Panoramic Video Stitching

by

Wei Xu

B.S., Wuhan University, 2000

M.E., Wuhan University, 2003

M.Sc., York University, 2005

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2012

This thesis entitled:
Panoramic Video Stitching
written by Wei Xu
has been approved for the Department of Computer Science

Jane Mulligan

Henry M. Tufo

Roger A.(Buzz) King

Nikolaus J. Correll

Willem A.(Vlakkies) Schreüder

Min-Hyung Choi

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Xu, Wei (Ph.D., Computer Science)

Panoramic Video Stitching

Thesis directed by Prof. Jane Mulligan

Digital camera and smartphone technologies have made high quality images and video pervasive and abundant. Combining or stitching collections of images from a variety of viewpoints into an extended panoramic image is a common and popular function for such devices. Extending this functionality to video however, poses many new challenges due to the demand for both spatial and temporal continuity. Multi-view video stitching (also called panoramic video stitching) is an emerging, common research area in computer vision, image/video processing and computer graphics and has wide applications in virtual reality, virtual tourism, surveillance, and human computer interaction. In this thesis, I will explore the technical and practical problems in the complete process of stitching a high-resolution multi-view video into a high-resolution panoramic video. The challenges addressed include video stabilization, efficient multi-view video alignment and panoramic video stitching, color correction, and blurred frame detection and repair.

Specifically, I propose a continuity aware Kalman filtering scheme for rotation angles for video stabilization and jitter removal. For efficient stitching of long, high-resolution panoramic videos, I propose constrained and multi-grid SIFT matching schemes, concatenated image projection and warping and min-space feathering. These three approaches together can greatly reduce the computational time and memory requirement in panoramic video stitching, which makes it feasible to stitch high-resolution (e.g., 1920x1080 pixels) and long panoramic video sequences using standard workstations.

Color correction is the emphasis of my research. On this topic I first performed a systematic survey and performance evaluation of nine state of the art color correction approaches in the context of two-view image stitching. My evaluation work not only gives useful insights and conclusions about the relative performance of these approaches, but also points out the remaining challenges and possible directions for future color correction research. Based on the conclusions from this evaluation work, I proposed a hybrid and scalable color correction approach for general n-view image stitching, and designed a two-view video color correction approach for panoramic video stitching.

For blurred frame detection and repair, I have completed preliminary work on image partial blur detection and classification, in which I proposed a SVM-based blur block classifier using improved and new local blur features. Then,

based on partial blur classification results, I designed a statistical thresholding scheme for blurred frame identification. For the detected blurred frames, I repaired them using polynomial data fitting from neighboring unblurred frames.

Many of the techniques and ideas in this thesis are novel and general solutions to the technical or practical problems in panoramic video stitching. At the end of this thesis, I conclude the contributions made by this thesis to the research and popularization of panoramic video stitching, and describe those open research issues.

Dedication

To my parents.

Acknowledgements

I would like to thank my supervisor, Prof. Jane Mulligan, for her kind support and advice during the course of my pursuing the PhD degree. I would also like to thank my colleague, Jaeheon Jeong, for his friendship during the course of my PhD study.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

Multi-view image stitching, also called image mosaicking or panorama stitching, is a basic and long-studied problem of computer vision, with applications in robotics, architecture, industrial inspection, surveillance, computer graphics, film and mobile devices. After decades of development, the techniques have become mature and there exist many successful panorama stitching programs such as Microsoft Photosynth [95], AutoStitch [10] and Nokia Panorama [102]. Figure 1.1 shows a set of overlapping images and a panorama stitched from them using the AutoStitch software. In this example, there are six input images each of which captures a part of the scene, and the panorama stitching software merges these into a panorama with a much wider field of view.

(a) Input images.

(b) Stitched panorama.

Figure 1.1: An example panorama stitched from six overlapping images of the same scene.

The goal of multi-view image stitching is to generate a panoramic image from multiple overlapping images of the same scene, possibly taken at (slightly) displaced locations. "Multi-view" means the camera moves to different 3D poses (i.e., with different 3D locations or orientations) while taking these images, thus the captured images are of

different views of the same scene. An important assumption adopted by most image stitching programs is the "flat scene" assumption [53] which means the distance between the scene and the camera is relatively much larger than the distance between different capture locations (viewpoints) of the camera. Under this assumption, because the scene is relatively "flat" the images can be directly stitched together in a 2D plane, and complex 3D back-projection and reconstruction of scene points are avoided.

Although multi-view image stitching has been a focus of interest in the computer vision community, much less attention has been paid to the closely related topic of *multi-view video stitching* (also called panoramic video stitching). The goal of multi-view video stitching is to generate a panoramic video from multiple overlapping video streams captured by relatively (slightly) displaced cameras. When capturing a multi-view video, the cameras can be fixed or moving while they are capturing the videos, but their relative geometry is unchanged during the capture procedure. The input to a multi-view video stitching program is multiple, overlapping videos captured from different camera views which are together called a multi-view video, the output is a panoramic video of wider field of view constructed by merging and stitching the individual videos of the input multi-view video.

There is a misunderstanding that multi-view video stitching is just stitching the frames of multiple overlapping videos captured by different cameras using well known multi-view image stitching techniques. As I will show in this thesis, the solution to the problem is actually not that straightforward. Multi-view video stitching shares many of the basic challenges of multi-view image stitching like frame alignment and color correction. However, it is also distinguished by problems arising from its temporal properties such as video synchronization and video stabilization, which are interesting computer vision problems by themselves. For multi-view video stitching, some of the sub-problems such as frame alignment (i.e., 2D relative geometry registration) can be solved more robustly using multiple frame correspondences from the temporal sequence. In addition, because the input multi-view video may be captured by a mobile device, one still faces video processing problems such as jitter removal for the multiple component videos. In multi-view video stitching, the temporal video processing problems are usually compounded with spatial, multi-view image stitching problems, resulting in even greater challenges.

Although multi-view video stitching has wide applications in computer graphics, virtual reality, surveillance, human computer interaction etc., mature and successful multi-view video stitching systems are rare. One example of successful multi-view video stitching systems is the Google Street View [52], which actually uses specially-designed,

well-calibrated, professional-level camera heads, the R2, R5 and R7 camera heads [3], to capture the multi-view video for a street environment. Because the cameras in these custom camera heads have a fixed relatively geometry and are synchronized for capturing, many challenging problems such as 2D relative geometry estimation and video synchronization are engineered or avoided. Also note the cameras composing these camera heads are static CMOS cameras rather than video cameras, so the multi-view videos are captured at pretty low sampling rates, which cause the Google Street View videos are not as continuous as regular videos are.

The ladybug2 [54] camera head is another commercial multi-view video capturing device developed and sold by Point Grey Research. This device integrates five firewire cameras in a camera head with each camera being able to capture videos of 1024x768 resolution at 30 frames per second (FPS). The provided SDK can stitch the captured multi-view videos into a spherical panoramic video quickly. Like the R2, R5 and R7 cameras, the cameras in the ladybug2 have a fixed relative geometry and are synchronized for capturing. The ladybug2 is also a relatively expensive, professional device that is sold at over $10,000 per device.

Commercial multi-view video stitching systems are usually very expensive and unaffordable to the general public. This hinders the popularization of multi-view video stitching systems and applications. Nowadays, with high-resolution commodity cameras becoming cheaper and more popular, the demand for stitching high-resolution panoramic video from multi-view video is increasing. To the best of my knowledge, there are no multi-view video stitching systems that use casually assembled commodity cameras – flexibility in assembling the capture device requires much greater effort in the post-capture video processing stage. From a research point of view, there are many unsolved problems and technical issues in multi-view video stitching especially with multi-view videos captured by common devices such as camcorders (e.g, the 3 camera rig shown in Fig. 3.8), including video synchronization, automatic video synchronization, efficient multi-view video alignment and stitching, color correction, and blurred frame detection and repair.

Next, I will introduce the unique problems faced by a general panoramic video stitching system. I will also introduce my solutions or designs of potential solutions to most of these problems as well as discuss related open research issues.

1.1 Problems and Solutions

Fig. 1.2 shows the general procedure of stitching a multi-view video into a panoramic video. There are five steps in the procedure: video pre-processing, multi-view video alignment, color correction, panoramic video stitching and display area extraction.

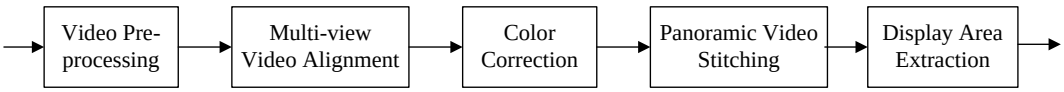


Figure 1.2: Procedure of multi-view video stitching.

This procedure poses five major research problems. They are respectively:

Video stabilization Because the video capture device may be moving while it records multi-view videos, motion jitter (mainly caused by uneven road surfaces and vibration of the capture device) usually exists in the captured videos. The video processing technique that uses various data smoothing techniques to remove this jitter is called *video stabilization*. Video stabilization by itself is a long-studied computer vision problem but as yet there does not exist an ideal solution.

Video synchronization The cameras of a multi-view video capture device can be synchronized or unsynchronized. Professional capture rigs (e.g., the Ladybug2 camera head) are synchronized but such devices are usually very expensive. As high-definition commodity cameras become more and more popular the demand to be able to stitch videos from cameras mounted in less formal configurations arises. This highlights the synchronization problem for individual videos captured by different component cameras. Such synchronization can be done manually, by tracking and matching the motion track of a target object (e.g., a yellow tennis ball). However, ideally synchronization would be done automatically by tracking and matching the trajectories of prominent spatio-temporal image features.

Efficient large-size multi-view video alignment and panoramic video stitching With high-definition and large storage cameras becoming more and more popular, the efficiency of stitching high-definition (e.g., input images are of 1920x1080 pixel resolution) and long duration panoramic videos has become a practical issue. There are two steps in the stitching procedure of Fig. 1.2 whose execution speed depend on video resolu-

tion: multi-view video alignment and panoramic video stitching. My experiments show traditional image manipulation algorithms for these two steps cannot satisfy the speed and memory requirements for stitching a high-definition and long panoramic video. More efficient algorithms must be developed for practical consideration.

Color correction In practice, the different view angles and exposure levels set by different cameras for capturing the multi-view video usually cause some of the constituent views in a panoramic frame to look quite different in color and brightness levels. If such color and brightness differences are not corrected before stitching, then the resulting panorama would look incoherent and unnatural. Color correction thus is an indispensable step in panoramic image and video stitching and worthy of research efforts.

Blurred frame detection and repair Motion blur can easily occur in recorded frames during multi-view video capture due to rapid motion of objects in the scene or rapid rig motion especially due to uneven road surface or device shaking. Blurred frames can make the above vision problems become even harder because it represents unreliable image information. The blurred frames in a multi-view video must be detected and repaired before they are used for the vision tasks listed above.

I will address most of the above research problems in detail in the following chapters of this thesis.

Chapter 2

Video Stabilization

2.1 Introduction and background

Video stabilization is a technique to remove jitter by smoothing the temporal neighborhood of the frames in a video. Video cameras which are handheld or on vehicles without physical anti-shake devices will capture multi-view videos which contain jitter. Video stabilization thus can be performed as a pre-processing technique for removing high-frequency jitter due to device shake before the video frames are aligned and stitched together.

There are typically three major stages constituting a video stabilization process: camera motion estimation, motion smoothing, and image warping. The video stabilization algorithms can be distinguished by the methods adopted in these stages. Video stabilization is achieved by first estimating the inter-frame motion of adjacent frames. The inter-frame motion describes the image motion which is also called global motion. An approach needs to assume a camera motion model for inter-frame motion, which is represented in the form of 2D [1] or 3D geometric transforms. Widely used motion models include 2D linear, affine or projective transforms (e.g., [56, 15, 35]), and full 3D motion models (e.g., [140]).

The accuracy of the global motion estimation is crucial as the first step of the stabilization. As concluded by [127], there are two major approaches for global motion estimation. One is the feature-based approach (e.g., [23, 61]), the other is the global intensity alignment approach (e.g., [8, 165]). Feature-based methods are generally faster than global intensity alignment approaches, while they are more prone to local effects. They work well when the assumed camera motion model is correct; however, it is preferable to use a more general camera motion model since there exist many situations where camera motion cannot be approximated by such simple models, e.g., handheld camera motion.

[1] An introduction to 2D planar transformations can be found in Sec.3.2.2

Estimated frame-to-frame global motion transforms for the video to be stabilized are chained to form a global transform chain. The second step is removing the irregular perturbations (i.e., the jitter) by smoothing the computed global motion chain. Litvin et al. [79] proposed to use the Kalman filter to smooth the general camera motion path. Pilu [106] proposed an optimal motion path smoothing algorithm with constraints imposed by a finite image sensor size. Matsushita et al. [93] used a Gaussian weighted function to smooth the global transformation chain in a temporal neighborhood of frames. Xu et al. [155] constructed a weak Gaussian smoother to smooth the global motion chain iteratively for adaptive smoothing. Choi et al. [30] again used the Kalman filter to smooth the global motion chain but specially developed an adaptive RANSAC scheme for outlier removal.

The last step is image completion or image compensation. In comparison to the original frames, the stabilized frames usually require some image warping operations based on the smoothed motion chain. However, such image warping may result in missing areas in the stabilized frames. Filling in missing image areas in a video is called video completion. There exist many video completion techniques. Specifically, in [79] mosaicking is used to fill up the missing image areas in the context of video stabilization. Wexler et al. [142] filled in the holes in a video by sampling spatial-temporal volume patches from different portions of the same video. Jia et al. [64] solved the problem by segmenting the video into two layers, i.e., a moving object layer and a static background layer, and complete the moving object layer according to periodic motion pattern. Another work that makes use of the periodic motion pattern is by Cheung et al.'s video epitomes framework [29] which requires a similar video patches in the different portions of the same video. More recently, Matsushita et al. [93] proposed the motion inpainting scheme which extrapolates the spatial-temporal dense motion field in a video to complete the missing areas which has become a milestone work on this topic. More recently, Hu et al. [61] used dynamic programming to fill up the missing areas in the stabilized frames.

2.2 Continuity aware Kalman filtering of rotation angles

2.2.1 The Kalman filter

The Kalman filter is an algorithm which operates recursively on streams of noisy input data to produce a statistically optimal estimate of the underlying system state. Kalman filters are based on linear dynamic systems

discretized in the time domain. They are modeled on a Markov chain built on linear operators perturbed by Gaussian noise. The state of the system is represented as a vector of real numbers. At each discrete time increment, a linear operator is applied to the state to generate the new state, with some noise mixed in, and optionally some information from the controls on the system if they are known. Specifically, the Kalman filter model assumes the true state at time k is evolved from the state at $(k-1)$ according to [123]:

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \tag{2.1}$$

where $\mathbf{F}_k$ is the state transition model which is applied to the previous state $\mathbf{x}_{k-1}$; $\mathbf{B}_k$ is the control-input model which is applied to the control vector $\mathbf{u}_k$; $\mathbf{w}_k$ is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance $\mathbf{Q}_k$: $\mathbf{w}_k \sim N(0, \mathbf{Q}_k)$. At time k an observation (or measurement) $\mathbf{z}_k$ of the true state $\mathbf{x}_k$ is made according to

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{2.2}$$

where $\mathbf{H}_k$ is the observation model which maps the true state space into the observed space and $\mathbf{v}_k$ is the observation noise which is assumed to be zero mean Gaussian white noise with covariance $\mathbf{R}_k$: $\mathbf{v}_k \sim N(0, \mathbf{R}_k)$. The initial state, and the noise vectors at each step $\mathbf{x}_0, \mathbf{w}_1, \cdots, \mathbf{w}_k, \mathbf{v}_1 \cdots \mathbf{v}_k$ are all assumed to be mutually independent.

