IMPROVING THE UTILITY OF EGOCENTRIC VIDEOS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Biao Ma

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Amy R. Reibman, Chair
School of Electrical and Computer Engineering

Dr. Charles A. Bouman
School of Electrical and Computer Engineering

Dr. Mireille Boutin
School of Electrical and Computer Engineering

Dr. Mary L. Comer
School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis
Head of the School Graduate Program

To my late grandmother. To my parents. To my beloved Pingge.

ACKNOWLEDGMENTS

Stories only start when people meet each other. It was 2015 when I met with Prof. Amy R. Reibman. I was hoping to transfer to the Ph.D. program and starting to work on research topics. I know both my communication skills and professional skills were awful at the point, but she still talked to me all the way from the classroom to her office. From that point, during all these 4 years, I received numberless of suggestions from her. She educated me ways of thinking, and engineering. Most importantly, she taught me how to express my ideas and communicate with others. This is critical not only for my research, the degree, but also for my life. I know this may be just a start, but I really appreciate it.

Ideas only come up when wisdom meets curious, craziness and failure. I love every project during this journey. They are crazy, and it seems they are unachievable. However, this gives me the motivation to challenge them. I would like to thank Prof. Reibman's support and suggestions for all my ideas. I will not forget the afternoon that we sat and played the GTAV in order to investigate the First-Person Motion. I would like to thank Prof. Boutine to offer me the chance to work on her project of stress prediction. I would like to thank Prof. Delp to let me work on his indoor 3D reconstruction project. I also want to thank the suggestions from Prof. Bouman and Prof. Comer for my Ph. D. research. And all my lab-mates, Chen Bai, He Liu, Chengzhang Zhong, and Haoyu Chen, thanks for the discussion and helps. Finally, I would like to thank the reviewers of our papers. Failure is a part of the journey. I really appreciate all the valuable suggestions.

The journey only continues when beloved support. My parents are definitely young parents. I was born in their early twenty's. I know clearly that is a fantastic age, but they spent those ages on me and did whatever they can to make me become a better man until now. So did my beloved Pingge. I know clearly that waiting under

uncertainty is tormenting, but you always support me and my dream. Words cannot express my love for all three of you.

The feast never ends when people leave. A few weeks before my final defense, my dearest grandmother passed away. Regrettably, I cannot stand by her side. Although she only graduated from a middle school, the education in my childhood from her is my greatest treasure. Reunion is always less than parting, and is only less for once. However, this will not be the end. I will keep fighting with your love.

TABLE OF CONTENTS

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABBREVIATIONS

BoW	Bag-of-Words
BD-BR	Bjøntegaard-Delta Bit-Rate
DCVs	Dashcam Videos
DPB	Decoded Picture Buffer
DFD	Displaced Frame Difference
DSO	Direct Sparse Odometry
FPVs	First-Person Videos
FPM	First-Person Motion
FPMI	First-Person Motion Information
FPF	First-Person Feeling
FPMR	First-Person Motion Range
FPE	First-Person Experience
FOV	Field Of View
GPS	Global Positioning System
HEVC	High Efficiency Video Coding
ITF	Inter-frame Transformation Fidelity
MAR	Minimum Angular Resolution
MV-MSE	Motion-Vector-based Mean Squared Error
NDV	Near-Duplicated Videos
ORB	Oriented FAST and Rotated BRIEF
PLCC	Pearson Linear Correlation Coefficient
PSNR	Peak Signal-to-Noise Ratio
RANSAC	Random sample consensus
SVE	Structure Viewing Experience

SfM	Structure from Motion
SURF	Speeded Up Robust Features
SIFT	Scale Invariant Feature Transform
SPEM	Smooth Pursuit Eye Movement
VE	Viewing Experience
VOR	Vestibulo-Ocular Reflex
VOs	Visual Odometers
vSLAM	visual Simultaneous Localization and Mapping

ABSTRACT

Ma, Biao PhD, Purdue University, August 2019. Improving the Utility of Egocentric Videos. Major Professor: Amy R. Reibman.

For either entertainment or documenting purposes, people are starting to record their life using egocentric cameras, mounted on either a person or a vehicle. Our target is to improve the utility of these egocentric videos.

For egocentric videos with an entertainment purpose, we aim to enhance the viewing experience to improve overall enjoyment. We focus on First-Person Videos (FPVs), which are recorded by wearable cameras. People record FPVs in order to share their First-Person Experience (FPE). However, raw FPVs are usually too shaky to watch, which ruins the experience. We explore the mechanism of human perception and propose a biometric-based measurement called the Viewing Experience (VE) score, which measures both the stability and the First-person Motion Information (FPMI) of a FPV. This enables us to further develop a system to stabilize FPVs while preserving their FPMI. Experimental results show that our system is robust and efficient in measuring and improving the VE of FPVs.

For egocentric videos whose goal is documentation, we aim to build a system that can centrally collect, compress and manage the videos. We focus on Dash Camera Videos (DCVs), which are used by people to document the route they drive each day. We proposed a system that can classify videos according to the route they drove using GPS information and visual information. When new DCVs are recorded, their bit-rate can be reduced by jointly compressing them with videos recorded on the similar route. Experimental results show that our system outperforms other similar solutions and the standard HEVC particularly in varying illumination.

The First-Person Video viewing experience topic and the Dashcam Video compression topic are two representations of applications rely on Visual Odometers (VOs): visual augmentation and robotic perception. Different applications have different requirement for VOs. And the performance of VOs are also influenced by many different factors. To help our system and other users that also work on similar applications, we further propose a system that can investigate the performance of different VOs under various factors. The proposed system is shown to be able to provide suggestion on selecting VOs based on the application.

Part I

Introduction

1. BACKGROUND AND CONTRIBUTIONS OF THIS THESIS

With the development of the CMOS sensor, mobile devices get higher and higher capabilities on image and video capturing. Digital Single Lens Reflex (DSLR) cameras are not the only source of high quality images and videos. And because of the relatively smaller size, lighter weight and cheaper price, mobile image capturing devices become popular. The variety of products keep increase. Examples are not only the smart phones, but also the sports cameras and Dashcams. GoPro launched the first sports camera in 2002 and by now, 7 generations of products have been launched and sell to all over the world. Sony, Nikon and other traditional camera factories also join in this trend. For the Dashcams and drone-mounted cameras, Garmin and DJI provide the leading products in the market.

This unstoppable trend of putting efforts into these mobile camera devices reflex the need of customers. People start to do video log for their daily life: use Dashcams to record on-road footage, use sports cameras to share their unique experience and even film a movie. We call these videos the egocentric videos.

The mobile feature of the camera brings both problems and possibilities to egocentric videos. One of the incoming problem is that light weight and small size make the egocentric videos have a lot of motions. The resulting egocentric videos are shaky to watch. To solve this problem, video stabilization techniques are developed. The other incoming problem is the data storage. Long time video logging requires large digital space to store. This pushes forward the developing of near-duplicated video detection and compression. The possibilities are also provided by the egocentric motion and the large amount of data. Techniques such as real-time visual-SLAM are developed to extract the motion information for applications such as autonomous driving and

Augmented Reality. Techniques such as Structure-from-Motion can utilize the large amount of data to generate dense reconstruction of the scene view.

The topic of this thesis is about egocentric video processing and is relative to computer vision algorithms. Our target is to contribute several missing parts in the mentioned computer vision techniques. For the video stabilization techniques, we believe the main subject is the First-Person Video and users care about the viewing experience of these videos. The entertainment feature is the keystone of First-Person videos. However, this is ignored by current stabilization techniques. As a result, the first part we believe is missing and aim to contribute is the subjective measurement of the video stability and the corresponding stabilization technique.

For the Dashcam video storage, current industry simply overwrites the historical data when the digital space is reaching the limit. In addition, the storage and management of the recorded data are done user by user offline and separately. However, consider the feature of Dashcam videos that they keep record similar content, and consider the need of large amount of training data for computer vision applications such as autonomous driving, we believe a video compression system is needed for this kind of documentation videos. A compression system for Dashcam videos is the second contribution we want to make.

Both the First-Person Video viewing experience topic or the Dashcam Video compression topic are related to Visual Odometer. These two topics are two representation of two kinds of applications that rely on VO: visual augmentation and robotic perception. During the research, we notice that different VOs usually provide us different final performance of our system. The optimal VO of the FPV topic may not be the best choice for the Dashcam Video topic. To help users that have the same problem with us, the third contribution we want to make is an evaluation methodology that can help users to investigate the performance of different VOs under various factors.

The whole thesis is present in three parts in the order of: Improving the First-Person Video viewing experience (Part II), Dashcam Video compression (Part III) and Visual Odometer evaluation methodology (Part IV). In fact, in our research

timeline, the Dashcam Video compression is the first topic instead of the First-Person Video. However, we are the most proud of our contribution in the First-Person Video viewing experience topic. Also, both the Fisrt-Person Video topic and Dashcam Video topic are based on Visual Odometry. We prefer to end the thesis with the Visual Odometer evaluation topic. As a result, we re-organize the order. Please kindly choose the reading order based on your interest and time. The detail of three topics are relatively independent. For the detailed motivation and backgrounding, please refer to corresponding Parts.

2. PUBLICATIONS RESULTING FROM THIS WORK

Journal Papers:

1. Biao Ma and Amy R. Reibman, Viewing Experience Model of First-Person Videos, J. Imaging 2018.;
2. Biao Ma and Amy R. Reibman, An Evaluation Methodology for Visual Odometry (Ready to submit).

Conference Papers:

1. Biao Ma and Amy R. Reibman, A Perceptually-inspired 2D Video Stability Estimator, Global SIP 2018;
2. Biao Ma and Amy R. Reibman, Measuring and Improving the Viewing Experience of First-person Videos, ACM Multimedia Thematic Workshops, 2017;
3. Biao Ma and Amy R. Reibman, Enhancing Viewability for First-person Videos based on a Human Perception Model, IEEE International Workshop on Multimedia Signal Processing, 2017;
4. Biao Ma and Amy R. Reibman, DashCam Video Compression using Historical Data, Picture Coding Symposium (PCS), 2016.

Part II

A Viewing Experience Model of First-Person Videos

3. INTRODUCTION

With the development of wearable cameras, people are starting to create First-Person Videos (FPVs) to record their life. From simple life-logs to playing sports, the content of FPVs varies significantly. Some producers even started to make First-Person films. By the time the recorders mount a camera on their head or nose to record FPVs, most of them share a same primary goal: sharing the First-Person Experience (FPE) with others. However, the fact is that the viewing experience of FPVs is quite different from recorders' FPE. In this work, we start with discussing the reason behind this fact and end up with a reliable solution that enables us to improve the viewing experience of FPVs based on a viewing experience model.

To help recorders to achieve their goal of sharing experience, first we need to know the components of the FPE. A solution can be found then, if we compare the components of both the FPE and the viewing experience of FPVs and accordingly adjust the component of the latter. Two components that recorders are eager to share consist of the FPE. First, recorders would like to share the **objective information** they perceive. For example, if a recorder capture a FPV when he/she is running in a park, he/she may want to share the trees or flowers with viewers. Second, recorders would like to share the **subjective information** of the objective world: the feeling of interacting in their First-Person perspective. Using the same example, the recorder may want to share his/her reaction that a dog barks behind him/her and he/she is surprised and quickly turns around and look at the dog.

FPVs record all visual information appears in First-Person perspective. The objective world is recorded in each frame and the First-Person reaction is recorded by the camera motion reflected by temporal frame changes. However, a strange thing happens: there is no visual information loss during the process of "recording", but

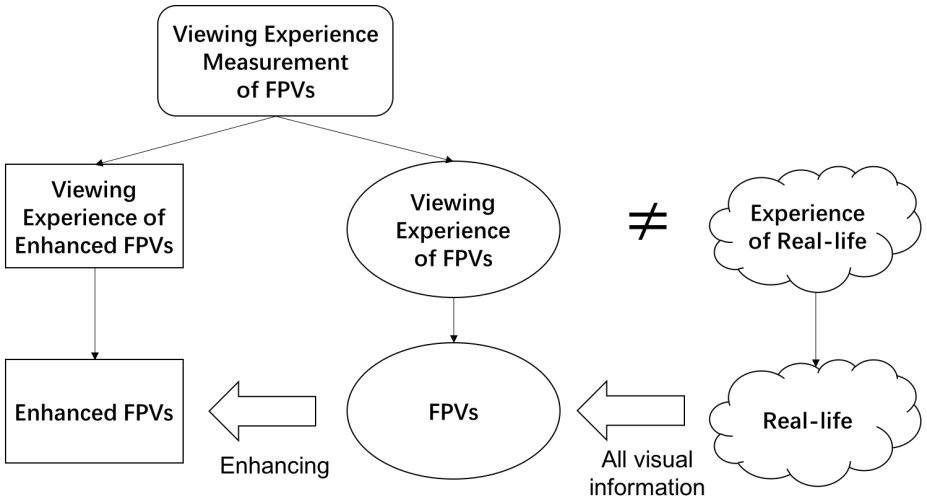


Fig. 3.1.: Experience of different versions of First-Person perspective

the viewing experience of FPVs is not identical with recorders' FPE. Fig.3.1 shows the logic of this problem.

The reason is the viewing experience of FPVs has different components with the FPE. It includes video stability and First-Person Motion Information (FPMI). The FPMI is corresponding to the FPE, which is the information conveyed by camera motion. It contains both the objective and subjective information the recorder may perceive. However, during FPE, people do not perceive instability, but it is present in the captured video. Recorders never feel dizzy when they record the video but viewers feel the FPVs are too shaky to watch. Later in this paper, we show that this is because the visual function that occurs when we watch the FPVs is different with the visual function we rely on when we are in the real-life.

As a result, to help recorders to share FPE, what we should do is to:

(1) Preserve the similar component: preserve FPMI;

(2) Fix the unpleasant component: increase video stability.

A direct solution would be construct a video enhancement system based on a measurement of the viewing experience of FPVs. To approach it, it is helpful to take the fact that the experience of watching FPVs and living in real-life are based on different visual function into consideration.

There are several off-the-shelf approaches which are related to our problem, such as traditional video stabilization techniques, video stability measurements and FPV watchability enhancement techniques. However, in the next Chapter, we show these related works are insufficient. In Chapter 21.3, we provide an overview of our system. Following the pipeline of our system, its subparts are introduced from Chapter 6 to Chapter 9. Chapter 6 focuses on the motion estimation algorithm, which is a modified visual odometer according to our application. The human perception model based on eye movements is the basis of our system, which is presented in Chapter 7. Applying the human perception model, we propose the quality metric of the viewing experience of FPVs in Chapter 8 and describe the improving method in Chapter 9. Chapter 10 includes three parts of our experimental results. Firstly, we design and use subjective

tests to finalize the parameter in our human perception model. Secondly, objective and subjective tests are employed to test the robustness of our viewing experience measurement. Thirdly, objective and subjective tests show the effectiveness of our system by comparing with other methods. Note that this whole work is based on our previous researches [1–3].

4. RELATED WORK

4.1 Video stabilization

As we discussed, the viewing experience of FPVs includes two parts: video stability and FPMI. Traditional video stabilization techniques can remove the shakiness within FPVs, but they either have limited performance or damage the FPMI with high probability. However, these traditional methods provide some valuable ideas.

Traditional video stabilization techniques include hardware-based methods and software-based methods. Hardware-based video stabilization techniques include the built-in function of wearable cameras such as GoPro Hero 5-7 and other hand-held stabilizers. The problem with this solution is that it has limited performance. The built-in function of GoPro Hero 5-7 can only remove small amounts of shakiness since the camera records in real-time and must avoid obvious stitching errors when applying the stabilization function. Hand-held stabilizers require users to hold the device, which limits the users' activities to riding or driving. Although wearable gimbals are available, they are large and heavy for users who want to have long-duration activities.

To stabilize a video using software-based method, three steps are performed: motion estimation, motion smoothing and frame/video reforming. Based on the motion type they work with, the video stabilization techniques are classified into 2D [4–7] and 3D solutions [8–11]. In the 2D solutions, the motion on the image plane is estimated using either local features or pixel intensity information. Then the frame transformation is calculated based on the smoothed 2D motions. In contrast, 3D solutions estimate the camera poses/3D camera motions, which include both camera orientation and position information in 3D space. The estimation approaches mainly rely on the methods of either Structure-from-Motion (SfM) [12] or visual-based Simultaneous

Localization And Mapping (vSLAM) [13]. The advantage of 3D solutions is that they have a full understanding of the physical camera motions. Given the 3D motion, it is easier to stabilize FPVs according to the human intention. However, 3D solutions are usually slower and more fragile than 2D methods because of the motion estimation step. The motion estimation step in 3D solutions requires enough matching points to be found across several frames and needs the translations between frames to be large. Otherwise, the triangulation process may fail or be inaccurate. In addition, pose optimization in 3D solutions is computationally expensive. The problem of all existing software-based video stabilization techniques is that they over-stabilize the FPVs, which will remove almost all the FPMI.

4.2 Stability measurements

To overcome the over-stabilization problem, measurements for both video stability and FPMI are needed. They can be used to inform us the extent of ongoing stabilization process and the amount of remaining FPMI. Precise measurements can help us accurately find the optimum stopping instance of stabilization so that the resulting video is stable while enough FPMI is preserved.

Currently, there is no related work on measuring FPMI while some works are related to measuring the video stability. Some stability measurements are proposed along with video stabilization techniques. In fact, many video stabilization techniques are not based on measurements [6, 9, 14–16]. The result of lacking a measurement is that their smoothing parameters are not adaptive and the stabilization process is not controllable. However, the rest of the video stabilization techniques [4, 5, 7, 8, 17] implicitly generate stability measurements as side-products. Some other stability measurements are independent studies based on video quality assessment. All these measurements compute stability scores based on the camera motion and can be classified into two categories: the motion types they analyze and the motion properties they measure.

Different motion types: The video stability measurements are based on analyzing 2D or 3D motions, which is similar to the stabilization techniques. 2D estimators [4–7, 18–24] either track local feature points or apply optical-flow algorithms to calculate the 2D motion. 3D methods [3, 8, 10, 11] estimate camera movements from frame to frame using the epipolar geometry.

When deal with the stability estimation, this time, 2D methods have more advantages than 3D methods. In general, 2D methods have two advantages. First, we have mentioned that the 2D motion is faster and more robust to compute. The whole process of 3D motion estimation is fragile and time consuming. Second, theoretically, video stability estimation based on 2D motions has a greater potential to be consistent with human perception. This is because what viewers actually perceive are 2D images with 2D motions. 2D motions include the information of both the camera and the objects in the video. However, the 3D motion only provides information about the camera itself; the scene structure information is missing. However, in our work, we adopt the 3D motion as the subject to measure the video stability when we stabilize FPVs. This is because we not only want to measure the stability of the video and stabilize the video, but also want to measure the FPMI and preserve the FPMI. The FPMI only correlates with 3D camera motion and is independent with 2D motion. Therefore, to achieve these two advantages, in this paper we measure the stability based on 3D motion.

Different motion characterizations: To analyze the stability or shakiness of camera motions, methods focus on two different characterizations of the motion: intensity and frequency. Studies that focus on the camera motion intensity assume that camera motion with higher intensity is shakier. Most of them use the camera motion amplitude as the measurement of the motion intensity. For example, [18, 19] uses the average motion amplitude of local feature points between frames as the measurement. The Inter-Frame Transformation Fidelity (ITF) [20] is a special case in this category, which does not estimate the actual motion but uses the PSNR between adjacent frames as the measurement of motion intensity. One problem of

these methods is that they do not consider the frequency of the camera motion. For instance, back-and-forth motion and single-direction motion may generate the same score under these algorithms but viewers would perceive a different degree of stability.

Therefore, many authors [21–24] consider the problem in the frequency domain. After obtaining the camera motion, some use the first, second and even the third derivative of the motion amplitude to measure the motion shakiness. Many video stabilization algorithms adopt these methods since they are easy to calculate. In [21–23], Motion-vector-based mean squared error (MV-MSE) is used. The original motion is first filtered to generate a smooth motion. Then the difference between the original motion and the smoothed motion is assumed to be the high frequency component, which is used as the shakiness measurement.

However, all these stability measurements are not precise enough. This is because they do not have a reliable human perception model. For the frequency-focus methods, the threshold between shaky motion and smooth motion in the frequency domain is ambiguous. For the intensity-focus methods, numerical relationship between motion size and instability is unknown. A recent work [24] tries to solve this problem by applying a machine learning process to analyze how different frequency bands of camera motion influence human perception.

In this work, we approach this problem using a human perception model learned from psychophysics. And we propose precise measurements for both video stability and FPMI, which can guide us to appropriately stabilize FPVs.

4.3 FPV watchability enhancement techniques

There are also some works about enhancing watchability of FPVs [8,25–27], which share the same target with us. However, they apply a different strategy to the problem of shakiness. Instead of stabilizing frames by smoothing the camera motion, they try to find and remove the frames which cause the shakiness. After the processing, only the semantic segments, for example the segments that contain human faces, are

preserved and construct of the resulting video. As a result, their resulting videos have reduced content and are fast-forwarded version of the original videos.

However, we believe that not only are the selected specific frames semantically meaningful but also there is semantic meaning in First-Person motions. Their strategy has the same problem of the traditional video stabilization techniques, which also ignores and discards FPMI.

5. SYSTEM OVERVIEW

We adopt the pipeline of traditional video stabilization techniques as shown in Fig. 5.1. As we discussed in Section 4.2, we choose to estimate and edit 3D motion. There are three main differences that make our system outperform traditional algorithms in solving the problem of FPVs.

First, the motion smoothing step is replaced by our motion editing since this is where traditional methods damage the FPMI. Within our motion editing step, we employ two procedures: measuring and improving. The failure of traditional video stabilization techniques informs us that stabilization is a too strong procedure for FPVs. We need to precisely measure the video stability and FPMI in order to carefully decide the stopping instance of stabilization.

Second, this motion editing step is based on a human perception model of First-Person Motion conveyed by FPVs, which is seldom considered in traditional solutions. We model the eye movements of viewers as a random process of choosing and tracking targets based on the research in physiology [28–35]. Based on it, we propose our viewing experience measurement which includes both video stability and FPMI. Using our measurement, we can design a new camera path whose goal is to provide more stable view while preserving the FPMI, i.e. improving the viewing experience. On the basis of the human perception model, the result of our viewing experience improvement has higher human preference than traditional video stabilization techniques.

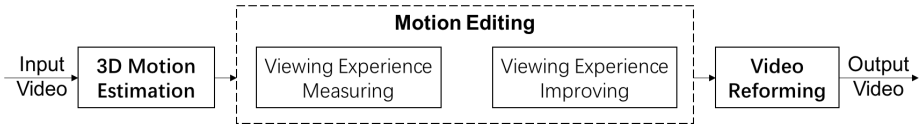


Fig. 5.1.: System pipeline

Third, our system only edits angular motions. Based on our hypothesis, the translations are necessary to convey the FPMI, which should be preserved. As a result, we simplify the traditional 3D motion estimation algorithm to only extract the angular motions. For monocular visual odometers, since there is no information for the scene true depth, the estimated translations between different frames have different unit, which is called the scale problem. To solve this problem, the detected local features used to estimate camera poses need to present in successively several frames. However, we only estimate camera rotations, which are independent with the scales. This relaxes the constraint that enough features must always be seen across time.

6. 3D MOTION ESTIMATION

3D motion estimation or so-called the camera pose estimation is an old problem in both the computer vision and the robotics communities where it is defined as Structure-from-Motion (SfM) or visual Simultaneous Localization And Mapping (vSLAM). Motion or the camera pose usually refers to both the translation and rotation of the camera. Traditional 3D video stabilization techniques apply these approaches and remove both the angular motion and the translation of the camera.

However, we do not remove and thus do not need to estimate the camera translations. As we state, the translations include significant FPMI. The amplitude and frequency of translations can reflect the moving speed and moving approach of the recorder. For example, if the amplitude of forward translation is high, and the amplitude and frequency of the translation in the vertical direction are also relatively high, then we may infer that the recorder is running. Applying traditional stabilization techniques, we may lose this important information since the video would look like it was recorded using a drone if the vertical translation is smoothed or removed.

Also, translations are preferred in FPVs. Their importance is evident from First-Person Video gaming, and watching a FPV is similar to playing a First-Person Video game, except the viewer cannot control the viewing angle. In First-Person Video games, the translations in vertical and horizontal directions are called "head bobbing". Recent First-Person Video games simulate head bobbing to make the games more realistic. For this topic, [36] provided a study and showed that most players prefer games that have head bobbing.

Since the translation is not estimated, the rotation between each two frames can be estimated independently during a first pass, which is called the initialization. After that, we build the pose graph and apply a graph optimization algorithm to reduce the error introduced during initialization step. The initialization step is operated based

on SURF [37] features with RANSAC (Random sample consensus) [38]. For each two frames, we first assume they share a large enough baseline that a triangulation can be performed successfully using epipolar geometry. When there are not enough inliers after the RANSAC, we believe the triangulation fails and the camera motion between these two frames is a pure rotation. In this situation, the fundamental matrix degenerates into a homography.

Suppose the estimated rotation from frame m to frame n is $R_{n,m}$. The estimated camera pose of the i^{th} frame is R_i:

$$R_i = R_{i,i-1}R_{i-1,i-2}\cdots R_{2,1}. \tag{6.1}$$

Using the graph optimization tools provided by [39], we refine the estimated rotations using the objective function:

$$\begin{aligned} &\min_{R_i,R_j\in SO(3)} \sum_{(i,j)\in L} \left\| Log(R_{i,j}^T R_i R_j^T) \right\|^2, \\ &L = \{(i,j) : i-j = l \ and \ i,j \in \mathbb{Z}^+\}. \end{aligned} \tag{6.2}$$

$SO(3)$ refers to the special orthogonal group, which is group of all 3D rotations. It restricts all R to be valid rotations during optimization. i and j are time indices of camera poses. L is the constraint for frame index distance. The larger the L is, the further the two frames are. In the experiments, we found $l = 5$ is large enough to have a large baseline for FPVs recorded in 30fps. Note that this approach requires the camera intrinsic parameters but relaxes the constraint of feature tracking.

These matrices in the special orthogonal group are mathematical expression of camera poses, which do not have a physical meaning to humans. To edit camera motions, we need to decompose these matrices in the same way that humans understand camera motions. After that, we can apply the human perception model to measure the viewing experience of the camera motions.

Humans understand camera motions in the sense of Euler angles. A camera motion is usually understood as combination of pitch, yaw and roll, which are the rotations around the x, y and z axes. Thus, we decompose a camera pose R_i into three Euler angles. However, there is no unique solution of the decomposition. In order to make the results have more realistic meaning and closer to the understanding of humans, the decomposition order should coincide with the importance of human motions. Consider the normal human activities, yaw is the primary and most important motion, which is performed to look around. Pitch has intermediate importance, which is performed to look up and down. Roll is seldom performed intentionally. So our goal is to remove it during our stabilization procedure. As a result, the camera pose of the i^{th} frame R_i is decomposed as:

$$R_i = R_z(\theta_z)R_x(\theta_x)R_y(\theta_y). \tag{6.3}$$

The corresponding angles are computed as:

$$\theta_y = \tan^{-1}\Big(\frac{-R_i(3,1)}{R_i(3,3)}\Big), \tag{6.4}$$

$$\theta_z = \tan^{-1}\Big(\frac{R_{zx}(2,1)}{R_{zx}(1,1)}\Big), \tag{6.5}$$

$$\theta_x = \tan^{-1}\Big(\frac{-R_{zx}(2,3)}{R_{zx}(2,2)}\Big), \tag{6.6}$$

where

$$R_{zx} = R_i R_y(\theta_y)^{-1}, \tag{6.7}$$

and $R(k,l)$ is the (k,l) entry of matrix R.

7. HUMAN PERCEPTION MODEL

The viewing experience of watching FPVs is different than our real life First-Person experience. This is because we rely on different visual functions. In this Chapter, we first introduce human visual functions used in these two situations, which are the Vestibulo-Ocular Reflex (VOR) and Smooth Pursuit Eye Movement (SPEM). Then we introduce our revised eye movement model of SPEM, which considers the practical situation.

