[77] constrain the dynamic 3D geometry using video depth estimation methods. Gao et al.[32] combine a static and a dynamic NeRF with regularization losses to render physically plausible novel views. Du et al.[27] apply NeRF in a multi-view setting to learn a 4D spacetime representation of a dynamic scene, while Peng et al.[60] reconstruct single human performers in the same vein.

All these methods produce impressive results outperforming previous approaches in their quality and wide applicability. However, they are computationally extremely expensive, usually requiring state of the art hardware and several hours, if not days to reconstruct a single scene. The original NeRF [53] implementation converged in approximately two days for a scene of average complexity, and newer approaches [26] promise up to six-fold speed-up at the scene specific training time.

In contrast, our method needs around two minutes to compute the 3D reconstruction of a time step of the video, and seconds to render a novel view. Additionally, it runs on a current average personal computer.

4.1 Future Work

While in this thesis we have demonstrated that our methods offer solutions to dynamic 3D scene acquisition, the limitations of our approaches provide immediate guidance to specific improvements.

Correcting Motion Distortions in Time-of-Flight Imaging The most severe limitation that we have identified working on this project was related to the poor quality of the depth videos captured by the consumer depth sensor. Hence, a sane fist step would be to swap out the depth camera to a newer model. Indeed, in the years since we've concluded our experiments, new technologies became widely available that promise a less noisy acquisition.

Following another route to tackle the same problem, we could mine the information provided by the color videos, captured at the same time, akin to Haefner et al. [36], to improve the quality of the depth maps.

Finally, our current registration process assumes rigid transformations only. One way to bypass the manual calibration of the consumer depth sensor would be to use the accurate knowledge of the scene provided by the LiDaR point cloud. One can imagine doing so using a non-rigid ICP algorithm, or one could explicitly model the distortion parameters and solve for them based on the LiDar scan.

Dynamic Scene Novel View Synthesis via Deferred Spatio-temporal Consistency Our current approach is aware of the captured scene being dynamic to the extent that it allows consequent frames to differ in areas where the input frames also differ. An idea to further leverage the motion in the scene would be to incorporate the optical flow in the rendering process. One can imagine doing so, more traditionally, by computing it beforehand and using the provided information to guide the optimization. More interestingly, one could solve for it in the screen space, using the same energy functional as for the novel depth and color images.

From another point of view, our current approach is somewhat restrictive regarding the acquisition setup and virtual camera path. For instance, we use synchronized video sequences. Incorporating the scene flow into our approach would also alleviate this constraint, possibly even lead to temporal super-resolution. Moreover, we could use it to fill in areas of the novel view not covered by frames from the same time step, which would allow the virtual camera greater freedom of movement around the scene.

4.1.1 Next Steps

As a broader perspective, we believe the casual capture of 3D dynamic scenes needs to be further addressed. Both acquisition devices, such as smart phones equipped with depth sensors and several cameras, and visualization technologies for virtual and augmented reality have become widely available in the recent years. Technologies to process temporal 3D data are in high demand.

For this reason, it is important not only to reconstruct and visualize dynamic scenes, but also to research ways to interact with them. One could take pointers from already existing applications working on static scenes, to inspire novel approaches that let end users create, edit, visualize, and interact with their own acquisitions. Both scene motion, and the size of temporal data make adapting existing methods nontrivial.

Bibliography

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, et al. *Slic superpixels*. Tech. rep. 2010 (cit. on p. 27).

[2] C. Ancuti, C. O. Ancuti, and P. Bekaert. "Video super-resolution using high quality photographs". In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, pp. 862–865 (cit. on p. 11).

[3] Rafael K dos Anjos, João Pereira, and José Gaspar. "A navigation paradigm driven classification for video-based rendering techniques". In: *Computers & Graphics* 77 (2018), pp. 205–216 (cit. on p. 28).

[4] Luca Ballan, Gabriel J Brostow, Jens Puwein, and Marc Pollefeys. "Unstructured video-based rendering: Interactive exploration of casually captured videos". In: *ACM Transactions on Graphics (TOG)* 29.4 (2010), p. 87 (cit. on pp. 28, 30).

[5] Dana H Ballard. "Generalizing the Hough transform to detect arbitrary shapes". In: *Pattern recognition* 13.2 (1981), pp. 111–122 (cit. on p. 15).