Given the above state transition model, the algorithm works in a two-step process: "prediction" and "update". In the prediction step, the Kalman filter produces estimates of the true unknown values, along with their uncertainties. Once the outcome of the next measurement is observed, these estimates are updated using a weighted average, with more weight being given to estimates with higher certainty. This method produces estimates that tend to be closer to the true unknown values than those that would be based on a single measurement alone or the model predictions alone. Specifically, the "prediction" step includes the following mathematical operations:

$$\text{Predicted state estimate:} \quad \hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k \tag{2.3}$$

$$\text{Predicted estimate covariance:} \quad \mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k \tag{2.4}$$

the "update" step includes the following mathematical operations:

$$\text{Measurement residual:} \quad \tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \tag{2.5}$$

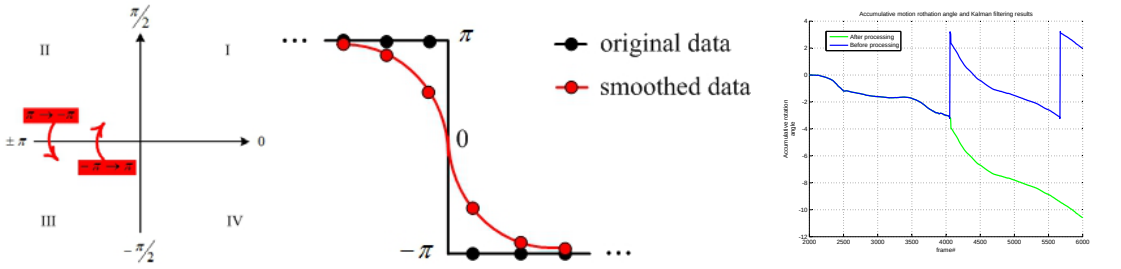
$$\text{Residual covariance:} \quad \mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k \tag{2.6}$$

$$\text{Updated state estimate:} \quad \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \tag{2.7}$$

$$\text{Updated estimate covariance:} \quad \mathbf{P}_{k|k} = (I - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_{k|k-1} \tag{2.8}$$

The "prediction" and "update" steps alternate, with the prediction advancing the state until the next scheduled observation, and the update incorporating the observation. Note the formula for the updated estimate and covariance above is only valid for the optimal Kalman gain.

2.2.2 Continuity aware Kalman filtering of rotation angles



(a) Problematic θ changes (b) Kalman filtering effect on problematic θ change (c) Real data (frames 2000 to 6000 of the right view)

Figure 2.1: Rotation angle temporal continuity needs to be protected during Kalman filter-based smoothing. (a) and (b) show the cause of the problem. (c) shows the rotation angle array for frames 2000 to 6000 of a video sequence from the right view. There are two sudden value falls (near frames 4050 and 5665) in the unprocessed array due to angle value conversion into the $(-\pi, \pi]$ range, which breaks the continuity of the sequence.

Kalman filter-based smoothing is one of the most popular choices for global motion chain smoothing (e.g. [79,

(a) Smoothed sequence without continuity protection

(b) Smoothed sequence with continuity protection

Figure 2.2: Rotation angle temporal continuity (a) shows the Kalman filter smoothed frames near frame 4050 as described in Figure 2.1 above. (b) After processing with Algorithm 1, this problem is solved.

30]). The standard procedure of using Kalman filter-based smoothing for video stabilization works as follows: first, it assumes there is a similarity transform relationship between two succeeding frames I_t and I_{t+1}:

$$\tilde{X}_{t+1} = S\tilde{X}_t \tag{2.9}$$

where $\tilde{X}_t = [x_i^{(t)} \quad y_i^{(t)} \quad 1]^T$ and $\tilde{X}_{t+1} = [x_i^{(t+1)} \quad y_i^{(t+1)} \quad 1]^T$ are homogeneous 2D coordinates of two pixels: $p_i^{(t)} = (x_i^{(t)}, y_i^{(t)})$ and $p_i^{(t+1)} = (x_i^{(t+1)}, y_i^{(t+1)})$, $p_i^{(t)}$ and $p_i^{(t+1)}$ are the projections of a world point P_i into the two temporal frames I_t and I_{t+1} respectively. S is the 3x3 similarity transform matrix with 4 degrees of freedom (DoF):

$$S = \begin{bmatrix} s_{00} & s_{01} & s_{02} \\ s_{10} & s_{11} & s_{12} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

There are three geometric transformation parameters in S: the scale factor s, the translation vector (t_x, t_y), and the rotation angle θ. These parameters are extracted from the S matrix, and accumulated from the first frame to the last, then the accumulated arrays of s, (t_x, t_y) and θ are respectively smoothed with Kalman filtering.

Algorithm 1 Compensation of the rotation angles to protect temporal continuity.

1: Input: The rotation angle array $\theta = (\theta^{(1)}, \theta^{(2)}, \cdots)$
2: Output: Smoothed rotation angle array θ'
3: Define rotation angle change limit $\delta = \pi/4$;
4: **for all** t **do**
5: $\quad \Delta\theta_t = \theta^{(t+1)} - \theta^{(t)}$;
6: $\quad$ **if** $\Delta\theta_t > 2\pi - \delta$ **then**
7: $\quad\quad \Delta\theta_t = \Delta\theta_t - 2\pi$
8: $\quad$ **else if** $\Delta\theta_t < -(2\pi - \delta)$ **then**
9: $\quad\quad \Delta\theta_t = \Delta\theta_t + 2\pi$
10: $\quad$ **end if**
11: $\quad \theta^{(t)'} = \theta^{(t)} + \Delta\theta_t$;
12: **end for**
13: Apply Kalman filtering on θ' array.

However, this standard process does not work well for a mobile multi-view video capture device. This is because while the mobile capture device is working, most of its motion is forward motion plus shaking composed of small left/right rotations in the fronto-parallel plane. The θ is extracted from S as:

$$\theta = \frac{\tan^{-1}\left(\frac{s_{10}}{s_{00}}\right) + \tan^{-1}\left(\frac{-s_{01}}{s_{11}}\right)}{2} \tag{2.11}$$

and $\theta \in (-\pi, \pi]$.

This style of egomotion will generate θ values close to 0 and $+/-\pi$ when there is a shake. Specifically, the θ values may go from the second quadrant to the third quadrant (i.e. go from $+\pi$ to $-\pi$) or the reverse from one frame to the next, which results in a sudden big fall/rise (see Fig. 2.2). Frame shifting artifacts will occur if Kalman filtering is directly applied on this θ array (see Fig. 2.2(a)). Thus special processing on the θ array is needed before filtering.

The technique for processing the θ array relies on the temporal continuity of the neighboring frames. Due to shaking there will be small changes in θ values for two succeeding frames, but this change should be small and a dramatic change like the value changing from $+\pi$ to $-\pi$ is unlikely. I detect dramatic changes, and compensate for them as described in Algorithm 1. I call this algorithm continuity aware Kalman filtering of rotation angles. With the application of this algorithm, the Kalman filtering-based smoothing of rotation angles can proceed more robustly than the standard procedure.

2.3 Summary

In this chapter I described my work on the video stabilization problem. I first introduced the three steps required for video stabilization, including camera motion estimation, motion smoothing, and image warping, and reviewed previous major work on each of these three areas. For the motion smoothing step, I proposed a new algorithm called continuity aware Kalman filtering of rotation angles which removes abrupt interruptions due to instability in rotation estimates in the stabilized videos associated with standard Kalman filtering.

Chapter 3

Efficient Multi-view Video Alignment and Panoramic Video Stitching

Driven by the availability and demand for high quality images and video, more and more capture systems have adopted high-resolution and large storage capability cameras. When many of traditional multi-view video alignment and panoramic video stitching algorithms were first proposed they did not consider video resolution and duration issues. The standard VGA 640x480 resolution and a few minutes length are probably the expected scale for video by such traditional computer vision algorithms. In contrast, nowadays commodity cameras can support the recording of an hour long video of 1920x1080 resolution easily. The efficiency of stitching high-resolution and long panoramic videos has become a practical issue.

There are two steps in the stitching procedure of Fig. 1.2 whose execution speed depend on video resolution: multi-view video alignment and panoramic video stitching. My experiments show traditional image manipulation algorithms for these two steps lack the efficiency in speed and memory to allow tractable stitching for high-definition and long panoramic videos. More efficient algorithms must be developed for practical consideration. In the following, I will first overview previous work and technically introduce basic concepts and operations in multi-view image and video alignment and stitching, then I will discuss the efficiency issue for high-resolution and long panoramic video stitching.

3.1 Previous Work

Aligning images and stitching them into seamless panoramas is a very old research topic in computer vision. Image stitching algorithms create high-resolution image mosaics and wide-angle panoramas that have found a lot applications in digital mapping (e.g. Google earth), satellite imaging [60], tele-reality [125], virtual environments [126],

distance learning (e.g, [47]), medical imaging (e.g. [69]) remote sensing (e.g. [27]) and photogrammetry [22]. Currently many digital cameras and mobile phones on the market already have built-in image stitching functionality.

As Szeliski concludes [127], there are two kinds of image alignment approaches: 1) Direct (pixel-based) alignment and 2) Image feature-based alignment. Most early alignment algorithms belong to the first category and work by directly minimizing pixel-to-pixel dissimilarities. In this category, Lucas and Kanade [84] developed a patch-based translational alignment approach using optical flow. Badra et al. [4] utilized Zernike moments to estimate the rotation and scaling differences among the images. Jia and Tang [66] proposed multi-scale luminance voting to repeatedly align the images.

Feature-based algorithms work by extracting a sparse set of features and then matching these between images. These sparse image features include Harris corners [166, 19], lines [94], SIFT features [11], MOPS features [13], and SURF features [70]. Feature-based approaches have the advantage of being more robust against scene movement, potentially faster, and able to automatically recognize the component images for stitching a panorama [11]. Nowadays many of the commercial image stitching programs are feature-based (e.g., AutoStitch [12] and Nokia Panorama [149]).

Recent research on image alignment and stitching focuses on computing globally consistent alignments [129, 115, 122], removal of "ghosts and artifacts" due to parallax and object movement [32, 122, 139, 1, 98], seam hiding by blending [85, 139, 75, 1, 110], dealing with exposure and color difference [66, 18, 154, 148], handling of misalignment [67, 158], and non-linear and non-uniform registration alignment approaches [131, 78].

3.2 Basic Concepts and Standard Techniques

3.2.1 The flat scene assumption

For building a panoramic image mosaic, Szeliski proved that panoramic images taken from the same view point with a stationary optical center are related by two-dimensional projective transformation [128]. The geometry of the multi-camera device approximately satisfies this condition when the scene is far away and the geometric relationship between any two adjacent camera views can be approximated by a two-dimensional projective transformation. Suppose (x_i^1, y_i^1) and (x_i^2, y_i^2) are images of a world point p_i in two adjacent camera views I_1 and I_2. The projective

transformation (i.e. 2D homography) from (x_i^1, y_i^1) to (x_i^2, y_i^2) is:

$$\tilde{X}_2 \backsim \tilde{H}\tilde{X}_1 \tag{3.1}$$

where $\tilde{X}_1 = [x_i^1 \quad y_i^1 \quad 1]^T$ and $\tilde{X}_2 = [x_i^2 \quad y_i^2 \quad 1]^T$ are homogeneous coordinates [58] of (x_i^1, y_i^1) and (x_i^2, y_i^2) respectively. $\tilde{H}$ is the projective transformation matrix of 3x3 size and 8 degrees of freedom (DoF):

$$\tilde{H} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix} \tag{3.2}$$

$\backsim$ denotes equals after normalization.

3.2.2 2D planar transformations

Multi-view video alignment is the process of computing the 2D geometric transformation between the video frames of two (spatially) neighboring camera views. If the videos are synchronized and each frame of a video is temporally aligned to its corresponding frame in another video, traditional image alignment algorithms can be used for frame alignment. Image alignment algorithms compute the correspondence relationships among images with varying degrees of overlap. It has two major applications: one is in multi-view video alignment where spatially neighboring frames are aligned in order to create panoramas; the other is in video stabilization where temporally neighboring frames are aligned in order to compute the camera global motion chain.

The goal of image alignment is to compute the 2D projective transform (or 2D homography) between two overlapping images. The 2D projective transform is described by a 3x3 transformation matrix which transforms the homogeneous coordinates of pixels in one image to those in the other. Specifically, assume I_1 and I_2 are the two overlapping images. $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$ are the projections of the same world point P into I_1 and I_2 respectively. The homography can be represented as:

$$\tilde{X}_2 \backsim \tilde{H}\tilde{X}_1 \tag{3.3}$$

where $X_1 = [x_1 \quad y_1]^T$ and $X_2 = [x_2 \quad y_2]^T$ are 2D Euclidean coordinates of p_1 and p_2 respectively, $\tilde{X}_1 = [x_1 \quad y_1 \quad 1]^T$ and $\tilde{X}_2 = [x_2 \quad y_2 \quad 1]^T$ are homogeneous 2D coordinates corresponding to X_1 and X_2 respectively. "$\backsim$" means equal up to scale (i.e., equal if normalized).

In Eq.(3.3), the homography matrix $\tilde{H}$ is the most complex form of 2D planar transformations. 2D planar transformations, in order of increasing complexity, include: translation, Euclidean transform, similarity transform, affine transform and projective transform (i.e. homography) [127]. Different transformations preserve different geometry properties.

Translation. 2D translations can be written as:

$$\tilde{X}_2 = \begin{bmatrix} I & t \\ 0 & 1 \end{bmatrix} \tilde{X}_1 \tag{3.4}$$

where I is the (2x2) identity matrix and $t = (t_x, t_y)^T$ is the (1x2) translation vector. $\tilde{X}_1 = (x_1, y_1, 1)^T$ and $\tilde{X}_2 = (x_2, y_2, 1)^T$ are homogeneous 2D coordinates.

Euclidean transform. This transformation is also known as 2D rigid body motion or the Rotation + translation transformation (since Euclidean distances are preserved). It can be written as:

$$\tilde{X}_2 = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tilde{X}_1 \tag{3.5}$$

where

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{3.6}$$

is an orthonormal rotation matrix with $RR^T = I$ and $|R| = 1$. And and $t = (t_x, t_y)^T$ is the (1x2) translation vector.

Similarity transform. Also known as the scaled rotation, this transform can be written as

$$\tilde{X}_2 = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \tilde{X}_1 \tag{3.7}$$

The similarity transform preserves angles between lines.

Affine transform. The affine transform is written as

$$\tilde{X}_2 = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ 0 & 0 & 1 \end{bmatrix} \tilde{X}_1 \tag{3.8}$$

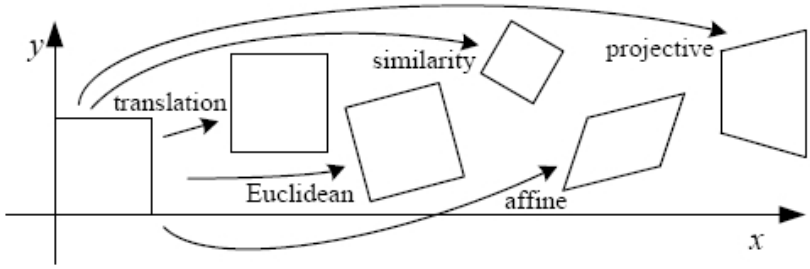


Figure 3.1: 2D planar transformations [127].

Parallel lines remain parallel under affine transformations.

Projective transform. This transform, also known as a perspective transform or homography, operates on homogeneous coordinates $\tilde{X}_1$ and $\tilde{X}_2$,

$$\tilde{X}_2 \backsim \tilde{H}\tilde{X}_1 \tag{3.9}$$

where $\backsim$ denotes equality up to scale and $\tilde{H}$ is an arbitrary 3x3 matrix. Note that $\tilde{H}$ is itself homogeneous, i.e., it is only defined up to a scale. The resulting homogeneous coordinate $\tilde{X}_2$ must be normalized in order to obtain an inhomogeneous result X_2, i.e.,

$$\begin{aligned} x_2 &= \frac{h_{00}x_1 + h_{01}y_1 + h_{02}}{h_{20}x_1 + h_{21}y_1 + 1} \\ y_2 &= \frac{h_{10}x_1 + h_{11}y_1 + h_{12}}{h_{20}x_1 + h_{21}y_1 + 1} \end{aligned} \tag{3.10}$$

The projective transform is the most general form of 2D planar transformation. It only preserves straight lines. It is the transform adopted by an image alignment algorithm when the overlap of two images is small (i.e. there are dramatic Field of View (FoV) differences between the two images).

Fig. 3.1 illustrates different 2D planar transformations. Generally speaking, different 2D planar transformations should be used for different cases depending on whether the flat scene assumption holds and whether the camera has a wide field of view. For unknown cases, usually it is the most general transformation, the projective transform (i.e. the homography) that is used.