7.1 Vestibulo-Ocular Reflex

When the recorder captures FPVs in real life, they never feel dizzy. This is because their bodies apply the Vestibulo-Ocular Reflex (VOR) to compensate for the rotations of their head [28]. The bio-signals from the semicircular ducts located in their ears carry the information of the amplitudes of yaw, pitch and roll and control the extraocular muscles to perform opposite eye motions to compensate the head motion. As a result, the images of interested objects can be maintained at the center of the visual field, i.e. the fovea. This process can be thought as an image stabilization function of the human body. The frequency of VOR can reach up to 100 Hz [28], which ensures the stabilization can be performed in real-time.

7.2 Smooth Pursuit Eye Movement

When watching a FPV, the viewer perform a different eye movement in order to accomplish the stabilization task. Note that the VOR is disabled. The motions that viewers perceive are not performed by their bodies so that their semicircular ducts provide no information. Consequently, to stabilize the images, viewers can

only rely on visual information and must perform Smooth Pursuit Eye Movement (SPEM) [40], which allows humans to closely track visual targets. If we believe the movement information used by VOR is the ground truth, then the SPEM is a process which tries to predict the ground truth of the motion information. Viewers feel dizzy when watching a FPV because this predicting process is not efficient enough.

Before the viewer starts to pursue the target, a catch-up saccade needs to be performed to catch the target, which takes nearly 150 ms [29]. A catch-up saccade may be also triggered when SPEM lag behind or over-shoot the target. Also, during a catch-up saccade, visual information is not processed. All these lead to an experience of instability.

In order to measure the viewing experience of FPVs, we need an approach to measure the stability and FPMI. This requires that we first understand and model the operating mechanism of the catch-up saccade based on [30]. The result in [30] indicates that a catch-up saccade is a reflex triggered when the eye-crossing time (T_{XE}) is outside the range of 40 ms to 180 ms. T_{XE} is defined as:

$$T_{XE} = \frac{-PE}{\omega_{rs}}, \tag{7.1}$$

where PE is the angular position error of the target with respect to the gaze. ω_{rs} is the relative angular speed (target's image speed on fovea) or so-called retinal slip. Intuitively, the ratio of angular position error and retinal slip is the time that human eyes need to catch the target.

However, this model is not enough for building a measurement of viewing experience of FPVs. Next, we discuss the basic geometry of watching FPVs. By considering the practical situations, we present our mathematical model of SPEM for FPVs. Our measurement of viewing experience is shown in Chapter 8.

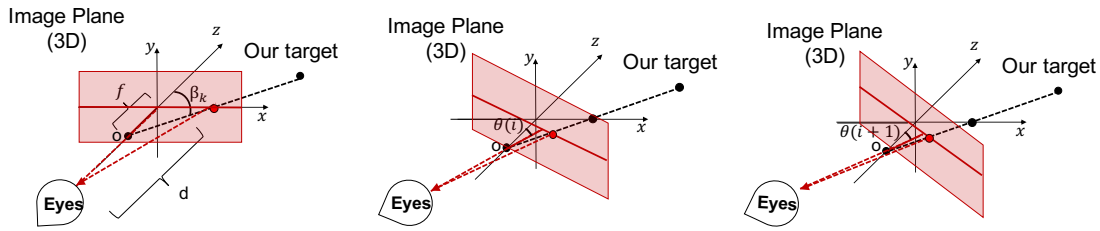


Fig. 7.1.: Images of target object under yaw motion in 3D. Images from the left to right illustrate when the camera yaws from angle 0 (aligned with the world coordinate) to angle $\theta(i)$, and to angle $\theta(i+1)$ and captures the object on the image plane. "Yaw" means the camera rotates about the y axis at the camera center o. Images for pitch motion are similar but the camera rotates about the x axis. β is the angular object position with respect to the world coordinate and is independent of the camera motion. It is only shown in the most left image for clarity.

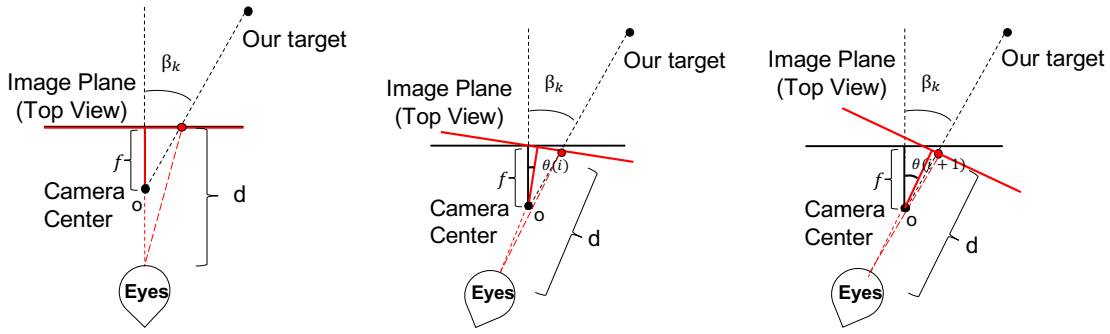


Fig. 7.2.: Images of target object under yaw motion from the top view.

7.3 Mathematical model of SPEM for FPVs

To simply the problem, we develop the model for yaw, pitch and roll motions separately. Suppose the viewer observes a video with viewing distance d and the camera is performing a yaw/pitch motion. Fig. 7.1 illustrates this in 3D. To help readers understand the 3D geometry, Fig. 7.2 shows the top view of Fig. 7.1.

Fig. 7.1 shows the geometric relationship among the target object, camera and viewer across the time. Images from the left to right illustrate when the camera yaws from angle 0 (aligns with the coordinates) to angle $\theta(i)$, and to angle $\theta(i+1)$ and captures the object on the image plane. Images for pitch motion are similar but the camera rotates along the x axis. For FPVs, the translation of the recorder within a short period is relatively small with respect to the depth of most objects. As a result, we assume the relative position of the target with respect to the camera center is fixed.

Suppose the target position with respect to the camera center is β_k at frame k while the camera focal length is f. The viewing distance of the viewer is d and the estimated camera position is θ. So the observation angle of the target for the viewer at frame k is:

$$\varphi_{obj}(k;\beta_k) = \arctan\big[\frac{f\tan\beta_k}{d}\big]. \tag{7.2}$$

At frame k', the observation angle under the coordinates of frame k changes to:

$$\varphi_{obj}(k';\beta_k) = \arctan\big[\frac{f\tan(\beta_k - \theta(k'-1) + \theta(k))}{d}\big], \tag{7.3}$$

The geometry of roll motion is different from yaw and pitch as shown in Fig. 7.3. Suppose the object is on the $x-z$ plane in the k^{th} frame. Then the position at frame k' is:

$$\varphi_{obj}(k';\beta_k) = 2\arcsin\Big[\frac{r\sin(\theta(k'-1)/2 - \theta(k)/2)}{\sqrt{d^2+r^2}}\Big], \tag{7.4}$$

$$r = f\tan\beta_k. \tag{7.5}$$

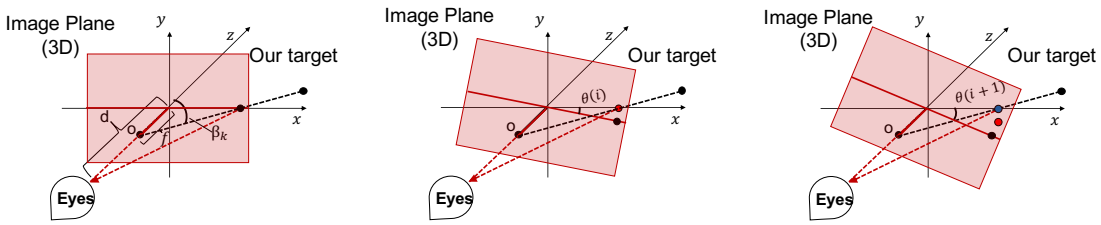


Fig. 7.3.: Geometry of roll motion in 3D. Images from the left to right illustrate when the camera rolls from angle 0 (aligned with the world coordinate) to angle $\theta(i)$, and to angle $\theta(i+1)$ and captures the object on the image plane. "Roll" means the camera rotates about the z axis at the camera center o. β is the angular object position with respect to the world coordinate and is independent of the camera motion.

Assume that the frame rate is 30 and the viewer performs a SPEM from frame k to frame $(k+1)$. Then the $PE(\beta_k)$ and $\omega_{rs}(\beta_k)$ in equation (7.1) at frame $(k+2)$ can be calculated as:

$$PE(\beta_k; k+2) = \varphi_{obj}(k+2; \beta_k) - 2\varphi_{obj}(k+1; \beta_k) + \varphi_{obj}(k; \beta_k), \tag{7.6}$$

$$\omega_{rs}(\beta_k; k+2) = 30 \cdot PE(\beta_k; k+2). \tag{7.7}$$

However, we found it is inaccurate to directly apply the condition given in equation (7.1), for two main reasons. First, the model in [30] ignores the predictive ability of SPEM. Second, the sensitivity of the human visual system needs to be taken into consideration in practice. We improve the model given in equation (7.1), considering each of these two reasons respectively in the following two new conditions:

$$0.04 \leq \frac{|PE(\beta_k; k+2)| + b}{|\omega_{rs}(\beta_k; k+2)|} \leq 0.18 \tag{7.8}$$

$$|PE(\beta_k; k+2)| < MAR, \tag{7.9}$$

where k is the frame index, MAR is the minimum angular resolution of human eyes, and b is the bias of position error estimation. If either one of these two conditions is satisfied, the catch-up saccade will not be triggered. In the next two paragraphs, we explain each of these two conditions.

The first condition, equation (7.8) addresses the fact that equation (7.1) treats the SPEM as an open-loop system. In the experiments in [30], the position errors are generated by changing the target position abruptly. This weakens the predictive ability of SPEM, which is a key closed-loop characteristic of SPEM [31]. In addition, [30] used laser spots or circles as the tracking target to test the SPEM properties of human eyes. This will also underestimate the predictive ability of SPEM, because according to [32, 33], the target shape can provide additional information for visual tracking. [31, 34, 35] also concluded that the predictive ability can be generated by scene understanding or the experiences of motion patterns. None of these factors are

taken into account in [30]. A direct illustration of the failure of their model is that, without the additional information, the target may precede the gaze in both position and velocity, which would create a negative value of T_{XE}.

In contrast, our modified constraint (7.8) treats the SPEM as a closed-loop system. We recognize that the viewer can utilize the additional information (i.e. target shape, scene content) to track targets. The eye movement can be accelerated before the next frame is shown, which makes the retinal slip change and makes the T_{XE} positive and fall into the desired region. This is the reason we introduce the absolute values in equation (7.8). However, there is no widely recognized mathematical model to adjust for retinal slip [31], although most studies agree that the procedure relates to a feedback control that has constant parameters based on visual content, scene understanding or the experiences of motion patterns [31–35]. As a result, to describe the characteristic of the feedback control and make our model compatible, we include an unknown bias parameter b. Later in Chapter 10, we design a subjective test based on real scenes to help us learn about the bias parameter. The learned parameter makes our model accurately describe the SPEM in practical situations.

The second condition, equation (7.9), models a limitation of our visual system. A position error, PE, less than the minimum angular resolution (MAR) cannot be perceived by human eyes. In addition, the human visual system introduces an estimation error when estimating the position error. This is another reason that introducing the bias b is helpful.

Equation (7.8) and (7.9) are conditions on β_k, the angular position of the target at frame k. Solving them for each frame and for each type of motion (yaw, pitch or roll) gives us an interval of β_k. Any object in the current frame that has an angular position within this interval can be tracked without having a catch-up saccade between the next two frames, if the SPEM has already been performed. We define this interval as $B(k)$ for future convenience. $B(k)$ contains all the information we need to compute the viewing experience of FPVs.

8. VIEWING EXPERIENCE MEASUREMENT

The camera motion of a FPV can be decomposed into three types of First-Person Motions (FPMs), which are yaw, pitch and roll. The viewing experience measures of all three types of FPMs are combined to be the viewing experience measure of the FPV. In this Chapter, we propose two viewing experience measurements for FPMs, which are all based on the object position interval $B(n)$. Either measurement can be used to measure the video stability and the FPMI.

The first measurement is called Viewing Experience (VE) score, which is based on a probabilistic model. It measures the fraction of frames in a video that can be tracked using SPEM. The second measurement is call Structure Viewing Experience (SVE) score, which is a speed-up algorithm inspired by the VE score, but does not have the same physical meaning. The approach of combining scores of different types of FPMs is presented after discussing VE and SVE scores. Additionally, we extend the human perception model into 2D motion and propose a 2D VE score. It can be quickly computed and has higher potential to be consistent to human perception to video stability. However, it can only measure the stability but not the FPMI.

8.1 3D Viewing Experience score

The VE score of a FPM is a fraction number indicating the amount of frames can be tracked using SPEM under this FPM. Applying it to different components of a FPM, we can obtain the stability or FPMI of a FPV, which is shown in Chapter 9. Here, we show how a single VE score is computed given a motion.

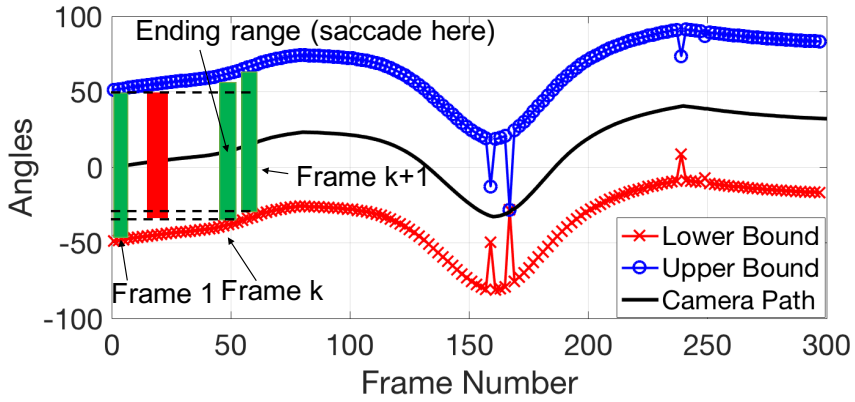


Fig. 8.1.: Example of $\tilde{B}(n)$

Suppose we want to compute the VE of a yaw or a pitch motion. $B(n)$ is the object position interval of each frame. To unify the coordinate of all frames, we add the camera position $\theta(n)$ to $B(n)$:

$$\tilde{B}(n) = B(n) + \theta(n). \tag{8.1}$$

For a roll motion, the object position is influenced by the yaw and pitch motion. So we decompose the $\tilde{B}(n)$ into vertical and horizontal components. And the corresponding $\tilde{B}(n)^V$ and $\tilde{B}(n)^H$ are computed using:

$$\tilde{B}^V(n) = B(n) + \theta_{pitch}(n), \tag{8.2}$$

$$\tilde{B}^H(n) = B(n) + \theta_{yaw}(n). \tag{8.3}$$

In Fig. 8.1, the upper and lower boundaries show an example of $\tilde{B}(n)$. Unlike $B(n)$, $\tilde{B}(n)$ not only includes the object position interval of each frame but also has the spatial relationship between the intervals across the whole video.

Consider the example in Fig. 8.1. The eye movements of the viewer is modeled as a random process. Suppose the viewer randomly chooses a target to track within the field of view (FOV) from the first frame. This target remains at the same object position within a short period with respect to the camera in the first frame. So the

trajectory of this target in Fig. 8.1 is a straight horizontal line. When this line intersects with the boundaries $\tilde{B}(n)$, the viewer loses tracking this target. In this situation, one of two things happens. To keep tracking the same target, the viewer can perform a catch-up saccade. Otherwise, a saccade eye movement is performed to randomly retarget a new object within the FOV. Either of these two procedures takes nearly 6 frames [29].

The procedure of targeting and smooth pursuit is defined as a trail of tracking. A possible path of watching a video consists of several trails of tracking. By calculating the length of each possible path and their probability, we can find the expected value of the fraction of frames for which the viewer performs SPEM. As a result, a wider, more open pathway between the upper and lower bounds shown in Fig. 8.1 will produce a higher expected value, i.e. a higher VE score.

First, we compute the probability of a single tracking trail. Define $V_{i,j}$ to be the event:

$$V_{i,j} = \{\text{Target can be tracked from frame i to j}\}.$$

Define $T_{i,j}$ to be the event:

$$T_{i,j} = \{\text{Target is tracked from frame i and lost at frame j}\}.$$

Then we have:

$$\begin{aligned} Prob(T_{i,i+k+1}) &= Prob(\overline{V}_{i,i+k+1}, V_{i,i+k}) \\ &= Prob(V_{i,i+k}) - Prob(V_{i,i+k+1}), \end{aligned} \tag{8.4}$$

where

$$Prob(V_{i,i+k}) = \frac{\left|\bigcap_{n=i}^{i+k} B(n)\right|}{FOV}. \tag{8.5}$$

Note that $|\cdot|$ computes the difference between the upper bound and the lower bound of a interval. Here we assume the viewer will choose any target in the frame within the FOV with equal chance.

The probability of a possible path is the cumulative product of the probability of all its tracking trails. Suppose for an N-frame video, a possible path $path_m$ has n trails of tracking. Then the probability of $path_m$ is:

$$Prob(path_m) = \prod_{i=1}^{n-1} Prob(T_{l_m(i),l_m(i+1)-6}) \cdot Prob(V_{l_m(n),N}), \tag{8.6}$$

where $l_m(i)$ encodes the start frame index of each trail of tracking in $path_m$. And the length of this possible path is:

$$L(path_m) = N - 6(n-1). \tag{8.7}$$

So our VE score of a video from frame i to frame $(i+N)$ can be computed as:

$$VE(i; N, \theta) = \frac{1}{N+1} \sum_m Prob(path_m) \cdot L(path_m), \tag{8.8}$$

where θ is the camera motion.

The reason we calculate a VE score for only a few frames instead of the whole video is that objects may only be visible for a short period. To compute the equation (8.18), we first identify all the possible paths for a length $(N+1)$ video. Then we check whether each path is feasible or not. A path is not feasible when any of its trails are not feasible, which is indicated when the probabilities in equation (8.4) and (8.6) are less than 0. Then we let:

$$Prob(path_m) = 0. \tag{8.9}$$

To reduce the computational complexity, we set $(N+1)$ to 10. As a result, for a K-frame video, $\tilde{B}(n)$ of yaw, pitch or roll is a $(K-2)$ by 2 vector, and their VE score has length $(K-12)$.

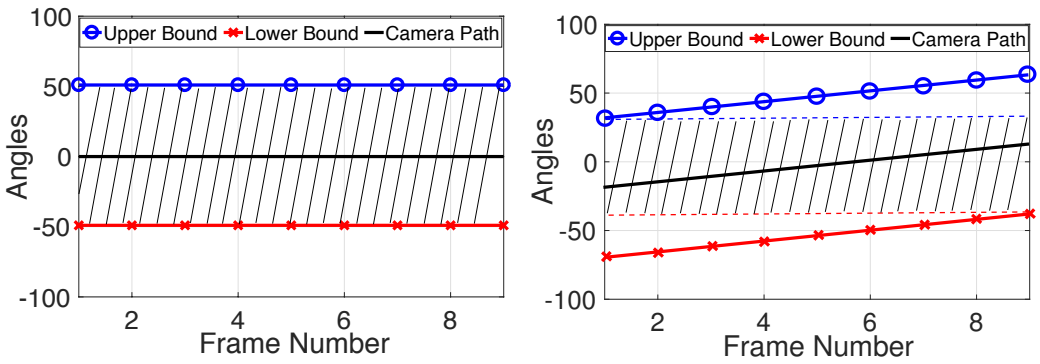


Fig. 8.2.: $\tilde{B}(n)$ of zero motion and motion with slope

8.2 Structure Viewing Experience score

The computing procedure of VE scores is a little time consuming. For $N + 1 = 9$, computing the VE of a 300-frame video requires around 0.425 seconds. On the contrary, the Structure Viewing Experience (SVE) score introduced in this section only needs 0.155 seconds. However, the price is that the SVE score does not have the physical meaning like the VE score.

The SVE score shares the same mechanism with the VE score. Fig. 8.2 shows the $\tilde{B}(n)$ of a zero motion and a motion with slope. The zero motion has the largest VE score since a target can be tracked across the whole video as long as it is within the FOV. However, the motion with slope yields the VE score that is smaller than 1. Intuitively, the shaded area in Fig. 8.2 decreases as the motion slope increases. This reflect the fact that the VE score not only depends on the interval value $\tilde{B}(n)$ of each frame but also depends on the shape of $\tilde{B}(n)$.

The more open the pathway of $\tilde{B}(n)$ is, the higher VE the video has, which is the inspiration of the SVE score. The most straight forward approach would be to compute the shaded area in Fig. 8.2. However, this becomes more complicated

when the motion amplitude varies more heavily or frequently. Instead, we apply the following equation to obtain a similar measurement, the SVE score, to the VE score:

$$SVE(i;N,\theta) = 1 - \frac{2}{(N+1)N} \cdot \frac{\sum_{m=0}^{N-1} \big(\Delta\tilde{B}(i+m)\big)\big(N-m\big)}{FOV}, \tag{8.10}$$

$$\Delta\tilde{B}(n) = \big|\tilde{B}(n+1)\big| - \big|\tilde{B}(n+1) \cap \tilde{B}(n)\big|. \tag{8.11}$$

where $|\cdot|$ computes the difference between the upper bound and the lower bound of a interval.

We apply equation (8.11) to compute the $\Delta\tilde{B}$ of each pair of adjacent frames, which quantifies the amount of objects we would lose tracking from frame n to frame $(n+1)$. Then we compute the SVE score for a short period of the video using equation (8.10). The main idea is to assign different weights to $\Delta\tilde{B}$ of different time instants. $\Delta\tilde{B}$ of earlier time instants have more influential to the openness of the object position interval $\tilde{B}$ across the time, which also can be explained using the right plot in Fig. 8.2. If a target is lost at the second frame, then this target cannot be tracked in totally 7 frames. If a target is lost at the 8^{th} frame, then it only influences the viewing experience for only 1 frame.

By considering the openness of the object position interval $\tilde{B}$ across the time, we make the SVE score has the similar property and measure performance with the VE score. Experiments of performance comparison are present in Chapter 10.

8.3 VE and SVE score of a video

Either the resulting VE score or the SVE score of a single FPM is a vector. Recall that each input video has three types of FPMs: yaw, pitch and roll. As a result, a FPV has 3 VE/SVE score vectors in total. To obtain a single measurement, we combine all three vectors into a single vector, which describes the fraction of frames

can be tracked using SPEM temporally. Suppose the measurement for the three FPMs are M_y, M_x and M_z. Then the measurement for the whole FPV is M_{all}:

$$M_{all}(i; N, \theta) = \min_{j=x,y,z} M_j(i; N, \theta). \tag{8.12}$$

Equation (8.12) suggests that the measurement of a FPV is also a vector. Each element of the vector is a VE/SVE score for a short period. This VE/SVE score is limited by the measurement of the most shaky motion among the yaw, pitch and roll.

In order to apply optimization procedures and do further analysis, in the following Chapters, the VE/SVE scores refer to the mean square value of M_{all}, which are noted as $\mathbb{M}$. For a K-frame video, M_{all} is a vector with length $(K - N - 2)$. As a result, $\mathbb{M}$ is computed as:

$$\mathbb{M}(\theta) = \frac{\|M_{all}(i; N, \theta)\|}{K - N - 2}. \tag{8.13}$$

8.4 2D Viewing Experience Score

Our proposed 3D Viewing Experience (VE) score and Structure Viewing Experience (SVE) score are both based on 3D camera motion. As we discussed in Section 4.2, 2D motion is more robust and faster to estimate than 3D motion. Theoretically, video stability estimated based on 2D motion has a greater potential to be consistent with human perception since it contains object motion information that 3D motion does not have. Motivated by this, we extend the model used by the 3D VE/SVE to 2D and propose a 2D Viewing Experience score. However, note that the 2D VE is suitable to measure the video stability for applications that need to quickly estimation, but it has nothing to do with the FPMI since FPMI can only be computed through 3D motion.

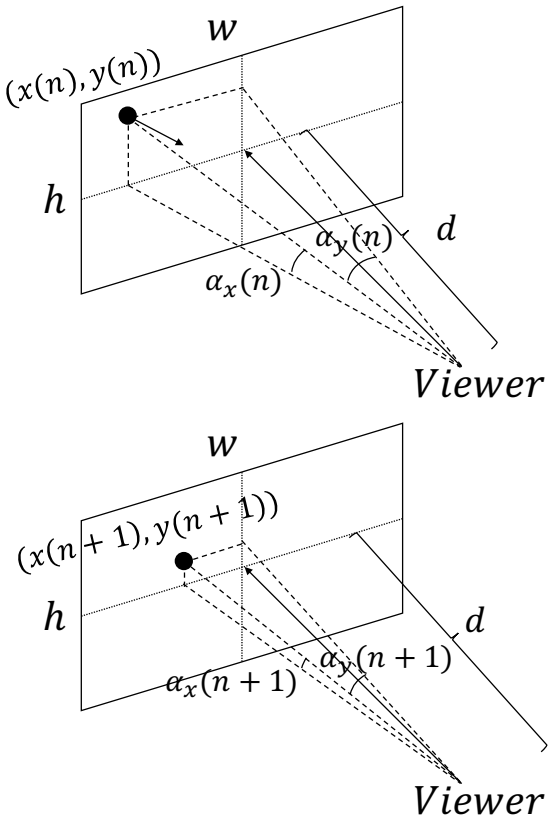


Fig. 8.3.: 2D image motion of a visual target from frame n to frame $(n+1)$.

8.4.1 2D-Mathematical model of SPEM

To create our 2D-VE, we recalculate condition (7.8) and (7.9) based on target position on the screen with respect to human eyes using the 2D image motion of interest points instead of target angular position β_n with respect the camera.

Assume a viewer tracks a target from frame n to $(n+1)$ and its 2D image motion is estimated as shown in Fig. 8.3. The target moves from $\big(x(n), y(n)\big)$ to $\big(x(n+1), y(n+1)\big)$ while this event is observed by the viewer with viewing distance d on an image with width w and height h. $\alpha_x(n)$ and $\alpha_y(n)$ are target angular positions in the viewer's eyes in the horizontal and vertical directions respectively. $\alpha_x(n)$ can be calculated as:

$$\alpha_x(n) = \arctan \frac{\big|x(n) - w/2\big|}{\sqrt{\big(y(n) - h/2\big)^2 + d^2}}. \tag{8.14}$$

$\alpha_y(n)$ and similar results can be derived for target angular positions in frame $(n+1)$ and $(n+2)$. We assume the viewer's eye movement has a constant speed after the target is tracked at frame n. Then the position error PE and retinal slip ω at frame $(n+2)$ in the horizontal or vertical direction can be calculated:

$$PE_k(n+2) = \alpha_k(n+2) - 2\alpha_k(n+1) + \alpha_k(n), \tag{8.15}$$

$$\omega_k(n+2) = \frac{PE_k(n+2)}{FPS}, \tag{8.16}$$

where $k = x$ or y indicating the horizontal and vertical components. FPS is the video frame rate. After substituting Equation (8.15) and (8.16) into condition (7.8) and (7.9), if the condition is satisfied for both vertical and horizontal motion, we can conclude that the current tracked target can be continually tracked at frame $(n+2)$. We use $C(\cdot)$ to denote the processing of checking condition (7.8) and (7.9) based on the input motion. The output of $C(\cdot)$ is binary: "1" indicates that the condition is satisfied for both vertical and horizontal motion. "0" indicates a catch-up saccade will be triggered. During the required 200ms (about 5 frames at 30fps) nearly no information can be perceived. In the next section, we introduce how the video stability is computed using this 2D mathematical model.

8.4.2 2D-VE of a video

Given a video, we first identify the interest points in each frame. Then we calculate the possible tracking length of each interest points using the model we propose above. The stability of a video is measured using the averaged possible tracking length of all interest points.