[6] Aayush Bansal, Minh Vo, Yaser Sheikh, Deva Ramanan, and Srinivasa Narasimhan. "4D Visualization of Dynamic Events from Unconstrained Multi-View Videos". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020) (cit. on pp. 29, 50).

[7] Wenbo Bao, Wei-Sheng Lai, Chao Ma, et al. "Depth-aware video frame interpolation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3703–3712 (cit. on p. 29).

[8] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing". In: *ACM SIGGRAPH 2009 Papers*. SIGGRAPH '09. New Orleans, Louisiana: Association for Computing Machinery, 2009 (cit. on p. 31).

[9] S. Berkiten, M. Halber, J. Solomon, et al. "Learning Detail Transfer based on Geometric Features". In: *Computer Graphics Forum, Proceedings Eurographics 2017*. 2017 (cit. on p. 11).

[10] P. J. Besl and N. D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256 (cit. on p. 16).

[11] Pravin Bhat, C. Lawrence Zitnick, Noah Snavely, et al. "Using Photographs to Enhance Videos of a Static Scene". In: *Proceedings of the 18th Eurographics Conference on Rendering Techniques*. EGSR'07. Grenoble, France: Eurographics Association, 2007, pp. 327–338 (cit. on p. 11).

[12] Derek Bradley, Wolfgang Heidrich, Tiberiu Popa, and Alla Sheffer. “High Resolution Passive Facial Performance Capture”. In: *ACM Trans. on Graphics (Proc. SIGGRAPH)* 29.3 (2010) (cit. on p. 10).

[13] Michael Broxton, John Flynn, Ryan Overbeck, et al. “Immersive Light Field Video with a Layered Mesh Representation”. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 39.4 (2020), 86:1–86:15 (cit. on p. 28).

[14] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. “Unstructured lumigraph rendering”. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM. 2001, pp. 425–432 (cit. on p. 27).

[15] D. Herrera C., J. Kannala, and J. Heikkilä. “Joint Depth and Color Camera Calibration with Distortion Correction”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.10 (2012), pp. 2058–2064 (cit. on pp. 22, 23).

[16] Yaron Caspi and Michal Irani. “Spatio-Temporal Alignment of Sequences”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 24.11 (Nov. 2002), pp. 1409–1424 (cit. on p. 11).

[17] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. “Depth synthesis and local warps for plausible image-based navigation”. In: *ACM Transactions on Graphics (TOG)* 32.3 (2013), p. 30 (cit. on p. 27).

[18] Shenchang Eric Chen and Lance Williams. “View interpolation for image synthesis”. In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM. 1993, pp. 279–288 (cit. on p. 27).

[19] I. Choi, O. Gallo, A. Troccoli, M. H. Kim, and J. Kautz. “Extreme View Synthesis”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 7780–7789 (cit. on pp. 28, 38, 40, 43).

[20] Sean Clarkson, Jonathan Wheat, Ben Heller, J Webster, and Simon Choppin. “Distortion correction of depth data from consumer depth cameras”. In: *3D Body Scanning Technologies, Long Beach, California, Hometrica Consulting* (2013), pp. 426–437 (cit. on p. 22).

[21] Alvaro Collet, Ming Chuang, Pat Sweeney, et al. “High-Quality Streamable Free-Viewpoint Video”. In: *ACM Trans. Graph.* 34.4 (2015) (cit. on p. 28).

[22] K. Cornelis, F. Verbiest, and L. Van Gool. “Drift detection and removal for sequential structure from motion algorithms”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.10 (2004), pp. 1249–1259 (cit. on p. 30).

[23] James Davis, Diego Nehab, Ravi Ramamoorthi, and Szymon Rusinkiewicz. “Spacetime Stereo: A Unifying Framework for Depth from Triangulation”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 27.2 (Feb. 2005), pp. 296–302 (cit. on p. 11).

[24] James Davis, Ravi Ramamoorthi, and Szymon Rusinkiewicz. “Spacetime stereo: A unifying framework for depth from triangulation”. In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 2. IEEE. 2003, pp. II–359 (cit. on p. 29).

[25] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. “Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach”. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 1996, pp. 11–20 (cit. on p. 27).

[26] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. “Depth-supervised NeRF: Fewer Views and Faster Training for Free”. In: *arXiv preprint arXiv:2107.02791* (2021) (cit. on p. 50).