3.2.3 Sparse features for image alignment

3.2.3.1 Interest points

An interest point is a distinctive image neighborhood or location which is characterized by: 1) a clear, preferably mathematically well-founded, definition, 2) a well-defined position in image space, 3) rich information content in the local image structure around the interest point, 4) stability under local and global perturbations in the image domain, including deformations such as those arising from perspective transformations, and 5) optionally, the notion of interest point should include an attribute of scale, to make it possible to compute interest points from real-life images as well as under scale changes. Interest points in an image are also called image feature points. Because their distribution is usually sparse in the image (compared to the dense distribution of image pixels), they are also called sparse image features or sparse features.

Interest point detection refers to the detection of interest points for subsequent processing (e.g. image alignment). Interest point detectors vary from standard feature detectors such as Harris corners [57] or DOG maxima to more elaborate methods such as maximally stable regions [90] and stable local phase structures [20]. Generally, interest point extraction and descriptor matching are considered as two basic steps, and there has been some progress in evaluating the various techniques with respect to interest point repeatability [117] and descriptor performance [96]. Other researchers have suggested that interest points should be located such that the solutions for matching position [121], orientation and scale [134] are stable. Compelling applications of invariant features based matching have been demonstrated for object recognition [82], structure from motion [116] and panoramic imaging [11].

More recently, feature detectors that are more invariant to scale ([83, 97] and affine transformations [97] have been proposed. These can be very useful when matching images that have different scales or aspects (e.g., for 3D object recognition). A simple way to achieve scale invariance is to look for scale-space maxima in a Difference of Gaussian (DOG) [83] or Harris corner detector [97, 134] over a sub-octave pyramid, (i.e., an image pyramid where the sub-sampling between adjacent levels is less than a factor of two). Lowe's original paper [83] uses a half-octave ($\sqrt{2}$) pyramid, whereas Triggs [134] recommends using a quarter-octave ($\sqrt[4]{2}$).

Of course, interest points are not the only kind of features that can be used for image alignment. Zoghlami et al. [166] use line segments as well as point-like features to estimate homographies between pairs of images, whereas

Bartoli et al. [5] use line segments with local correspondences along the edges to extract 3D structure and motion. Tuytelaars and Van Gool [135] use affine invariant regions to detect correspondences for wide baseline stereo matching. Matas et al. [91] detect maximally stable regions, using an algorithm related to watershed detection, whereas Kadir et al. [71] detect salient regions where patch entropy and its rate of change with scale are locally maximal.

3.2.3.2 The SIFT feature

The SIFT feature is a kind of low-level image feature proposed by David Lowe in 1999 to detect and describe invariant local features in images [82]. Lowe's method for image feature generation transforms an image into a large collection of feature vectors, each of which is invariant to image translation, scaling, and rotation, partially invariant to illumination changes and robust to local geometric distortion. These features share similar properties with neurons in the inferior temporal cortex that are used for object recognition in primate vision [118]. Key locations are defined as maxima and minima of the result of a difference of Gaussians function applied in scale-space to a series of smoothed and resampled images. Low contrast candidate points and edge response points along an edge are discarded. Dominant orientations are assigned to localized keypoints. These steps ensure that the keypoints are more stable for matching and recognition. SIFT descriptors are robust to local affine distortion are then obtained by considering pixels around a radius of the key location, blurring and resampling of local image orientation planes.

Given a pair of images and the computed SIFT feature set for each image, the SIFT algorithm builds inter-image feature correspondence as follows: For a localized SIFT feature point in one image, the algorithm finds its correspondences in the other images by comparing it to all of the features in the SIFT feature set of the other image. This is implemented by a modified k-d tree algorithm called the Best-bin-first (BBF) search method [7] that can identify the nearest neighbors with high probability using only a limited amount of computation.

In a recent performance evaluation and comparison of local descriptors by Mikolajczyk and Schmid [96], SIFT generally performed the best. Thus in this work, I adopt the SIFT feature for 2D homography estimation and image alignment.

3.2.4 Homography estimation based on sparse feature correspondences

The alignment of a pair of overlapped images first needs to estimate the 2D homography between them. Given a set of n ($n \geq 8$) corresponding image feature matches (e.g. SIFT feature matches) in the overlapping region of the images, the optimal values of the elements in the 2D homography matrix $\tilde{H}$ are estimated by minimizing the total transformation error:

$$E = \sum_{i=1}^{n} e_i. \tag{3.11}$$

where e_i is the transformation vector error (residual) for the i'th feature correspondence. It is computed as:

$$e_i = \sqrt{(x_i^2 - \frac{h_{00}x_i^1 + h_{01}y_i^1 + h_{02}}{h_{20}x_i^1 + h_{21}y_i^1 + 1})^2 + (y_i^2 - \frac{h_{10}x_i^1 + h_{11}y_i^1 + h_{12}}{h_{20}x_i^1 + h_{21}y_i^1 + 1})^2} \tag{3.12}$$

There are a total 8 unknown variables in the $\tilde{H}$ matrix (i.e. $h_{00}, \cdots, h_{21}$ in Eq.(3.12)), so at least 8 feature correspondences are needed to compute them. Given more than 8 feature correspondences, the optimal value of the 2D homography matrix $\tilde{H}$ can be solved as a minimum least squares (MLS) solution to Eq.(3.11).

3.2.5 Outlier removal using RANSAC

Once an initial set of feature correspondences has been computed, one needs to find a subset that will produce a high-accuracy alignment. One possible approach is to simply compute a least squares estimate, or to use a robustified (iteratively re-weighted) version of least squares. However, in many cases, it is better to first find a good starting set of inlier correspondences, i.e., points that are all consistent with some particular motion estimate.

One widely used solution to this problem is called RANdom SAmple Consensus (RANSAC) [46]. The technique starts by selecting (at random) a subset of k correspondences, which is then used to compute the homography H. The residuals of the full set of correspondences are then computed following Eqs.(3.12) and (3.11). The RANSAC technique then counts the number of inliers that are within ϵ distance to their predicted location, i.e., whose $|r_i| < \epsilon$. ϵ is an application dependent parameter, whose value is often around 1-3 pixels. The random selection process is repeated S times, and the sample set with largest number of inliers is kept as the final solution. Either the initial parameter guess p or the full set of computed inliers is then passed on to the next data fitting stage.

To ensure that the random sampling has a good chance of finding a true set of inliers, there must be a sufficient number of trials S. Let p be the probability that any given correspondence is valid, and P be the total probability of

(a) Input Image (b) Forward Projection

(c) Backward Projection

Figure 3.2: Different projection options for an input image.

success after S trials. The likelihood in one trial that all k random samples are inliers is p^k. Therefore, the likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S \tag{3.13}$$

and the required minimum number of trials is

$$S = \frac{\log(1 - P)}{\log(1 - p^k)} \tag{3.14}$$

The above equation shows the number of trials grows quickly with the number of sample points used. This provides a strong incentive to use the minimum number of sample points k possible for any given trial, which in practice is how RANSAC is normally used.

For the particular task of robust homography estimation, one can run RANSAC on the input feature correspondences with proper ϵ and S settings. Because there are eight unknown elements in $\tilde{H}$, $k = 8$. After the outlier feature correspondences are removed, then the true transformation is estimated from the inlier set using minimum mean squares error (MMSE) estimation.

3.2.6 Image warping

After the projective transform matrix $\tilde{H}$ is estimated from inlier feature correspondences (denoted by $\hat{H}$), it can be used to transform every pixel in one image to the coordinate system of the other image so that one can stitch the two images together in the same coordinate system. This process is called image warping. There are two ways to do the warping of the image. The first way is called **forward projection**, which means one uses $\hat{H}$ as a projection matrix to project every pixel in I_1 to a position in I_2 via matrix-vector multiplication and normalization. The input is the image I_1 and the output is another image I_1' which is the projection of I_1 into the coordinate system of image I_2. A problem of forward projection is when the projection is a scattered projection, there will exist seams in the projected image (see Fig. 3.2(b) for an example), which are intolerable artifacts for high-quality image stitching.

A better way to generate a seamless image warping result is to use **backward interpolation** instead. The goal of backward interpolation is still to project the image I_1 into the coordinates system of image I_2 and thus generate the projected image I_1', but it approaches the goal in a reverse way. Backward interpolation uses the projection formula:

$$\tilde{X}_1 \backsim \hat{H}^{-1}\tilde{X}_2 \tag{3.15}$$

For every output pixel position in the coordinate system of image I_2 whose value is unknown, Eq.(3.15) is used to project the position back to the corresponding position in I_1. The projection result may reside in a place within four neighboring pixels in I_1. Then, the pixel values of the output pixel in I_1' is interpolated from the values of the four neighboring pixels in I_1. Fig. 3.2(c) shows a backward interpolation example.

Backward interpolation avoids the seam artifacts of forward projection at the cost of more dense computation (because every output pixel value in I_1' needs an additional interpolation operation).

3.2.7 The compositing space

Once one has registered all of the input images with respect to each other, he/she must also choose an output compositing space onto which to warp and place all of the aligned images, so that the final panorama can be stitched there. This compositing space can be simply the Euclidean coordinate space of one of the input image (which is called a reference image), or be a different geometric space like the spherical or cylindrical space.

3.2.7.1 The flat (Euclidean) compositing space

The coordinate system of an input image is a Euclidean (orthogonal) space. The user can select the Euclidean space of the reference image as the output compositing space, and project and warp other input images into this space in order to stitch the final panorama there. Or, the user can select a different Euclidean coordinate space as the output compositing space and project all the input images into this space. The panorama stitched in the Euclidean space is called a rectangular panorama.

3.2.7.2 The cylindrical compositing space

An alternative to using homographies or 3D motions to align images is to first warp the images into cylindrical coordinates and to then use a pure translational model to align them [25]. Unfortunately, this only works if the images are all taken with a level camera or with a known tilt angle.

Assume the camera is in its canonical position, i.e., its rotation matrix is the identity so that the optic axis is aligned with the z axis and the y axis is aligned vertically. The 3D ray corresponding to an (x, y) pixel is therefore (x, y, f). One wishes to project this image onto a cylindrical surface of unit radius [125]. Points on this surface are parameterized by an angle θ and a height h, with the 3D cylindrical coordinates corresponding to (θ, h) given by

$$(\sin\theta, h, \cos\theta) \propto (x, y, f) \tag{3.16}$$

From this correspondence, one can compute the formula for the warped or mapped coordinates [129],

$$x' = s\theta = s\tan^{-1}\frac{x}{f} \tag{3.17}$$

$$y' = sh = s\frac{y}{\sqrt{x^2 + f^2}} \tag{3.18}$$

where s is an arbitrary scaling factor (sometimes called the radius of the cylinder). It can be set to a larger or smaller value for the final compositing surface, depending on the desired output panorama resolution. The inverse of the mapping equations (3.17) and (3.18) is given by

$$x = f\tan\theta = f\tan\frac{x'}{s} \tag{3.19}$$

$$y = fh\sec\theta = f\frac{y'}{s}\sec\frac{x'}{s} \tag{3.20}$$

Fig. 3.3 shows a rectangular panorama and its cylindrical version.

(a) A rectangular panorama

(b) Corresponding cylindrical panorama

Figure 3.3: A rectangular panorama and its cylindrical version.

3.2.7.3 The spherical compositing space

If the composting space is a spherical space, one can map image coordinates $p = (x, y)$ onto 2D spherical screen location $u = (\theta, \phi)$, $\theta \in (-\pi, \pi]$ using

$$x' = s\theta = s\tan^{-1}\frac{x}{f} \tag{3.21}$$

$$y' = s\phi = s\tan^{-1}\frac{y}{\sqrt{x^2 + f^2}} \tag{3.22}$$

where f is the average focal length of the CCD (average of the focal lengths in x and y directions). s is the scaling factor set by the user.

The reverse of Eqs.(3.21) and (3.22) are:

$$x = f\tan\theta = f\tan\frac{x'}{s} \tag{3.23}$$

$$y = f\tan\phi\sec\theta = f\tan\frac{y'}{s}\sec\frac{x'}{s} \tag{3.24}$$

Eqs.(3.23) and (3.24) mean $(f\tan\theta, f\tan\phi\sec\theta, f) \propto (x, y, f)$, that is

$$(\sin\frac{x'}{s}\cos\frac{y'}{s}, \sin\frac{y'}{s}, \cos\frac{x'}{s}\cos\frac{y'}{s}) \propto (x, y, f) \tag{3.25}$$

Figure 3.4: A spherical panorama.

Fig. 3.4 shows a spherical panorama, corresponding to the panoramas shown in Fig. 3.3.

Note no matter what the final output compositing space is (either flat, cylindrical or spherical), usually during the stitching process, backward interpolation (Eq. 3.25) is used to avoid the occurrence of seams in the output space.

3.2.8 Image blending

Image blending means to create a new, composite image from two overlapping images whose color at a pixel position is generated as a mixture of the colors of the two input image at the same pixel position. From the point of view of the whole image, spatially-varying weighting is often be used to accomplish a gradual transition appearance in order to hide dramatic color changes from one image to the other. There are only a few image blending techniques existing in the literature, I will introduce the most popular ones here.

Feathering Image blending using a gradually changing weighting function is called feathering [127]. A simple but effective weighting function is proposed by Brown et al. [11] $w(x,y) = w(x)w(y)$ where $w(x)$ varies linearly from 1 at the center of the image to 0 at the edge (Fig. 3.5). Suppose there are n images $I^i(x,y)(i \in \{1..n\})$ overlapping at a pixel position (x,y). If the final compositing space is cylindrical (or spherical), then both the images and their weight functions are projected and resampled in cylindrical coordinates yielding $I^i(\theta,h)$ and $W^i(\theta,h)$ (or in spherical coordinates yielding $I^i(\theta,\phi)$ and $W^i(\theta,\phi)$). A simple approach to blending is to perform a weighted sum of the image intensities at each pixel position (θ,ϕ) using these projected weight functions (here spherical coordinates are used to illustrate the idea).

$$I^{composite}(\theta,\phi) = \frac{\sum_{i=1}^{n} I^i(\theta,\phi)W^i(\theta,\phi)}{\sum_{i=1}^{n} W^i(\theta,\phi)} \tag{3.26}$$

where $I^{composite}(\theta,\phi)$ is a pixel in the composite spherical image formed using linear blending. This simple blending

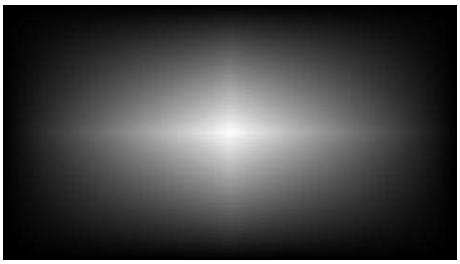


Figure 3.5: Visualization of the weight function in Euclidean coordinates.

approach works pretty well in practice when the image alignment error is small. It is one of the blending options of the image stitching software AutoStitch [12].

Laplacian pyramid blending An attractive solution to this problem was developed by Burt and Adelson [16]. Instead of using a single transition width, a frequency-adaptive width is used by creating a band-pass (Laplacian) pyramid and making the transition widths a function of the pyramid level. The process operates as follows. First, each warped image is converted into a band-pass (Laplacian) pyramid, which involves smoothing each level with a 1/16(1, 4, 6, 4, 1) binomial kernel, subsampling the smoothed image by a factor of 2, and subtracting the reconstructed (low-pass) image from the original. This creates a reversible, over-complete representation of the image signal. Invalid and edge pixels are filled with neighboring values to make this process well defined. Next, the mask (valid pixel) image associated with each source image is converted into a low-pass (Gaussian) pyramid. These blurred and subsampled masks become the weights used to perform a per-level feathered blend of the band-pass source images. Finally, the composite image is reconstructed by interpolating and summing all of the pyramid levels (band-pass images).

Gradient domain blending An alternative approach to multi-band image blending is to perform the operations in the gradient domain. The paper by Levin et al. [75] examines several different variants on this approach, which they call Gradient-domain Image STitching (GIST). The techniques they examine include feathering (blending) the gradients from the source images, as well as using an $L1$ norm in performing the reconstruction of the image from the gradient field, rather than using an $L2$ norm. Their preferred technique is the $L1$ optimization of a feathered (blended) cost function on the original image gradients (which they call GIST1-l1).

Fig. 3.6 shows an example of image blending.

(a) Left image. (b) Right image. (c) Without blending (d) With blending.

Figure 3.6: An image blending example. The blending algorithm is Laplacian pyramid blending [16].