For a N-frame length video segment, we split its frames into I by J regions (vertically and horizontally). For each region, we choose its center as the interest point. Then we calculate the optical-flow of each frame. Suppose we start to track the target from frame k and focus on the $(i,j)^{th}$ interest point. Its pixel location at frame $n(k < n \leq N)$ is interpolated using the computed optical-flow of frame n, denoted as

$m_{i,j,k}(n)$. We define the stability S of the $(i,j,k)^{th}$ interest point at frame $(n+2)$ to be:

$$S_{i,j,k}(n+2) = C\big(m_{i,j,k}(n); m_{i,j,k}(n+1)\big) \tag{8.17}$$

$S_{i,j,k}(n+2)$ represents whether the $(i,j,k)^{th}$ target can be tracked using SPEM at frame $(n+2)$. This is calculated only based on the motion of frame n and $(n+1)$. However, to know about the stability of this target across the time, we need also consider the catch-up saccade periods. For example, if $S_{i,j,k}$ is "1011111", then it needs to be converted to "1000001" to account for the fact that once a viewer can no longer track at the second frame, and that a catch-up saccade lasting 5 frames is required. For the converted resulting vector, we use the notation $S^*_{i,j,k}$.

To calculate the stability of an N-frame video segment, we need to calculate equation (8.17) for all IJ interest points and for all possible starting frame index k $(1 \leq k < N)$. The overall video stability is estimated using our 2D-VE score, which is calculated as:

$$VE_{2D} = \sqrt{\frac{1}{K-N+1}\sum_{k=1}^{K-N+1}\Big(mean_{i,j}\frac{|S^*_{i,j,k}|}{N}\Big)^2}. \tag{8.18}$$

K is the total number of frames while $|\cdot|$ computes the L1 norm. $\frac{|S^*_{i,j,k}|}{N}$ calculates the fraction of frames over which the $(i,j,k)^{th}$ interest point can be tracked using SPEM. And the overall VE score is the average over all interest points. In the experiments, we set N to 30 frames, I and J to 10 and 20.

9. VIEWING EXPERIENCE IMPROVEMENT

To improving the viewing experience of a FPV, we edit its First-Person Motions (FPM) and reconstruct frames to create a new version of the video. In this Chapter, we start from systematically defining the FPM including both structure and properties. As we mentioned, the properties are the stability and FPMI. Then we introduce the approach of measuring the properties using our VE/SVE scores. Again, the 2D VE can only measure the stability but not the FPMI. As a result, we will ignore it in this chapter. At the end, we demonstrate the optimization procedure that helps editing the FPM given the measurements of the stability and FPMI.

9.1 First-Person Motion

First-Person Motion (FPM) is the angular camera motion estimated from FPVs, which has three types: yaw, pitch and roll. We first introduce its properties which are the intuitive feelings that viewers can directly perceive. After that, we introduce its structure, which controls the properties. Following this order, reader may have better understanding of the logic of our measurement assignment of each property in Section 9.2.

9.1.1 Properties of FPM

The FPM has two properties: stability and First-Person Motion Information (FPMI), which consist of the viewing experience of FPVs. The stability describes the comfort extent of watching the FPV. A low stability makes it difficult for the viewer to perceive the content of the video and may even causes dizziness.

The FPMI is the information conveyed by FPM. It has two sub-parts: First-Person motion range (FPMR) and First-Person feeling (FPF). FPMR is the objective information, which is produced when the recorder looks around. The damage to FPMR when doing motion editing makes the viewer lose any chance to observe some particular objects. FPF is the subjective information, which is the sense of watching activities captured from the First-Person Perspective. It is produced by three conflicts. The first and main conflict for the viewer is that the motion he/she perceives by watching FPVs is not consistent with what he/she perceives in his/her own First-Person experience. The more obvious the conflict is, the more obvious the FPF is. Actually, in real life, viewer eyes do not perceive many large motions using SPEM (but using VOR).

The second and third conflicts have limited influence for the viewing experience of FPVs but do exist. The second conflict is between the feeling of watching FPVs and traditional videos (such as cinematographic videos). The viewer finds that the current video (FPV) is not like the video he/she usually watches. By watching more FPVs, this conflict may be eliminated. The third conflict happens in the vestibular system caused by mismatched motions: the visual motions make humans feel that their body is moving while the body has no physical motion. It causes disorder in the visual system, which is called vestibular illusion. However, this effect varies with the strength of visual cues as illustrated in [41]. Stronger visual clues make the viewer more confident about the mismatched motions. Compared with VR systems, this effect has limited influence for 2D screens where FPVs are played back.

Fig. 9.1 shows the relationship between the stability and FPMI. In general, there is a trade-off between them. When the FPM has large amplitude, the FPMI increases (in both FPMR and FPF) while the stability decreases. However, it is still possible to increase the stability while preserving the FPMI, i.e. increasing the viewing experience of a FPV. This is because the trade-off between stability with FPF and stability with FPMR occur at different timing and in different manner. More detailed discussion are in Section 9.2.

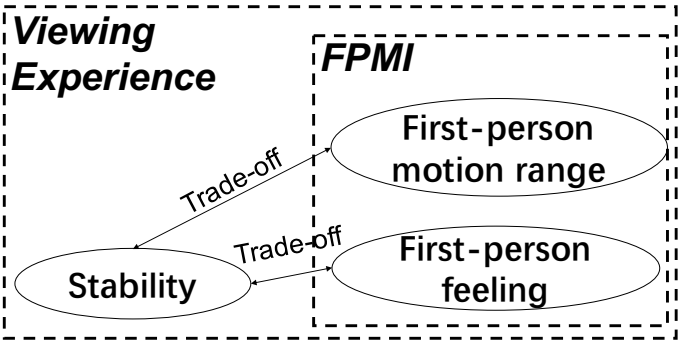


Fig. 9.1.: Properties of First-Person Motion

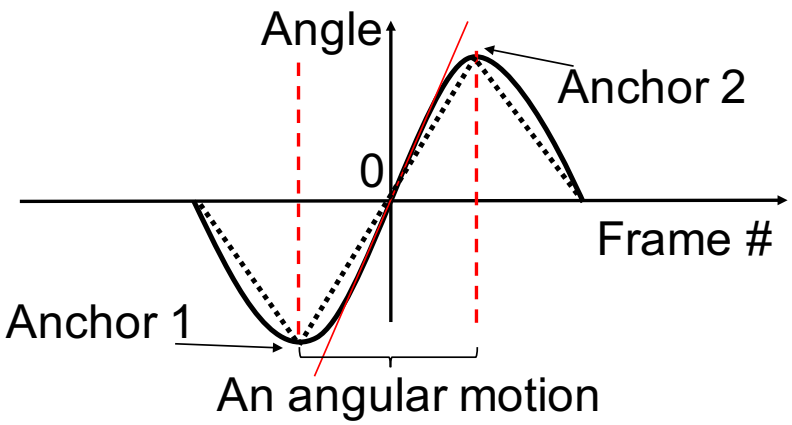


Fig. 9.2.: Example of a First-Person Motion

9.1.2 The structure of FPM

Two structures consist of a FPM: motions anchors and motion shape. An example of a FPM is shown in 9.2. Motion anchors are the local extremes of the FPM. It can be thought as the skeleton of the FPM. When the motion anchors are fixed, the motion amplitude and motion frequency of the FPM is fixed, which make the FPMR and FPF fixed respectively. The motion shape is the path between motion anchors, which only influences the stability of FPVs. Note that the stability is also influenced by the motion anchors since the possibilities of motion shape is limited by the motion amplitude and motion frequency.

So the general strategy of our system is a two-step iteration. First, we modify the motion amplitude of original motion anchors, which increases the video stability while still preserving certain amount of FPMI. Second, we find the optimal motion shape based on the new motion anchors, which yields the highest video stability. The iteration terminates when we find the desirable motions that preserve an adequate amount of FPMI and maximize the video stability.

Note that in the iteration, to simplify the problem, we only modify the motion amplitude and motion shape of the FPM while preserving the motion frequency.

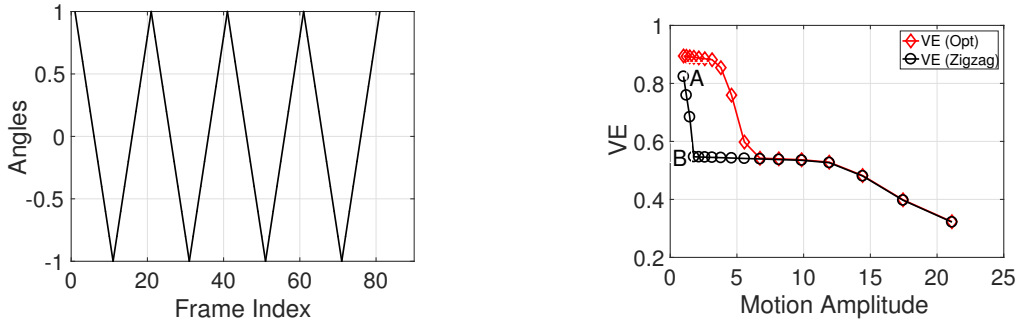
To apply optimization algorithms to perform the iterations, we need a measurement for the motion anchors. The problem is that motion anchors is just a set of points while the measurements we proposed in previous Chapters are only suitable for motion curves. As a result, we define a motion which is one-to-one relationship with motion anchors: the pure FPM. The pure FPM is the zigzag path that connects all motion anchors. An example of pure FPM is shown in Fig. 9.2 using dotted lines. The pure FPM is uniquely determined by motion anchors, and vice versa. As a result, the VE/SVE score of the pure FPM is a descriptor of motion anchors. Since the FPMI is only determined by motion anchors, any measurement of the pure FPM is a measurement of the FPMI.

9.2 Measurements assignment

To quantify the properties of the FPM, we extract corresponding structures and assign different measurements to them. The aim here is demonstrating the logic of the assignment.

Our proposed VE/SVE score measures both the stability and FPMI of FPVs. Given the estimated FPM of a FPV, the stability is measured using the absolute value of VE/SVE score since their definition align with each other. The higher the VE/SVE score of the FPM is, the higher fraction of frames the viewers can perceive and the higher stability the FPV has. The FPMI is measured by the negative VE/SVE score of the pure FPM of the original FPM. This is because, firstly, any measurement of the pure FPM is a measurement of the FPMI. Secondly, the FPMI increases when the main conflict discussed in Section 9.1.1 becomes more obvious, which happens when then FPM has higher frequency and larger motion amplitude. In this situation, the VE/SVE score of the pure FPM decreases.

To better understanding the logic of the measurement assignment, consider the example in Fig. 9.3. Fig. 9.3 (a) shows an example of pure motion: a zigzag path connects motion anchors which have fixed frequency and amplitude. We perform



(a) Example of pure motion

(b) VE scores

Fig. 9.3.: VE scores based on pure motions

the easing function introduced in the next section to find the optimal motion shape that produces the highest VE/SVE scores. This process is performed repeatedly by varying the motion amplitude from 1 to 20. The VE/SVE scores of the pure motion and their corresponding optimal motion are shown in Fig. 9.3 (b). In this example, the ratio of viewing distance with respect to the focal length is 6. Increasing or reducing this ratio only shifts the curves to the right or left while the shapes do not change.

This example shows that when the motion amplitude increases, the VE score of the optimal motions decrease since the camera/view shakes more heavily. Thus, it is reasonable to assign the VE/SVE score to be the measurement of FPVs' stability. The things happen on the VE score of the pure motion is more interesting. First, there is a sharp decrease from point A to point B in Fig. 9.3 (b). Since we proposed the negative VE/SVE score of the pure motion to be the measurement of the FPMI, FPMI increases from point A to point B. This is because point B is where the FPF becomes noticeable, which means the main conflict in viewers' visual system become obvious. It aligns with the fact that from point A to point B, the amplitude of FPM gets larger. As the motion amplitude increase beyond point B, the VE scores of the pure motion decrease gradually, which indicates the increasing of FPMR caused by increasing angular motion variation.

Based on Fig. 9.3 (b), we realize that it is possible to increasing the stability of FPVs while preserving their FPMI. Suppose the original FPV has large motions and situation locates at the most right point on curves in Fig. 9.3 (b). When we start to stabilize the FPV, the situation shifts on the curves and to the left. During this period, the video stability increases while the FPMR decreases. So optimal stopping instance point B, where the video loses no FPF while has large enough video stability. However, note that this does not mean we have to lose all FPMR until we get to point B. A more specific explanation and mathematical solution is given in the next section.

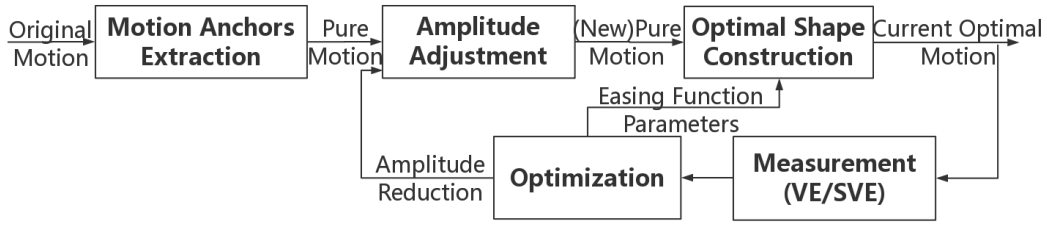


Fig. 9.4.: General procedure of motion editing

9.3 Optimization procedure

Our optimization procedure operates based on the two-step iteration mentioned in Section 9.1.2. In this section, we first systematically introduce the general procedure and the mathematical definitions of parameters need optimizing. Then we propose and discuss our objective function which enables us to increase the video stability while preserving the FPMI. Finally, we supplement the speeding up method of the optimization procedure.

9.3.1 General procedure and optimization parameters

Fig. 9.4 show the pipeline of the general optimization procedure. Given the original motion, we first extract the pure FPM. The optimization module focus on two tasks: adjusting the motion amplitude and searching the optimal motion shape that yields the highest VE/SVE score given the amplitude adjusted motion anchors. This means there are two sets of optimization parameters: motion amplitude reduction rates and motion shape controlling parameters.

Suppose the frame index of the i^{th} motion anchor is $A(i)$, the original camera motion is θ and the new camera motion is $\tilde{\theta}$. Their pure motions are θ_p and $\tilde{\theta}_p$.Then the motion amplitude reduction rates of the i^{th} motion anchor is defined as $D(i)$:

$$D(i) = \frac{\left|\tilde{\theta}_p\big(A(i)\big) - \theta_p\big(A(i)\big)\right|}{\left|\theta_p\big(A(i)\big) - \theta_p\Big(A\Big(\arg\min_{i\neq j}\big|A(j) - A(i)\big|\Big)\Big)\right|}. \tag{9.1}$$

Equation (9.1) groups the nearest two motion anchors and believes that they consist of a FPM. The absolute amplitude difference between these two anchors is set to the base size of the amplitude. The reduction rate is the ratio of the reduction amount with respect to the based size.

We apply the easing function method to search the optimal motion shape. Easing function methods are popular in the computer graphics community [42,43], which are usually used to construct smooth motions. However, if we adopt the n-dimensional polynomial easing function to search motion shapes, there will be n unknown parameters need optimizing for only one motion anchor pair. Usually, a 300-frame FPV has over 30 single FPM. For longer videos, this parameter amount is unacceptable due to the computational cost. As a result, we construct our own easing function in in equation (9.2) which only requires 2 parameters for a motion anchor pair.

$$\tilde{\theta}(n; k_i, s_i) = \tilde{\theta}_p(n) + s_i \cdot \Big[\frac{n - \big(A(i) + A(i+1)\big)/2}{\big(A(i+1) - A(i)\big)/2}\Big]^{k_i} \Delta\tilde{\theta}_p(n), \tag{9.2}$$

$$\Delta\tilde{\theta}_p(n) = \tilde{\theta}_p\Big(\arg\min_{A}\big(A(i) - n, n - A(i+1)\big)\Big) - \tilde{\theta}_p(n). \tag{9.3}$$

Our easing function considers the constraint that there is no local extreme between two input motion anchors. In equation (9.2), k controls the degree of the shape while s controls the scale.

9.3.2 Objective function

Our primary goal is to increase the video stability while preserving an adequate amount of FPMI. Our conclusion is that the theoretical optimal stopping instance of stabilizing is the point B in Fig. 9.3. At point B, the FPF just becomes noticeable and the stability of the optimal motion we can get is at a high level. So the optimization strategy is find the point where the VE/SVE score of the optimal motion is high (to have a high stability) and the VE/SVE score of the corresponding pure motion is low (to have necessary amount of FPMI). However, this is not enough.

We also need to pay more attention to the FPMR of some important motion. We can increase the video stability and preserve an adequate amount of FPMI. This damages the FPMR since reducing motion amplitude will definitely lose part of the image. However, this process treats unintentional motion and intentional motion the same. We prefer to keep the FPMR of intentional motions. To solve this problem, we make compensation to the motion anchors where the FPMR is important. For example, for the FPMs caused by vibrations or unintentional head motions, the original FPMR contains no information of the recorder's personal interests. For the FPMs that are larger than the FOV, we believe that the recorder perform the motion intentionally and the FPMR has significant information. As a result, our objective function is constructed as:

$$\min_{D,k,s} \left[1 - \mathbb{M}(\tilde{\theta}) + \mathbb{M}(\tilde{\theta}_p)\right] + \alpha_{FPMR} \cdot \Delta FPMR^T, \tag{9.4}$$

where $\tilde{\theta}$ is the edited motion and $\tilde{\theta}_p$ is the corresponding pure motion. $\mathbb{M}(\cdot)$ is either $VE(\cdot)$ or $SVE(\cdot)$. α_{FPMR} is the vector of weights for motions that are larger than FOV:

$$\alpha_{FPMR}(i) = \frac{\Delta\theta(i) \cdot \mathbb{1}_{\{x>FOV\}}\Delta\theta(i)}{\sum_j \Delta\theta(j) \cdot \mathbb{1}_{\{x>FOV\}}\Delta\theta(j)}, \tag{9.5}$$

$$\Delta\theta(i) = \theta\big(A(i)\big) - \theta\big(A(i-1)\big), \tag{9.6}$$

And $\Delta FPMR$ is the vector of distortions:

$$\Delta FPMR(i) = 1 - \frac{\tilde{\theta}\big(A(i)\big)}{\theta\big(A(i)\big)}. \tag{9.7}$$

9.3.3 Speeding up method of optimization

Our optimization procedure is performed baed on particle swarm, which enables us to find the global optimizer. For a camera motion which has T motion anchors, the proposed objective function has $T + 2(T-1)$ variables, where T is needed by the reduction rate and $2(T-1)$ is needed by the motion shape searching. To find the global optimizer, when T is large, more particles and longer convergence time are required. As a result, we construct a look-up table which produces the values of motion shape variables k and s if we provide the motion amplitude.

To construct the look-up table, we pre-find all the k and s using the objective function below and vary the motion amplitude from 0.1 to 30.

$$\min_{k,s} 1 - \mathbb{M}(\tilde{\theta}). \tag{9.8}$$

This look-up table enables us to eliminate the parameter k and s in equation (9.4), which reduce the number of variables by 3 times. Since the yaw and pitch motion have different mathematical motion with the roll motion, their training process need to be taken separately.

10. EXPERIMENTAL RESULTS

In this Chapter, experimental results are present in two parts: the performance of our proposed measurement and our whole system. Each part includes both objective tests and subjective tests.

10.1 Tests for viewing experience measurement

So far we proposed our viewing experience measurements based on the model of viewers' eye movement. As we discussed, the measurements could help us finding the optimal stopping instance of stabilization, where the video stability is improved and also an adequate amount of FPMI is preserved. As a result, it is necessary to test the performance of the measurements.

First, we design a subjective test to help us complete the design of our eye movement model, which is the basis of the measurements. After that, an objective test is performed to test the robustness of the measurements under noise and a subjective test is applied to test its effectiveness in real FPVs.

10.1.1 Perception model parameter estimation

The bias parameter is the only thing we do not know in our human perception model, which is used for both the 3D VE and 2D VE. Our target here is to find an accurate bias parameter so that the measurement has a higher correlation coefficient with the human subjective scores. The higher the correlation coefficient is, the more accurate the stopping instance we can find.

The general idea is obtaining subjective scores of the stability of FPVs and adjusting the bias parameter. The optimal bias parameter is found when the Pearson

(a) Lobby

(b) Market

(c) University Street (US)

(d) Park Walkway (PW)

Fig. 10.1.: Example frames of source videos

Linear Correlation Coefficient (PLCC) between subjective scores of the stability and the VE scores is maximized. This process can also be performed using FPMI instead of video stability, since they are based on the same measurement. The reason we choose video stability is that it is easier to explain to subjects the content they are asked to compare. As a result, the subjective test result would be more accurate. Note that we also perform this experiment for the 2D VE score.

The designed subjective test is based on paired comparison, which includes 19 subjects. The test videos include 4 scenes. For each of the scene, there are 9 version of videos, each having a different combination of motion amplitude in both yaw and pitch to create different levels of video stability. Each video is 5 seconds long and is played back on a 27-inch, 82 PPI screen at full resolution. The ratio of viewing distance with respect to the equivalent focal length is 4. During the test, for each comparison, only one question is asked: **Which video is more stable?**

Fig. 21.1 shows sample frames of the test videos. The resolution of all videos is 1080p and the frame rate is 30 fps. These test videos are generated by adding synthesized motions into 360° videos which are captured using a 360° camera-set on a wheeled tripod. To capture the 360° videos, we slowly and smoothly move the

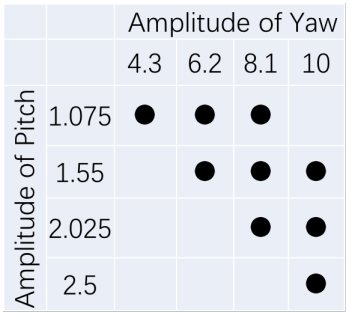


Fig. 10.2.: Motion amplitude settings

tripod forward and record the scene. The obtained six videos are stitched using Kolor Autopano Video 3 [44] with D-warp, color normalization, video stabilization and other options on. Each frame of the resulting video is a equirectangular image. Based on the synthesized motion of each frame, we centralize the camera view at a particular part of the equirectangular image to create a perspective image. Collecting all resulting perspective images, we generate a video with synthesized motions. The camera-set consists of 6 GoPro Hero Session 4 cameras. According to the motion model used in First-Person video games [45], yaw and pitch motions are synthesized using sinewaves while the rate of pitch is twice as that of yaw and the motion amplitudes are shown in Fig. 10.2. All the synthesized videos are designed to mimic First-Person running videos. If the normal running speed is R meters per second, the step size is S meters and the frame rate of recorded video is F_{rate}, then the period of pitch motion T is $\frac{F_{rate}}{R} \cdot S$. Suppose the original video records M meters' moving in N frames, then the fast-forward multiple of the original video is $\frac{N \cdot R}{F_{rate} \cdot M}$.

There are two main reasons for using this synthesizing process. First, it enables easy and accurate access to the camera motion since we actually generate it. Second, it enables us to create videos that do not contain several potential distortions. For example, since we move the tripod smoothly and slowly, videos are free from rolling shutter. And since we use the 360° videos, all the test videos are free from black areas.

Table 10.1.: Bradley-Terry scores of all videos

			Yaw Motion Amplitude			
			4.3	6.2	8.1	10
Lobby	Pitch Motion Amplitude	1.075	0 (±0.345)	-3.057 (±0.422)	-5.105(±0.550)	
		1.55		-4.027 (±0.533)	-5.987 (±0.622)	-7.813 (±0.809)
		2.025			-6.464 (±0.652)	-9.021 (±0.956)
		2.5				-9.857 (±1.095)
Market		1.075	0 (±0.387)	-1.262 (±0.431)	-2.184 (±0.495)	
		1.55		-1.268 (±0.468)	-2.683 (±0.555)	-4.445 (±0.720)
		2.025			-3.199 (±0.580)	-4.906 (±0.799)
		2.5				-5.298 (±0.870)
US		1.075	0 (±0.370)	-3.057 (±0.463)	-5.041 (±0.588)	
		1.55		-3.734 (±0.551)	-5.761 (±0.671)	-7.087 (±0.796)
		2.025			-6.948 (±0.763)	-7.981 (±0.897)
		2.5				-8.641 (±1.006)
PW		1.075	0 (±0.434)	-1.885 (±0.468)	-3.202 (±0.515)	
		1.55		-2.083 (±0.455)	-3.792 (±0.566)	-5.101 (±0.658)
		2.025			-4.530 (±0.624)	-5.950 (±0.729)
		2.5				-6.516 (±0.827)

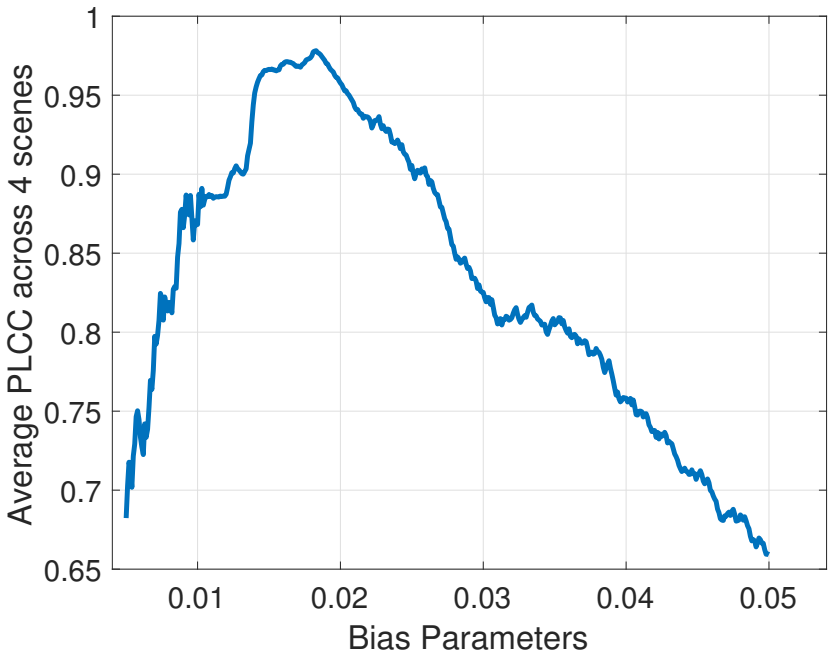


Fig. 10.3.: Average Pearson linear correlation coefficients of different bias parameters across 4 scenes

The subjective test results are expressed using Bradley-Terry scores and are shown in Table 10.1. To find the optimal bias parameter that maximizes the correlation between subjective scores and our measurement, suppose the bias parameter is b, BT scores of i^{th} scene are stored in vector BT_i and the corresponding VE scores is VE_i. Then we use following equation to find the optimal bias parameter:

$$\max_b J = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(corr(BT_i, VE_i))^2}, \tag{10.1}$$

where $corr(\cdot)$ computes the Pearson linear correlation coefficient (PLCC).

For each possible bias parameter, we computes the mean squared value of PLCC between the BT scores and VE scores across 4 scenes. The results under different bias parameters are shown in Fig. 10.3. As can be seen, the optimal bias parameter is 0.0183 for 3D motion model. And for the 2D motion model, the optimal bias parameter is around 0.038.

10.1.2 Measurement robustness

Our measurements VE and SVE are subjective measurements. Usually, the performance of this kind of measurements are evaluated using subjective test, but objective tests are also useful. We apply objective tests to evaluate the robustness of our measurements, since objective tests are easy to perform massively.

The robustness here refers to the property of the measurements that similar objects which human subjects cannot distinguish should theoretically have same subjective measurement result. In practical, the less the difference between the measurement results, the more robust the measurement is. However, this is not absolute. Using the objective tests in this section and subjective test in the next section, we will show that high robustness will degrade the effectiveness.

Our objective tests are based on similar enough motions that are expected to have nearly equivalent viewing experience for viewers. First, we use the sine-wave as our base motion. Then we modify the sine-wave motion by making small changes. The small changes are bounded by the minimum angular resolution (MAR) so that the viewer should not be able to perceive apparent difference between the base motion and the synthetic motions. After that, we compute the VE/SVE scores of the base motion and the synthetic motions. Then we are able to compute the measurement errors caused by the difference between the base motion and the synthetic motions using:

$$e = \frac{1}{r}\sum_{i=1}^{r}\left|\frac{\mathbb{M}(\theta_{syn}^{i}) - \mathbb{M}(\theta)}{\mathbb{M}(\theta)}\right| \cdot 100\%, \tag{10.2}$$

where r is the number of experiment trials. The sine-wave we use has amplitude 1, period 20 and 12 periods. θ_{syn} is the synthetic motion and θ is the base motion. We perform the experiment by increasing the amplitude of the sine-wave from 1 to 20. For each amplitude the number of trials r is set to 100 and MAR is set to be 0.02.