[27] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. “Neural Radiance Flow for 4D View Synthesis and Video Processing”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021 (cit. on p. 50).

[28] Richard O Duda and Peter E Hart. “Use of the Hough transformation to detect lines and curves in pictures”. In: *Communications of the ACM* 15.1 (1972), pp. 11–15 (cit. on p. 14).

[29] D. Fleet and Y. Weiss. “Optical Flow Estimation”. In: *Handbook of Mathematical Models in Computer Vision*. Ed. by Nikos Paragios, Yunmei Chen, and Olivier Faugeras. Boston, MA: Springer US, 2006, pp. 237–257 (cit. on p. 13).

[30] John Flynn, Michael Broxton, Paul Debevec, et al. “DeepView: View synthesis with learned gradient descent”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2367–2376 (cit. on p. 27).

[31] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. “Deepstereo: Learning to predict new views from the world’s imagery”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5515–5524 (cit. on p. 27).

[32] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. “Dynamic View Synthesis from Dynamic Monocular Video”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2021 (cit. on p. 50).

[33] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. “The lumigraph”. In: *Siggraph*. Vol. 96. 30. 1996, pp. 43–54 (cit. on p. 27).

[34] Kaiwen Guo, Peter Lincoln, Philip Davidson, et al. “The Relightables: Volumetric Performance Capture of Humans with Realistic Relighting”. In: *ACM Trans. Graph.* 38.6 (2019) (cit. on p. 28).

[35] Ankit Gupta, Pravin Bhat, Mira Dontcheva, et al. “Enhancing and Experiencing Spacetime Resolution with Videos and Stills”. In: *International Conference on Computational Photography*. IEEE, 2008 (cit. on p. 11).

[36] Bjoern Haefner, Yvain Quéau, Thomas Möllenhoff, and Daniel Cremers. “Fight Ill-Posedness With Ill-Posedness: Single-Shot Variational Depth Super-Resolution From Shading”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on pp. 11, 51).

[37] A. Hamdi-Cherif, J. Digne, and Chaine R. “Super-resolution of Point Set Surfaces using Local Similarities”. In: *to appear in Computer Graphics Forum* 37.1 (2018), pp. 60–70 (cit. on p. 11).

[38] Peter Hedman, Julien Philip, True Price, et al. “Deep Blending for Free-Viewpoint Image-Based Rendering”. In: *ACM Trans. Graph.* 37.6 (Dec. 2018) (cit. on pp. 27, 38, 40, 41).

[39] Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. “Scalable inside-out image-based rendering”. In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), p. 231 (cit. on p. 27).

[40] E. Herbst, X. Ren, and D. Fox. “RGB-D flow: Dense 3-D motion estimation using color and depth”. In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 2276–2282 (cit. on p. 13).

[41] Aleksander Holynski and Johannes Kopf. “Fast depth densification for occlusion-aware augmented reality”. In: *SIGGRAPH Asia 2018 Technical Papers*. ACM. 2018, p. 194 (cit. on pp. 26, 27).

[42] F. Huguet and F. Devernay. “A Variational Method for Scene Flow Estimation from Stereo Sequences”. In: *2007 IEEE 11th International Conference on Computer Vision*. 2007, pp. 1–7 (cit. on p. 13).

[43] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. “Learning-based view synthesis for light field cameras”. In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), p. 193 (cit. on p. 27).

[44] Yong Joo Kil, Boris Mederos, and Nina Amenta. “Laser Scanner Super-resolution”. In: *Proc. Point-Based Graphics*. SPBG’06. Boston, Massachusetts, 2006, pp. 9–16 (cit. on p. 11).

[45] T. P. Koninckx and L. Van Gool. “Real-time range acquisition by adaptive structured light”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.3 (2006), pp. 432–445 (cit. on p. 10).

[46] Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. “Image-based rendering in the gradient domain”. In: *ACM Transactions on Graphics (TOG)* 32.6 (2013), p. 199 (cit. on p. 27).

[47] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. “Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021 (cit. on p. 50).

[48] Christian Lipski, Christian Linz, Kai Berger, Anita Sellent, and Marcus Magnor. “Virtual Video Camera: Image-Based Viewpoint Navigation Through Space and Time”. In: *Computer Graphics Forum* 29.8 (2010), pp. 2555–2568 (cit. on pp. 28, 40–42, 46).