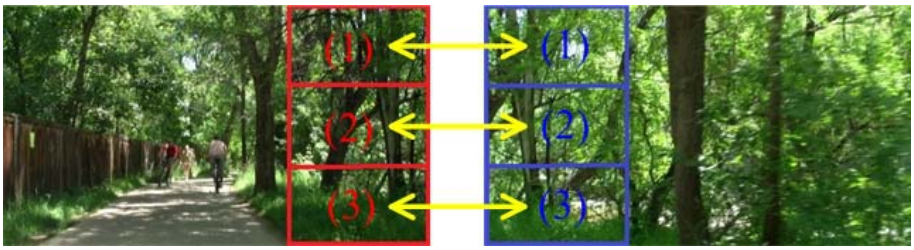


Figure 3.7: Illustration of the constrained SIFT feature matching scheme. The original SIFT matching scheme matches the SIFT features across the whole left image to those in the right image. Because the image size is large (1920x1080 pixels), the left image has 27921 SIFT features and the right has 26568 features, the global search strategy of original SIFT matching thus has to evaluate $27921 \times 26568 \approx 741.8$ million possible feature pairs to find the optimum matches, which is infeasible in practice. The constrained SIFT matching scheme restricts the matching to between divided sub-blocks of certain parts of the images (shown by yellow arrows in the figure, and $N = 3$ for this particular example), which reduces the computation by $1 - (1 - 8/9) \times (1 - (3 - 1)/3) = 26/27$ at the cost of loss of some cross-region SIFT feature matches.

3.3 Efficient high-resolution multi-view video alignment and stitching

In the above I introduced the basic concepts and operations in multi-view video alignment and stitching. Given a multi-view video of small or median resolution (e.g., 640x480 pixels for each single view), one can first synchronize the individual videos using video synchronization techniques. Then, the synchronized frames can be aligned and stitched into panoramic frames using the multi-view image alignment and stitching techniques introduced above [11]. However, if the multi-view video is of a high-resolution, say 1920x0180 pixels, or of a long length, say a few hours, simply applying the above techniques may have time and efficiency issues. In the following I will describe the new techniques I specially developed for stitching high-resolution and long panoramic videos.

3.3.1 Constrained and multi-grid SIFT matching

As I described earlier, SIFT features are widely used in multi-view image stitching to estimate the 2D homographies between the views (e.g. [12]). For high-resolution (e.g., HDTV resolution of 1920x1080 pixels) multi-view

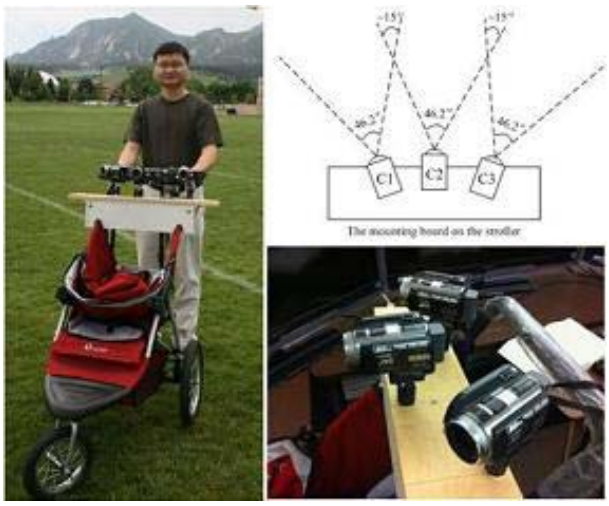


Figure 3.8: A multi-view video capturing devices composed of high-definition commodity cameras.

image stitching, one problem with SIFT features is that their distribution is affected by the content of the image. For blank image areas, there are usually only a few SIFT features, while for areas of abundant texture there can be tens of thousands. This may result in an efficiency problem for matching SIFT features between two high-definition images of an outdoor scene that has abundant texture from tree, bushes, grass and other natural surfaces.

The matching strategy of SIFT features is global. That is, given a pair of images whose SIFT features are already computed, for each SIFT feature of one image a search is performed over the whole set of SIFT features of the other image in order to find the best match from this set. This can become a very time-consuming process for matching large SIFT feature sets. Fig. 3.7 shows an example. It contains a pair of high-resolution (1920x1080 pixels) images of an outdoor scene where the left and right images are spatial neighbors in the horizontal direction. Because the scene is of a woodland with abundant texture, there are many SIFT features generated in each view. Particularly for this example, the left image has 27921 SIFT features and the right has 26568 features. When these two SIFT feature sets were matched to each other, the computation took more than 10 minutes.

My solution to speed up this time-consuming global searching process is to exploit device spatial layout to restrict the search space in SIFT feature matching. For multi-view video alignment, the local motion constraint is determined by the spatial layout of the cameras in a multi-view video capturing device. Fig. 3.8 shows the multi-view video capturing device used to capture the images shown in Fig. 3.7. Note that this device actually represents the most typical layout of multi-view video capture rigs in which the cameras are placed at the same horizontal level in a

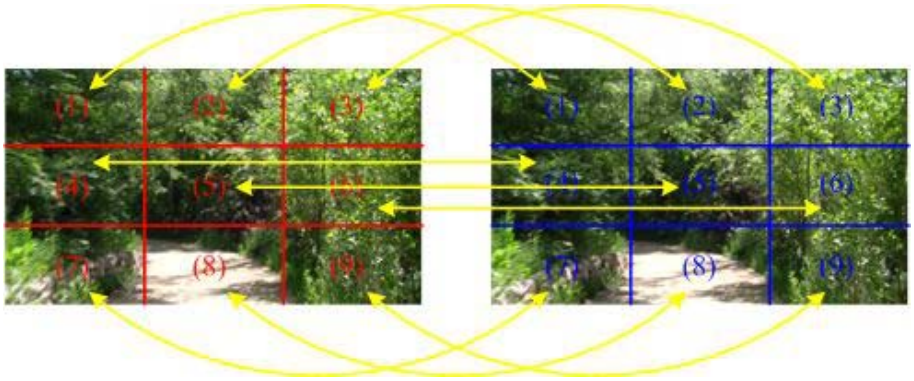


Figure 3.9: Illustration of the temporally constrained SIFT feature matching scheme. The original SIFT matching scheme matches the SIFT features over the whole left image to those in the right image (the left and right images are temporal neighbors). Because the image size is large (1920x1080 pixels), the left image has 38028 SIFT features and the right has 28174 features, the global search strategy of original SIFT matching thus has to evaluate $38028 \times 28174 \approx 1071.4$ millions possible feature pairs to find the optimum matches, which is slow in practice. The multi-grid SIFT matching scheme restricts the matching to between equally divided patches of the images (shown by yellow arrows in the figure, and $M = N = 3$ for this particular example).

circle so that the captured images can be aligned side-by-side to form a panorama. This spatial layout of the cameras constrains the overlap regions of the captured images to only certain parts of the whole body of the images. This spatial constraint can be exploited to restrict the search space of SIFT matching.

I designed a constrained SIFT matching scheme considering the special layout of the multi-view video capturing device shown in Fig. 3.8. The scheme has two parts. First, given a pair of images captured by two neighboring cameras in Fig. 3.8, I limit corresponding SIFT feature matching to between the right 1/M part of the left view and the left 1/M part of the right view. Comparing to full-image SIFT matching, this reduces the computation of the matching process by $1 - (1/M) \times (1/M) = (M^2 - 1)/M^2$. Second, for the restricted regions to be matched to each other, I divide each of them vertically into N equally-large blocks, and limit the matching to between horizontal pairs of the divided blocks. This further reduces the computation by $1 - (1/N) \times (1/N) \times N = (N - 1)/N$. Particularly for the example image pairs in Fig.3.7, when using my piece-wise SIFT matching scheme with $M = 3$ and $N = 3$, there are in total $4180 \times 3972 + 3338 \times 3633 + 1639 \times 2363 \approx 32.6$ million possible feature pairs to be evaluated and the total matching process takes < 25 seconds to finish.

Note not only device spatial layout can be explored for (spatial) multi-view image alignment, similarly a local motion constraint can be explored for video stabilization in which the 2D planar transformations between temporal frames are computed to form the global transform chain (see Chapter 2).

For temporal frame matching, I have designed a multi-grid SIFT matching scheme considering the high sampling speed (30 fps) of typical video cameras. High sampling speed usually means low image motion from one temporal frame to the next so that I can require every SIFT feature in one temporal frame to search for its correspondence in the next temporal frame only within its local image neighborhood. My multi-grid scheme equally divides the images into MxN patches, and applies SIFT matching for each of the i'th ($i = 1, \cdots, M \times N$) inter-frame pairs of patches. Particularly for the example image pair in Fig.3.9, the original SIFT localization and matching scheme takes 566.6 seconds and the 3x3 multi-grid scheme takes only 147.4 seconds.

The negative effect of these two locality-constrained SIFT matching schemes is that some true SIFT matches across the blocks will be excluded from the matching process. However, considering the extremely large number of SIFT features existing in the images of a typical outdoor scene, such loss is usually negligible.

3.3.2 Video alignment using multiple frame correspondences

A problem of SIFT features is that their distribution over the image depends on the content of the image. For example, for an image whose content has both textured areas (e.g. trees and grass) and almost blank areas (e.g. the sky), the computed SIFT features of this image may only distribute in the textured area. This kind of uneven distribution of SIFT features may bias the frame alignment and generate large alignment errors for the views of a multi-view video. This introduces the following problem: given a multi-view video containing tens of thousands of frames, how can one find the frame correspondence which best represents the relative geometry between the views of the multi-view video, and thus allow estimation of a 2D homography with minimum alignment error?

Here, I approach the problem in a different way. I propose to use many randomly selected frames for aligning the views in a multi-view video, rather than using a single frame. I make the assumption that per-frame alignment errors (due to uneven SIFT feature distribution) would cancel out given a sufficiently large number of randomly selected frames. In other words, if I can use a sufficiently large number of multi-view video frames to estimate the 2D homography between the views, although there exists individual alignment error for each frame, the accumulated error cancels out at the end.

To implement this strategy, I first union the set of SIFT feature correspondence for each of the selected frames,

and then use this combined set of SIFT feature correspondences for estimating the 2D homography. [1] RANdom SAmple Consensus (RANSAC) [46] is used to process the combined set of SIFT feature correspondences in order to remove the outlier correspondences. The final 2D homography is computed from the RANSAC inliers.

3.3.3 Concatenated projection and warping

Algorithm 2 Efficient two-view spherical panorama stitching.

Input: High-definition images I_1 and I_2, the 2D homography H_{12} from I_1 to I_2, and the scaling factors s_θ and s_ϕ
Output: A seamless spherical panorama I' stitched from I_1 and I_2

Step 1: Forward projections.
1.1) Create two coordinate arrays (x_1, y_1) and (x_2, y_2) recording all possible pixel coordinates of I_1 and I_2 respectively;
1.2) Use forward projection $X_1' = H_{12} * X_1$ to find the projection of $X_1 = [x_1, y_1, 1]^T$ in I_2's coordinate system, getting $X_1' = [x_1', y_1', w_1']^T$ (un-normalized); Also let $X_2 = [x_2, y_2, 1]^T$;
1.3) Use forward projection $\theta = \tan^{-1}\frac{x}{z}$ (Eq.(3.21)) and $\phi = \tan^{-1}\frac{y}{\sqrt{x^2+z^2}}$ (Eq.(3.22)) where $z = W_1'$ for X_1' and $z = 1$ for X_2 to project X_1' and X_2 into spherical coordinates, getting (θ_1', ϕ_1') (for X_1') and (θ_2, ϕ_2) (for X_2) respectively.

Step 2: Create the output image.
2.1) Let $\theta_{min} = min\{\theta_1', \theta_2\}$, $\theta_{max} = max\{\theta_1', \theta_2\}$, $\phi_{min} = min\{\phi_1', \phi_2\}$, $\phi_{max} = max\{\phi_1', \phi_2\}$; then $R_{size} = ceil((\phi_{max} - \phi_{min}) * s_\phi)$, $C_{size} = ceil((\theta_{max} - \theta_{min}) * s_\theta)$
2.2) Create two empty images I_1' and I_2', both of size (R_{size}, C_{size}).

Step 3: Backward interpolation.
3.1) For each pixel $p' = (x', y')$ in I' compute $\theta' = (x' - 1)/s_\theta + \theta_{min}$ and $\phi' = (y' - 1)/s_\phi + \phi_{min}$; Compute $X_{binterp} = [\sin\theta'\cos\phi', \sin\phi, \cos\theta\cos\phi]^T$ (Eq.(3.25));
3.2) Compute $\hat{X}_1 = H_{12}^{-1} * X_{binterp} = [x_1'', y_1'', w_1'']^T$; let $\hat{X}_2 = X_{binterp} = [x_2'', y_2'', w_2'']^T$;
3.3) Normalize $\hat{X}_1$ and $\hat{X}_2$ to get $\hat{X}_1' = [\frac{x_1''}{w_1''}, \frac{y_1''}{w_1''}, 1]^T$ and $\hat{X}_2' = [\frac{x_2''}{w_2''}, \frac{y_2''}{w_2''}, 1]^T$;
3.4) Interpolate I_1' out from I_1 by (x,y) coordinates of $\hat{X}_1'$; Interpolate I_2' out from I_2 by (x,y) coordinates of $\hat{X}_2'$.

Step 4: Blending.
4.1) Blend I_1' and I_2' to get the final panorama I'.

If one chooses the cylindrical or the spherical space as the output compositing space, then there will be two image projection and warping processes. One is to project and warp all the input images to the Euclidean coordinate system of the reference view, the other is to project the intermediate results of the last projection to the cylindrical or spherical space. For high-resolution stitching, it is desirable to concatenate these two projections into a single step in order to save the time and space for storing the intermediate results. Also, as I described earlier, there are two kinds

[1] An alternative approach is to compute a 2D homography from each of the selected frames using its own SIFT feature correspondences, and then average the estimated homography matrices for all of the frames. This method is not effective in practice.

of image warping operation: forward projection and backward interpolation. Forward projection is generally faster than backward interpolation but may leave black seams for large images (see Fig. 3.2). For high-resolution panorama stitching, both speed of the stitching process and the quality of the output panorama need to be considered, so the question of how to apply forward projection and backward interpolation in an optimized way has become a problem. I have developed the following procedure that carefully combines forward projection and backward interpolation for fast and good-quality stitching of high-resolution panoramas. The procedure is described in Algorithm 2.

Algorithm 2 describes the optimized stitching procedure for high-resolution panoramas by choosing the spherical space as the output compositing space. If the cylindrical space is chosen as the output compositing space, one just needs to modify Steps 1.3 and 3.1 in Algorithm 2 by substituting Eqs. (3.21), (3.22) and (3.25) with Eqs. (3.17), (3.18) and (3.16) respectively.

3.3.4 Min-space feathering of high-resolution images

For efficient stitching of high-resolution panoramic video, I also propose a min-space feathering approach. The approach is an improvement over Brown's linear blending approach described in Sec. 3.2.8. As a traditional approach designed for blending regular size images, Brown's approach takes all projected images into computer memory before blending them together. I have found this scheme has a very large memory footprint for HDTV (i.e. 1920x1080) images which easily results in out-of-memory errors.

My solution to this problem is to only load the (projected) overlap area into memory for blending, and directly copy the non-overlapped areas to the corresponding areas in the final panorama. This can be implemented via a set of mask operations. Given a pair of input images, the spatial mask of the overlapped area is determined by the intersection of the (projected) non-zero masks of the blending weight function of each image, the mask of the non-overlap areas are determined by the difference of the mask of the image and that of the overlapped area. Denoting the masks for a pair of input images as M_{I_1} and M_{I_2} respectively, and $proj(.)$ as the projection of an image to the output compositing

space, and denoting their overlap area as Ω_{12} with masks $M_{\Omega_{12}}$:

$$M_{\Omega_{12}} = proj(M_{I_1}) \cap proj(M_{I_2}) \tag{3.27}$$

$$M_{\omega_1} = proj(M_{I_1}) - M_{\Omega_{12}} \tag{3.28}$$

$$M_{\omega_2} = proj(M_{I_2}) - M_{\Omega_{12}} \tag{3.29}$$

The areas denoted by ω_1 and ω_2 are non-overlap areas in the compositing space, and M_{ω_1} and M_{ω_2} are their spatial masks. The output values in these areas are directly copied from the projections of I_1 and I_2 without blending. The area denoted by Ω_{12} is the overlap area between I_1 and I_2 in the compositing space, and $M_{\Omega_{12}}$ is the corresponding spatial mask. The output value of a pixel p_i in this area is linearly blended from the corresponding values at p_i in the projections of I_1 and I_2:

$$p'_i = \frac{proj(I_1(p_i)) \cdot proj(W_1(p_i)) + proj(I_2(p_i)) \cdot proj(W_2(p_i))}{proj(W_1(p_i)) + proj(W_2(p_i))} \tag{3.30}$$

where W_1 and W_2 are the weight functions of images I_1 and I_2 respectively. $W(x, y) = w(x)w(y)$ where $w(x)$ varies linearly from 1 at the center of the image to 0 at the edge [12] (see also Sec. 3.2.8).