The synthetic motions are generated using 3 operators: flipping, adding Gaussian noise and adding slope. The flipping operator simply flips the sine-wave from left

to right or up-side down. The operator of adding noise adds normally distributed Gaussian noise using the following equation:

$$\theta_{syn}(n) - \theta_{syn}(n-1) = \theta(n) - \theta(n-1) + \mathcal{N}\Big(0, \big(\frac{MAR \cdot d}{3f}\big)^2\Big), \tag{10.3}$$

where f is the camera focal length and d is the viewing distance in pixels. The adopted distribution bounds the amplitude of the noise so that the noise added to the position error is smaller than MAR in human eyes with probability 99.73%. It is equivalent to:

$$\theta_{syn} = \theta + \mathcal{N}\Big(0, \big(\frac{MAR \cdot d}{12f}\big)^2\Big). \tag{10.4}$$

The operator of adding slope is the same with adding accumulated noise based on equation (10.4), which models the drifting in motion estimation.

We assume the resolution of the video of the synthetic motion is 1080p, the viewing distance is 3240 and focal length is 830. The error of VE and SVE scores are shown in Fig. 10.4. From Fig. 10.4, we can see that our two measurements VE and SVE are both robust under different operators. There are three main observations. Firstly, the flipping operator does not influence the scores at all. Secondly, VE and SVE have similar performance when the motion amplitude is large. Meanwhile, they are more robust at high amplitude than at low amplitude. This is simply because when amplitude is large, the noises are relatively small. Intuitively, that is when videos become more shaky so that it is harder for viewers to distinguish the difference between similar motions. Thirdly, in the low amplitude region, SVE is more stable than VE under all operators except flipping. However, we cannot assert that SVE is better than VE. On the contrary, larger changes under noise implies that VE is a more sensitive measure, especially in the low amplitude region is where the FPF increases dramatically according to Fig. 9.3 (b). We prefer a measurement that has a higher sensitivity while it is robust enough. This is supported by the tests in the next section.

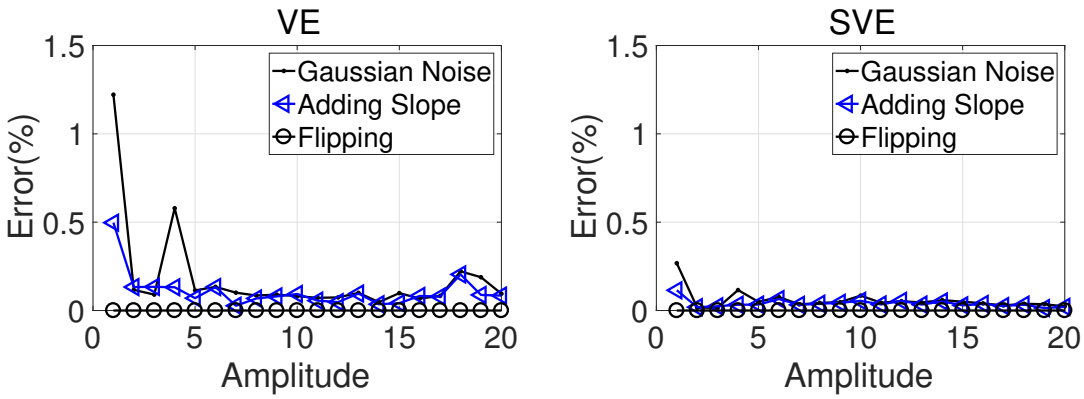


Fig. 10.4.: Error caused by synthetic motions

10.1.3 Measurement effectiveness

To evaluate the effectiveness of our measurements, we treat them as quality estimators of video stability and FPMI. We use videos and subjective test results from our previous work [1] to test if the results of our measurements align with the subjective scores. Since [1] uses a different method of video stabilization, this experiment is independent of our motion editing step and simply considers how effective the VE/SVE is to measure the viewing experience. Note that the videos from [1] are real videos, which makes the experiment result closer to the practical situation.

The resulting videos in [1] include three versions: the original ones, results of their algorithm and the results from Microsoft Hyperlapse [8, 46]. Their subjective scores are computed based on the Bradley-Terry model [47]. To test the effective of measurements in the sense of subjective measurements, we apply the model proposed in [48].

The general idea of this model is to examine the ordinal scale of the quality estimator scores. As illustrated in [48, 49], if the Bradley-Terry scores of three version of the videos have the relationship: $BT_1 < BT_2 < BT_3$, then the scores of a quality estimator that has a good ordinal scale should be: $QE_1 < QE_2 < QE_3$. Meanwhile, the distance between the subjective scores and the distance between the quality estimator scores should be similar. As a result, the model [48] yields the following conditions:

$$sign(BT_3 - BT_2) = sign(QE_3 - QE_2), \tag{10.5}$$

$$sign(BT_3 - BT_1) = sign(QE_3 - QE_1), \tag{10.6}$$

$$sign(BT_2 - BT_1) = sign(QE_2 - QE_1), \tag{10.7}$$

$$sign(BT_3 - 2BT_2 + BT_1) = sign(QE_3 - 2QE_2 + QE_1). \tag{10.8}$$

By calculating the number of conditions that are satisfied, we can use the fraction of correctness to evaluate the effectiveness of our measurements. The results are shown in Table. 10.2. We can see that for the first three conditions, VE and SVE

Table 10.2.: Fraction of correct distance conditions

	VE				
Conditions	Cdn 1	Cdn 2	Cdn 3	Cdn 4	Total
Stability	1	1	1	0.8	0.95
FPMI	0.8	1	1	1	0.95
	SVE				
Conditions	Cdn 1	Cdn 2	Cdn 3	Cdn 4	Total
Stability	1	1	1	0.6	0.9
FPMI	0.8	1	1	0	0.7

have the same performance. It shows that they are both robust for the simple ranking tasks of stability or FPMI. However, for the condition 4, the performance of VE is much better than SVE, which supports the demonstration in the objective tests. VE is a more sensitive measure, which ensures the distance similarity between subjective scores and the quality estimator scores. This conclusion is more solid for measuring the FPMI, since VE is more sensitive when the motion amplitude is small, where FPF (part of FPMI) varies dramatically.

10.2 Tests for overall system

To evaluate our overall system, we first apply an objective test based to compare our stabilization function of our system with existing stabilization techniques. After that, we construct a subjective test to show our system can effectively improve the stability of FPVs while preserving their FPMI. According to the experimental results shown in previous section, we adopt VE as the measurement used in the tests of this section.

10.2.1 Stabilization performance

Our stabilization performance test is based on 5 video sets, which are recorded in 5 different scenes (available at [50]). Each set includes 6 different versions of the same video: an original video, an output of our system, an output result from Microsoft

Table 10.3.: ITF scores of different video versions

	Orig	Ours	HL	DS	[5]	[6]
Yard	29.2	33.7 (0.6%)	33.7	33.7 (17.2%)	32.9	**34.9** (37.0%)
Cave	29.3	33.6 (0.03%)	33.8	33.8 (7.3%)	33.5	**34.1** (7.8%)
Beach	26.6	30.8 (0.74%)	**30.9**	**30.9** (15.5%)	30.5	30.8 (10.0%)
Climb1	28.0	32.9 (0.5%)	32.6	32.6 (19.0%)	32.3	**33.2** (22.6%)
Climb2	28.4	33.4 (0.76%)	**33.6**	**33.6** (13.0%)	32.3	33.0 (13.4%)
Average	28.3	32.8 (0.5%)	32.9	32.9 (14.4%)	32.3	**33.2** (18.2%)

Table 10.4.: Revised ITF scores

	Orig	Ours	HL	DS	[5]	[6]
Yard	29.2	**33.7**	**33.7**	33.0	32.9	32.3
Cave	29.3	33.6	**33.8**	33.4	33.5	33.7
Beach	26.6	30.8	**30.9**	30.4	30.5	30.3
Climb1	28.0	**32.9**	32.6	32.2	32.3	32.0
Climb2	28.4	33.4	**33.6**	32.5	32.3	32.3
Average	28.3	32.8	**32.9**	32.3	32.3	32.1

Hyperlapse (HL) [46], an output from Deshaker (DS) [51], an output from Youtube stabilizer [5] and an output from [6]. The original videos are 10 seconds and recorded by a GoPro Hero Session 4 with 1080p. To minimize the black area of all results, each output is cropped to 1280×720.

The objective measurement of video stability is based on inter-frame transformation fidelity (ITF) [20]. The test results are shown in Table 10.3 where a larger value indicates higher video stability. According to Table 10.3, all stabilization methods successfully stabilize the videos. Although our method does not have the highest value among all methods, the difference between all 5 methods are significantly smaller than the improvements.

As we mentioned in Section 4.2, ITF scores cannot predict subjective stability scores well. One of the reason is that ITF only measures the ability of a stabilization method to smooth the camera motion; and, ITF cannot measure the amount of black pixels at the edges of the image. Table 10.3 shows, in parentheses, the percentage

of black area for each method. Hyperlapse and Youtube stabilizer do not have black area while [51] and [6] have significant amount of black area. Note that when we obtain the results from [6, 51], we disable the option to remove the black area. When this option is enabled, it either scales the frames or introduces significant stitching errors. However, this black area issue is not reflected by the ITF scores, which will significantly degrade the stability in practical situation. For example, [6] has the highest ITF score while its resulting videos are obviously less stable than our method, which can be verified in the database [50].

As a result, we modified the ITF by taking the black area issue into account. The revised ITF is calculated as:

$$ITF = \frac{1}{N-1}\sum_{k=1}^{N-1} PSNR(k), \tag{10.9}$$

where N is the number of frames, and:

$$PSNR(k) = 10\log_{10}\Big(\frac{255^2}{MSE(k)}\Big). \tag{10.10}$$

To deal with the black area issue, we compute the mean square error (MSE) based on the average of non-black area S of adjacent frames:

$$MSE(k) = \frac{1}{S}\sum_{i}\sum_{j}\Big(I_k(i,j) - I_{k-1}(i,j))\Big)^2. \tag{10.11}$$

The revised ITF scores are shown in Table 10.4, where Hyperlapse has the highest score and our method is in the second place, which shows that our system can effectively stabilize FPVs.

10.2.2 Overall performance

To further subjectively evaluate the performance of our system on improving stability while preserving FPMI, we construct and discuss another subjective test in this section.

This subjective test uses the same source videos with the previous section (section 10.2.1), which includes 5 different scenes shot by GoPro Hero Session 4 with 1080p. Three versions are prepared for each video: the original one, our resulting video and Microsoft's result. All the test parameters are the same: videos are played back on a 27-inch, 82 PPI screen and the ratio of viewing distance with respect to the equivalent focal length is 4.

Our test uses paired comparison, which includes 21 subjects. Subjects are asked the following questions after shown each comparison: (1) **Which video is more stable**; (2) **In which video you can recognize more First-Person motion**; (3) **If your friend tries to share his/her First-Person experience with you, which one do you prefer**. The subjective scores (with 95% confidence interval) calculated using the Bradley-Terry model [47] are shown in Fig. 10.5, which also includes the result from [1]. A higher score indicates higher stability, more FPMI or higher preference.

As expected, the original videos have the highest score for FPMI and Microsoft's videos have the highest stability. Our system is in the second place for these two scores. Meanwhile, the scores of our videos are closer to the desired videos: our FPMI score is closer to the original videos' while our stability score is closer to the Microsoft's. This illustrates that our system can effectively increase the stability while preserving an adequate amount of FPMI. [1] has a similar FPMI score to ours. However, their stability score is lower than ours. Moreover, the preference score of our videos is the highest. As a result, we can conclude that our system is more effective than the one in [1]. This is because our system is based on a more precise

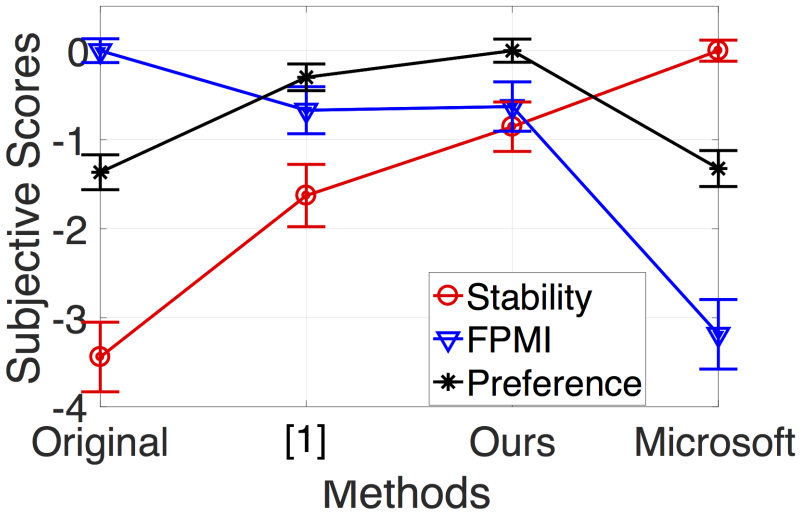


Fig. 10.5.: Result comparison of different version videos

human perception model and has carefully designed measurements for the viewing experience.

11. CONCLUSION

In this work, we proposed two measurements (VE and SVE) that can quantify both the stability and the First-Person feeling of a FPV. Based on the measurements, we further proposed a system that can improve the viewing experience of FPVs. To accomplish these two items, we described the human perception model, analyzed the perceptual geometry of watching FPVs and defined the First-Person motion and its properties. We also apply a subjective test to further improve the basic perception model. Finally, we systematically evaluate our measurement and overall system using objective tests and subjective tests.

The objective test of robustness show that our measurements are robust under different operators that can create visually equivalent motions. The objective test of stabilization performance shows that our system is as effective as existing works. The subjective tests show that both measures of VE or SVE highly align with the subjective scores. Also, our system can effectively increase the stability of FPVs while preserving an adequate amount of FPMI.

Our system still has places can be improved facing the practical situation. Our current pipeline is based on the 3D video stabilization procedure, which requires the estimation of camera 3D motions. In the practical situation, the problem is that the speed and accuracy cannot satisfy the real-time processing at the same time. One solution would be enhance the 3D motion estimation algorithm based on a recent work [52]. The other solution would be implementing the whole pipeline according the 2D video stabilization procedure, which leads to many advantages. For example, in Section 10.1.1, we can see that the fitting model of VE scores and subjective scores depends on scenes content. Under the 2D solution, instead, we can obtain an unique model which enables us to build a unified criteria of video stability and the video content is no longer restricted as FPVs. Also, this also provide us convenience of

removing the rolling shutter distortion as suggested by [6]. We will implement both solutions in the future work and analyze their performance. In addition, we plan to test our system on First-Person Sports Videos, which requires a more robust system.

Part III

Dashcam Video Compression

12. INTRODUCTION

As mobile devices are developing rapidly, more and more mobile videos are captured to record daily life. One popular kind of mobile videos is the dash camera video (DCV), which is captured by cameras mounted on the vehicle dashboard or built-in cameras of vehicles. DCVs are used to record driving history and provide effective evidence about traffic accidents. Of these DCVs, the ones that do not contain any abnormal situations are usually deleted or overwritten. However, with the development of intelligent vehicles [53–56], discarding these historical data seems inadvisable. These works use dashcam videos (DCVs) as training samples to detect vehicles or model vehicle behavior. However, to model complex traffic and vehicle behavior, a substantial amount of DCVs is necessary; these can be collected by vehicle-video-sharing [57] or vehicle-to-broadband cloud (V2B) platforms [58]. For these platforms, an effective DCV compression approach is desired. Our aim is to find a storage mechanism for DCVs, which systematically collect them and efficiently compress them.As a result

A common idea of efficiently storing data is jointly compressing the data that are clustered. Clustering means to collect a group of objects that have similar properties. Inside a cluster, the relationship between objects is measured by defined distance. The more the objects are close to each other the higher correlation they have. As a result, jointly compression which exploits the redundancy within clusters can achieve a better compression result. Meanwhile, clustering also gets the objects sorted well. Multi-view extension of HEVC (MV-HEVC) [59] is a recent example that has been designed for efficiently encoding multi-view videos, which are recorded simultaneously by different cameras. As a result, the video pairs are significantly correlated in spatial and can be thought as naturally clustered. Being jointly compressed, their bit-rate can be greatly reduced. This idea is available for DCVs.

DCVs are possible to be clustered well. When a vehicle travels the same route on two occasions, the two DCVs contain both static objects and new or moving objects. Because vehicles may have the same location in different DCVs, the static objects may be similar, which makes the corresponding frames highly correlated. As a result, instead of compressing these videos separately, it is reasonable to consider compressing correlated videos jointly.

However, DCVs are not multi-view videos (naturally clustered), so it is difficult to use MV-HEVC directly. In some cases, it is hard to find redundancy between two DCVs. Unlike in multi-view videos, for instance, static objects in DCVs may be recorded under different illumination and the relative viewpoints on different days may not be fixed, which are challenging situation for inter-view prediction algorithms. And in some other cases, the redundancy is not explicit. Users may drive at a different speed on different days, so the corresponding frames will not match temporally. There is a set of best reference frames for the video to be encoded, but the current standard coding tool does not have any way to find it. So our goal is to take advantage of the existing framework of MV-HEVC/3D-HEVC tools as much as possible, by applying several preprocessors to identify the DCVs that have correlations and create a specific sequence that matches camera locations, orientations and illumination conditions across time for the DCV to be encoded. Specifically, two problems need solving: clustering DCVs and finding their temporal relationships.

13. RELATED WORKS

There is little research about DCVs considering the two tasks: clustering and finding temporal relationships. However, some approaches proposed for usual videos can provide some valuable ideas.

13.1 Image set and Near-duplicated videos

Some related approaches focus on jointly compressing clustered images and videos. These approaches are first applied on images. Zou et al. [60] proposed a system to compress personal photos that have similar occasions. The sum of square difference (SSD) computed during motion estimation is used to measure the similarity between photos. Based on the similarity, photos are stored in a tree structure and compressed using HEVC. In [61], a system using the same idea was proposed. However, local features are used to measure the similarity instead of pixel values. Before using the block-based motion compensation to compress the photos, geometric deformations and photometric transformations are used to exploit the redundancy between photos.

Some of these approaches focus on near-duplicated videos (NDVs). NDVs [62] are generated by replicating or editing a video in a variety of ways, such as changing frame order, using different compression parameters or modifying image scales. Most works of NDVs citecheung2005fast,shang2010real,shen2007uqlips,chou2015pattern,chou2014near are concentrated on key-frame-based video retrievals. Given the results, not only is a matched video found but also a general timeline alignment can be created. This is because the changes between key-frames are based on temporal resampling, such as equally-spaced sampling (e.g. halving or doubling the frame rate). Part of this research applies global features [63–65] to achieve fast retrieval while preserving the

ability to generalize. Another part uses the local features [66, 67] and applies the Bag-of-Words (BoW) technique or similar visual vocabulary approach to reduce the computational cost. Inspired by the photo-oriented systems [60, 61], Wang et al. [68] proposed a compression system for NDVs. Although similar NDVs can be found, some preprocessing is required before applying the video coding tools for compression. Wang et al. first use an analyzer to retrieve the NDVs, and then apply the Bag-of-feature to search the reference frames for each I frame. After that, a light regulator is proposed to remove the effect of different lightness across different videos. Also a homographic transform is applied to solve the resizing problem. Experiments show that their system can effectively compress the NDVs.

However, the aforementioned clustering methods and preprocessors are not sufficient and appropriate for DCVs. First, the photo-oriented methods only solve the clustering problem, which makes them do not scale well to video processing. To cluster similar DCVs, it would be inefficient to directly apply them to each of the frame in DCVs. Normally, a DCV may have over 10 minutes content, which means there are more than 20,000 frames to process. Approaches used for NDVs solve both problems but do not perform well on DCVs. They take the advantage of the consistent manipulation applied to create NDVs, whereas variation between DCVs are not consistent. The key-frame-based video retrieval techniques used for NDVs are developed from the photo-oriented methods. Instead of processing all the frames, these approaches only generate features based on a set of key-frames. Unlike NDVs, DCVs are recorded independently. The contents of DCVs vary every day due to the changing vehicle speed, weather conditions and driving routes. As a result, the correspondence between two DCVs cannot be modeled using regular temporal resampling. Thus, key-frame-based algorithms cannot provide an accurate timeline alignment. In addition, DCVs are also difficult to be distinguished temporally since there are many on-road scenes that are similar and do not have distinguishing features. So global features or quantized local features cannot provide a confident video retrieval. Since most dash cameras

have an associated GPS module, a much easier way to retrieve DCVs is to use the GPS information recorded along with the videos.

The preprocessing algorithms mentioned above should also be reconsidered. Methods such as light regulator [68] are useful for stable illumination changes. A good example is the manually changed lightness in NDVs. In DCVs, illumination conditions are influenced by several factors and may vary locally within one frame. For instance, the leaves of road-side trees may create irregular illumination changes within some frames when vehicles drive along the road. The light regulator does not perform well in this kind of scenario. Also, the homographic transformation does not always provide benefits for compressing DCVs. Although the transformation can rectify the corresponding frames in different view angles, when the difference between viewpoints is large, as is frequently the case in DCVs, the transformation may actually decrease the image quality.

13.2 Video synchronization

Video synchronization methods provide some idea to the problem of finding temporal relationships. Instead of using local features, some algorithms for video alignment or video synchronization provide different ideas for timeline alignment. [69, 70] are based on image intensity and use heavily downsampled images as the descriptors to save computations. This enables the application of global optimization algorithms. In [69], a dynamic Bayesian network is used to find an optimal correspondence relationship between frames. [70] tries to find the relationship by minimizing the cost function based on image difference. Note that [69,70] are based on a strong condition that the relative viewing angle between different videos cannot be too large and the vehicle must maintain the same lane. When this is not satisfied, experimental results demonstrate our system has significantly improved compression efficiency relative to the matching algorithm in [70]. Further, we have a different goal. [69] uses an interval containing 3 to 6 frames to measure the goodness of each matching result. If the

matching result falls into the interval, it is considered a true correspondence. However, we are interested in finding an exact frame correspondence that gives the highest compression efficiency.

In this work, we proposed a system, which enables easy categorization of incoming DCVs to allow effective compression and database construction. Two preprocessors are developed: video retrieval and video alignment. In the video retrieval preprocessor, GPS information is used to query the reference videos for the new video. In the video alignment preprocessor, a point-wise frame matching algorithm based on ORB (Oriented FAST and Rotated BRIEF) features is used to find the best reference frame for each current frame and use them to construct a new reference video. Experiments showed that our previous system performs well on compressing DCVs. In the next Chapter, our system structure is demonstrated. In Chapter 15, along with the data structure, two preprocessors and more algorithm details are presented. Performance of our system and comparison with others system are discussed in Chapter 16. Finally, we will conclude our work in Chapter 23. Note that this work is based on our previous research [71].

14. SYSTEM DESCRIPTION

To efficiently store DCVs, we propose a system based on clustering and compressing. The general idea is illustrated in this Chapter. We first introduce the structure used for classifying and clustering DCVs. Then, we show our video alignment preprocessor, which further exploit the redundancy between clustered DCVs. In addition, we also discuss the encoding and decoding structure of our system.

14.1 General framework

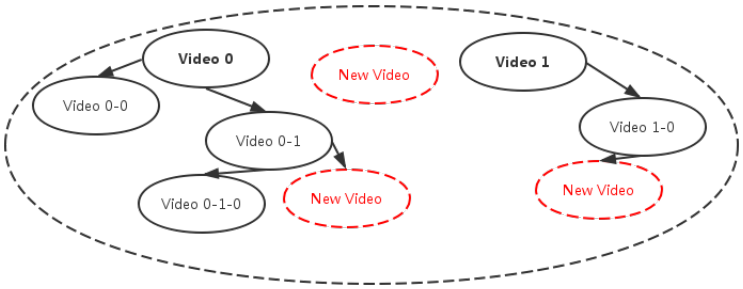


Fig. 14.1.: The cluster for a specific route

First, DCVs are classified according to the GPS information. The classification structure is based on several paths that are pieces of driving routes. Every path has its own standard GPS feature, which means there is no overlap between two different paths. A single DCV is divided into video sequences according to these paths. Different sequences are assigned to different classes by comparing their GPS feature to the standard GPS feature of paths, which is illustrated in Chapter 15 in detail.

In each path class, video sequences that record the same path are clustered depending on the illumination condition. When a new video sequence comes in, our illumination matching algorithm is applied to find the video sequence which has the most similar illumination condition with it. As the database developing, video sequences in the same class are clustered. Fig 14.1 shows the general idea of our clustering structure.

Based on the clustering result, video sequences can be efficiently compressed. In Fig 14.1, video 0 and video 1 are root nodes of the clustering structure, which are compressed by standard HEVC. If the algorithm cannot find a similar video for the new video, the new video will be set as the root node. Otherwise, the new video will be set as a child node for an existing node and jointly compressed based on its parent node.

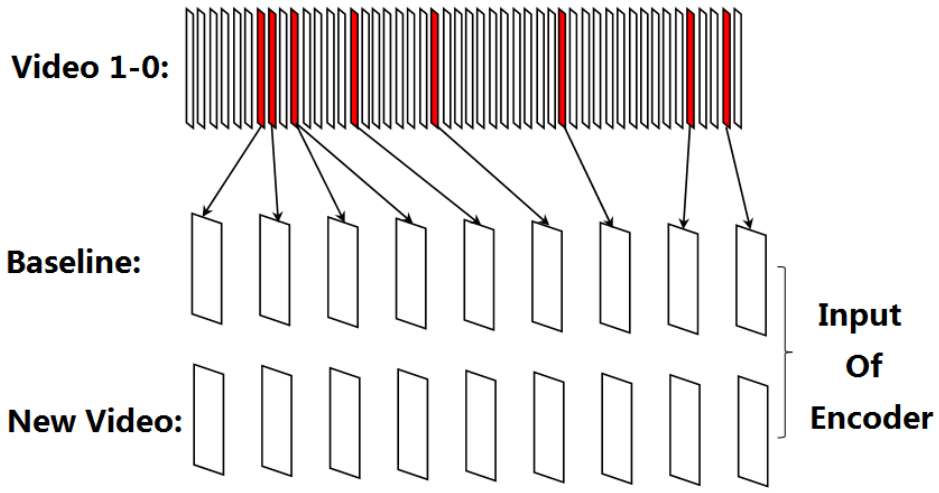


Fig. 14.2.: The general idea of video alignment

Before jointly compressing the new video, an artificial reference video (called the"baseline") needs to be generated. This is finished by the video alignment. For instance, suppose the matching video for the new video is video 1-0. The general idea of the video alignment is shown in Fig 14.2. Since the vehicle speed is different in the new video and video 1-0, the frames that can effectively compress the frames of the new video are irregularly distributed in video 1-0. The task of the video alignment

is to search and assemble these reference frames to construct the baseline. As a result, the baseline matches the camera locations of the new video across time and can provides a good inter-view prediction.

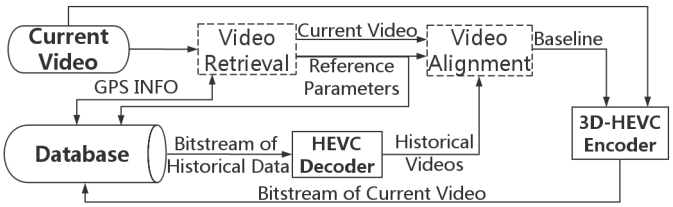


Fig. 14.3.: The cluster-compression system of DCVs

Above discussed GPS classifier, illumination cluster and video alignment comprise the main part of our whole system which is shown in Fig 21.2. The video retrieval module includes the GPS classifier and illumination cluster structure. It is responsible for identifying the new video and retrieving the video sequences that can provide as much information as possible for compressing the new video. The video alignment module is then used to store the reference video parameters into the database and decode the corresponding historical video. Offered the indicating parameters and the new video, it applies a frame-wise matching algorithm to find the specific reference for each frame in the new video. After storing the reference frame parameter for each frame, a baseline is constructed and used to compress the new video.