[49] D. Liu, J. Gu, Y. Hitomi, et al. “Efficient Space-Time Sampling with Pixel-Wise Coded Exposure for High-Speed Imaging”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.2 (2014), pp. 248–260 (cit. on p. 11).

[50] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. “Consistent Video Depth Estimation”. In: *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* 39.4 (2020) (cit. on p. 29).

[51] Kevin Matzen, Michael F. Cohen, Bryce Evans, Johannes Kopf, and Richard Szeliski. "Low-Cost 360 Stereo Photography and Video Capture". In: *ACM Trans. Graph.* 36.4 (2017) (cit. on p. 27).

[52] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, et al. "Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines". In: *ACM Trans. Graph.* 38.4 (July 2019) (cit. on pp. 27, 38, 40, 41, 44).

[53] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: 2003.08934 [cs.CV] (cit. on pp. 27, 50).

[54] Niloy J. Mitra, Simon Flöry, Maks Ovsjanikov, et al. "Dynamic Geometry Registration". In: *Proc. Symposium on Geometry Processing*. SGP '07. Eurographics, 2007, pp. 173–182 (cit. on p. 13).

[55] Pierre Moulon, Pascal Monasse, and Renaud Marlet. "Adaptive Structure from Motion with a Contrario Model Estimation". In: *Proceedings of the Asian Computer Vision Conference (ACCV 2012)*. Springer Berlin Heidelberg, 2012, pp. 257–270 (cit. on p. 30).

[56] Pierre Moulon, Pascal Monasse, Romuald Perrot, and Renaud Marlet. "Openmvg: Open multiple view geometry". In: *International Workshop on Reproducible Research in Pattern Recognition*. Springer. 2016, pp. 60–74 (cit. on pp. 29, 47).

[57] A. Mustafa, H. Kim, J. Guillemaut, and A. Hilton. "Temporally Coherent 4D Reconstruction of Complex Dynamic Scenes". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4660–4669 (cit. on pp. 26, 28, 30, 38).

[58] Armin Mustafa, Marco Volino, Hansung Kim, Jean-Yves Guillemaut, and Adrian Hilton. "Temporally Coherent General Dynamic Scene Reconstruction". In: *CoRR* abs/1907.08195 (2019). arXiv: 1907.08195 (cit. on pp. 26, 28, 30, 38).

[59] OpenNI organization. *OpenNI User Guide*. 2017 (cit. on p. 19).

[60] Sida Peng, Yuanqing Zhang, Yinghao Xu, et al. "Neural Body: Implicit Neural Representations with Structured Latent Codes for Novel View Synthesis of Dynamic Humans". In: *CVPR*. 2021 (cit. on p. 50).

[61] Eric Penner and Li Zhang. "Soft 3D reconstruction for view synthesis". In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), p. 235 (cit. on p. 28).

[62] Albert Parra Pozo, Michael Toksvig, Terry Filiba Schrager, et al. "An Integrated 6DoF Video Camera and System Design". In: *ACM Trans. Graph.* 38.6 (2019) (cit. on p. 28).

[63] J. Quiroga, T. Brox, F. Devernay, and J. Crowley. "Dense semi-rigid scene flow estimation from RGBD images". In: *European Conference on Computer Vision (ECCV)*. 2014 (cit. on p. 13).

[64] Gernot Riegler and Vladlen Koltun. "Free View Synthesis". In: *European Conference on Computer Vision*. 2020 (cit. on p. 27).

[65] Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer. "Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-looking Wrinkles". In: *ACM SIGGRAPH Asia 2010 Papers*. Seoul, South Korea: ACM, 2010, 157:1–157:8 (cit. on p. 11).

[66] Johannes L Schonberger and Jan-Michael Frahm. "Structure-from-motion revisited". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4104–4113 (cit. on p. 31).

[67] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. "Layered depth images". In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 1998, pp. 231–242 (cit. on p. 27).

[68] E. Shechtman, Y. Caspi, and M. Irani. "Space-time super-resolution". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.4 (2005), pp. 531–545 (cit. on p. 11).

[69] Noah Snavely, Steven M Seitz, and Richard Szeliski. "Photo tourism: exploring photo collections in 3D". In: *ACM transactions on graphics (TOG)*. Vol. 25. 3. ACM. 2006, pp. 835–846 (cit. on p. 27).