Given the case that an input image I_1 overlaps with a set of other input images $S = \{I_i, i = 2, ..., n\}, n \geq 2$, one needs to repeat the mask operations in Eq.(3.29) for $n - 1$ times, each time only considering the overlap between the image I_1 and an image non-recurrently selected out from S.

The proposed approach processes one of the areas of $\Omega_{12}, \omega_1, \omega_2$ at a time to achieve minimum memory footprint for high-resolution images. In practice, I find the min-space blending not only has a much smaller memory footprint than Brown's original blending technique, but is also much faster. For example, for stitching the panorama shown in Fig. 3.4 whose input images are of 1920x1080 pixels, using my blending approach took about 10 seconds while using Brown's approach took 50+ seconds. This is because there is no blending computation but a direct copy operation for non-overlap areas and there are fewer large-size memory load/save operations, which makes my approach much faster than Brown's. This good property of my approach is critical to the task application of stitching a high-resolution multi-view video composed of tens of thousands of frames.

3.4 Summary

In this chapter, after briefly reviewing the previous work in multi-view video alignment and stitching, I introduced the basic conception and standard techniques for the problem. Then, considering the special demands of high quality panoramic video, I proposed a sequence of optimized approaches for efficient stitching of high-resolution and long duration panoramic videos and showed their superior efficiency over traditional approaches.

Chapter 4

Color Correction

4.1 Introduction

A multi-video capturing device like that shown in Fig. 3.8 usually has cameras facing in diverging directions, and as a result automated brightness and color balance functions can produce color differences between two neighboring cameras and cause the captured images to look different even at the overlap areas. These color variations cause the stitched panoramas look banded and unnatural. Although blending can somewhat "hide" such differences by mixing the two images over the overlap area, it is not a solution of the root problem and usually cannot fully remove the artifact when the difference is large. Figure 4.1 shows an example.

The color variation may come from two sources: different device settings or different viewing conditions. If the color difference comes from device settings, it may be corrected by camera color calibration beforehand. However, especially in outdoor scenes where brightness can vary sharply, the camera autosettings are essential to capturing coherent video. I thus have to rely on image post-processing techniques to balance the color differences (including color difference from both sources). I call this color correction/balancing for multi-view image stitching [154].

Color correction or color balancing in automatic multi-view image and video stitching is the process of correcting the color differences between neighboring views which arise due to different exposure levels and view angles. Compared to the other major steps in image stitching of registration and blending, color correction has received less attention and relatively simpler treatment. Image blending, which has a similar end effect to color correction, has concealed the the role of the latter. Only recently, with the growing demand and popularity of high definition images and video, have people begun to recognize that image blending alone cannot remove all the color difference between

(a).  (b). 

(c). 

Figure 4.1: An example mosaic image pair that have dramatic color difference. (a) and (b) are the mosaic image pair, (c) is the stitching result of AutoStitch [11, 12] with the multi-band image blending [16] functionality enabled. (Note that AutoStitch projects the stitching result into the cylindrical plane). Color incoherence is still obvious in the scene even after multi-band image blending is performed.

different views under all situations.

Color correction or color balancing in automatic multi-view image and video stitching is the process of correcting the color differences between neighboring views which arise due to different exposure levels and view angles. Compared to the other major steps in image stitching of registration and blending, color correction has received less attention and relatively simpler treatment. Image blending, which has a similar end effect to color correction, has concealed the the role of the latter. Only recently, with the growing demand and popularity of high definition images and video, have people begun to recognize that image blending alone cannot remove all the color difference between different views under all situations.

In the computer vision and multi-view video processing communities, the initial efforts on solving the color balancing problem for multi-view stitching used exposure compensation (or gain compensation) [99, 11, 139]. This approach adjusts the intensity gain level of component images to compensate for appearance differences caused by different exposure levels. Although it works for some cases, it may fail to completely compensate for color difference between different views when the lighting conditions vary dramatically. Later work compensated for differences using

all three color channels rather than via the single intensity channel [132, 59, 66, 68, 161, 156]. At the same time the image processing and computer graphics communities were developing similar color manipulation methods they called *color transfer techniques* (e.g., [112, 130, 146, 145, 108]). Technically speaking, there is no difference between color balancing and color transfer, except that the latter generally does not have to be restricted to the overlapped area — if I restrict color transfer techniques to operate using only information from the overlapped area, then they can be easily used to solve the color balancing problem for multi-view image and video stitching. Actually, one of the current research topics in color transfer is how to exploit local spatial constraints/matches to guide the color transfer (e.g., the local color transfer technique [130]). From this perspective research on color balancing and color transfer exhibit significant overlap.

In this chapter, I will discuss in depth color balancing and color transfer techniques in the context of automatic multi-view image and video stitching. In this context, I use the term "color correction approaches" to refer to the union of applicable color balancing and color transfer approaches. I use the terms "color correction', "color alteration", "color transfer", "color mapping", and "color balancing" interchangeably in the following parts of this chapter.

4.2 Basic Concepts

4.2.1 The image formation process

To better understand image color correction, it is necessary to understand the mechanism of the image formation process first. Here I focus on the image formation process in digital cameras. Modern digital camera systems are usually composed of two parts: an optical lens system and a CCD (Charge Coupled Device). The energy of the incoming light from a scene point changes when it passes through the lens system, in a manner determined by the optical parameters such as aperture size and focal length. After the light reaches the CCD, it is further converted into electronic signals that are the output pixel brightness. This conversion follows a radiometric response function that is the mapping between the irradiance falling on the CCD and the pixel brightness. Fig. 4.2 outlines the procedure of the image formation process for a general digital camera system. Specifically, the scene brightness is defined by the term radiance (L in Fig. 4.2), which is the power per unit foreshortened area emitted into a unit solid angle by a surface [48]. After passing through the lens system, the power of radiant energy falling on the image plane (i.e.

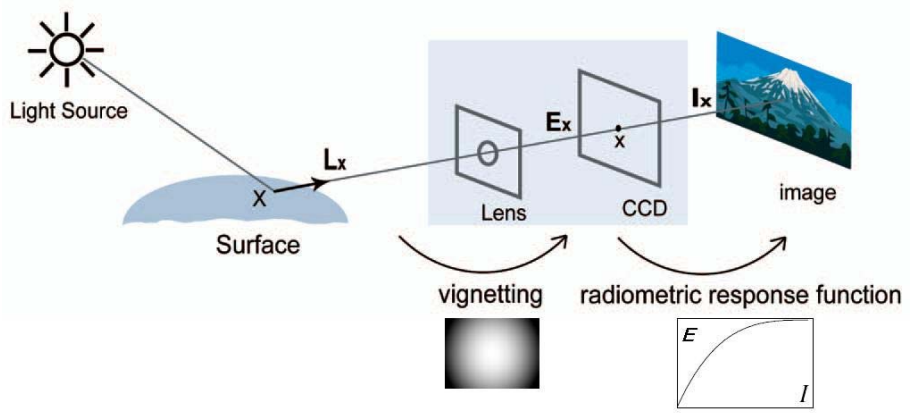


Figure 4.2: Radiometric image formation process. Vignetting affects the transformation from the scene radiance (L) to the image irradiance (E). The radiometric response function explains the nonlinear relationship between the image irradiance (E) and the image brightness (I).

the CCD) is called irradiance (E in Fig. 4.2). Irradiance is then transformed to image brightness (I in Fig. 4.2) via a light-to-electronic-signal conversion process following the radiometric response function. The parameters of the lens system and the radiometric response function co-determine the relationship between the scene radiance and the corresponding image intensity.

Most color correction algorithms operate in the image domain and aim at computing the end-to-end mapping of the intensities/colors of one image to those of the other. But with hardware pre-calibration information, some color correction algorithms (e.g., [37]) can also operate in the radiance or irradiance domain for mapping in more original level of the image formation process.

4.2.2 Image representation convention

In computer vision and image processing, images are usually represented by a matrix after they are loaded from files. There are two conventions for indexing an image matrix, either using a $(row, column)$ coordinate system or using an (x, y) coordinate system.

Many commercial image manipulation programs such as the InfanView software [124] use the (x, y) coordinate system. In this system, the original is at the top-left corner of an image, and the positive direction of the x axis is from left to right and the positive direction of the y axis is from up to down. Fig. 4.3 illustrates this coordinate system. Note

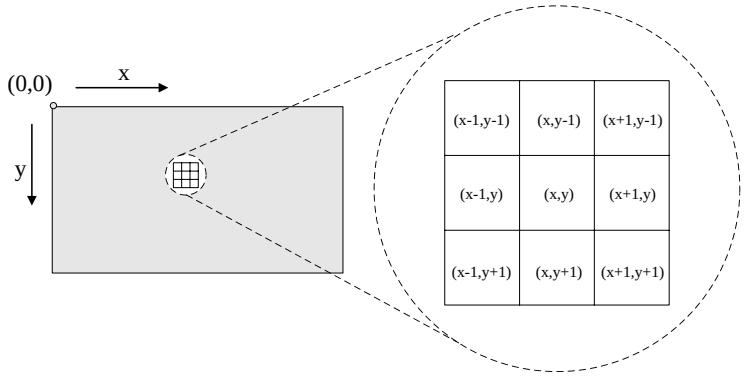


Figure 4.3: The (x, y) coordinate system for images. To the right is an enlarged view of the local neighborhood centered at the pixel position (x, y).

in this (x, y) coordinate system for images, the y axis is pointing downward, which is different to the typical depiction of a Euclidean coordinate system in mathematics where the y axis is pointing upward.

The (x, y) coordinate system of images is conceptual. For example, when we say an image of 640×480 resolution, we mean the image width in the x axis direction is 640 pixels and the image height in the y axis direction is 480 pixels. The popular scientific computation software MATLAB [92] adopts the $(row, column)$ coordinate system. This is because MATLAB uses column major matrices as its primary computational unit. The convention between the (x, y) coordinate system and the $(row, column)$ coordinate systems is that $row = y$ and $column = x$.

Most of the discussion of the image operations in this thesis is based on the conceptual (x, y) indexing of images. In the case where MATLAB implementation details use $(row, column)$ indexing I will specify in advance.

4.2.3 Color spaces

A color space specifies how colors in our physical world (i.e. the visible spectrum) are represented numerically by a man-made digital system (e.g. a computer system). It is the numerical space in which a color model matches physical colors using its associated color matching function(s). A color model is an abstract mathematical model describing the way physical colors can be represented as tuples of numbers, typically as three or four values or primary color components (e.g. RGB and CMYK are color models). The matching function represents a kind of interpretation of physical colors. Adding a certain matching function between the color model and a certain reference color space not only embodies a kind of color interpretation but also results in a definite "footprint" within the reference color

space. This “footprint” is known as a gamut. The gamut and the color model defines a color space.

Given the visitable spectrum of our eyes, a color space must have all the information needed for the storage, processing and generation (visualization) of a perceptually equivalent spectrum. Different color spaces have different definition and organization of such information, which characterizes the color spaces themselves. Color correction algorithms can choose to run in different color spaces to achieve their best end effects.

A detailed list and explanation of popular color spaces used by color correction algorithms can be found in Appendix A.

4.2.4 Contrast, dynamic range and gradient manipulation techniques

Contrast is the difference in luminance and/or color that makes an object (or its representation in an image or display) distinguishable. In visual perception of the real world, contrast is determined by the difference in the color and brightness of the object and other objects within the same field of view. Because the human visual system (HVS) is more sensitive to contrast than absolute luminance, we can perceive the world similarly regardless of the huge changes in illumination over the day or from place to place. The maximum contrast of an image is the contrast ratio or dynamic range.

Contrast can be represented by the gradient of the luminance/color of an image. Such contrast is called device contrast. Researchers argue that such simple representation of contrast does not obey the perceptual characteristics of the HVS, and thus various, more complex representations of contrast which obey the perceptual characteristics of the HVS are proposed. These latter representations of contrast are called perceptual contrast [86].

Many contrast processing techniques operate in the gradient domain which means the gradient fields (dx, dy) either of the input image or of some converted version of the image. Computer vision techniques operating in the gradient domain are called gradient manipulation techniques.

4.3 Color correction approaches

The essence of all color correction algorithms is transferring the color (or intensity) palette of a source image to a target image. In the context of multi-view image and video stitching, the source image corresponds to the view selected as the reference by the user, and the target image corresponds to the image whose color is to be corrected.

Rather than giving an historic review of color balancing and color transfer respectively, here I will categorize the techniques used according to their basic approaches. At the highest level there are two classes of color correction approaches: parametric and non-parametric.

4.3.1 Model-based parametric approaches

4.3.1.1 Global color transfer

Model-based color correction techniques are parametric, and include global and local modeling approaches. Global modeling approaches assume the relation between the color of the target image and that of the source image can be described by a transform: $I_s = M \cdot I_t$, where M is a 3x3 matrix representing the mapping of the three color channels. Here M can be a diagonal matrix, an affine matrix or an arbitrary 3x3 matrix, corresponding to the diagonal model, the affine model and the linear model respectively [132, 50]. Various approaches can be used to estimate M, depending on applications and inputs.

Exposure compensation (or gain compensation) is the technique initially employed to address the color balancing problem in panorama stitching where the inputs are partially overlapped images. Nanda and Cutler first incorporated gain adjustment as part of the "AutoBrightess" function of their multi-head panoramic camera called RingCam [99]. Then, Brown and Lowe employed it in their well-known automatic panorama stitching software "AutoStitch" [11, 12], and Uyttendaele et al. [139] applied it on a block-by-block basis followed by spline interpolation of piece-wise gain parameters. Since the gain compensation technique only operates in the intensity channel but not in full color space, it actually corresponds to a particular diagonal model where the values in the main diagonal of M have to be same. This particular diagonal model was also adopted in some later work that combines exposure compensation and vignetting correction [51, 80].

Other more general approaches in global modeling include Tian et al.'s work [132] using histogram mapping over the overlap area to estimate the transformation matrix M, and Zhang et al.'s work [161] using the principal regions mapping to estimate M where the highest peaks in the hue histogram are designated as principal regions.

Given two general images where there is no overlap, Reinhard et al. [112] proposed a linear transformation based on the simplest statistics of global color distributions of two images: $g(C_t) = \mu_s + \frac{\sigma_s}{\sigma_t}(C_t - \mu_t)$, where (μ_s, σ_s) and (μ_t, σ_t) are the mean and standard deviation of the global color distributions of the source and target images in

the uncorrelated $l\alpha\beta$ color space and C_t is the color of a pixel in the target image. This work was widely used as the baseline approach by other color correction approaches [161, 146, 107, 130]. Xiao et al. [146] proposed an ellipsoid mapping scheme which extends Reinhard et al.'s work to correlated RGB color space. An et al. [2] discussed the linear transformation in YUV space.

4.3.1.2 Local color transfer

Global modeling usually provides only a rough mapping between the color of two images. In practice, many applications require a more deliberate mapping, which suggests local color transfer approaches. Tai et al. [130] proposed a local color transfer scheme based on probabilistic image segmentation and region mapping using Gaussian mixture models (GMM) and the EM algorithm. Xiang et al. [145] improved this work in the case that multiple source images are available for selection. For both of these approaches, after the local regions of the two images are matched, region-wise color transfer is performed using a weighted version of Reinhard's method [112]. Oliveira et al. [103] improved Tai et al.'s approach by using mean shift segmentation and color influence maps for local color transfer instead.

4.3.2 Modeless non-parametric approaches

Non-parametric methods assume no particular parametric format of the color mapping function and most of them use a look-up table to directly record the mapping of the full range of color/intensity levels. This look-up table is usually computed from 2D joint histogram of image feature correspondences or pixel pairs in the overlapped area of two images. Two points need to be kept in mind when one estimates a mapping function from the histogram: First, robust methods are usually needed because the data are prone to outliers and noise due to different lighting conditions, capture angles and reflection properties of scene objects. Second, the monotonicity property of the color/intensity levels in the estimated mapping function has to be maintained . All of the existing non-parametric mapping approaches can be distinguished from each other in how they accomplish these two points.