14.2 Encoding and decoding structures

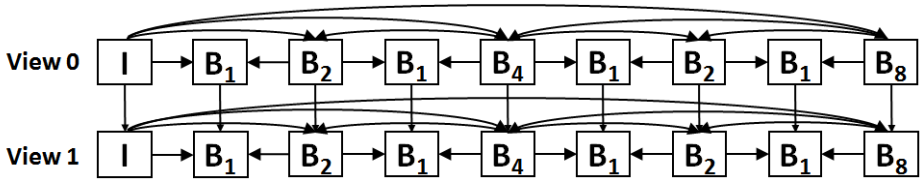


Fig. 14.4.: Multiview prediction structure of HEVC [72]

The 3D-HEVC encoder adopts the prediction structure shown in Fig 14.4. This structure assumes the videos in both views share a common timeline, and enables predictions between frames that are at the same time instances. In our case, the baseline is set as the view 0 which provides information for view 1. The current video is encoded as view 1 using inter or intra predictions and interview predictions. Since the baseline is specifically constructed for the current video, the bit-rate of the current video can be significantly reduced. While MV-HEVC also has the same prediction structure, we apply the 3D-HEVC in our system so that we can take advantage of its illumination compensation algorithm.

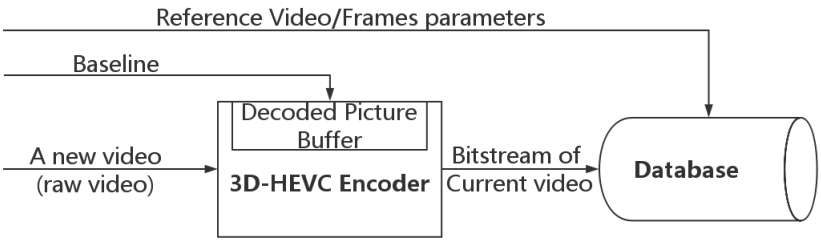


Fig. 14.5.: The encoding process

Fig 14.5 shows the encoding process of our system. The baseline should not be recompressed during this stage, because it was already compressed once. It can be reconstructed at any time using the historical data which is already stored in the database. To encode the new video, the baseline should be used by the decoded picture buffer (DPB) directly. Thus, only the bitstream of the new video and its compression parameters need to be stored. Meanwhile, no motion information is generated for the baseline. Consequently, two coding tools of 3D-HEVC which both rely on the motion information of the baseline, the inter-view motion prediction and interview residual prediction, must be disabled in the encoder.

Fig 14.6 shows the process of decoding DCVs. To decode the new video that was encoded in Fig 14.5, it decodes the historical data including the baseline frames and apply the frame reference parameters to construct the baseline. Since we never

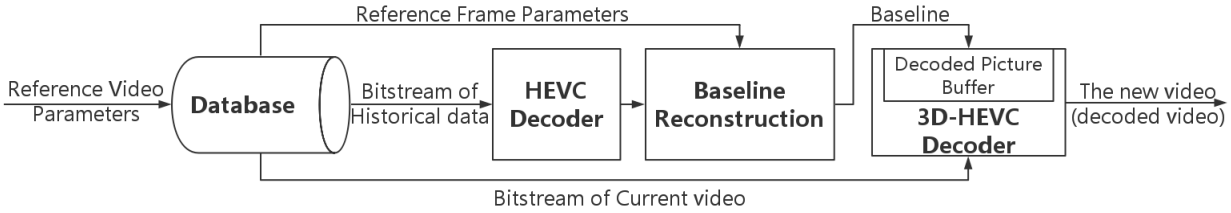


Fig. 14.6.: The decoding process

construct the bitstream of the baseline, the 3D-HEVC decoder has to use the reconstructed baseline directly as the reference.

15. PROPOSED DASHCAM VIDEO COMPRESSION METHOD

In this work, we improve the two preprocessors: video retrieval and video alignment. In this Chapter, we first illustrate the video retrieval preprocessor, which includes a GPS-based video classification algorithm, an illumination-feature-based video clustering algorithm and an adaptive switch algorithm (decides the coding mode of each frame). Then we show the video alignment preprocessor. Based on the experiments, the homographic transformation is disabled to achieve better compression performance. Also, new searching strategies that can boost the point-wise frame matching algorithm are demonstrated.

15.1 Video retrieval

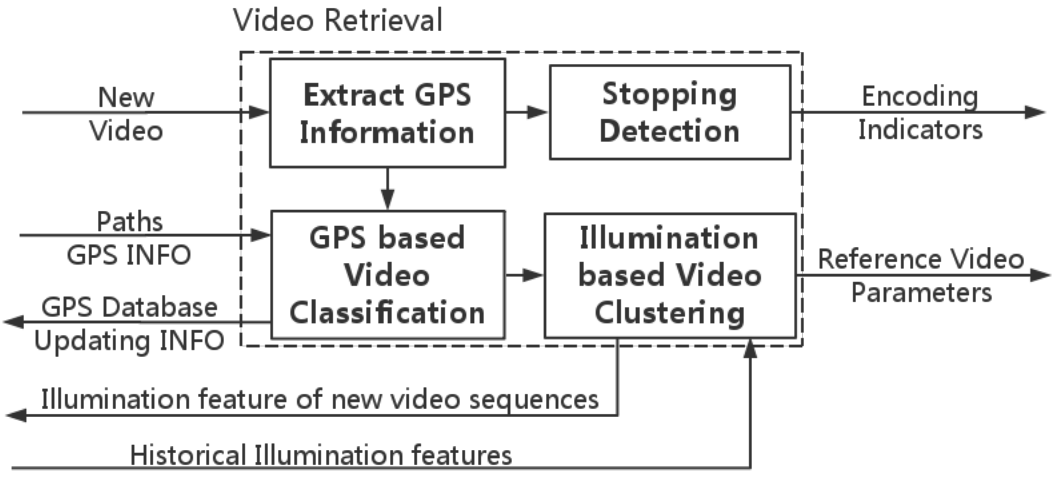


Fig. 15.1.: The video retrieval preprocessor

Fig 15.1 shows the structure of the video retrieval preprocessor. When a new video comes into the video retrieval preprocessor, its GPS information is first exacted and used to find the matched geometric paths in the current database. According to matching result, the new video is then divided into sequences are respectively classified and encoded. Meanwhile, the current GPS database is updated.

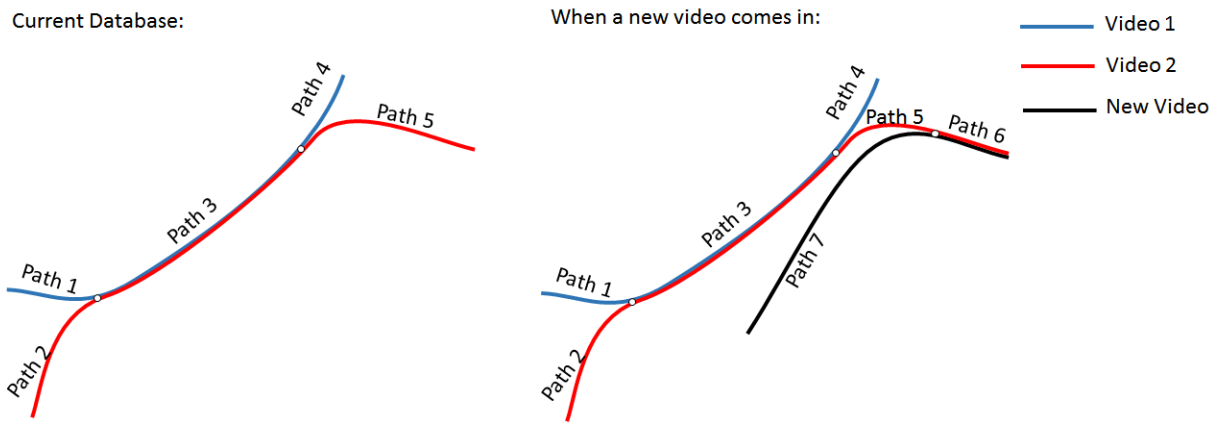


Fig. 15.2.: Path matching and video dividing

Fig 15.2 shows a example. The current database includes 5 mutually independent paths. Based on these paths, video 1 and 2 are divided into 3 sequences respectively. Note that, their second sequences have the same geometric location, which makes themselves classified into the same path class. When a new vide comes in, the database is updated. Path 7 is added as a new class and the original path 5 is divided into two separated paths. To clarify the operations here, we first introduce our data structure.

For each new video, the database first assign it a video index V_c. Its GPS information is represented as $G_c = (G_c^1, G_c^2, \cdots, G_c^n, \cdots, G_c^N)$. And its deduced reckoning (DR) information is stored as $DR_c = (DR_c^1, DR_c^2, \cdots, DR_c^n, \cdots, DR_c^N)$. Note that, G_{pi}^n is a GPS coordinate (x_c^n, y_c^n). For the i^{th} geometric path, the database includes its GPS information $L_{pi} = (L_{pi}^1, L_{pi}^2, \cdots, L_{pi}^n, \cdots, L_{pi}^M)$. The database also has the index set of its class members $\{V_{pi}\}$. For instance, in Fig 15.2, the index set of path 3 is $\{V_1, V_2\}$.

To find the match part of the new video and the geometric path i, a indicator sequence is computed as follow.

$$I_i(n) = \mathbb{1}_{\{x:\ x \leqslant T+e\}} \left[\min_{S(n-1) \leqslant m \leqslant M} \|(x_c^n - x_{pi}^m, y_c^n - y_{pi}^m)\| \right]$$

$S(n)$ is the closest point of n^{th} second of the new video in the geometric path i.

$$S(n) = \underset{S(n-1) \leqslant m \leqslant M}{\arg\min} \|(x_c^n - x_{pi}^m, y_c^n - y_{pi}^m)\|$$

$T + e$ is the threshold that indicates whether the two location are identical while T is the geometric constraint and e is the GPS error.

$$T = \frac{\|(x_{pi}^n - x_{pi}^m, y_{pi}^n - y_{pi}^m)\|}{2}$$

$$e = \frac{Accuracy\ of\ GPS(km)}{Distance\ per\ degree(km/degree)} \approx 1.35 \times 10^{-4}$$

After this matching algorithm has been run for all the geometric paths in the current database, the new video is divided into sequences which belong to different path classes. Note that, instead of storing the GPS information of the new video, we store the closest point of each second of matched sequences. At the same time, the GPS information of the isolated part (non-matched part) becomes the standard GPS information of the new path class. So the parameter set of a classified video includes two part: path index set and GPS matching result. The path index set indicates the component of the video. For example, the path index set of video 1 in Fig 15.2 is $\{P2, P3, P5\}$. The GPS matching result includes the closest point of each second. For example, in Fig 15.2, the GPS matching result of video 1 is $\{\{0, length\}, \{S_{P3}(n)\}, \{0, length\}\}$. Here, we use "0" to indicate that these parts of video are isolated while it is also necessary to record the corresponding length.

Table 15.1.: Changing mechanism of parameter set

	Paths	**Video Index Set**	**GPS Info**
Before New Video Inputed	Path 5	{V2}	$\{L_{P5}^1, L_{P5}^2, \cdots, L_{P5}^M\}$
After New Video Inputed	Path 5	{V2}	$\{L_{P5}^1, L_{P5}^2, \cdots, L_{P5}^D\}$
	Path 6	{V2; New Video}	$\{L_{P5}^{D+1}, L_{P5}^{D+2}, \cdots, L_{P5}^M\}$
	Videos	**Path Index Set**	**GPS Matching Result**
Before New Video Inputed	Video 2	{P2,P3,P5}	$\{\{0, length\}, \{S_{P3}(n)\}, \{0, length\}\}$
After New Video Inputed	Video 2	{P2,P3,P5,P6}	$\{\{0, length\}, \{S_{P3}(n)\}, \{0, length\}, \{0, length\}\}$
	The New Video	{P7,P6}	$\{\{0, length\}, \{S_{P3}(n)\}\}$

As discussed before, the classification result of the new video may also update the database. Use Fig 15.2 as a example, Table 15.1 shows the changing of the parameter sets.

After the new video is classified, each path class operates the illumination clustering algorithm to select a reference video sequence that has the most similar illumination condition to the current video sequence. At this step, three situations are included. First, new path classes which only have one video sequence (isolated sequence) do not operate the illumination clustering algorithm. Meanwhile, the isolated video sequences are decided to be encoded using standard HEVC. Second, video sequences that cannot be assigned to the current illumination clustering structures are also decided to be encoded using standard HEVC. Third, the rest of the video sequences which are able to find the corresponding clusters are encoded using 3D-HEVC. As a result, the encoding modes of every video sequence of the new video are decided. Reference video parameters including the information of encoding mode and reference video indexes are transfer to the video alignment preprocessor. The illumination clustering algorithm is illustrated in the section 15.3 in detail.

Along with the GPS information, the DR information which includes bearing and speed of the vehicle is also recorded by the dashcam. This information is then employed by our stopping detection algorithm (an adaptive switch algorithm) which decides whether our video alignment algorithm should be applied on the current video sequence. For instance, if the vehicle is stopped or moving slowly, the best reference frame is surely in the current video instead of in any prior video. To be computationally efficient, our system will not use the proposed algorithms in these

situations. However, if the bearing of the vehicle changes significantly, the recorded scene will change a lot even if the vehicle speed is low. Our proposed should still be used under this situation. Finally, the stopping detecting model generates a sequence of indicators which implies the time we switch off the video alignment algorithms.

Note that, the encoding mode of each video sequence is fixed and only determined by the illumination clustering result. A video sequence decided to be jointly encoded may include some "stopping frames". Although these frames do not apply our algorithms to find reference frames, they are still encoded using 3D-HEVC by being assigned a default picture as their reference frames.

15.2 Video alignment

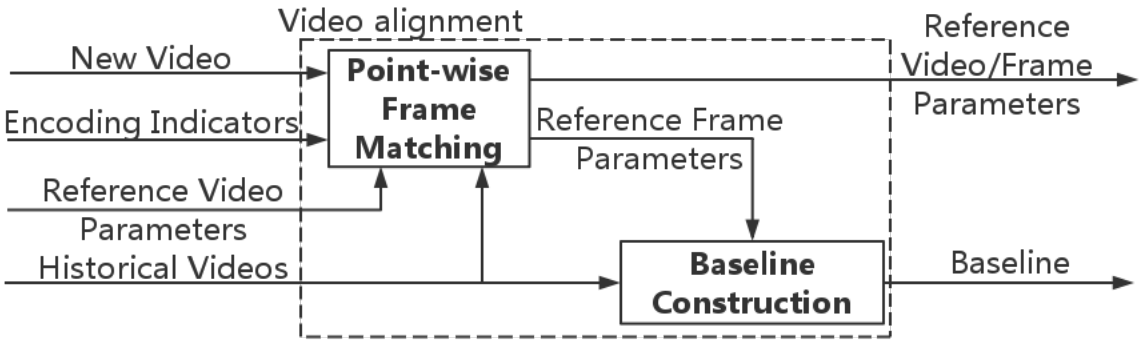


Fig. 15.3.: The video alignment preprocessor

The video alignment preprocessor simultaneously finishes two tasks which are the timeline alignment and image alignment. For each video sequence, given the encoding indicators and the reference video parameters, the video alignment preprocessor uses an ORB-based point-wise frame matching algorithm to complete a precise timeline alignment. The timeline alignment guarantees every frame indicated to apply the matching algorithm in the new video is matched with a reference frame (chosen from the reference sequence) that is the closest to it. Based on the timeline alignment

result, a rough image alignment is automatically completed because the same location ensures the objects in corresponding frames look similar. In previous work [REF], we used homographic transforms to further geometrically align the corresponding frames. Later we will show that this is not necessary. Last, according to the matching result, the final baseline is constructed using the decoded historical data.

For convenient, in this section, we assume all of the frames in the new video are applied our algorithm.

15.2.1 Timeline alignment

GPS can provide location information for DCVs. However, its 1 Hz signal frequency restricts its accuracy. DCVs have 30 frames per second so that frames' relationship within 1 second cannot be accurately estimated. Instead, the local visual feature is used to construct the matching sequence that aligns the current video temporally.

Our goal is to find a reference frame that gives the highest compression rate for the current frame. To achieve this, the objects' scale in the reference frame should be similar to the ones in the current frame. We use the number of matching points between two frames to measure their scale similarity. In the GPS and illumination matched video, the frame that has the most matching points is selected to be the reference frame for a current frame. This is because when similar scale is achieved, the resolution is similar. And key-points are matched only when the resolution is similar. The more area has similar scale, the more matching points the image pair has.

However, not all local features are equally effective. Large-patch-based local features such as ORB have higher accuracy. Scale-invariant features have lower accuracy, such as SIFT [73]. Although scale-invariance increases features' robustness, it also increases the number of matching points that have dissimilar scale. Thus, we select the ORB feature since it has a larger patch-size and is not completely scale-invariant [74].

We use RANSAC [38] to remove unstable matches. Note that the initial set of matching points may include many unstable matches. To obtain a more precise result, the RANSAC algorithm [38] is used to remove those outliers.

Since there are many frames in the new video, what we desire is a feature that can be computed quickly for extraction and matching. To further speed up the algorithm, the frames are down sampled. Consequently, the feature should also be robust to down sampling, which means the matching results using this feature need to have more matching points after RANSAC. Based on the matching experiments of most of the local features, we choose a feature by quantifying the "matching number after RANSAC per second". Our results demonstrate that the ORB [74] feature outperforms other features.

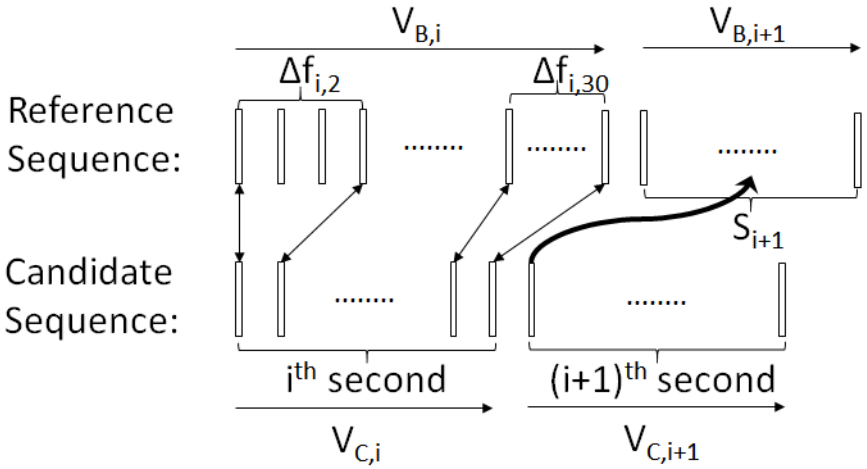


Fig. 15.4.: The searching strategies

To search efficiently, two strategies are also developed. The first one is to search only in one direction. For instance, in Fig 7, suppose the best match for the first frame in the i^{th} second of the current video is the first frame in the reference video sequence. Since the vehicle is always moving forward, the matching frame for the next frame in the current video must be after the previous match result temporally within the reference video sequence.

The second strategy is to estimate a small search range before starting. The matching frame can be found within a small range S, since the reference video video sequence has been roughly matched to the current video using GPS information. After each iteration, the matching distance Δf is calculated, which is the difference between the frame number of the current matching result and the previous one. It is directly proportional to the speed ratio between the reference and current video. Thus to estimate this S, the average Δf of the previous second and the velocity information is used:

$$S_{i+1} = \lceil \hat{d}_{i+1} \rceil = \lceil \frac{V_{B,i+1}/V_{C,i+1}}{V_{B,i}/V_{C,i}} \cdot \overline{d}_i \rceil = \lceil \frac{1}{29} \frac{V_{B,i+1}/V_{C,i+1}}{V_{B,i}/V_{C,i}} \sum_{n=2}^{30} \Delta f_{i,n} \rceil$$

In [70], a set of algorithms called SeqSLAM was proposed to solve a similar problem. It was shown to have good performance even when the data is captured in different weather or seasons. In SeqSLAM, to match the frames in two video sequences, instead of using local features, all frames are down-sampled and frames in one sequence are subtracted from frames in the other one. The frame pairs that have small differences are considered to be matching pairs. Finally, they find the trajectory that has the minimum sum of subtraction values.

However, this set of algorithms does not perform well for our database. This is because their system is based on two assumptions which our database does not satisfy. The first assumption is that the velocities of the same path in different videos are repeatable. Otherwise, the trajectory in the difference matrix will be a curve and cannot fit their straight line model. The second assumption is that the viewpoints of the videos must be close. To have a more precise matching result, it requires the contrast enhancement algorithm to maximize the difference between the subtracted values. Consequently, they try to achieve a similar field of view for the videos by using a single rectangle to crop the frames. However, this approach cannot work well when the vehicle may operate on different lanes or have a different viewing angle. The true matching frames will still have large differences although they are

cropped. Meanwhile, our searching algorithm is robust to varying speed (and indeed, the varying speed even helps us to search.). The local feature that we adopt is also robust for different viewpoints.

15.2.2 Image alignment

To reduce the bit-rate of compressing one NDV using another, [61,68] align matching frames at the pixel level using one or more homographic transformations. The main idea is to use a 3 by 3 homographic transformation to align the matching frame to a frame that was captured with a different viewpoint. As a result, by using multi-view video compression, the motion vectors and displaced frame difference (DFD) can be reduced. However, this is not necessary and sometimes it decrease the final performance. Our performance results shown in Chapter 16 demonstrate that this approach usually decreases the performance of our system by about 4%.

In the bitstream of a compressed video, motion vectors have much less bits than the DFD. So we only need to consider how to reduce the DFD. One of reasons that the MV or 3D HEVC encoder has better compression performance than normal HEVC is that the reference block and the current block have the similar scale, which reduces the DFD. As a result, we only need to ensure that similar scale content can be found as much as possible, which is ensured by the timeline alignment. Some content does not have exact the same scale, but has more similar scale than when using normal HEVC. In HEVC, prediction units are allowed to be divided into smaller blocks. These smaller blocks can deal with the content that does not have same but similar scale.

Moreover, the homographic transformation can be thought as a non-linear filter. Some parts of the original image are downsampled and some parts are upsampled. The accuracy of the transformation mainly depends on the number and distribution of the matching points. As long as the number is not enough or matching points are

concentrated at a local area, the calculated transformation will destroy the image. On the other hand, the timeline alignment is robust for these problems.

There is another approach for compressing two NDVs considering align frames in pixel level, which is view synthesis [75]. This technique uses the camera parameters and the depth map of the videos to synthesize the "aligned" frames for the current frame. However, this approach is not practical for our case. This method is mainly proposed for 3D videos which already have the depth maps and fixed relative position between the pair of cameras, neither of which are available in our system.

15.3 Illumination Clustering

In Chapter 14, we point out that our system includes three main components: GPS classifier, illumination cluster and video alignment. The first and third components are already illustrated in section 15.1 and 15.2. These two components can classify the DCVs according to the GPS information and align the relevant DCVs to make them become quasi-multiview videos which can then be efficiently encoded by 3D-HEVC. Nevertheless, to reach a higher efficiency, the illumination clustering is necessary, which is introduced in this section.

Illumination clustering includes two parts: illumination compensation and illumination matching. If the illumination conditions of different videos are similar enough, the illumination compensation model in 3D-HEVC is sufficient to complete the clustering task. However, in most of the case of DCVs, illumination conditions for two random DCVs are quite different. Our illumination matching algorithm is necessary to leverage the performance of illumination compensation.

15.3.1 Illumination compensation

3D-HEVC includes a block-based illumination compensation model [76] which uses a scale parameter and an offset parameter to compensate the reference block. These parameters are calculated using the rows and the columns just before the

reference and current blocks. However, the block matching algorithm is prior to the compensation. When the illumination difference is large, there is a high probability that objects are mismatched during this compensation. These mismatchings have the minimum displaced frame difference (DFD). However, in these cases, illumination compensation is unlikely to further reduce the displaced frame difference. As a result, the compensation model in 3D-HEVC only works well when the illumination difference is small.

Because our application creates videos that may have significantly different illumination conditions, applying only the 3D-HEVC illumination compensation algorithm is not very effective. Therefore, secondly, we design an illumination matching algorithm, applied prior to 3D-HEVC compression, to leverage its illumination compensation algorithm.

15.3.2 Illumination matching algorithm

Since our compression system is based on a continually growing database, we assume the current video has several matching videos stored in the database. Among these videos, the one that has most similar illumination as the current video is considered to be the best reference. To find it, we develop an illumination matching algorithm which uses global features to recognize the illumination condition of video sequences.

The stimulus of the camera $S(\lambda)$ can be modeled as:

$$S(\lambda) = I(\theta; i; \lambda) \cdot R(\lambda)$$

where $I(\theta; i; \lambda)$ is the illumination with the angle of incidence θ, intensity i and wave length λ, and $R(\lambda)$ is the reflectivity. This algorithm is operated over each path class. Thus, the videos that it process are captured at the same location. As a result, the reflectivity function of the same object is approximately unchanged, since stationary

objects in the videos can be assumed as still. Thus, similar $I(\theta; i; \lambda)$ will produce the similar stimulus, i.e. similar color distributions.

Consequently, to match those videos that have the similar illumination conditions, a feature based on color distributions may be useful. In [77], it is claimed that the skewness of the pixel values is an effective feature to rate the lightness of surfaces. However, [78, 79] disagree that simple statistical features cannot effective predict the illumination conditions, stating that the geometric information of the light intensity is also required. For instance, the video captured in the morning and the video captured in the afternoon may have the same color histogram even though the positions of the shadows are totally different. Considering the position of the cloud, sun, and the shadows, we designed the feature as follow.

First, the features we adopted are the statistical mean and the contrast feature proposed in [80]. This contrast feature was shown to accurately describe the distribution of gray values. It is calculated as:

$$C = \frac{\sigma}{(\alpha_4)^{\frac{1}{4}}}$$

where σ is the standard deviation and α_4 is the fourth moment of pixel values.

Second, to use the geometric information, each frame is equally divided into four quadrants. Frames are divided into upper quadrants and lower quadrants to eliminate the influence of the clouds and sun, since the clouds and sun only appear in the sky. Frames are divided into left and right quadrants to separately describe the position of shadows and sun. For a n-frame sequence, we calculated the average statistic for every quadrant: d_{ul}, d_{ur}, d_{dl}, d_{dr}. Then the illumination feature of one quadrant d of a sequence is constructed as: $Ilum(d) = (mean(d), C(d))$. When a new video sequence i comes into the current path class which has K historical videos, its illumination feature of each quadrant is calculated. Based on each quadrant d, the k^{th} historical video sequence in the current path class calculates its distance to the new video:

$$D(k) = \|Ilum(d^i) - Ilum(d^k)\|_2$$

For each quadrant, a matching score is given to the historical video k based on the rank of $D(k)$ within the set $\{D(k) : 1 \leqslant k \leqslant K\}$. The smaller the $D(k)$ is, the higher the score is. The final matching score of a video sequence is the sum of score of its all four quadrants. Finally, the video sequence that has the highest score is chosen as the best reference. This procedure is designed to ensure the algorithm is robust when large unexpected objects appears in the video. For example, the white or dark vehicle that tries to overtake.

16. EXPERIMENTAL RESULTS

To test our system, we built a DCV database was built that contains 86 videos recorded in 3 different scenarios: downtown(21 videos), highway(35 videos) and boulevard(24 videos). In addition, videos of the same scenario were captured under 4 different illumination conditions: cloudy days, and clear days' morning, noon and afternoon. All of the DCVs are recorded by KDLINKS X1 Digital Recorder with resolution 1920×1080. The DR information is also obtained. Each video will get through the video retrieval and video alignment preprocessors.

We process each video with the video retrieval and alignment preprocessors. To show the compression results, we use 300 frames of each output video of the video retrieval preprocessor. Each video is predicted using one historical video which is chosen by our illumination matching algorithm. To encode the DCVs, we utilize the reference software of HEVC and its 3D extension. We encode each video using quantization parameters, 30, 35, 40, 45, and use BD-BR (Bjøntegaard-Delta Bit-Rate) [81] to quantify the results.