[70] J. Song, X. Chen, and O. Hilliges. "Monocular Neural Image Based Rendering With Continuous View Control". In: *ICCV 2019*. 2019, pp. 4089–4099 (cit. on p. 28).

[71] Pratul P. Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. "Learning to Synthesize a 4D RGBD Light Field from a Single Image". In: *International Conference on Computer Vision (ICCV) 2017* (2017) (cit. on p. 28).

[72] Gary K.L. Tam, Zhi-Quan Cheng, Yu-Kun Lai, et al. "Registration of 3D Point Clouds and Meshes: A Survey from Rigid to Nonrigid". In: *IEEE Transactions on Visualization and Computer Graphics* 19.7 (2013), pp. 1199–1217 (cit. on p. 2).

[73] Minh Vo, Srinivasa G Narasimhan, and Yaser Sheikh. "Spatiotemporal bundle adjustment for dynamic 3d reconstruction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1710–1718 (cit. on pp. 26, 29).

[74] T. Weise, B. Leibe, and L. Van Gool. "Fast 3D Scanning with Automatic Motion Compensation". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8 (cit. on p. 10).

[75] Thibaut Weise, Sofien Bouaziz, Hao Li, and Mark Pauly. "Realtime Performance-based Facial Animation". In: *ACM Trans. Graph.* 30.4 (July 2011), 77:1–77:10 (cit. on p. 10).

[76] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, et al. "High Performance Imaging Using Large Camera Arrays". In: *ACM Trans. Graph.* 24.3 (2005) (cit. on p. 28).

[77] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. "Space-time Neural Irradiance Fields for Free-Viewpoint Video". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 9421–9431 (cit. on p. 50).

[78] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. “Novel View Synthesis of Dynamic Scenes with Globally Coherent Depths from a Monocular Camera”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020) (cit. on pp. 29, 36–38, 40, 41, 45, 50).

[79] Cha Zhang and Tsuhan Chen. “A survey on image-based rendering—representation, sampling and compression”. In: *Signal Processing: Image Communication* 19.1 (2004), pp. 1–28 (cit. on p. 27).

[80] Li Zhang, Brian Curless, and Steven M. Seitz. “Spacetime Stereo: Shape Recovery for Dynamic Scenes”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Madison, WI, 2003, pp. 367–374 (cit. on p. 11).

[81] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. “Spacetime Faces: High Resolution Capture for Modeling and Animation”. In: *ACM Trans. Graph.* 23.3 (Aug. 2004), pp. 548–558 (cit. on p. 10).

[82] Song Zhang. “Recent progresses on real-time 3D shape measurement using digital fringe projection techniques”. In: *Optics and Lasers in Engineering* 48.2 (2010). Fringe Projection Techniques, pp. 149 –158 (cit. on p. 10).

[83] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. “Stereo magnification: Learning view synthesis using multiplane images”. In: *arXiv preprint arXiv:1805.09817* (2018) (cit. on p. 27).

[84] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. “View synthesis by appearance flow”. In: *European conference on computer vision*. Springer. 2016, pp. 286–301 (cit. on p. 27).

[85] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. “High-Quality Video View Interpolation Using a Layered Representation”. In: *ACM Trans. Graph.* 23.3 (Aug. 2004), 600–608 (cit. on pp. 28, 37, 40–42, 46).

List of Figures

1.1 A consumer depth sensor (e.g. Kinect) acquires dense low-resolution scans at a high rate while a LiDaR scanner acquires sparse high resolution scans containing time distorsions. We recover details by undistorting the LiDaR scan. 3

1.2 Given a small set of video sequences of a performance, our method computes camera poses and sparse points, then optimizes those points into a novel video sequence following a user-defined camera path. Our space-time SfM intentionally does not compute temporal consistency for points on dynamic objects and instead defers spatio-temporal consistency in both depth and RGB reconstructions to the novel view synthesis stage via our variational formulation. 5

2.1 A consumer depth sensor acquires dense low-resolution scans at a high rate (right) while a LiDaR scanner acquires sparse high resolution scans containing time distorsions (left). We recover details by undistorting the LiDaR scan (bottom right). 9

2.2 Overview of our high resolution dynamic point set acquisition and processing algorithm 12

2.3 Registration result. For 5 different time values t, we select LR and HR points that lie in a small temporal neighborhood after the spacetime registration. The LR points are oriented and displayed in grayscale values while the HR points are shown in red. The LiDAR scanline acquisition process results in vertical lines. 17