Yamamoto et al. [156] proposed using the joint histogram of SIFT feature matches between two neighboring views in a multi-view camera network. An energy minimization scheme in the 2D histogram space was proposed to get a robust estimation of the color mapping function and meanwhile maintain its monotonicity.

Jia and Tang [66] proposed a two-stage approach to handle robustness and monotonicity separately: in the first stage 2D tensor voting was used to suppress the noise and fill in the data gaps (i.e. places where no correspondences are available for some color levels). This produced an initial estimate of the mapping function. In the second stage, a heuristic local adjustment scheme was proposed to adjust the initial estimate and make the mapping monotonically increasing. In other work by the same authors [65], a Bayesian framework was used to recover the mapping function between a poorly exposed image and a blurred image.

Similar to Jia's work, Kim and Pollefeys [72] proposed a likelihood maximization scheme for robust estimation of the Brightness Transfer Function (BTF) from the 2D joint intensity histogram of two overlapped images. The approach operated in each of the three color channels separately. Dynamic programming was used to find a robust estimate under the monotonicity constraint. The estimated BTF was further used to estimate and remove the exposure difference and vignetting effect in the images.

Fecker et al. [40] proposed the use of cumulative color histogram mapping for color correction. They used a closest neighbor mapping scheme to select the corresponding color level of the source image to each level of the target. Using cumulative histogram-based mapping automatically satisfies the monotonicity constraint. The authors also suggested some special adjustment to the mapping of the border bins (i.e. the first and last bin) to avoid possible visual artifacts.

Pitié et al. [107, 108] proposed a quite different approach for color correction, called iterative color distribution transfer. This approach does not employ any explicit mapping function of the global color distribution, but relies on applying a sequence of simple conversions with respect to randomly projected marginal color distributions. Specifically, it treats the colors of an image as a distribution in a high dimensional space (usually of order 3), and repeatedly projects this high dimensional distribution into a series of random 1D marginal distributions using the Radon Transform. The color distribution of the target image is converted to that of the source image by repeatedly mapping its 1D marginal distributions to those of the source image until convergence. In [108], a post-processing technique for reducing the grain artifacts over the converted image was also proposed.

4.4 Performance evaluation of color correction approaches

4.4.1 Selection of approaches

To better understand the behavior of color correction approaches, I performed an extensive performance evaluation of popular color correction approaches in the context of multi-view image stitching. In this context, there is a trade-off of effectiveness and extendability for any color correction approach: *effectiveness* measures how genuinely the color mapping function (which is usually estimated from the overlapped area) transfers the color palette of the source image to the target image, and *extendability* measures how well this mapping extends to the non-overlapped areas of the target image without creating visual artifacts. I want to determine through evaluation how different approaches behave with respect to effectiveness and extendability given mosaic image pairs captured under different conditions. I focus on automatic approaches in my evaluation and exclude those approaches that need human invention or guidance (e.g. [101]) to complete the task. I also focus on techniques operating in the image domain for maximum generality, and thus those approaches that request pre-calibration information to operate in the radiance or irradiance domain (e.g. [37]) are not included. The inputs to my evaluation system are images of different appearance and unknown capture conditions, so those approaches for calibration of multi-view camera systems (e.g. [62, 157]) are also excluded.

Although various color correction techniques have been proposed in the last decade, there does not exist an extensive evaluation comparing the performance of these approaches. Most authors either only demonstrated their systems on a few self-selected example images or compared with very simple baseline approaches. My evaluation results should be of interest not only to the computer vision community, but also to other communities including computer graphics, image processing, and multi-view video processing.

Table 4.1 shows a list of nine color correction algorithms I selected for performance evaluation and comparison. These selected algorithms are either a widely used standard baseline in color alteration (e.g., [112]), or represent the most recent progress in color correction techniques. The selection includes both model-based parametric approaches and modeless non-parametric approaches, both approaches using global image statistics for estimating the color mapping function and approaches using local matches in the overlapped area, as well as approaches operating in various color spaces.

Index#	Name of Approach	Reference	Parametric/Non-Parametric	Local/Global	color space
1	gain compensation	Brown 2007 [12]	parametric	local	intensity
2	principal regions mapping	Zhang 2004 [161]	parametric	local	CIECAM97
3	tensor voting in joint image space	Jia 2005 [68]	non-parametric	local	RGB
4	global color transfer	Reinhard 2001 [112]	parametric	global	$l\alpha\beta$
5	global color transfer in correlated color space	Xiao 2006 [146]	parametric	global	RGB
6	local color transfer	Tai 2005 [130]	parametric	local	$l\alpha\beta$
7	brightness transfer function	Kim 2008 [72]	non-parametric	local	RGB
8	iterative color distribution transfer	Pitie 2005 [107]	non-parametric	global	RGB
9	cumulative histogram mapping	Fecker 2008 [40]	non-parametric	local	YCbCr

Table 4.1: Color correction approaches selected for performance evaluation and comparison.

The details about the various color spaces appearing in Table 4.1 can be found in Appendix A.

4.4.2 Test image sets

Both synthetic images and mosaic pairs selected from real image and video stitching tasks are used in my evaluation. The synthetic test image set includes 40 source/target image pairs. Each image pair is created in three steps: First, I selected images of poor exposure-levels from the Berkeley BSDS300 image segmentation dataset [88]. Then, for each of these selected images/frames an image processing tool [124] was used to auto adjust its color. This produces a new image of the same scene but differing in color properties (see Fig. 4.4 (c)-(d)). Finally, I visually compared the quality of the original image and the new image, and designated a clip from the image with better quality as the source image and another clip from the other image as the target image (If the two images are of similar quality then the assignment is random). When a color correction algorithm is executed with these synthetic image pairs, its ability to increase the quality of an image by altering its color distribution is thus tested.

The real test image set includes 30 example mosaic image pairs collected from various sources, including image frames from multi-view video applications, scenic or object photos taken with/without flash lighting or under different capture modes, and aerial image clips of the same place but at a different time (Fig. 4.4 (a)-(b)). For each of these real image pairs, the image which looks more natural is designated the source image and the other one the target image.

Each source/target image pair in the test image sets is partially overlapped and AutoStitch is used to find the geometric registration between them before color correction is performed. Figure 4.4 shows a few examples from the two test image sets.

(a) Real scene#6 (b) Real scene#13

(c) Synthetic scene#13 (d) Synthetic scene#38

Figure 4.4: Example test image pairs. (a)-(b) example real mosaic image pairs, (c)-(d) example synthetic image pairs. For each pair, left is source image and right is target image.

4.4.3 Evaluation criteria

A recently proposed theory on image quality evaluation is that from the perceptual point of view the goodness of a color altered target image should show both color coherence and structural coherence, since color correction may not only change the color of the target image but also the structure [141]. Based on this theory, I propose a criterion to evaluate the quality of transferring the color of a source image s to a target image t, which results in a converted image r. The proposed evaluation criterion includes two components: color similarity $CS(r, s)$ between the source image s and the transferred image r, and structure similarity $SS(r, t)$ between the target image t and the transferred image r. The color similarity $CS(r, s)$ is defined as:

$$CS(r, s) = PSNR(\hat{r}, \hat{s}) \tag{4.1}$$

where $PSNR = 20 \cdot \log_{10}(L/RMS)$ is the peak signal-to-noise ratio [36]. L is the largest possible value in the dynamic range of an image, and RMS is the root mean square difference between two images. $\hat{r}$ and $\hat{s}$ are the overlapped area of r and s respectively. The higher the value of $CS(r, s)$ the more similar the color between the two images r and s.

The structure similarity $SS(r, t)$ is defined as:

$$SS(r, t) = SSIM(r, t) \tag{4.2}$$

where $SSIM(r,t) = \frac{1}{N}\sum_{j=1}^{N} SSIM(a_j, b_j)$ is the Structural SIMilarity (SSIM) index [141]. SSIM divides an input image into a set of non-overlapping, evenly distributed local patch windows, and N is the number of local windows for that image. a_j and b_j are the image contents at the jth local window of r and t respectively. SSIM of a local window is defined as a combination of local luminance, contrast and structure components [141]:

$$SSIM(a,b) = [l(a,b)]^{\alpha} \cdot [c(a,b)]^{\beta} \cdot [s(a,b)]^{\gamma} \tag{4.3}$$

where $l(a,b) = \frac{2\mu_a\mu_b + A_1}{\mu_a^2 + \mu_b^2 + A_1}$, $c(a,b) = \frac{2\sigma_a\sigma_b + A_2}{\sigma_a^2 + \sigma_b^2 + A_2}$, $s(a,b) = \frac{\sigma_{ab} + A_3}{\sigma_a\sigma_b + A_3}$. μ_a and μ_b are the mean luminance values of windows a and b respectively; σ_a and σ_b are the standard variance of the of windows a and b respectively; σ_{ab} is the autocovariance between a and b. Here A_1, A_2 and A_3 are small constants to avoid divide-by-zero error, α, β and γ control the weighting between the three components. In my implementation I use the default settings recommended by [141]: $A_1 = (0.01 * L)^2$, $A_2 = (0.03 * L)^2$, $A_3 = A_2/2$, $L = 255$ for images of dynamic range $[0, 255]$ and $\alpha = \beta = \gamma = 1$. The higher the value of $SS(r,t)$ the less difference between the structure of r and t, and $SS(r,t) = 1$ if there is no structure difference.

The color and structure similarities measure the effectiveness and extendability of a color correction approach respectively. To give the reader a perception of these measures, Figure 4.5 shows a real example.

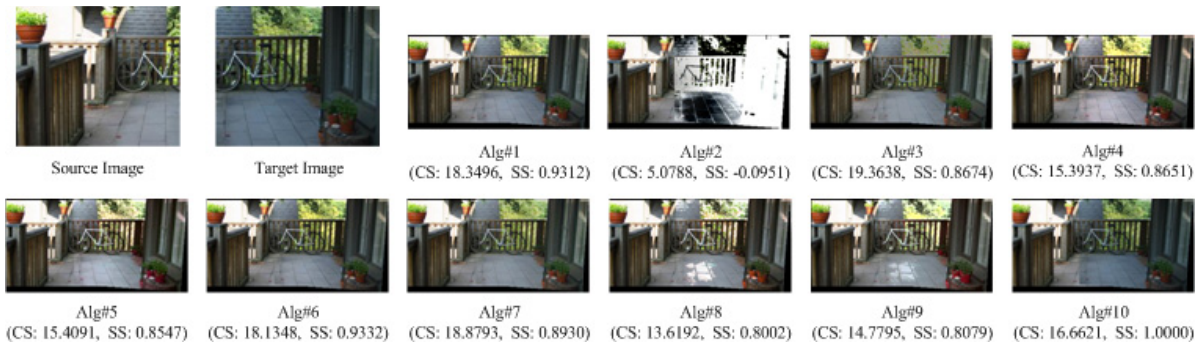


Figure 4.5: Color correction results on a real test image pair (real scene#16) and the corresponding CS and SS scores. The baseline approach alg#10 shows the basic CS and SS scores corresponding to directly stitching the source and target image together without taking any color correction measures. In this example, alg#8 and alg#9 produce out-of-gamut errors in the overlapped area, so they obtain lower CS scores than the baseline. alg#1 and alg#3 make the color in the overlapped area more balanced (seams become less obvious), so they obtain higher CS scores than the baseline. However, comparing to alg#1, the structure of the transferred image of alg#3 (over the tree area above the bike) is more distorted with respect to the original target image, so it gets a lower SS score than does alg#1. Note here and in my evaluation no advanced image blending techniques but simple averaging is applied over the overlapped area in order for the color correction effect to be evaluated independently. This source/target image pair is originally from [14] and used in my evaluation with permission.

4.4.4 Pre-processing

One problem that must be considered in my evaluation is that vignetting may exist in the test images. The traditional approach is to solve vignetting removal and color correction simultaneously, but recent progress allows the two to be decoupled so vignetting can be removed separately for a single image [163, 164]. Since the focus of this study was color correction, I used Zheng's approach [164] to remove vignetting effects that might exist in the test images before using the images to test selected color correction approaches.

4.4.5 Implementation details and parameter settings

I downloaded the source code for the iterative color distribution transfer approach [107] and SSIM [141] from the authors' websites and used them directly in my evaluation. I also implemented the other eight color correction approaches, and the proposed "color+structure" evaluation criterion using MATLAB 7.7.0 (R2008b). I used the open source code OpenTVF [138] for the 2D tensor voting technique in my implementation of the tensor voting-based color correction approach [68].

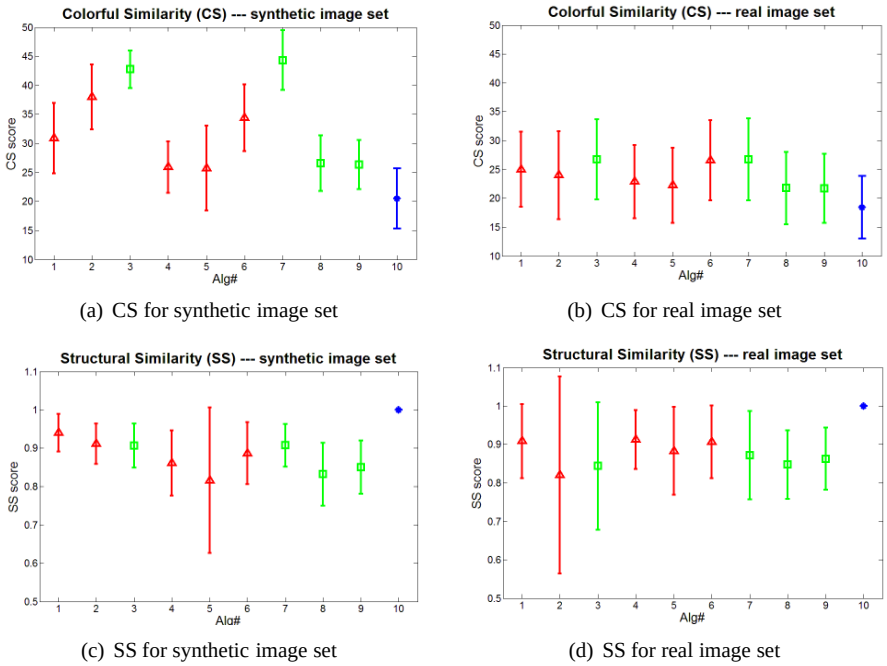
In my implementation, most of the approaches and evaluation criteria use the same parameters as stated in the original papers. The only exception is the principal regions mapping approach [161]: the original paper prefers to use three principle regions to construct order-2 polynomial mapping functions, while in my implementation I conservatively used only two principle regions which simplifies the mapping functions to an affine model. This is because in practice I found higher degree mapping functions are more prone to out-of-gamut errors (i.e. some transferred colors go out of the display range of the RGB color model) [73].

4.4.6 Evaluation results

4.4.6.1 Experimental data and analysis

I tested all of the nine selected approaches on both the synthetic image set and the real image set by computing both CS and SS scores of the correction results. Figure 4.6 and Table 4.2 show statistics of the these scores over the 40 synthetic image pairs and 30 real image pairs respectively. Here 'alg#10' is a "virtual baseline approach" added in for purely comparison purposes. It corresponds to "no correction performed", that is, computing the CS/SS scores

between the source image and the target image directly with no color adjustment.



(a) CS for synthetic image set

(b) CS for real image set

(c) SS for synthetic image set

(d) SS for real image set

Figure 4.6: Errorbar statistics of CS and SS scores for all of the nine selected color correction algorithms. Red triangle: parametric approach; Green Square: non-parametric approach; Blue Star: the baseline approach.

		Alg#1	Alg#2	Alg#3	Alg#4	Alg#5	Alg#6	Alg#7	Alg#8	Alg#9	Alg#10
syn	μ_{CS}	30.8825	38.0145	42.7686	25.9149	25.6988	34.3836	44.3281	26.5597	26.3066	20.5134
	σ_{CS}	6.0941	5.5917	3.2322	4.4301	7.2960	5.7530	5.1852	4.7586	4.2376	5.2041
real	μ_{CS}	24.9928	23.9749	26.6989	22.8586	22.2535	26.5897	26.7329	21.7544	21.6938	18.4429
	σ_{CS}	6.5170	7.6458	6.9387	6.3581	6.5077	6.9391	7.1134	6.2852	5.9989	5.4203
syn	μ_{SS}	0.9404	0.9115	0.9064	0.8609	0.8159	0.8866	0.9075	0.8320	0.8507	1.0000
	σ_{SS}	0.0492	0.0523	0.0573	0.0847	0.1897	0.0808	0.0554	0.0818	0.0695	0
real	μ_{SS}	0.9085	0.8205	0.8440	0.9127	0.8830	0.9064	0.8716	0.8478	0.8625	1.0000
	σ_{SS}	0.0962	0.2561	0.1660	0.0771	0.1144	0.0947	0.1149	0.0893	0.0810	0

Table 4.2: Mean (μ) and standard deviation (σ) statistics of CS and SS scores for the nine selected color correction algorithms.