The 86 videos are divided into 2 test sets. Test set I includes videos that are recorded on different days(21 videos for boulevard, 31 videos for highway and 17 videos for downtown). Test set II includes video pairs that are recorded at times that differ by a few minutes only. There are 3 videos recorded in boulevard scenario, 4 videos in highway and 4 videos in downtown scenario. This ensures that the road-side objects and illumination condition are nearly identical.

16.1 Performance of our system

Compared to standard HEVC, the performance of our system over test set II are 62.7% (downtown), 65.0% (highway) and 53.4% (boulevard). Since the difference

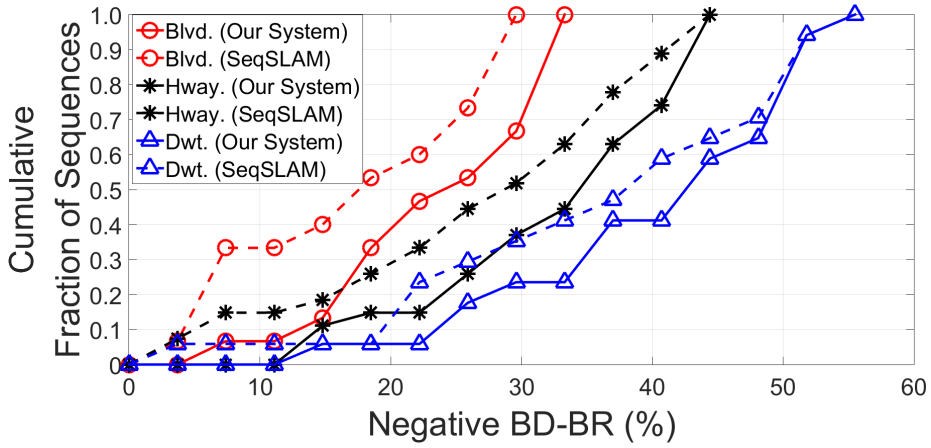


Fig. 16.1.: The bit-saving across sequences

between the reference video and current video are minimum, this is the upper-bound performance.

Fig. 16.1 shows the performance of our system under different scenarios using test set I. The x-axis represents the percentage of bit-saving in BD-rate. Every point in this figure represents the fraction of sequences on which our system does not exceed the corresponding percentage of bit-saving. Performance improves to the right. It shows that our algorithm performs well in all three scenarios, although it performs better in the downtown and highway scenario than in the boulevard scenario. The average amount of bit-saving for the downtown, highway and boulevard scenarios are 39.7%, 31.8% and 23.5% respectively. The downtown scenario has the highest performance because it is insensitive to illumination conditions. It has small sky area and a regular shadow shape, which is easy for the illumination compensation module. The boulevard scenario has the lowest performance. It is more sensitive to illumination conditions and has a large sky area like the highway scenario and a more complex shadow. It requires more videos to construct a dense database so that the best reference video can have a similar enough illumination condition to the current video.

Fig. 16.2 shows the average performance of our system for each type of frame using all videos. Here the subscript denotes the distance away from the reference frame in the current view. This result is also based on test set II. The average efficiency for P, B_8, B_4, B_2 and B_1 frames are 48.1%, 41.5%, 32.9%, 18.6% and 6.0%. The further the frame is away from the reference frame in the current video, the higher the compression efficiency. This is because for those frames far away from the reference frame in the current video, the aligned reference frame in baseline can provide a better prediction. However, if the timeline alignment is not optimal, the compression efficiency will decrease correspondingly. This can be visualized using Figure 16.3, which shows the usage of each type of frame, where the black area indicates the inter-view prediction.

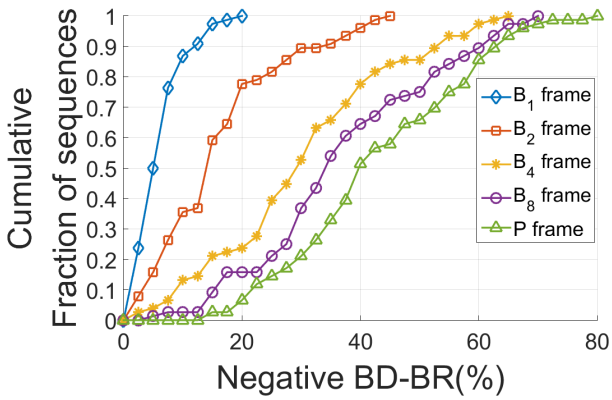


Fig. 16.2.: The bits saving of each type of frame

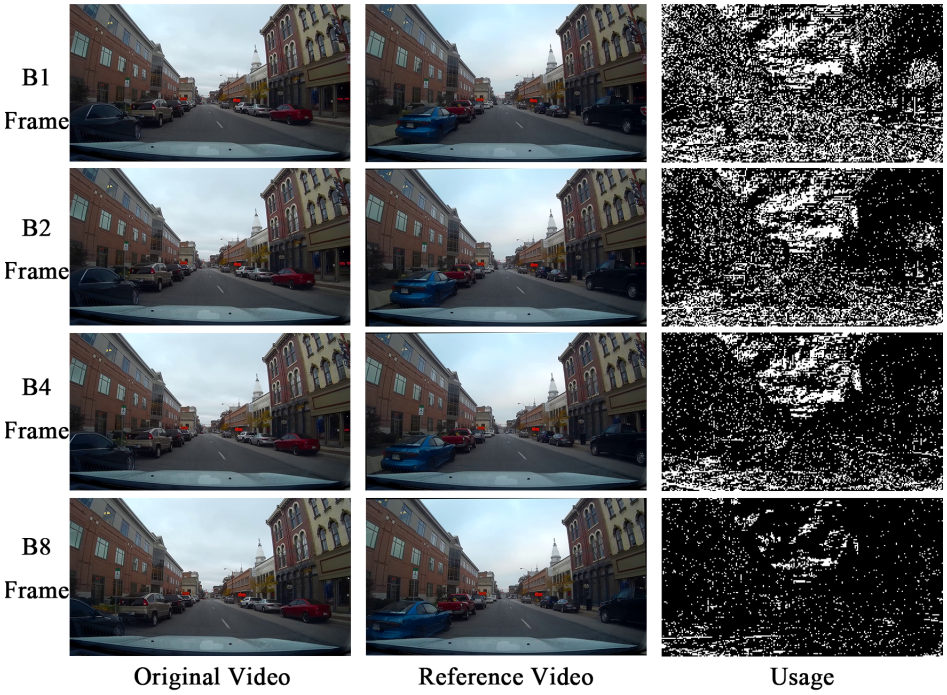


Fig. 16.3.: The usage of inter-view prediction

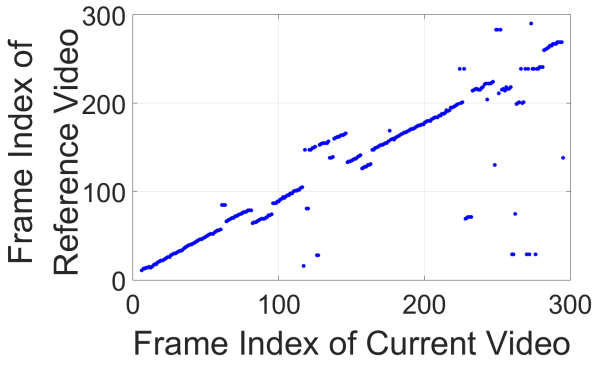
16.2 Comparison to other methods

In Fig. 16.1, we also present the performance using SeqSLAM [70], which uses image intensity for timeline alignment. Its average performance for the downtown, highway and boulevard scenarios are 35.1%, 25.5% and 16.7%. The main reason

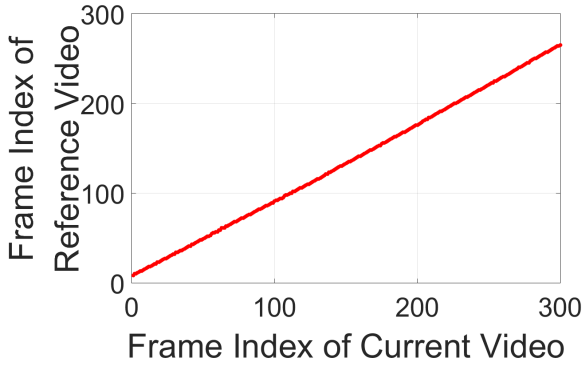
that our algorithm outperforms SeqSLAM is that we have different definitions of "best matching". We define the best matching frame to be the one with the most similar scale as the current frame. SeqSLAM chooses the frame that has the smallest difference to the current frame as the optimal one. These two approaches are the same only if the difference between viewing angles is quite small. In Fig. 16.1, our algorithm is much better than SeqSLAM when the compression efficiency is lower. This is because the strong assumption in SeqSLAM is not satisfied in these cases because the viewing angles are not quite similar. Although our algorithm also cannot achieve high performance in these cases, it is more powerful than SeqSLAM when viewing angles are not quite similar. For cases which have similar viewing angles, our algorithm and SeqSLAM both have high and similar compression efficiency. SeqSLAM performs as well as our algorithm for only 3 sequences in test set I, although the upper-bound performance on test set II are nearly identical.

Fig. 16.4a and Fig. 16.4b show the timeline alignment results of our algorithm and SeqSLAM on a sample sequence from the highway scenario. In this sequence, the viewing angles are a little different, which makes the result of SeqSLAM unstable. This also causes the two algorithms to have quite different results. We take the difference of these two matching results and plot the distribution in Fig. 16.4c. Among 300 frames, around 50% of frames have more than a 4 frames difference. And around 20% of frames have more than a 20 frames difference. Note that 4 frames is already a large distance for video compression. Because the timeline alignment of SeqSLAM is not optimal, its compression efficiency is lower than ours.

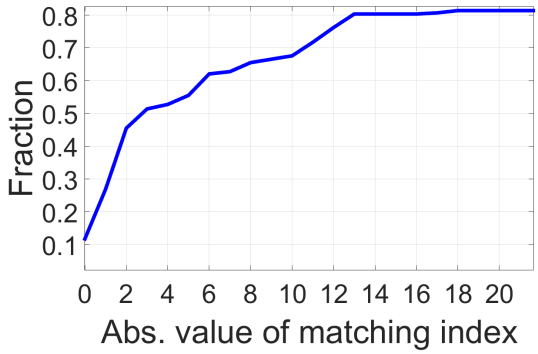
Figs. 16.4d to 16.4f show an example of matching frames for a P-frame. The frame selected by our algorithm shares the same scale across most of the frame, while the frame selected by SeqSLAM matches well only on the left side of the frame. Large differences appear on the right side, which requires more bits for the DFD. For this sequence, our algorithm achieves 29.4% average compression improvement while SeqSLAM achieves 21.0%. For this specific frame, our algorithm achieves a 49.7% savings while SeqSLAM only achieves 37.1%.



(a) SeqSLAM Matching Result



(b) Our Matching Result



(c) Matching Difference

(d) SeqSLAM Result Example

(e) Example of Current Frame

(f) Our Result Example

Fig. 16.4.: Matching results of SeqSLAM and our algorithm

If we use homographic transformations discussed in section 15.2.2, the performance of our system decreases 4.2% (downtown), 4.7% (highway) and 2.2% (boulevard) while SeqSLAM decreases 2.7%, 2.9% and 1.0%. The inaccurate matches of SeqSLAM are less affected by the homographic transform than our more accurate matches are.

17. CONCLUSION

In this work, we construct a DCV compression system that takes advantages of the historical data recorded from previous trips on the same route. The video retrieval and alignment preprocessors we design to construct a reference video using the historical data enables us to leverage the power of the 3D-HEVC standard. Our experiments show that significant compression improvements are achieved when DCVs are jointly compressed with a carefully-formed reference video. On average, we achieve around 30% bit-rate savings.

While we described the system assuming it would be used by a single vehicle, driving repeated routes on different days, our compression framework is also relevant for multiple-vehicle systems. For example, it can be used by a bus company to archive DCVs on a regular route.

Our future work will mainly focus on developing relevant applications. For example, we are interested in data-driven solutions for road pedestrian detection. Based on our system, huge amount of DCVs can be systematically stored. These videos may provide nearly complete information of the recorded scenes. By identifying the common visual information in the videos of a same scene, we may be able to remove the abnormal objects, for example other vehicles or pedestrians, to create a "clean" scene. By doing subtraction, we can easily detect the pedestrians in new incoming videos in real-time.

Part IV

An Evaluation Methodology for Visual Odometry

18. INTRODUCTION

The Visual Odometer (VO) is a hot topic in the computer vision community. The main purpose of visual odometry is to use one or more cameras as the only input device to estimate the 6 DoF (Degree-of-Freedom) of the camera(s). By using only visual inputs, the VO becomes a significant tool for many other research and applications.

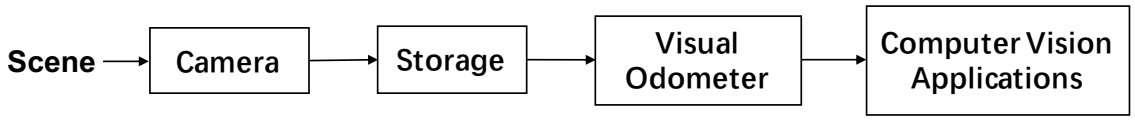


Fig. 18.1.: The data flow of computer vision applications

Fig. 18.1 shows the data flow of a typical computer vision application where users apply the VO. The visual information is gathered and provided to the VO. The motion estimation result of the VO is then used to enable various applications, such as robotic navigation or video stabilization. In this situation, users are eager to know both how the performance of the application is affected by using the current VO and which VO provides the best application performance.

It is important to have a methodology that can help the user automatically choose an optimal VO for their application. Currently, many VOs have been proposed. It is challenging to choose an optimal VO manually because users must consider with two variables: the method used by the VO and the type of input data used in the application. Each VO differs not only in terms of the detailed operation (for example, which specific features are used by a method based on local features), but also in terms of the basic algorithmic and mathematical approach. Meanwhile, users may have different conditions or assumptions for their input data, for example whether the image frames have more camera sensor distortion or motion distortion. These two variables lead to many possible combinations of data assumptions versus VO

methods. Each VO algorithm has different strengths and weaknesses for each input data type. Therefore, given the wide variety of options for both VO algorithms and input data, it is important to have a systematic approach to select a VO for a given type of input data.

To address this problem, current methods focus on creating a benchmark [82, 83] based on a large dataset where they apply metrics to measure the pose error between the estimation from VO and the ground-truth. In this work, we propose a VO evaluation methodology instead of a benchmark. Our methodology includes an operational protocol, which helps us to investigate the VO performance based on 5 influence factors, and includes various metrics designed for measuring VO performance in different applications. Our methodology has advantages relative to a benchmark. First, it does not rely on a large dataset, so it is not restricted by the data distribution, and second, it can provide suggestions for the particular application that users are interested in.

In the next section, we propose a categorization of the applications that rely on VOs, review current effective VOs, and review current evaluation methods for VOs. In Section 20, we propose the concept of our evaluation methodology in contrast to a benchmark. By jointly comparing the advantages and disadvantages of VOs and all the possible situations of different applications in Section 21, we describe how our methodology enables a thorough evaluation. Although our methodology can be applied to any VO method, to show its effectiveness, in Section 22, we apply our methodology to evaluate and analyze 4 existing VOs: Mono-VO, VisualSfM, Direct Sparse Odometer (DSO) and SfmLearner. Finally, we conclude our work in Section 23.

All in all, our main contributions are (1) an evaluation methodology for VOs that is more comprehensive than a benchmark; (2) new metrics to evaluate the performance of VOs on visual augmentation applications; and (3) a detailed operational protocol of the methodology, which is based on rigorous VO test conditions so that the performance of VOs can systematically and effectively be distinguished.

19. RELATED WORK

19.1 Applications of Visual Odometers

The goal of a VO is to estimate both the translation and rotation of the camera using only visual information. Based on how this information is used, the target applications for VOs can be categorized into two main parts: robotic perception and visual augmentation. Robotic perception provides information for robotic systems, and visual augmentation is relevant for human visual entertainment systems. The goal of robotic systems is that the estimates of camera pose are numerically the same as the ground truth. Example applications of robotic systems are visual-based navigation systems [84, 85]. The goal of human visual entertainment systems is to use the pose estimation results from the VO to perform further visual editing on the images and videos. Example applications include Augmented Reality [86] or video stabilization [8, 9, 11].

19.2 Existing Visual Odometers

Current state-of-art VOs can be classified into 3 types: local-feature-based methods, direct methods and deep-learning-based methods. The differences of their core theories make them have different advantages and disadvantages. Local-feature-based methods estimate the camera pose based on feature matching. Well-known features include SIFT [73], SURF [37] and ORB [74]. These features are usually located at the point on the image where the gradient is extreme in different directions and across different scales. The advantage of this kind of method is that it is robust under most image/video content since most images and video frames have sufficient gradient changes. The disadvantage of these methods is that they are influenced by the

quality of the feature points. When there is noise, motion blur or rolling shutter distortion in the image, the location of the detected feature points or the computed feature descriptors are inaccurate. Note that the influence of these distortions on the local feature accuracy depends on the feature type and is hard to quantify.

Direct methods do not compute any feature descriptors and only predict the camera poses based on the pixel values. Well-known algorithms include SVO [87], LSD-SLAM [88] (the front-end part) and DSO [52]. The advantage of these methods is that they are more robust under different image/video content than the local-feature-based methods. As long as there is illumination change across the frames, direct methods work fairly well, while local-feature-based methods may fail to detect any feature points. In our evaluations, direct methods are also robust to noise and blur in the image. However, direct methods are fragile when there is sudden or fast camera motion.

Deep-learning-based methods are being developed recently. The most famous method is called SfMLearner [89]. Deep neural networks usually require a large amount of data to train, and these data are also required to have accurate labels. As a result, these methods usually rely on several specific public datasets, such as the KITTI dataset [82]. This limits their accuracy in those scenarios not included in the datasets, for example indoor scenarios. However, as we show below, deep-learning-based methods are also robust to noise and distortions in the image. This may because the input to the network is usually a downsampled image and the downsampling significantly reduces the influence of the noise and distortions.

19.3 Existing evaluation benchmarks of Visual Odometers

Current visual odometer evaluation benchmarks mainly focus on computing the average performance on real scene data using pose error. Based on constructed datasets, the evaluated VO yields the motion estimation results. Then the pose error between the estimated motion and the ground truth motion is computed. Be-

tween robotic perception applications and visual augmentation applications, current benchmarks focus on the previous one, because VOs with small error values produce numerically accurate pose estimations relative to the ground truth.

One of the most frequently used benchmarks is the KITTI benchmark suite [82]. The on-road real scene data is captured by a recording system mounted over a vehicle. The camera system mainly includes a GPS module, two RGB cameras, and a Li-dar. The ground truth camera pose and RGB images are provided and synchronized. However, since the cameras are mounted on a vehicle, the camera motion is constrained to have only yaw motion and translations. To evaluate the performance of the VO, the Relative Pose Error [90] is used as a metric. On the contrary, the other popular benchmark, the TUM dataset [83], provides indoor scenes with more random camera poses. In [83], the Kinect camera is used to capture indoor scenes. The camera motion is captured using a high-accuracy motion-capture system with eight high-speed tracking cameras. Since the ground truth pose of the camera is provided, the Relative Pose Error is again used to evaluate the performance of tested VOs. [91] applies this dataset to further evaluate different VOs under different challenges. The challenging situations include videos with a corridor, videos containing various scene changes, videos with illumination changes, and videos with fast motion.

There are also evaluation benchmarks that are based on different metrics than the Relative Pose Error. In [92], the author proposes to evaluate VOs based on 5 different criteria: (1) the repeatability of the VO, which is measured by the covariance between pose estimation in different runs; (2) the loop closure error, which is the distance between the end point and start point when there is a loop closure in the video; (3) the sparse bundle adjustment (SBA) re-projection error, which is the difference between the re-projection of the inliers before and after the SBA; (4) the number of RANSAC iterations, which indicates how much effort the RANSAC method needs to find a solution; and (5) the percentage of the inliers with respect to original feature points. The drawback of this metric is that criteria (3) to (5) only apply to local feature based methods. Another benchmark [93] aims to evaluate the performance only for

learning-based VOs. The proposed benchmark [93] is based on three different datasets [82, 94, 95]. However, the adopted evaluation metrics are the Mean Square Error (MSE) of the predicted label versus the real one, Normalized Root MSE (NRMSE) and the Standard MSE (SMSE), which have been claimed in [90] to be ineffective for comparison.

On the contrary, our evaluation methodology focuses on both robotic perception and visual augmentation applications. Also, it does not rely on large dataset and is effective for all VO methods.

20. OUR CONTRIBUTION: AN EVALUATION METHODOLOGY

In this paper, we propose an evaluation methodology for VOs, which is broader than a benchmark. A complete evaluation methodology includes three parts: (1) a dataset or a collection of datasets; (2) a metric or metrics that define goodness and (3) an operational protocol which is a suggested strategy for using the datasets. The main idea of our methodology is to evaluate the performance of the VOs under 5 influence factors based on our operational protocol.

A benchmark does have both a dataset and a metric, but it does not have an operational protocol, or say, its protocol is too trivial. For example, in the KITTI benchmark [82], the dataset is the KITTI dataset and the metric is the Relative Pose Error. The operational protocol is just performing the evaluated VO over all data and using the average metric value to represent its performance. The drawback of a benchmark is that it is based on the assumption that its dataset has a good distribution, in that it covers all the scenarios for the application.

We propose a methodology, which is more user-oriented and does not rely on assumptions of the data distribution. First, we propose two metrics for different applications instead of a single pose error. This is because some VOs may perform well on visual augmentation applications but not on robotic perception applications. The proposed metrics are pose error and visual error. The pose error is designed for robotic perception applications. VOs with small pose error produce numerically accurate pose estimations relative to the ground truth. The visual error is designed for visual augmentation applications. Applications incorporating VOs that have a small visual errors produce visual results that are more comfortable for people to perceive. Second, we develop the operational protocol rather than constructing and relying on a large dataset. We consider 5 factors that may influence the performance

of VOs, which are described in Section 21.1 below. Our protocol evaluates VOs based on the 5 factors separately. And the dataset is also designed and constructed based on the factors, which is shown in Section 21.

The advantage of our methodology is that depending on what the main factor is for the users' application environment, our methodology can provide suggestions on which VO is more robust. And when users study a single VO, our methodology can help find which the bottleneck factor is. Users can get the corresponding information based on their application.

21. METHODOLOGY DESIGN

In this section, we first discuss the 5 evaluation factors, then we discuss the operational protocol of our methodology, namely, how we use existing datasets to evaluate VOs based on each of the 5 factors. After that, our entire evaluation process is introduced. The evaluation is accomplished using one factor at a time. To achieve this isolation of factors and avoid the influence of other factors that are not being evaluated, we use synthetic videos as our main input. We describe in detail how we synthesize videos. Finally, the metrics used to evaluate the performance of VOs are discussed.

21.1 Evaluation factors

As we reviewed in the previous section, the performance of each VO method is influenced by different factors of the input data. Our main goal here is to carefully control these factors and construct rigorous VO test conditions so that the performance of VOs can systematically and effectively be distinguished.

Fig. 18.1 shows the data flow of a typical computer vision application. A camera with 3D motion captures the scene content. The resulting video is compressed, transferred, and stored. Finally, the user selected visual odometer extracts the video data, estimates, and provides the 3D camera motion to the computer vision applications. Any step before the motion estimation step may have factors that influence the accuracy of the VO. And the accuracy of the VO affects the final performance of the computer vision application. For example, rolling shutter distortion may lead to drifting in the pose estimation so that virtual objects may shift across time in the augmented reality application.

We describe the 5 factors in the order of the data flow. There are 2 factors at the "scene" step: (1) whether the image frame has few features or rich features; and

(2) whether or not the image frame is from a learned scenario. For different scene content and structure, different VOs may perform differently. Having enough feature points is critical for local feature based methods while it has less influence on direct methods and deep-learning based methods. The deep-learning based methods are influenced by whether the training set contains any content that is similar to the content currently being processed.

There are also two factors at the camera capturing step: (3) whether the image frame has noise from the sensor level; and (4) whether the image frame suffers distortions caused by motions. Extreme illumination conditions may introduce noise and distortions on the CMOS sensor. For instance, over-exposure may happen when there is too much light. When under low light, high settings of ISO can introduce more noise than usual. Frames taken with wide angle and wide aperture lens suffer from optical vignetting. Also, the camera motion can introduce motion blur and rolling shutter distortion on the frame.

Factor (5) is whether the image frame suffers compression distortions. It comes from the last step before motion estimation: the storage or transmission. The lossy compression steps in video codecs introduce more artifacts and blur as the compression rate increases. Any image-based VO may be influenced by compression.

21.2 Operational protocol

There are 3 aspects we want to control during the evaluation: camera motion, compression rate, and image content. Amid the 5 factors, factor (4) concerns camera motion, factor (5) concerns video compression, and the other three are about image content. As a result, to isolate the influence of the 5 factors, we set up 3 general test conditions on the test videos: (1) Synthetic motion + Synthetic content under different compression rates; (2) Synthetic motion + Real content under different compression rates; (3) Real motion + Real content under different compression rates. In general, using synthetic or real motion, we can isolate the influence of camera motion.

Table 21.1.: Detailed settings of test factors

Factor		Test Video Conditions			
		Motion Type	**Image Content Type**	**Image Content Detail**	**Video Dataset**
(5)	**(1)**	Synthetic	Synthetic	Less feature / Rich feature	Virtual KITTI / Virtual KITTI
	(2)	Synthetic	Synthetic	Learned / Unseen	Virtual KITTI / Synthetic Indoor photo
	(3)	Synthetic	Synthetic / Realistic		Virtual KITTI / KITTI
	(4)	Synthetic / Realistic	Realistic		Univerisity Street / Walk

(a) University Street

(b) KITTI

(c) Walk

(d) Virtual KITTI (Fog)

(e) Synthetic Indoor

(f) Virtual KITTI (original)

Fig. 21.1.: Example frames of test videos

Using synthetic or real content images, we can isolate the influence of factors (1) to (3), which are about image content. Using different compression rate, we can study the influence of compression distortion. The detailed evaluation settings of each factor can be found in Table 1. Note that we assume the deep learning method to be evaluated has been trained based on the KITTI dataset, which is true for most cases. In the experiments, we chose the SfMLearner [89].

To evaluate the performance of VOs under factor (1), the number of feature points, we aim to set it as the only variable. To achieve this, we use synthetic camera motion and image content so that no distortion is introduced by the camera capturing step. As a result, factors (3) and (4) are fixed. Also, the frames are from Virtual KITTI dataset, which ensures the deep-learning method will have an input consistent with its training input (factor (2) is fixed). Specifically, we choose a pair of frames in the Virtual KITTI dataset. The two frames are from the same time instance but have different synthetic illumination conditions: the frame from the original illumination

condition (Fig. 21.1 (f)) has more feature points than the frame from the "fog" illumination condition (Fig. 21.1 (d)) .

Factor (2) is used to evaluate if the performance of VO relies on scene structure. Again, we aim to set the scene structure as the only variable. We evaluate the performance of VOs with image frames from learned and excluded scenarios in the dataset. We vary the scene structure of the two videos. One of the videos is synthesized using frames in the Virtual KITTI dataset (Fig. 21.1 (b)). The other video is synthesized using virtual indoor photos (Fig. 21.1 (e)), which are different than the data used to train the deep-learning method. Also, we use synthetic camera motion and image content to fix factors (3) and (4), and this time we control the number of feature points to be similar to fix factor (1).

Factor (3) and (4) concern the camera capturing step. As a result, we correspondingly vary the image content and camera motion to be synthetic or realistic. For factor (3), we aim to set image distortion from the sensor level as the only variable. Images from Virtual KITTI (Fig. 21.1 (f)) and KITTI dataset (Fig. 21.1 (b)) at the same time instance are used to generate the video. We verify that the number of feature points in both of the images are close to ensure factor (1) is fixed. Note that we still use the outdoor images in order to avoid unfairness to the deep-learning based methods (factor (2) fixed), and we use synthetic motion to avoid the motion distortion (factor (4) fixed).