2.4 Synthetic dataset along with reconstructed frames for validating various steps of our method as described in Table 2.1 18

2.5 Our experimental setup: a Kinect and a LiDaR acquire the same scene from different viewpoints. No calibration is required. 19

2.6 Undistorting LiDaR scans (left) from consumer depth camera sequences (center). Our result (right) show higher spatial accuracy than the consumer depth camera while respecting the global motion. Video results can be seen in supplemental materials. 20

2.7 Functional schematic of the LiDaR Scanner. 21

2.8 Failure case: large motion are not handled well by our method. Left: Lidar Scan. Depth sensor frames (top row) and corresponding corrected frames (bottom row). 22

2.9 Sample color (top row) and disparity (bottom row) images used to calibrate consumer depth sensors. Image taken from Herrera et al. [15]. 23

3.1 Given a small set of video sequences of a performance, our method computes camera poses and sparse points, then optimizes those points into a novel video sequence following a user-defined camera path. Our space-time SfM intentionally does not compute temporal consistency for points on dynamic objects and instead defers spatio-temporal consistency in both depth and RGB reconstructions to the novel view synthesis stage via our variational formulation. 25

3.2 *Top row:* Sparse reconstructed 3D points (left) and their weights $w_{\hat{D}}$ (right) projected into the virtual view. Red indicates areas of empty space; depth map is bright green in far depth regions. *Bottom row:* Points diffused into a full depth map D (left) according to the weight map w_D (right). Note how the how the color edges are correctly identified via Eq. 3.5, and how the occluded points from behind the head of the character on the left are given no weight by Eq. 3.6 (top right) and so do not corrupt the depth. 32

3.3 Four closest input images $I_{s,t}$ projected onto the virtual camera's view point alongside their corresponding weight maps w_P (Eq. 3.8). 33

3.4 *Top row:* Color image I_{t-1} and depth map D_{t-1} of a previous time step. *Bottom left:* I_{t-1} reprojected to the camera view by D_{t-1} of the current time step. *Bottom right:* Weight map w_T (Eq. 3.11) modulates consistency, notably around the moving mouth of the character on the left. 33

3.5 Color and depth results for Cat&dog and Elephant-wiggle scenes. . . . 37

3.6 Rendering Ablation Study. We show our rendered result without temporal consistency (1st column), without weights on the projected sparse 3D points (2nd column), without depth weights and inverse image gradients (3rd column), only without inverse image gradients (4th column), without weights on the projected input images (5th column), together with the full result (6th column) and the groundtruth (7th column), for both the depth (first row) and the color view (2nd row). . 39

3.7 Comparison with Deep Blending [38] on the Cat-Dog and Elephant-wiggle sequences. 41

3.8 Comparison with Extreme View Synthesis [19] on the Cat-Dog, Jumping and Elephant-wiggle sequences. Our method produces fewer artifacts than EVS. 43
3.9 Our method (left) in comparison with Local Light Field Fusion [52] (right) on the Jumping, Playground, Umbrella and Skating sequences. 44
3.10 Comparison with Monocam [78] on the Jumping, Playground, Umbrella and Skating sequences. Stronger temporal inconsistencies in results computed with Monocam can be seen in the accompanying video. . . . 45
3.11 Comparison with [85] and [48] on the Breakdancers sequence. 46
3.12 *Limitations:* In the drone sequence, the scene has thin features which makes geometry reconstruction difficult. Here, our consistent propagation has trouble correcting for missing sparse feature points. 46

List of Tables

2.1 Average point-wise distances between reconstructed and generated ground truth point clouds using either known (GT) registration parameters and motion field or computing them using our method (E). All errors are below 2% of the shape height (2.5 units) 19

3.1 Ablation Study. We compare estimated camera pose accuracy for naive SfM, naive SfM using a ground truth mask for the static parts of the scene, and an ablated version of our space time SfM without camera smoothing. While adding smoothing slightly has little effect on the positional error (PE), it reduces orientation error (OE). Our approach also minimizes the median reprojection error (RE) of the feature points. SfM minimizes reprojection errors by construction which explains why using ground truth camera poses increases reprojection errors, but results in perfect camera poses by construction. 38

3.2 Comparisons regarding scene type, minimum number of input views, and speed. LLFF and EVS use pre-trained networks, so we did not re-train them. 40