First of all, the data tell us the synthetic image set and the real image set are pretty different. The CS score range of alg#10 (the baseline) is 20.5134 ± 5.2041 for the synthetic set, and is 18.4429 ± 5.4203 for the real image set. Since CS is built upon PSNR which is built upon RMS errors, this data shows there are more intra-pair and inter-pair differences in the real image set, and thus it is more challenging than the synthetic image set. Considering this factor that my test sets are different and my goal is to serve real applications, in the following I use the data from the real image set as main reference. From the data I find:

As a very simple and widely used approach in image and mosaic stitching, alg#1 (gain compensation) performs pretty well: this is reflected as relatively high (rank 4) mean SS scores with small variance, and good (rank 2) CS scores.

The biggest problem for alg#2 (principle regions mapping) is stability. It has pretty good performance on the synthetic image set, but very poor performance on the structural score on real image set. The possible explanation for this is that it simply designates peaks in hue histograms as principle regions, which might be too simple to work well for real scenes with complex contents.

Alg#5 (global color transfer in correlated color space) and alg#6 (local color transfer) are both variants to alg#4 (global color transfer). Compared to alg#4 which operates in uncorrelated $l\alpha\beta$ color space, alg#5 operates in correlated RGB color space, which leads to deteriorations in both color correction effectiveness and extendability. Alg#6 makes use of local spatial information to guide color transfer, which leads to a gain in color correction effectiveness and similar extendability.

Alg#3 (tensor voting in joint image space) and alg#7 (brightness transfer function) are representatives of non-parametric approaches that build the color transfer function upon exact mapping of the full range of color/intensity levels. Compared to alg#8 (iterative color distribution transfer) and alg#9 (cumulative histogram mapping) that use implicit or rough mapping, they show not only much better color correction effectiveness, but also similar (or slightly better) extendability.

From the perspective of class-level comparison, non-parametric approaches have better color transfer effectiveness but less extendability than parametric ones in general. But this is by no means absolute for individual approaches: some parametric approaches that make use of local information, such as alg#6 (local color transfer), have quite close performance in color transfer effectiveness as that of the most capable non-parametric approaches such as alg#3 (tensor voting in joint image space) and alg#7 (brightness transfer function).

4.4.6.2 Analysis of the worst cases

There are two questions of interest to us: 1) is there a common factor in practice that may affect the performance of all the approaches, and 2) what is the most challenging scene for all of the approaches. To answer these questions, I have found the five real scenes (i.e. image pairs) on which the nine selected approaches achieve the lowest average

CS scores and SS scores (see Table 4.3).

	CS					SS				
scene#	13	16	7	6	19	13	6	23	16	22
average score	13.0501	15.4465	15.9166	16.1193	17.5904	0.5909	0.6690	0.7155	0.7622	0.7627
baseline score	16.7105	11.9117	12.6996	16.6621	12.9240	1.0000	1.0000	1.0000	1.0000	1.0000
gain (percentage)	-28.05%	22.88%	20.21%	-3.37%	26.53%	-40.91%	-33.10%	-28.45%	-23.78%	-23.73%

Table 4.3: The real test scenes on which the nine selected approaches achieves the lowest average CS and SS scores.

Table 4.3 shows that scenes #6, #13 and #16 are in the worst case lists of both CS and SS scores. Especially, on scene #13 and #6 the nine selected approaches on average suffer deterioration of both CS and SS scores. Figures 4.4(a), 4.4(b) and 4.5 show these three scenes. It is easy to discover that all of them are affected by extreme lighting conditions: The target image of scene #6 contains saturated areas, which may cause problems in both calculating the color mapping function and in extending this mapping to the non-overlapped area. Scene #13 has saturated parts in the non-overlapped region of the target image, which may cause problems when extending the color mapping function to this area. The saturated area in scene#16 is in the non-overlapped part of the source image, which might not cause any problems in calculating and extending the color mapping function, but at least partly shows that the lighting conditions are very different between the source and target images.

4.4.7 Discussion and conclusions

To the best of my knowledge, my work is the first work so far that performs an extensive, systematic and quantitative evaluation of the performance of color correction approaches in the context of automatic multi-view image and video stitching. My evaluation and comparison of the approaches, has yielded a number of useful observations and conclusions.

From the perspective of color transfer effectiveness, both the non-parametric approaches of alg#3 (tensor voting in joint image space) and alg#7 (brightness transfer function) and the parametric approaches of alg#1 (gain compensation) and alg#6 (local color transfer) are superior according to the experimental data. From the perspective of extendability, parametric approaches (including alg#1 (gain compensation), alg#4 (global color transfer) and alg#6 (local color transfer)) are generally better and more stable than non-parametric ones. It is also worth mentioning that non-parametric approaches are much more complex than parametric ones. Alg#3 (tensor voting in joint image

space) in particular is much slower than the other eight, according to my un-optimized implementation.

Considering all the above factors (effectiveness, extendability, stability and speed), I think alg#1 (gain compensation) and alg#6 (local color transfer) could be the first options to try for a general image and video stitching application in practice. Both of these two approaches are simple, fast, effective, and general. It is interesting to notice that to my best knowledge alg#1 may be one of the earliest color correction approaches developed for mosaic and panorama stitching [127], while the direct predecessor of alg#6, alg#4 (global color transfer), is widely used as the baseline approach in color transfer research. After alg#1 and alg#6, alg#3 (tensor voting in joint image space) and alg#7 (brightness transfer function) may also be good choices.

Based on my experience on studying various color correction approaches and implementing and evaluating nine of them, I think that future work on color correction approaches faces the following problems: 1) How to process extreme inputs like over-saturated parts of the input images that may affect the calculation of the color mapping function, 2) the handling of out-of-gamut errors, and 3) how to intelligently extend the color mapping function calculated from the overlapped area to non-overlapped regions. On the last problem, making use of image segmentation results to selectively extend the mapping might be a good exploration direction.

4.5 Color correction for n-view image stitching

Although two-view image stitching with a designated reference view may suffice for many applications (e.g. 1D horizontal panorama stitching with a rotating camera), in practice there are many cases of more general stitching tasks with multiple input images ($n \geq 2$) and unknown reference view. When the reference view is unknown and each view is free to change its color under correction, a globally optimized solution is required to reach the balancing point between all the input images. The number of input images can potentially be large ($\geqslant 2$) and good scalability becomes the primary consideration. For those approaches described above, only Brown's [12] and Xiong's [149] approaches are designed bearing these two requirements in mind, and are thus tailored for general n-view image stitching. Surprisingly, both approaches are based on the simplest diagonal model. One thus cannot help thinking: is it possible to increase the power of the model without sacrificing scalability?

Traditional color correction approaches mainly operate in the intensity/color value domain, but ignore the dynamic range difference (i.e. contrast difference) between two input images. If one wants to take the contrast difference

of the input images into account and perform a contrast balancing, gradient domain manipulation techniques must be used. There is extensive prior work on dynamic range compression/expansion using gradient domain manipulation techniques. For example, Fattal et al. [39] compressed the contrast of an input image using an exponential function, and then reconstructed the output image by solving a Poisson equation with Neumann boundary condition. Levin et al. [75] discussed how to apply gradient domain reconstruction for image blending, however, the gradients of each input image are not changed. When the contrast difference between two images is too large, the reconstruction may fail to get reasonable results [159]. In view of this deficiency, I propose to balance the dynamic range of the input images before blending to avoid any reconstruction artifacts. To this end, I propose to do large-scale balancing in the perceptual framework of contrast processing proposed by Rafal et al. [86] in which the relationship between the perceptual contrast of different input images is multiplicative.

In this section, I describe a new hybrid color correction approach for general n-view image stitching that performs color and contrast correction simultaneously and with good scalability. My approach performs color correction in the intensity/color domain, but has a more powerful affine model to replace the simple diagonal model used by Brown and Xiong. The adoption of the affine model increases the power of the color transform model without sacrificing scalability. In addition, my method performs contrast correction in a perceptual space of contrast processing [86] where image dynamic range adjustment is readily scalable to the general n-view case. Finally, I merge the intermediate results of these two steps and reconstruct the final output by solving an extended Poisson equation with Neumann boundary condition. This operation is performed for each view, and the final mosaic/panorama is stitched together from the n merged results using standard image registration and stitching operations.

4.5.1 Approach overview

Given n input images, my approach performs color correction and contrast correction for each image separately. Color correction operates in the image domain and gives n color-corrected output images; contrast correction operates in the gradient domain and gives n corrected gradient maps. I use an extended Poisson reconstruction [147] to merge these two kinds of intermediate results, to yield n output images. Finally, these results are stitched and blended to form the final panorama. Fig. 4.7 shows the outline of my approach.

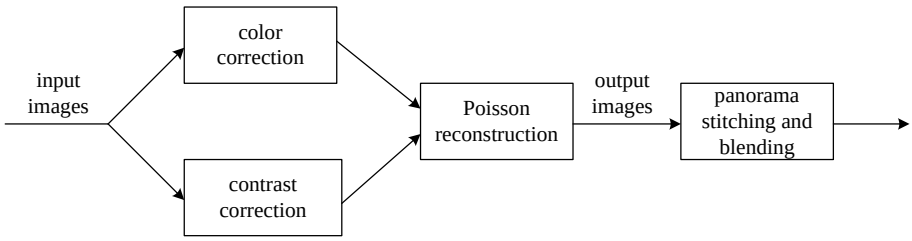


Figure 4.7: Outline of my approach.

4.5.2 Affine-transform based multi-view color correction

Probably the most widely used color correction approach for multi-view image stitching is gain compensation which is the technique used by several commercial products including AutoStitch [12] and Nokia Panorama [149]. Gain compensation assumes there is a pure scaling relationship between the intensity of the same pixel u in two overlapping images:

$$g_i \cdot I_i(u) = g_j \cdot I_j(u) \tag{4.4}$$

where $I_i(u)$ and $I_j(u)$ are the intensities of the pixel in the two images I_i and I_j, and g_i and g_j are the corresponding gain values. This gain compensation model provides the simplest way to balance the colors of the input images in two-view image stitching. However, its performance in practice is just average, as shown in my work on performance evaluation of color transfer methods (see Sec. 4.4). In my evaluation, the best performance was achieved by Tai et al.'s local color transfer approach [130] which performs statistical color transfer [112] within inter-frame region-pairs. However, a problem with the statistical color transfer technique is that it was developed for color transfer between only two views. For general image stitching I need to extend it to multiple views.

The original statistical color transfer approach adjusts the color of the target image I_t to be close to that of the source I_s as follows [112]:

$$\widehat{C}_t(u) = \bar{C}_s + \frac{\sigma_s}{\sigma_t}(C_t(u) - \bar{C}_t) = \frac{\sigma_s}{\sigma_t}C_t(u) + (\bar{C}_s - \frac{\sigma_s}{\sigma_t}\bar{C}_t) \tag{4.5}$$

where u is a pixel, $C_t(u)$ and $\widehat{C}_t(u)$ are its colors before and after the transfer. $\bar{C}_s$, $\bar{C}_t$, σ_s and σ_t are mean and standard deviation of the colors of the source and target images. This is an **affine transform** whose parameters are composed from basic image statistics. Note that this transform is very general and does not require the images I_s and I_t to have any spatial correspondence. Given that my target application is image stitching in which the spatial overlap

between neighboring views can be exploited to guide color transfer, I localize the computation of the parameters in the transform as follows:

$$\widehat{C}_t(u) = \bar{C}_s(\Omega_{st}) + \frac{\sigma_s(\Omega_{st})}{\sigma_t(\Omega_{st})}(C_t(u) - \bar{C}_t(\Omega_{st})) \tag{4.6}$$

where $\Omega_{st} = \Omega(s,t)$ represents the overlap region between I_s and I_t. $(\bar{C}_s(\Omega_{st}), \sigma_s(\Omega_{st}))$ and $(\bar{C}_t(\Omega_{st}), \sigma_t(\Omega_{st}))$ are the mean and standard deviation of the colors of I_s and I_t in Ω_{st}.

I choose the local statistical color transfer technique described by Eq.(4.6) as my base technique for direct color correction in the image domain. First, I try to extend it to n-view image stitching with $n \geq 2$. Under the affine transform the relationship between the colors of the same pixel u in two overlapped images is:

$$g_i \cdot I_i(u) + k_i = g_j \cdot I_j(u) + k_j \tag{4.7}$$

$I_i(u)$ and $I_j(u)$ represent the value of a pixel u in I_i and I_j in one of the color channels. (g_i, k_i) and (g_j, k_j) are corresponding gain factors and offsets in the channel.

Based on Eq. (4.7), I define the following global optimization problem for affine transform-based color correction of n views:

$$\min E_1 = \frac{1}{2}\sum_{\substack{i,j=1 \\ j\neq i}}^{n} \sum_{\substack{u_i \in \Omega(i,j) \\ u_j \backsim H_{ij} u_i \\ \Omega(i,j)\neq\emptyset}} (g_i I_i(u_i) - g_j I_j(u_j) + k_i - k_j)^2 \tag{4.8}$$

where H_{ij} is the geometric homography from I_i to I_j, $\backsim$ means equals after normalization, $\Omega(i,j)$ is the overlap between I_i and I_j. Note I have the condition $j \neq i$ in my model because transferring the color of an image to itself is meaningless. As in [12, 149], I want to approximate pixel-wise correspondence by region-wise correspondence in the above formulation in order to both simplify the computation and make the algorithm robust to spatial alignment errors. Thus I get the following approximately equivalent problem:

$$\min E_2 = \frac{1}{2}\sum_{\substack{i,j=1 \\ j\neq i}}^{n} N_{ij}\left(\frac{(g_i \bar{I}_i(\Omega_{ij}) - g_j \bar{I}_j(\Omega_{ij}) + k_i - k_j)^2}{\sigma_N^2} + \frac{(1-g_i)^2}{\sigma_g^2} + \frac{k_i^2}{\sigma_k^2} \right) \tag{4.9}$$

where $(1-g_i)^2/\sigma_g^2$ and k_i^2/σ_k^2 are prior terms to avoid trivial solutions. σ_N, σ_g and σ_k are normalization factors. Compared to Eq. (4.8), the addition of $N_{ij} = |\Omega_{i,j}|$ in Eq. (4.9) incorporates the $\Omega(i,j) \neq \emptyset$ case and gives more weight to prominent images with larger overlap regions. This is a quadratic objective function in the affine coefficients

(g_i, k_i), $(i = 1, 2, \cdots, n)$ which can be solved in a closed form by setting the derivatives to zero. Specifically,

$$\frac{\partial E_2}{\partial g_i} = \left(\sum_{\substack{j=1 \\ j \neq i}}^{n} N_{ij} \left(\frac{\bar{I}_{ij}^2}{\sigma_N^2} + \frac{1}{\sigma_g^2} \right) \right) \cdot g_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(\frac{N_{ij} \bar{I}_{ji} \bar{I}_{ij}}{\sigma_N^2} \cdot g_j \right) + \left(\sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{N_{ij} \bar{I}_{ij}}{\sigma_N^2} \right) \cdot k_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(\frac{N_{ij} \bar{I}_{ij}}{\sigma_N^2} \cdot k_j \right) - \frac{1}{\sigma_g^2} \sum_{\substack{j=1 \\ j \neq i}}^{n} N_{ij},$$

$$(\text{for } i = 1, \cdots, n) \quad (4.10)$$

$$\frac{\partial E_2}{\partial k_i} = \left(\sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{N_{ij} \bar{I}_{ij}}{\sigma_N^2} \right) \cdot g_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(\frac{N_{ij} \bar{I}_{ji}}{\sigma_N^2} \cdot g_j \right) + \left(\sum_{\substack{j=1 \\ j \neq i}}^{n} \left(N_{ij} (\frac{1}{\sigma_N^2} + \frac{1}{\sigma_k^2}) \right) \right) \cdot k_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(\frac{N_{ij}}{\sigma_N^2} \cdot k_j \right),$$

$$(\text{for } i = 1, \cdots, n) \quad (4.11)$$

By setting $\frac{\partial E_2}{\partial g_i} = 0$ and $\frac{\partial E_2}{\partial k_i} = 0$ I obtain a linear system with $2n$ equations in total to solve for the $2n$ unknown variables (g_i, k_i), $(i = 1, 2, \cdots, n)$. There is an exact solution for this system which can be solved via standard matrix factorization routines (e.g. LU decomposition). The input images are then corrected using the obtained parameters:

$$I_i' = g_i \cdot I_i + k_i \quad (4.12)$$

This procedure is run separately for each individual color channel.