For factor (4), we set the influence of motion distortion as the only variable. We vary the camera motion to be synthetic and realistic. However, it is hard to synthesize a video with realistic motion distortion using a synthetic image. As a result, we make a concession to allow the image content to be realistic but captured by the same camera sensor. The video from dataset [3] (Fig. 21.1 (a)) is the video with synthetic motion and real content. The video from dataset [96] (Fig. 21.1 (c)) is the video with real motion and real content. Note that these two videos are both captured by the GoPro, which helps to fix factor (3). Factors (1) and (2) are fixed using the same method we apply when investigating factor (3).

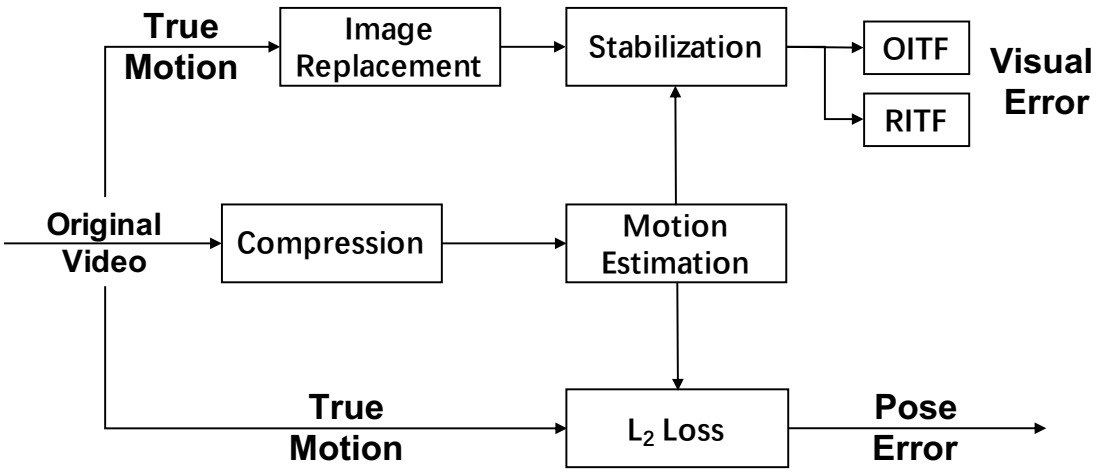


Fig. 21.2.: Our evaluation process

Factor (5) concerns the compression rate of the video codec. We consider that the compression rate may have different effects on how the factors (1) to (4) affect VO performance. As a result, when experimenting with factors (1) to (4), we change the compression rate at the same time to investigate how performance changes.

21.3 Evaluation process

Fig. 21.2 shows the diagram of our evaluation process. The input of our process is a video with known motion. The "motion" here indicates the rotations of the camera, not translations. This is because we focus on evaluating the performance of VOs to estimate rotations. There are two reasons. First, for most computer vision applications, the accuracy of rotation estimation is much more important than translation estimation. For example, in the augmented reality application, an added virtual object needs to rotate according to the camera motion. While the size of the object also needs to change accordingly when the translation is changed, this can be achieved by comparing the size of the local patch where the virtual object is located. Also, in video stabilization applications, rotational motion introduces more perception of shakiness than translations [1]. Second, for existing VOs, the camera

poses on the special Euclidian Group are usually first estimated using a error function. Then the rotations and translations are decomposed from the resulting pose matrices. Therefore, it is sufficient to evaluate the performance of VOs based on rotation results.

Given the input video, we first compress the video using H. 264 [97] with different compression rates where we use the FFMPEG to compress the video. In total there are 7 versions of each video: the original video and compressed versions with Constant Rate Factor (CRF) 18, 24, 30, 36, 42 and 48. Then we employ the desired VO to estimate the motion of the camera. The resulting estimated camera motion is then sent to the upper and lower branches in the block diagram to compute two performance measurements: pose error and visual error. The visual error is designed for visual augmentation applications, and is one of the main contributions of this work. The pose error is designed for robotic perception applications. It is based on the L_2 error between the ground truth motion and the estimated motion. Detailed descriptions of both are provided in Section 21.5, but the principles underlying the visual error are explained next.

The visual error is designed to measure the performance of VOs for two main sample applications for visual augmentation: video stabilization and Augmented Reality. We propose two measurements to quantify visual error: the Ordered-Inter-frame-Transformation-Fidelity (OITF) and Random-ITF (RITF). Both OITF and RITF are based on ITF [20], whose goal is to measure video stability. Both OITF and RITF are computed based on the pixel values of the frames, where OITF is computed between adjacent frames, and RITF is computed between random frames. In Section 21.5, we will show that they can both be interpreted as a weighted measure of the 2D motion amplitude between the frames. They are effective for choosing VOs for video stabilization algorithms, as well as for Augmented Reality systems where the goal is to ensure any attached virtual objects do not have drift when the camera is moving around.

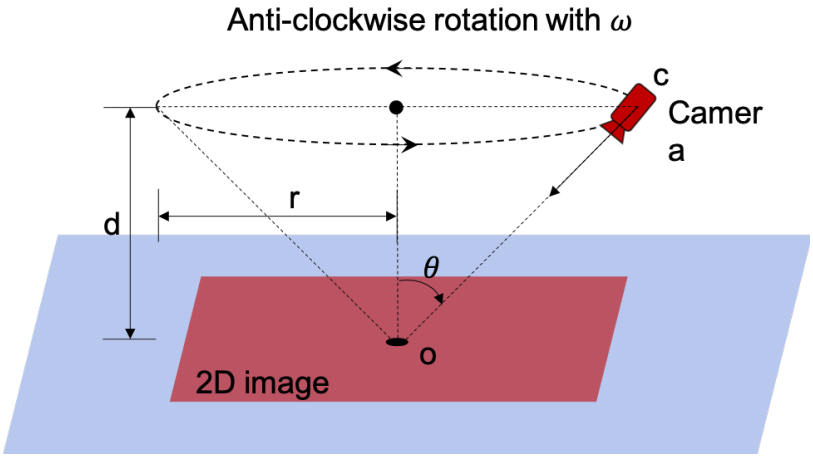
In the context here for Fig. 21.2, we compute both visual errors of a stabilized video that has been constructed specifically to isolate the impact of motion, and to

be independent of each images exact pixel values. This will allow the measures to focus only on the impact of the stabilization result. To accomplish this, we apply an image replacement step as shown in Fig. 21.2. Specifically, we create a stabilized video by taking a static image, applying a ground truth rotational motion, and then stabilizing it using the estimated motion from the VO being evaluated. The final measures average each of OITF and RITF on stabilized videos created from a variety of 2D images.

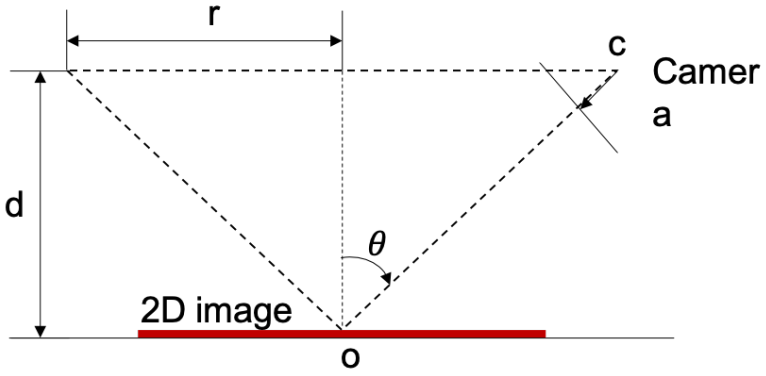
21.4 Video synthesis for our methodology

In this section, we introduce two approaches to generate the synthetic videos that are used as the original input video in Fig. 21.2 for factors (1) to (4). Again, the reason is we want to use synthetic videos with synthetic motion to control the influence of motion distortion over VOs. One approach is newly proposed in this work. It takes one image as input and provides a video that records this 2D image with translations and rotations. This approach is used for factors (1) to (3). Another approach is proposed in our previous related work [3]. It takes a video from a 360° video that is slowly moving forward as input and generates a video with designed rotations. This approach is used for factor (4).

We first introduce our newly proposed method. We use this method to generate videos with synthetic motions for factor (1) to (3) in Table 21.1. In this method, we take one 2D image as input and synthesize its appearance by assuming the camera is rotating around with 3D translations. The 3D plot of the camera motion is illustrated in Fig. 21.3 (a) and the side view of the plot is shown in Fig. 21.3 (b). The virtual camera rotates on a circle with radius r and speed ω rad/s. Its operating orbit has distance d with respect to the 2D image. The camera points straight to the center o of the 2D image with no rotation around its z axis.



(a) 3D view of the synthesized motion



(b) Side view of the synthesized motion

Fig. 21.3.: Illustration of synthezied camera motion

It can be shown that the translation between the camera center to the 2D image center in the XYZ world coordinate is:

$$t_{oo'} = \left[rcos(\omega t), -rsin(\omega t), d\right]^T, \tag{21.1}$$

and the rotation matrix from the world coordinates to the camera coordinates is:

$$R_{cw} = \left[\frac{R_x}{|R_x|}, \frac{R_y}{|R_y|}, \frac{R_z}{|R_z|}\right], \tag{21.2}$$

where

$$R_x = \begin{bmatrix} d^2 + r^2 sin^2(\omega t) \\ r^2 cos(\omega t) sin(\omega t) \\ drcos(\omega t) \end{bmatrix}, \tag{21.3}$$

$$R_y = \begin{bmatrix} 0 \\ d \\ -rsin(\omega t) \end{bmatrix}, \tag{21.4}$$

$$R_z = \begin{bmatrix} -rcos(\omega t) \\ rsin(\omega t) \\ d \end{bmatrix}. \tag{21.5}$$

Suppose the intrinsic parameter of the virtual camera is K. To obtain the 2D coordinates of four corners of the 2D image during the synthetic rotation, we have the following state transfer function:

$$x(t) = K \begin{bmatrix} R_{cw}^T & R_{cw}^T t_{oo'} \\ \mathbf{0} & 1 \end{bmatrix} K^{-1} x(0). \tag{21.6}$$

In our methodology, we also use another video synthetic approach which is proposed in our previous related work [3]. We apply this approach to generate a video with synthetic motion for the factor (4) in Table 21.1. The reason the synthetic video for factor (4) needs to be generated using this approach is that the comparison video

"walk" is a realistic recorded video which has forward translation and depth changes in the frames. We want to remove the influence of these two issues, and our newly proposed approach in the previous paragraph cannot solve the problem. As a result, we apply the approach we proposed in [3].

In [3], we mount 6 GoPro cameras on a tripod with wheels and record 360-degree videos by slowly and smoothly moving the tripod forward. Then we use sine-waves to synthesize yaw and pitch motions and apply the motion to the 360-degree video to get a video with regular resolution. The resulting video looks like a First-Person video that is recorded when the recorder is running forward. In this way, the resulting video has synthetic rotation motions, forward translation and depth change.

21.5 Evaluation metrics

As we discussed in Section 19.2, different VOs are suitable for different applications. General applications are classified into robotic perception and visual augmentation. For both kinds of applications, we provide measurements of the motion estimation result. Pose error and visual error correspond to robotic perception applications and visual augmentation applications, respectively.

21.5.1 Pose error

Robotic perception applications numerically prioritize the accuracy of the motion estimation because their goal is to localize the robot across time. Given a time instance they want to know the accurate pose of the robot. As the time scale of the estimation problem increases, they want the accumulated estimation error to be small.

A straightforward measurement of the pose accuracy is to compute the L_2 error between the ground truth pose and the estimated pose across time. However, in [90], the author shows that this is suboptimal. Assume the ground truth pose at frame i

is p_i and the estimated pose is $\widehat{p}_i$. The error from frame i to frame j can be expressed as:

$$e_{i,j} = \sum_{t=i}^{j} (p_t \ominus \widehat{p}_t)^2, \tag{21.7}$$

where $\ominus$ denotes the inverse compositional operator [90]. When the poses are represented using rotation matrices, this operation is equivalent to:

$$p_t \ominus \widehat{p}_t = R_t^{-1} \cdot \widehat{R}_t. \tag{21.8}$$

In [90], the author demonstrates that there are special cases that equation (21.7) cannot handle. Assume there is only an estimation error between frame 1 and 2, and there are no other estimation errors in the rest of the frames. This estimation error is incorporated repeatedly until the last frame, if equation (21.7) is applied. Consider two VOs that both only have one time error. The VO whose error occurs later in the timeline has less error than the one whose error occurs in the first frame. This is unfair. To solve this, [90] proposed to use the difference between the relative poses as the error, which is computed using:

$$e_{i,j} = \sum_{i \le m < n \le j} (p_m \ominus p_n) \ominus (\widehat{p}_m \ominus \widehat{p}_n). \tag{21.9}$$

To compute this error function from frame 1 to the end frame K, the KITTI dataset benchmark updates this equation to:

$$e_{1,K} = \sum_{t=1:s:K-1} \angle[(p_t \ominus p_K) \ominus (\widehat{p}_t \ominus \widehat{p}_K)], \tag{21.10}$$

where $\angle[\cdot]$ is the rotation angle, and s is the step size.

For our metric, we adopt the similar error function. However, the KITTI dataset contains only on-road data, which only have the yaw motion of the camera. In our

case, we have yaw, pitch and roll motion. As a result, we modify the equation (21.10) to be:

$$e_{1,K} = \sum_{t=1:s:K-1} (p_t \ominus p_K) \circledcirc (\widehat{p}_t \ominus \widehat{p}_K), \tag{21.11}$$

where $R_m \circledcirc R_n$ denotes the sum of rotation error on x, y, z three axes. Assume we have a basis e_x, e_y, e_z for the three axes. Then $R_m \circledcirc R_n$ can be expressed as:

$$R_m \circledcirc R_n = \sum_{i=x,y,z} \|R_m^{-1} R_n e_i - e_i\|_2. \tag{21.12}$$

21.5.2 Visual error

Visual augmentation applications prioritize the motion estimation result based on the performance of their algorithms. For example, the video stabilization algorithms require the resulting videos to be smoothed based on the estimated motion; the augmented reality applications desire that the attached virtual objects do not drift when the camera is moving around.

There are two corresponding ideal cases of the motion estimation result: (1) the estimation has small drift across time, but the drift accumulates to a large value; (2) the estimation has errors around the ground truth motion, but there is no noticeable accumulated error. Case (1) is desired for video stabilization applications since the motion between adjacent frames is small. Case (2) is desired for augmented reality applications since virtual objects do not drift across time. We create two visual errors to evaluate the performance of VOs based on these two cases.

The proposed visual errors are Ordered-Inter-frame-Transformation-Fidelity (OITF) and Random-ITF (RITF), which are used to measure the performance of VOs for cases (1) and (2), respectively. They are based on the Inter-frame-Transformation-Fidelity (ITF) [20], which has been widely used to evaluate video stability [98–101]. The original paper [20] and the subsequent citations [98–101] only describe it as a Peak-Signal-to-Noise-Ratio (PSNR) value. However, as we will show below, ITF essentially is a weighted 2D motion amplitude measurement. In the following, we first

demonstrate this new interpretation and then provide the details of how we improve the original ITF to obtain our new measurements, OITF and RITF, which are more effective to evaluate VO performance on both video stabilization and Augmented Reality applications.

Here, we re-interpret the ITF based on the Lucas-Kanade optical flow [102]. Assume we have a video $I(t)$, which is a sequence of images. The pixel value at location (x, y) is $I(x, y, t)$. Suppose video $I(t)$ has N frames and the maximum pixel value is 255. The original ITF proposed in [20] is computed using:

$$ITF(I) = \frac{1}{N-1}\sum_{k=1}^{N-1} PSNR(k), \tag{21.13}$$

where

$$PSNR(k) = 10\log\frac{255^2}{MSE(k)}. \tag{21.14}$$

The $MSE(k)$ computes the Mean-Squared Error between two frames at times k and $(k+1)$. We reformulate equation (21.13) and (21.14):

$$ITF(I) = 20\log 255 - \frac{10}{N-1}\sum_{k=1}^{N-1}\log MSE(k). \tag{21.15}$$

Now let's consider the Lucas-Kanade optical flow. Under the brightness constancy constraint, we have:

$$\frac{\partial I(x,y,t)}{\partial x}v_x + \frac{\partial I(x,y,t)}{\partial y}v_y = -\frac{\partial I(x,y,t)}{\partial t}. \tag{21.16}$$

If the frame size is W by H, the Mean-Squared Error can be rewritten as:

$$\begin{aligned} MSE(k) &= \frac{1}{W\cdot H}\sum_x\sum_y(\frac{\partial I(x,y,k)}{\partial t})^2 \\ &= \frac{1}{W\cdot H}\sum_x\sum_y(\frac{\partial I(x,y,k)}{\partial x}v_x + \frac{\partial I(x,y,k)}{\partial y}v_y)^2 \end{aligned} \tag{21.17}$$

Combining equations (21.15) and (21.17), we can see that the original ITF is just a function of motion amplitude v_x and v_y weighted by the local pixel gradient in the vertical and horizontal directions. Ideally, the smaller the motion is, the larger ITF becomes. In our metric, the smaller the motion in the stabilized video, the larger ITF becomes and the more accurate the motion estimation is.

However, there are two main problems with this original ITF. First, the brightness constancy constraint may not be satisfied. The video stabilization we perform in our methodology (see Fig. 21.2) is a naive method without any image stitching techniques. It is inevitable to have black regions in the resulting images. In this situation, the gradient or derivative in equation (21.16) and (21.17) is meaningless. To solve this problem, we proposed to use a revised ITF [1] which is only computed based on the non-black region. To make it more precise, in this work, we propose to compute ITF based on a Region-Of-Interest (ROI) and name the new metric Ordered-Inter-frame-Transformation-Fidelity (OITF). When we compute the MSE, we apply a mask $\delta(x, y, k)$:

$$MSE(k) = \frac{1}{\sum_{x,y} \delta(x, y, k)} \sum_x \sum_y (\delta(x, y, k) \cdot \frac{\partial I(x, y, k)}{\partial t})^2, \tag{21.18}$$

where

$$\delta(x, y, k) = \begin{cases} 0 & \text{if } I(x, y, k) \text{ or } I(x, y, k+1) \in \text{black region} \\ 1 & \text{otherwise} \end{cases}. \tag{21.19}$$

Note that to determine whether a pixel is inside the black region, we do not rely on the RGB value of the resulting video frame. Given the homography that is used to stabilize the video, we have the precise binary mask of the black or non-black region of each frame.

OITF is the first visual error we propose. The second visual error is called Random-ITF (RITF). Recall that there are two cases for the motion estimation result. Case (1) is ideal for video stabilization applications while case (2) is ideal for aug-

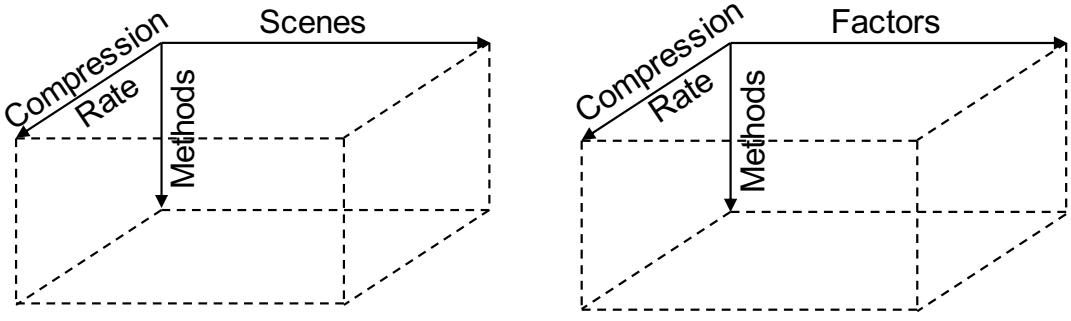
mented reality applications. The OITF is only sufficient for evaluating case (1) since it only measures the motion between adjacent frames. However, the resulting stabilized videos in both cases have small motion between adjacent frames. This means the OITF cannot distinguish these two cases. To solve this problem, we propose the RITF to make the measurement more thorough.

The RITF is computed using the same equation as OITF. However, before the computation, we randomly shuffle the frames in the stabilized video. In this situation, for case (1), the expected error between resulting adjacent frames is larger than in case (2). Using the OITF and RITF together, we can sufficiently evaluate the visual error of the motion estimation result. If both values are large, it indicates that the algorithm not only can provide a good result for video stabilization but also can be helpful for Augmented Reality applications. Otherwise, the corresponding VO may only be suitable for one of the applications.

22. EXPERIMENTAL RESULTS

In this work, we apply our proposed metric to evaluate 4 different VOs: Direct-Sparse-Odometer (DSO) [52], VisualSfM [103], Mono-VO [104] and SfMLearner [89]. They typify different algorithmic approaches of VOs. The DSO is the state-of-art method for direct approaches. Mono-VO and VisualSfM are local feature based methods. However, VisualSfM includes the bundle adjustment that Mono-VO does not have. SfMLearner is representative among the Deep-Learning-based methods. Theoretically, DSO is sensitive to camera motion; SfMLearner relies on the distribution of its training dataset; the performance of Mono-VO and VisualSfM depends on feature numbers; all of them are affected by the camera sensor distortion and compression effects.

The output of our evaluation methodology is a three-dimensional matrix, which is illustrated in Fig. 22.1. For each VO method, we have its evaluation scores under 6 different scenes across 7 different compression rates. The absolute performance matrix (Fig. 22.1 (a)) is helpful to know the absolute performance of a VO under a certain



(a) Absolute performance matrix

(b) Factor-based performance matrix

Fig. 22.1.: The outputs of our evaluation methodology

situation. To investigate the influence of a single factor on VOs, the factor-based performance matrix (Fig. 22.1 (b)) is computed.

22.1 Computation of factor-based performance matrix

To compute the factor-based performance matrix, we focus on VO method i, compression rate CRF_j and factor (k). Recall that for each factor, we have the ideal situation and a challenge situation. Suppose the ideal and challenge situations correspond to scene m and n. Suppose the absolute score of these two situations are $E(i, CRF_j, m)$ and $E(i, CRF_j, n)$ respectively, where E can be the pose error e or the visual error OITF/RITF. Then the measurement of the method under compression rate CRF_j and factor (k) is computed as:

$$E_{factor}(i, CRF_j, (k)) = E(i, CRF_j, m) \sim E(i, CRF_j, n), \tag{22.1}$$

where $\sim$ represents a general difference of the measurement, and can be a fractional relationship or a subtraction relationship.

22.1.1 Relative influence of factors

For users, one interesting question is: which factor influences the current VO the most? It is useful when users want to analyze the bottleneck of the performance of the current VO. Based on equation (22.1), we propose a relative influence measurement. For each VO, the resulting value is the performance change relative to the absolute performance under the influence of factors.

However, to compare the impact of different factors on a single VO, we need an approach to normalize the relationship between the variations in a given factor and their influence on the performance metric (either pose error or visual error). For example, when we investigate the influence of the number of features, we have an ideal case with rich features and a challenging case with fewer features. When we

investigate the influence of camera motion distortion, we have an ideal case with synthetic motion and a challenging case with realistic motion. We need to unify the variation of the number of features and the one of the camera motion distortion.

To address this problem, we propose to use the difference between OITF and RITF as the normalizer of the performance difference between the ideal case and the challenging case. High values of OITF only require the error between adjacent frames to be small while high values of RITF require the error to be small for any two random frames. It is harder for VOs to achieve high values of RITF than OITF. As a result, the difference between OITF and RITF increases along with the variation of the influence factor between the ideal case and the challenging case.

Therefore, we compute the relative influence of factors as:

$$E_{factor}(i, CRF_j, (k)) = \frac{E(i, CRF_j, m) - E(i, CRF_j, n)}{\Delta ITF(i, CRF_j, n)}, \tag{22.2}$$

where $\Delta ITF(i, CRF_j, n)$ is computed as:

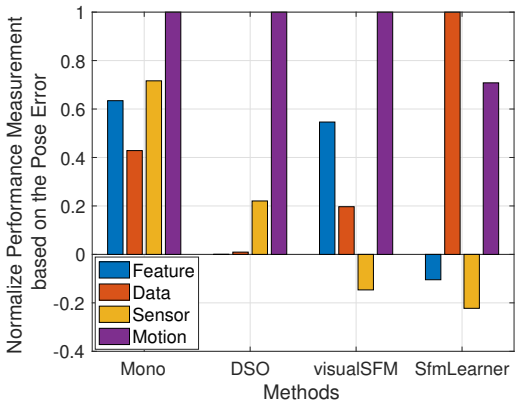
$$\Delta ITF(i, CRF_j, n) = OITF(i, CRF_j, n) - RITF(i, CRF_j, n). \tag{22.3}$$

For the pose error, a smaller value indicates better performance. For OITF and RITF, larger values indicate better performance. E_{factor} is a fractional value. To keep $E_{factor} > 0$, when computing the pose error, scene n corresponds to the ideal case. When computing the OITF/RITF, scene m corresponds to the ideal case.

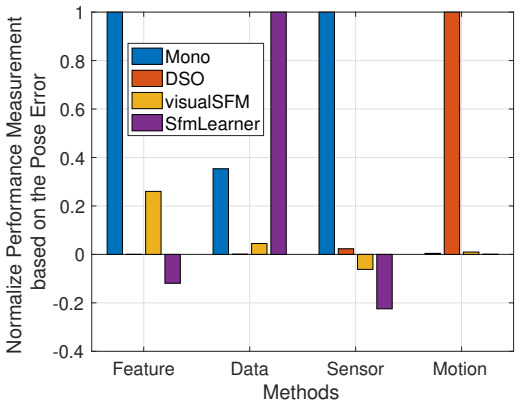
To compare the influence of different factors of a targeted VO, we further normalize $E_{factor}(i, CRF_j, (k))$ by dividing it using the maximum difference obtained when exploring any of the 5 factors:

$$E_{factor}(i, CRF_j, (k)) = \frac{E_{factor}(i, CRF_j, (k))}{|\max_k E_{factor}(i, CRF_j, (k))|}. \tag{22.4}$$

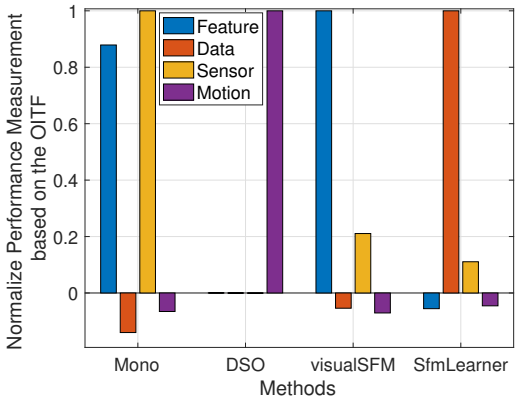
Since we have 3 measurements, we have 3 absolute performance matrices and 3 factor-based performance matrices.



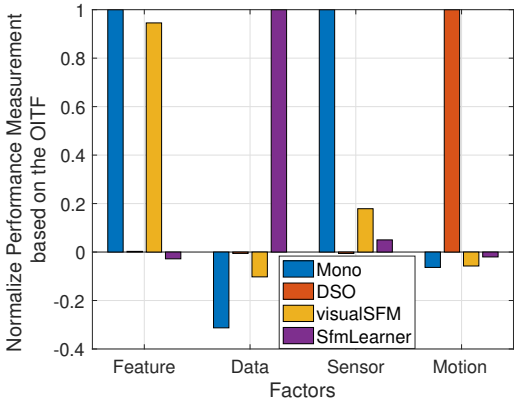
(a) Relative influence (pose error)



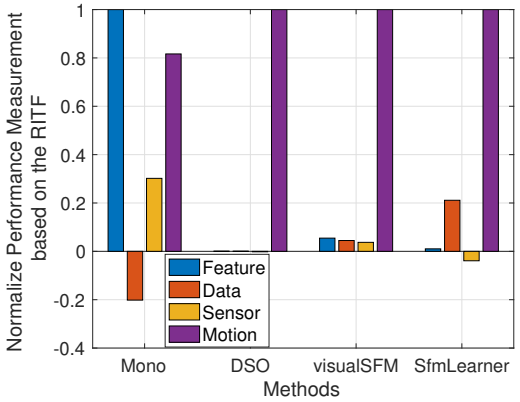
(b) Absolute influence (pose error)



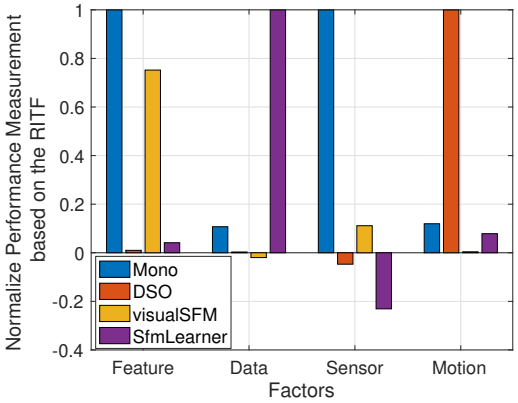
(c) Relative influence (OITF)



(d) Absolute influence (OITF)



(e) Relative influence (RITF)



(f) Absolute influence (RITF)

Fig. 22.2.: Resulting performance measurements with no compression

22.1.2 Absolute influence of factors

Another question users may be interested in is: which VO is the most sensitive to the current factor? The relative influence measurements only indicate whether one factor has higher or smaller influence relative to the other factors. To answer the new question, we consider the absolute changes of the measurements computed based on a concrete form of equation (22.1):

$$E_{factor}(i, CRF_j, (k)) = E(i, CRF_j, m) - E(i, CRF_j, n), \tag{22.5}$$

where the normalization is performed over different VOs with the same factor:

$$E_{factor}(i, CRF_j, (k)) = \frac{E_{factor}(i, CRF_j, (k))}{|\max_i E_{factor}(i, CRF_j, (k))|}. \tag{22.6}$$

22.1.3 Analysis of factor-based performance matrix

In Fig. 22.2 (a), (c) and (e), we show the relative performance measurements based on the pose error and the visual error with no compression. In Fig. 22.2 (b), (d) and (f), we show the corresponding absolute performance measurements. Fig. 22.2 (a), (c) and (e) help to investigate which factor influences the current VO the most. Fig. 22.2 (b), (d) and (f) are used to investigate which VO is the most sensitive to the current factor. In general, in Fig. 22.2, the larger the value is, the higher the influence from the factor is.

Note that when the factor has less influence on the VO, the measurements under the ideal case and the challenge case should be close. As a result, ideally all values in Fig. 22.2 should be greater than or equal to 0. However, there are still several negative values. This is because among those situations, the influence of factors other than current investigated one have not been completely disabled. For example, unless the input 2D images are the same, it is hard to completely disable the influence of feature numbers. Increasing the testing dataset may be helpful to remove this effect but currently, we focus on analyzing the positive signals.

We analyze Fig. 22.2 (a), (c) and (e) one VO at a time. Mono-VO is mainly influenced by the Feature and Sensor factor. One interesting point is that in the figure of OITF, the Motion factor is much smaller than in RITF and the pose error. This indicates that the main error caused by camera motion distortion of the Mono-VO is the estimation drift. For the video stabilization applications, the factor of camera motion distortion may not be critical.

The main influence factor for DSO is the camera motion distortion for all kinds of applications. As we discussed in Section 19.2, DSO relies on minimizing the photometric error between frames. If there is a sudden motion between frames, or there are rolling shutter distortions, DSO usually fails.

VisualSfM is also based on local feature matchings. However, the influence of the camera sensor distortion is relatively smaller than the one in Mono-VO. This may be due to the bundle adjustment. SfMLearner is a deep-learning-based method; Fig. 22.2 (a) (c) (e) indicate the Data factor is the main influence of its performance.

We analyze Fig. 22.2 (b), (d) and (f) factor by factor. It can be seen that the local-feature-based methods, Mono-VO and VisualSfM, are the most affected by the number of features, while DSO and SfMLearner exhibit nearly no influence. For factor (2), whether the scene is from a learned scenario or an excluded scenario, the SfMLearner is the most influenced. Note that there is a negative value for Mono-VO in the OITF figure. This may result because not all factors have been completely neutralized by our methodology. For factor (3), the camera sensor distortion, only the Mono-VO is affected. Again, the reason that VisualSfM is not influenced may be the existence of bundle adjustment. For the camera motion distortion, the direct-method, DSO is the most fragile.

Based on Fig. 22.2, users can draw some interesting conclusions for their scenario. To determine the optimal VO, it is ideal to know what the application is and what the major influence factor of the system is. For example, if we know the application is video stabilization, and the application scenario is a hallway with white walls, this

means the major factor is Feature, so we will focus on the Factor column in Fig. 22.2 (d). In this case, the optimal VO is the DSO.

However, consider the situation that we may not know the target application or we may not know the major factor. If we do not know the major factor and do know the target application, one strategy is to check which VO is influenced by the fewest factors. For example, if we assume the likelihood of different factors are equal, DSO is the best choice even if we do not know the target application. In addition, from the point of view of computer vision application system designers, if we know the application is augmented reality or robotic navigation, according to Fig. 22.2 (a) and (e), we may focus on preventing the camera motion distortion since it is a major problem for all VOs. If we do not know the target application but know what the major factor is, we may compare Fig. 22.2 (b), (d) and (f). For example, if the major factor is Feature, on average DSO is the most likely to have the best performance. If we can only choose from Mono-SLAM and VisualSfM, the better choice is VisualSfM.

22.2 Influence of compression rate

In addition to factors (1) to (4), we also have factor (5), which is the compression rates. We want to investigate how the influence of each factor (1) to (4) changes under compression.

We find there is no regular ordering pattern of the performance scores by varying the CRF, so we use the standard deviation to measure the fluctuation of the performance of a VO across different CRF. For each VO and each factor (1) to (4), we compute the standard deviation of their absolute performance across different compression rates. For method i under factor (k), the standard deviation of the measurement E_{factor} in equation (22.5) is computed. The value is redefined as Influence of Compression rate $IC_{i,(k)}$.

The resulting ICs for the pose error, OITF, and RITF, are shown in Tables 22.1 to 22.3, respectively. We prefer the method that has a low value for $IC_{i,(k)}$. The

Table 22.1.: Influence of compression rate on OITF

	Feature	Data	Sensor	Motion
Mono-VO	0.05293	0.00736	0.02862	0.00125
DSO	0.00099	0.00086	0.00014	0.22410
VisualSfM	0.03250	0.09120	0.01484	0.00546
SfMLearner	0.00027	0.00214	0.00070	0.00043

Table 22.2.: Influence of compression rate on RITF

	Feature	Data	Sensor	Motion
Mono-VO	0.01157	0.03851	0.03605	0.22373
DSO	0.00086	0.00022	0.00099	0.13416
VisualSfM	0.00202	0.00685	0.00823	0.04553
SfMLearner	0.00036	0.00082	0.00068	0.01697

Table 22.3.: Influence of compression rate on the pose error

	Feature	Data	Sensor	Motion
Mono-VO	0.26128	0.12951	0.05232	0.00181
DSO	0.00034	0.00033	0.00012	0.00459
VisualSfM	0.12961	0.02294	0.01704	0.03358
SfMLearner	0.01301	0.00787	0.00088	0.00053

larger the IC value is, the more influence compression rate has for the investigated factor. In other words, the compression effect makes a challenge situation even more challenging.

In general, across the three Tables, DSO is the most stable method under compression. All of its IC values are relatively lower than those of the other methods. VisualSfM is the most fragile method among itself, DSO and Mono-VO. There are two reasons VisualSfM is shown to be even worse than Mono-VO. First, without compression, the absolute performance of VisualSfM is better than Mono-VO, so its performance has more room to drop. Second, in order to do the bundle adjustment, local features are preferred to be seen across different frames. This may not be satisfied when compression increases.

Note that low values of IC need to be carefully treated. Among all 4 methods, the SfMLearner also has relatively small ICs. However, this is because it has the worst performance for motion estimation. Under compression, it is usually fully broken and yields useless motion estimation results.

23. CONCLUSION

In this work, we propose an evaluation methodology that can help users to investigate the performance of different VOs under various factors. The performance of VOs are measured for two main applications: robotic perception and visual augmentation. A pose error is proposed to measure the performance of VOs for the robotic perception applications, while Ordered-ITF and Random-ITF are proposed to measure the performance of VO for video stabilization and Augmented Reality applications, which both belong to visual augmentation applications.

We considered 5 factors here that influence the performance of VOs. These 5 factors are the richness of feature points in image frames, whether the scene content is consistent in the training set, camera sensor distortion and camera motion distortion. Furthermore, our methodology can also investigate how the compression rate influences the performance of a VO when the previous factors are present. Our methodology is designed to create comparisons in which 4 of these factors are held constant and only one is varied.

Two main questions can be answered by applying our methodology: (1) which factor influences the current VO the most, and (2) which VO is the most sensitive to the current factor. The first question helps users to find the bottleneck of the current VO. The second question provides suggestions when users want to select an appropriate VO given a main influence factor.

Part V

Summary

In this report, we present the current progress of our work on improving the utility of egocentric videos. We split the problem into 2 parts: improvement of viewing experience of entertainment videos and compression of documenting videos. The research subjects are First-Person Videos (FPVs) and DashCam Videos (DCVs) respectively.

The first part of this report (Chapter 1) focuses on the viewing experience model of FPVs. Our primary motivation is that FPVs are created to share experience and information but shakiness makes them uncomfortable to watch. The viewing experience model is proposed to improve the video stability while preserving the recorded experience and information unlike the existing video processing techniques. The model includes a perceptually-motivated viewing experience measurement which measures both the video stability and First-Person Motion Information (FPMI). The proposed measurement is based on a mathematical model of the Smooth Pursuit Eye Movement (SPEM), which human would perform when watching FPVs. The viewing experience model also includes a motion editing method, which uses the proposed measurement as the guidance to find the balance between video stability and FPMI in order to improve the overall viewing experience. We apply both objective and subjective tests on the viewing experience model. The tests show that viewing experience measurement is robust and effective; the motion editing method can effectively stabilize FPVs while preserving more FPMI than existing video processing techniques.

The second part of this report (Chapter 2) introduces our DashCam Video compression system. Our goal is to efficiently collect and compress these documenting videos. Our system jointly compresses incoming videos with historical DCVs that record the same route based on 3D-HEVC and two proposed preprocessors: video retrieval and video alignment preprocessors. The video retrieval preprocessor is performed to detect matching videos for the incoming DCV based on the GPS information and illumination condition. Given the matching videos, the video alignment preprocessor carefully creates a baseline video by assembling corresponding frames of incoming video frames that record the identical location based on local feature

matching. Experimental results show that our system achieves 30% more bit-savings with respect to the standard HEVC and outperforms other similar approaches.

The third part of this report (Chapter 3) works on an evaluation methodology for Visual Odometry. We propose an evaluation methodology that can help users to investigate the performance of different VOs under various factors. The system is especially designed for users to choose an optimal VO based on their applications. We use a pose error to measure the performance of VOs for the robotic perception applications. And two visual errors are proposed to measure the performance of VO for the visual augmentation applications. The influence factors include the richness of feature points in image frames, whether the scene content is covered in the training set, camera sensor distortion and camera motion distortion. Compared to traditional benchmarks, our methodology does not rely on a large dataset.

By the time we finished this thesis, the egocentric video processing has became one of the most popular topics in the industry. AR, vlog, and autonomous driving, either one of these may change and has changed our life. We hope our work contributes to our future.

REFERENCES

REFERENCES

[1] B. Ma and A. R. Reibman, "Enhancing viewability for First-Person Videos based on a human perception model," in *IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, 2017, pp. 1–6.

[2] B. Ma and A. R. Reibman, "Measuring and Improving the Viewing Experience of First-person Videos," in *ACM Multimedia Thematic Workshops*. ACM, 2017.

[3] B. Ma and A. R. Reibman, "Estimating the Subjective Video Stability of First-Person Videos," *Electronic Imaging*, vol. 2018, no. 14, pp. 1–7, 2018.

[4] K.-Y. Lee, Y.-Y. Chuang, B.-Y. Chen, and M. Ouhyoung, "Video stabilization using robust feature trajectories," in *IEEE International Conference on Computer Vision*, 2009, pp. 1397–1404.

[5] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust L1 optimal camera paths," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 225–232.

[6] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, "Subspace video stabilization," *ACM Transactions on Graphics*, vol. 30, no. 1, p. 4, 2011.

[7] H. Qu and L. Song, "Video stabilization with L1–L2 optimization," in *IEEE International Conference on Image Processing*, 2013, pp. 29–33.

[8] J. Kopf, M. F. Cohen, and R. Szeliski, "First-person hyper-lapse videos," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 78, 2014.

[9] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," *ACM Transactions on Graphics*, vol. 28, no. 3, p. 44, 2009.

[10] G. Zhang, W. Hua, X. Qin, Y. Shao, and H. Bao, "Video stabilization based on a 3D perspective camera model," *The Visual Computer*, vol. 25, no. 11, pp. 997–1008, 2009.

[11] E. Ringaby and P.-E. Forssén, "Efficient video rectification and stabilisation for cell-phones," *International Journal of Computer Vision*, vol. 96, no. 3, pp. 335–352, 2012.

[12] J. L. Carrivick, M. W. Smith, and D. J. Quincey, "Background to Structure from Motion," *Structure from Motion in the Geosciences*, pp. 37–59.

[13] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015.

[14] C. Jia and B. L. Evans, "Online motion smoothing for video stabilization via constrained multiple-model estimation," *EURASIP Journal on Image and Video Processing*, vol. 2017, no. 1, p. 25, 2017.

[15] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, "Full-frame video stabilization with motion inpainting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 7, pp. 1150–1163, 2006.

[16] Z. Wang and H. Huang, "Pixel-wise video stabilization," *Multimedia Tools and Applications*, vol. 75, no. 23, pp. 15 939–15 954, 2016.

[17] H.-C. Chang, S.-H. Lai, and K.-R. Lu, "A robust and efficient video stabilization algorithm," in *IEEE International Conference on Multimedia and Expo*, vol. 1, 2004, pp. 29–32.

[18] T. Mei, X.-S. Hua, C.-Z. Zhu, H.-Q. Zhou, and S. Li, "Home video visual quality assessment with spatiotemporal factors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 6, pp. 699–706, 2007.

[19] Y. Hoshen, G. Ben-Artzi, and S. Peleg, "Wisdom of the crowd in egocentric video curation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 573–579.

[20] L. Marcenaro, G. Vernazza, and C. S. Regazzoni, "Image stabilization algorithms for video-surveillance applications," in *IEEE International Conference on Image Processing*, vol. 1, 2001, pp. 349–352.

[21] K. M. Alam, M. Saini, D. T. Ahmed, and A. El Saddik, "VeDi: A vehicular crowd-sourced video social network for VANETs," in *IEEE 39th Conference on Local Computer Networks Workshops (LCN Workshops)*, 2014, pp. 738–745.

[22] M. K. Saini, R. Gadde, S. Yan, and W. T. Ooi, "Movimash: online mobile video mashup," in *Proceedings of the 20th International Conference on Multimedia*. ACM, 2012, pp. 139–148.

[23] M. Tanakian, M. Rezaei, and F. Mohanna, "Camera motion modeling for video stabilization performance assessment," in *Machine Vision and Image Processing (MVIP)*. IEEE, 2011, pp. 1–4.

[24] Z. Cui and T. Jiang, "No-reference video shakiness quality assessment," in *Asian Conference on Computer Vision*. Springer, 2016, pp. 396–411.

[25] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool, "Creating summaries from user videos," in *European Conference on Computer Vision*. Springer, 2014, pp. 505–520.

[26] Y. Poleg, T. Halperin, C. Arora, and S. Peleg, "Egosampling: Fast-forward and stereo for egocentric videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4768–4776.

[27] M. M. Silva, W. L. S. Ramos, J. P. K. Ferreira, M. F. M. Campos, and E. R. Nascimento, "Towards semantic fast-forward and stabilized egocentric videos," in *European Conference on Computer Vision*. Springer, 2016, pp. 557–571.

[28] S. Aw, G. Halmagyi, T. Haslwanter, I. Curthoys, R. Yavor, and M. Todd, "Three-dimensional vector analysis of the human vestibuloocular reflex in response to high-acceleration head rotations. II. Responses in subjects with unilateral vestibular loss and selective semicircular canal occlusion," *Journal of Neurophysiology*, vol. 76, no. 6, pp. 4021–4030, 1996.

[29] S. de Brouwer, M. Missal, G. Barnes, and P. Lefèvre, "Quantitative analysis of catch-up saccades during sustained pursuit," *Journal of Neurophysiology*, vol. 87, no. 4, pp. 1772–1780, 2002.

[30] S. De Brouwer, D. Yuksel, G. Blohm, M. Missal, and P. Lefèvre, "What triggers catch-up saccades during visual tracking?" *Journal of Neurophysiology*, vol. 87, no. 3, pp. 1646–1650, 2002.

[31] E. Kowler, "Eye movements: The past 25 years," *Vision Research*, vol. 51, no. 13, pp. 1457–1483, 2011.

[32] B. R. Beutter and L. S. Stone, "Human motion perception and smooth eye movements slow similar directional biases for elongated apertures," *Vision Research*, vol. 38, no. 9, pp. 1273–1286, 1998.

[33] B. R. Beutter and L. S. Stone, "Motion coherence affects human perception and pursuit similarly," *Visual Neuroscience*, vol. 17, no. 01, pp. 139–153, 2000.

[34] A. T. Bahill and J. D. McDonald, "Smooth pursuit eye movements in response to predictable target motions," *Vision research*, vol. 23, no. 12, pp. 1573–1583, 1983.

[35] G. Rotman, N. F. Troje, R. S. Johansson, and J. R. Flanagan, "Eye movements when observing predictable and unpredictable actions," *Journal of Neurophysiology*, vol. 96, no. 3, pp. 1358–1369, 2006.

[36] S. Foley, "Camera movement in First Person Games," Ph.D. dissertation, Worcester Polytechnic Institute, 2010.

[37] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *European Conference on Computer Vision*. Springer, 2006, pp. 404–417.

[38] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[39] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert, "Initialization techniques for 3D SLAM: a survey on rotation estimation and its use in pose graph optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4597–4604.

[40] D. Purves, G. J. Augustine, D. Fitzpatrick, L. C. Katz, A.-S. LaMantia, J. O. McNamara, and S. M. Williams, "Types of eye movements and their functions," *Neuroscience*, pp. 361–390, 2001.

[41] S. M. LaValle, "Virtual reality," *Champaign (IL): University of Illinois*, 2016.

[42] R. Penner, "Motion, tweening, and easing," *Programming Macromedia Flash MX*, pp. 191–240, 2002.

[43] L. Izdebski and D. Sawicki, "Easing functions in the new form based on bézier curves," in *International Conference on Computer Vision and Graphics*. Springer, 2016, pp. 37–48.

[44] Kolor, "Autopano video." [Online]. Available: http://www.kolor.com

[45] A. Thorn and M. S. arer, *Pro Unity Game Development with C#*. Springer, 2014.

[46] N. Joshi, W. Kienzle, M. Toelle, M. Uyttendaele, and M. F. Cohen, "Real-time hyperlapse creation via optimal frame selection," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 63, 2015.

[47] J. C. Handley, "Comparative analysis of bradley-terry and thurstone-mosteller paired comparison models for image quality assessment," in *PICS*, vol. 1, 2001, pp. 108–112.

[48] A. M. Demirtas, A. R. Reibman, and H. Jafarkhani, "Full-reference quality estimation for images with different spatial resolutions," *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2069–2080, 2014.

[49] A. Leontaris, P. C. Cosman, and A. R. Reibman, "Quality evaluation of motion-compensated edge artifacts in compressed video," *IEEE Transactions on Image Processing*, vol. 16, no. 4, pp. 943–956, 2007.

[50] "Test-set of enhancing viewability of FPVs," https://engineering.purdue.edu/VADL/resources/Enhancing_Viewability/testset_for_enhancingFPV.zip.

[51] G. Thalin, "Deshaker–video stabilizer," *Online at: http://guthspot.se/video/deshaker.htm*, 2014.

[52] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[53] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives," *Robotics and Autonomous systems*, vol. 32, no. 1, pp. 1–16, 2000.

[54] L. Wei, C. Cappelle, Y. Ruichek, and F. Zann, "GPS and stereovision-based visual odometry: application to urban scene mapping and intelligent vehicle localization," *International Journal of Vehicular Technology*, vol. 2011, 2011.

[55] S. Nedevschi, V. Popescu, R. Danescu, T. Marita, and F. Oniga, "Accurate ego-vehicle global localization at intersections through alignment of visual data with digital map," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 673–687, 2013.

[56] M. Bertozzi, L. Castangia, S. Cattani, A. Prioletti, and P. Versari, "360° Detection and tracking algorithm of both pedestrian and vehicle using fisheye images," in *IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 132–137.

[57] C.-Y. Chiang, S.-M. Yuan, S.-B. Yang, G.-H. Luo, and Y.-L. Chen, "Vehicle driving video sharing and search framework based on GPS data," in *Genetic and Evolutionary Computing*. Springer, 2014, pp. 389–397.

[58] M. Faezipour, M. Nourani, A. Saeed, and S. Addepalli, "Progress and challenges in intelligent vehicle area networks," *Communications of the ACM*, vol. 55, no. 2, pp. 90–100, 2012.

[59] G. Tech *et al.*, "MV-HEVC Draft Text 6. Doc. JCTVC-F1004," in *6th meeting: Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V) of ITU-T SG*, vol. 16.

[60] R. Zou, O. C. Au, G. Zhou, W. Dai, W. Hu, and P. Wan, "Personal photo album compression and management," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2013, pp. 1428–1431.

[61] Z. Shi, X. Sun, and F. Wu, "Photo album compression for cloud storage using local features," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 4, no. 1, pp. 17–28, 2014.

[62] J. Liu, Z. Huang, H. Cai, H. T. Shen, C. W. Ngo, and W. Wang, "Near-duplicate video retrieval: Current research and future trends," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, p. 44, 2013.

[63] S.-C. S. Cheung and A. Zakhor, "Fast similarity search and clustering of video sequences on the world-wide-web," *IEEE Transactions on Multimedia*, vol. 7, no. 3, pp. 524–537, 2005.

[64] L. Shang, L. Yang, F. Wang, K.-P. Chan, and X.-S. Hua, "Real-time large scale near-duplicate web video retrieval," in *Proceedings of the International Conference on Multimedia*, 2010, pp. 531–540.

[65] H. T. Shen, X. Zhou, Z. Huang, J. Shao, and X. Zhou, "UQLIPS: a real-time near-duplicate video clip detection system," in *Proceedings of the 33rd International Conference on Very Large Databases*, 2007, pp. 1374–1377.

[66] C.-L. Chou, H.-T. Chen, and S.-Y. Lee, "Pattern-Based Near-Duplicate Video Retrieval and Localization on Web-Scale Videos," *IEEE Transactions on Multimedia*, vol. 17, no. 3, pp. 382–395, 2015.

[67] C.-L. Chou, H.-T. Chen, C.-C. Hsu, C.-P. Ho, and S.-Y. Lee, "Near-duplicate video retrieval by using pattern-based Prefix tree and temporal relation forest," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2014, pp. 1–6.

[68] H. Wang, M. Ma, and T. Tian, "Effectively compressing Near-Duplicate Videos in a joint way," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2015, pp. 1–6.

[69] F. Diego, D. Ponsa, J. Serrat, and A. M. López, "Video alignment for change detection," *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 1858–1869, 2011.

[70] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 1643–1649.

[71] B. Ma and A. R. Reibman, "Dashcam video compression using historical data," in *Picture Coding Symposium (PCS)*. IEEE, 2016, pp. 1–5.

[72] G. J. Sullivan, J. M. Boyce, Y. Chen, J.-R. Ohm, C. A. Segall, and A. Vetro, "Standardized extensions of high efficiency video coding (HEVC)," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 1001–1016, 2013.

[73] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[74] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision*, 2011.

[75] S. Yea and A. Vetro, "View synthesis prediction for multiview video coding," *Signal Processing: Image Communication*, vol. 24, no. 1, pp. 89–100, 2009.

[76] H. Liu, J. Jung, J. Sung, J. Jia, and S. Yea, "3D-CE2. h: Results of illumination compensation for inter-view prediction," *ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCT3V-B0045*, 2012.

[77] I. Motoyoshi, S. Nishida, L. Sharan, and E. H. Adelson, "Image statistics and the perception of surface qualities," *Nature*, vol. 447, no. 7141, pp. 206–209, 2007.

[78] B. L. Anderson and J. Kim, "Image statistics do not explain the perception of gloss and lightness," *Journal of Vision*, vol. 9, no. 11, p. 10, 2009.

[79] M. Olkkonen and D. H. Brainard, "Perceived glossiness and lightness under real-world illumination," *Journal of Vision*, vol. 10, no. 9, p. 5, 2010.

[80] H. Tamura, S. Mori, and T. Yamawaki, "Textural features corresponding to visual perception," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 8, no. 6, pp. 460–473, 1978.

[81] G. Bjøntegaard, "Calcuation of average PSNR differences between RD-curves," *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4 April 2001*.

[82] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[83] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[84] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *Journal of Intelligent and Robotic Systems*, vol. 53, no. 3, pp. 263–296, 2008.

[85] S. U. Kamat and K. Rasane, "A Survey on Autonomous Navigation Techniques," in *2018 IEEE Second International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, 2018, pp. 1–6.

[86] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 1–10.

[87] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE international conference on robotics and automation (ICRA)*, 2014, pp. 15–22.

[88] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[89] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[90] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of SLAM algorithms," *Autonomous Robots*, vol. 27, no. 4, p. 387, 2009.

[91] Z. Fang and Y. Zhang, "Experimental evaluation of RGB-D visual odometry methods," *International Journal of Advanced Robotic Systems*, vol. 12, no. 3, p. 26, 2015.

[92] H. Alismail, B. Browning, and M. B. Dias, "Evaluating pose estimation methods for stereo visual odometry on robots," in *the 11th International Conference on Intelligent Autonomous Systems (IAS-11)*, vol. 3, 2010, p. 2.

[93] T. A. Ciarfuglia, G. Costante, P. Valigi, and E. Ricci, "Evaluation of non-geometric methods for visual odometry," *Robotics and Autonomous Systems*, vol. 62, no. 12, pp. 1717–1730, 2014.

[94] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. G. Sorrenti, and P. Taddei, "Rawseeds ground truth collection systems for indoor self-localization and mapping," *Autonomous Robots*, vol. 27, no. 4, p. 353, 2009.

[95] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Asian Conference on Computer Vision*. Springer, 2010, pp. 25–38.

[96] H. Ovrén and P.-E. Forssén, "Gyroscope-based video stabilisation with auto-calibration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2090–2097.

[97] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H. 264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[98] D.-Y. Koh, Y. K. Kim, K.-S. Kim, and S. Kim, "Bioinspired image stabilization control using the adaptive gain adjustment scheme of vestibulo-ocular reflex," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 2, pp. 922–930, 2016.

[99] M. Favorskaya and V. Buryachenko, "Fuzzy-based digital video stabilization in static scenes," in *Intelligent Interactive Multimedia Systems and Services in Practice*. Springer, 2015, pp. 63–83.

[100] W. Hong, D. Wei, and A. U. Batur, "Video stabilization and rolling shutter distortion reduction," in *IEEE International Conference on Image Processing (ICIP)*, 2010, pp. 3501–3504.

[101] K. L. Veon, M. H. Mahoor, and R. M. Voyles, "Video stabilization using SIFT-ME features and fuzzy clustering," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 2377–2382.

[102] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., pp. 674–679.

[103] C. Wu, "Towards linear-time incremental structure from motion," in *IEEE International Conference on 3D Vision*, 2013, pp. 127–134.

[104] A. Singh and K. Venkatesh, "Monocular Visual Odometry," *Undergraduate Project of Indian Institute of Technology Kanpur*, vol. 2, 2015.

VITA

VITA

Biao Ma was born in Beijing, China. He received his Bachelor Degree in Automatic Control from Beijing Institute of Technology, China, in 2014. Then he joined the Master program at School of Electrical and Computer Engineering, Purdue University (West Lafayette). After the first semester, in 2015, he transferred to the Ph. D. program under the advice of Prof. Amy R. Reibman. At the same year, Video Analytics for Daily Living (VADL) lab was founded. Mr. Ma was also the designer of the logo of VADL lab. His research interests are image processing, computer vision and deep learning. His main contribution during his Ph.D. is the viewing experience model of First-Person Videos.