4.5.3 Multi-view contrast correction in a conceptual space

The goal of contrast correction among different views is to unify the dynamic ranges of different input images before stitching the images into a panorama. It may not be easy for color correction operating in the image domain to achieve this goal [109], so that contrast corrections operating in the gradient domain have been employed (e.g., [39]). There are many gradient-domain approaches to compress the dynamic range of an image, but I follow the approach operating in the perceptual space of contrast processing proposed by Rafal et al. [86]. The benefit of operating in this contrast space, but not directly on the device contrast [86], is that this is a linearized space in which the nontrivial dynamic range compression operation is reduced to a linear scaling. For n-view image stitching, the (perceptual) contrast difference between different views can also be modeled by a relative scaling relationship. Specially, given n input images $I_i (i = 1 \cdots n)$, I first compute the contrast map of each color channel $C_{ij} (j = 1, 2, 3)$ of I_i. The contrast is defined as

$$(G_x, G_y) = \text{gradient}(log_{10}(C_{ij})) \quad (4.13)$$

(a) Image stitching

(b) Color corrected stitching

(c) Color and contrast corrected

Figure 4.8: Contrast domain correction. From left to right panoramas are constructed: (a) without color and contrast correction, (b) with color but not contrast correction and (c) with both color and contrast correction.

where G_x and G_y are computed by forward difference. Then, I convert the device contrast map to the corresponding perceptual contrast map: $(R_x, R_y) = T(G_x, G_y)$, where $R = T(G) = 54.09288 \cdot G^{0.41850}$ as described in [86]. The advantage of conversion to this perceptual space is that any compression of the dynamic range of an image can be modeled by a simple scaling relationship. That is, to increase/decrease the dynamic range of I_i, I just need to multiply (R_x, R_y) by a scale factor s:

$$(R_x{}', R_y{}') = s \cdot (R_x, R_y) \tag{4.14}$$

and then apply $T^{-1}(\cdot)$ on $(R_x{}', R_y{}')$ to get the output gradient map. In the original paper of [86], the target application is mono-view image manipulation and s is designated by the user. Here, my target application is multi-view image stitching, and I want the dynamic range of all the input images to be scaled to the same level by determining s automatically for each view. To achieve this I compute the scale factor s_{ij} from the mean absolute values of $\{R_x, R_y\}$ given images I_i and I_j:

$$s_{ij} = \frac{mean(|R_i|)}{mean(|R_j|)} \tag{4.15}$$

where $R = \{R_x, R_y\}$ is the union of R_x and R_y, $|\cdot|$ means absolute value. As for Eq.(4.8), computing the scale factors for n input images is a global optimization process:

$$\min E_3 = \frac{1}{2}\sum_{\substack{i,j=1\\ j\neq i}}^{n} \sum_{\substack{u_i\in\Omega(i,j)\\ u_j \backsim H_{ij} u_i\\ \Omega(i,j)\neq\emptyset}} (s_i R_i(u_i) - s_j R_j(u_j))^2 \tag{4.16}$$

Again I use region mean to approximate pixel values, and get the following approximately equivalent problem:

$$\min E_4 = \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} N_{ij}\left(\frac{(s_i\bar{R}_i(\Omega_{ij}) - s_j\bar{R}_j(\Omega_{ij}))^2}{\sigma_N^2} + \frac{(1-s_i)^2}{\sigma_s^2}\right) \tag{4.17}$$

where $(1-s_i)^2/\sigma_s^2$ is a prior term to avoid trivial solutions. σ_N and σ_s are normalization factors. The above optimization problem can be solved using the linear system formed by setting $\frac{\partial E_4}{\partial s_i} = 0$. Using the computed $s_i (i = 1\cdots n)$, the adjusted perceptual contrast map is $R' = s \cdot R$ (Eq.(4.14)), and the corresponding device contrast map is $(G'_x, G'_y) = 10^{T^{-1}(R'_x, R'_y)}$, where $G = T^{-1}(R) = 7.2232\cdot 10^{-5}\cdot R^{2.3895}$ as described in [86]. Fig. 4.8 illustrates the idea.

4.5.4 Reconstruction of the output

The last step is to merge the intermediate color-correction results I' (Sec. 4.5.2) and the adjusted gradient fields $(G_x{}', G_y{}')$ (Sec. 4.5.3) together and reconstruct the final output image O. This can be done by solving an extended Poisson equation including both the data term and the first-order derivative term, as in [147, 9]:

$$\min E = \lambda_1 \sum_p (O_p - I_p{}')^2 + \lambda_2 \sum_p [(\frac{\partial O_p}{\partial x} - G_x{}')^2 + (\frac{\partial O_p}{\partial y} - G_y{}')^2] \tag{4.18}$$

which means the output image O is expected to preserve the color of the intermediate image I' and the gradient fields of $(G_x{}', G_y{}')$. The two sums are taken over pixels in the relevant images. This minimization problem can be defined as the solution of the following linear system with Neumann boundary conditions:

$$\lambda_1 O + \lambda_2 \triangle O = \lambda_1 I' + \lambda_2 div(G_x{}', G_y{}') \tag{4.19}$$

where the Laplacian

$$\triangle O = O(x+1, y) + O(x-1, y) + O(x, y+1) + O(x, y-1) - 4O(x, y) \tag{4.20}$$

and the divergence

$$div(G_x{}', G_y{}') = G'_x(x,y) - G'_x(x-1,y) + G'_y(x,y) - G'_y(x,y-1) \tag{4.21}$$

The detailed construction and solution of the linear system of Eq.(4.19) is explained in Appendix B. The reconstruction described above is performed for each input image, and the final mosaic/panorama is stitched from the n reconstructed images using standard image registration and blending operations.

4.5.5 Stitching examples and comparison

(a) Left image. (b) Right image. (c) Opt. seam

Figure 4.9: A challenging two-view image stitching example. (a)-(b) are input images, and (c) is the optimum seam computed using dynamic programming from the difference map of (a) and (b).

I have tested the proposed approach with various multi-view image stitching examples, and compared it to gain compensation [12] and Xiong's approach [149]. For all the following examples, I ran both gain compensation and my approach in the RGB color space, and used the same common parameters: $\sigma_N = 10.0$ and $\sigma_g = 0.1$ as suggested in [12]. For Xiong's approach, I ran it in the YCbCr color space as proposed and set $\sigma_N = 10.0$ and $\sigma_g = 0.5$ for all examples. Particular parameters of my approach were set as $\sigma_k = 10, \lambda_1 = 1.0$ and $\lambda_2 = 0.1$ for all experiments.

I start my evaluation from the simpler case of two-view image stitching with no reference. Figs. 4.9 and 4.10 show a stitching example in which no reference view is designated. This is a very challenging example because the color difference between the two images is very large. As expected, none of the three image blending approaches illustrated (linear blending (Fig. 4.10(a)), multi-layer blending [16] (Fig. 4.10(b)) nor Poisson blending [75](Fig. 4.10(c))) can produce satisfactory results by itself. The large color difference even distorts the computation of the optimum boundary seam (Fig. 4.9(c)), which causes an abrupt transition in the Poisson blending result (Fig. 4.10(c)). This

(a) Linear blending. (b) Multi-layer blending. (c) Seam cut/Poisson.

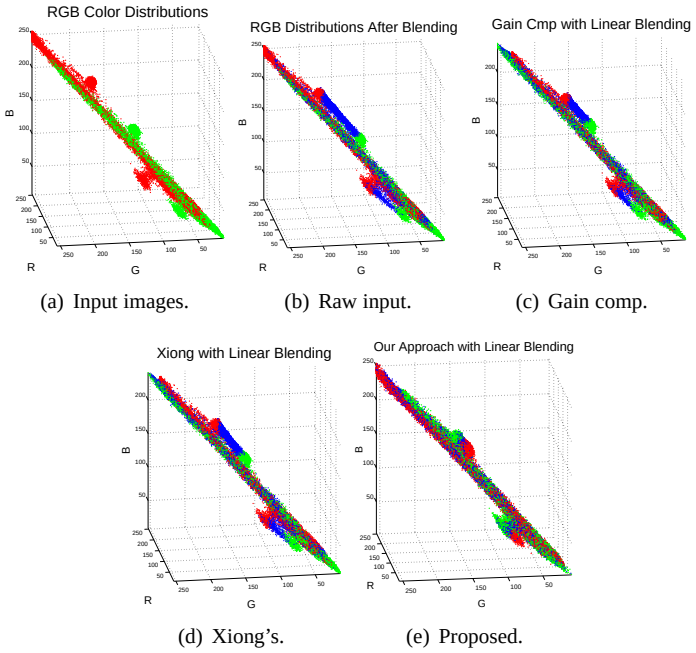
(d) Gain comp/lin blend. (e) Xiong/lin blend. (f) Proposed/lin blend.

Figure 4.10: (a)-(c) show the results using various blending approaches directly. (d)-(f) show the results obtained by combining different color correction approaches with simple linear blending.

figure further shows that a combination of color correction and blending can give more visually appealing results (compare Figure 4.10(d) to Fig. 4.10(f)). Here I compare the proposed approach with Brown's gain compensation approach [12] and with Xiong's approach [149]. As I described earlier, both of these approaches are based on the diagonal model, which is less effective at modeling the color difference than the affine model of my approach. The superiority of using the affine model can be seen by comparing the final panorama results.

Fig. 4.11 plots the color distributions in the overlap of the images in Fig. 4.9 before and after color correction. Although the two point clouds (in red and green respectively) have a similar shape, they are some distance away from each other which causes the images look different. Color correction should help reduce this distance and make the two distributions better matched. Comparing the distributions in Fig. 4.11(c), Fig. 4.11(d) and Fig. 4.11(e), we see that the proposed approach is more effective than gain compensation or Xiong's approach at improving the overlap of the point clouds. My approach is most successful at reducing the difference between the two input point clouds (red and green),and therefore achieves the most compact blending (i.e. leaves least burden to the blending algorithm).

Figs. 4.12 to 4.15 show more stitching examples with multiple input images. In Fig. 4.12, the rightmost input image is brighter than the other two which causes sharp transitions if directly blended. Also, the strong sunlight makes the outline of some far-field buildings fade away. My approach successfully corrects the color difference and leads to a



(a) Input images. (b) Raw input. (c) Gain comp.

(d) Xiong's. (e) Proposed.

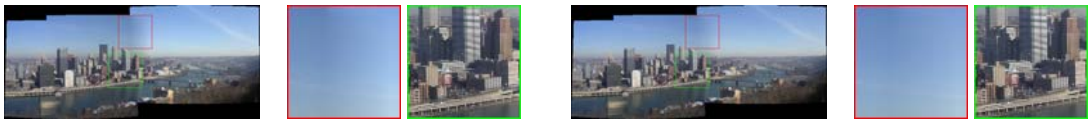
Figure 4.11: Overlap color distributions. Red and green point clouds are color distributions of the left and right images respectively. The blue point clouds in (b)-(e) are distributions for linear blending.

more coherent blending result (see the red-rectangled region and its enlarged view). My integrated contrast correction enhances the green-rectangled region.

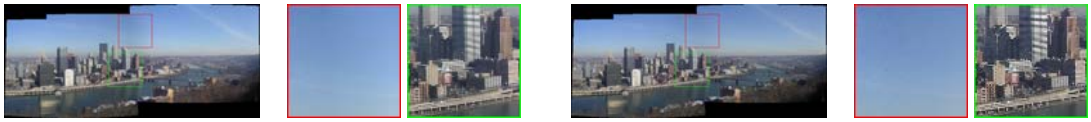
In Fig. 4.13, the leftmost input image is a zoomed-in close-up view of the city buildings, thus it is darker than the other four input images. If directly blended, this dark image causes sharp transitions around its boundaries. In this example, my approach (Fig 4.13(e)) is more effective than gain compensation (Fig 4.13(c)) and Xiong's approach (Fig 4.13(d)) at reducing this color difference and leads to a more natural output. In addition, simultaneous contrast correction enhances the contrast within the green-rectangled region.

Fig. 4.14 is an example illustrating that different capture angles cause significant color difference in the input images. From left to right, the brightness of the images gradually increased. Although this does not cause any boundary issues when blended, it causes the stitched scene to have a gradual color transition across the panorama, which is not what the scene looks like in reality (the real scene has a pretty even brightness distribution). This kind of color difference can only be solved in a global optimization framework, and my approach (Fig 4.14(e)) does a better job than gain compensation (Fig 4.14(c)) or Xiong's approach (Fig 4.14(d)) at setting an even the brightness

(a) Input images



(b) linear blending only. (c) Gain comp + linear blend.



(d) Xiong + linear blend. (e) My approach + linear blend.

Figure 4.12: Three-view stitching example. For (b)-(d), regions marked by red and green rectangles are enlarged on the right.

distribution for the whole panorama.

Fig. 4.15 is another example where the contrast of the stitching result is enhanced.

Method	Fig. 4.10	Fig. 4.12	Fig. 4.13	Fig 4.14	Fig. 4.15
Input image	14.2210	19.6846	18.8278	24.3637	16.7918
Gain comp.	17.4996	21.6171	21.2088	25.4696	21.8353
Xiong	16.6463	22.6176	21.0619	25.3819	19.4471
Proposed	20.3476	22.6580	21.9916	25.4521	21.7287

Table 4.4: Average PSNR scores for the stitching examples before and after color correction.

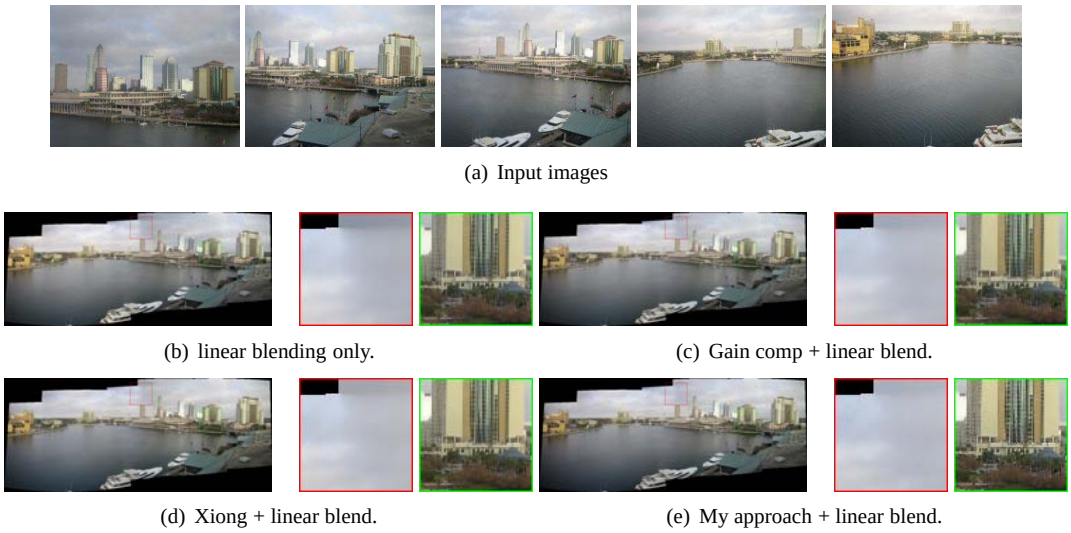
Method	Fig. 4.10	Fig. 4.12	Fig. 4.13	Fig 4.14	Fig. 4.15
Input image	0.6136	0.8320	0.7392	0.8541	0.6455
Gain comp.	0.6516	0.8440	0.7533	0.8594	0.6647
Xiong	0.6392	0.8427	0.7476	0.8501	0.6523
Proposed	0.6260	0.7679	0.6584	0.7847	0.5624

Table 4.5: Average $SSIM$ scores for the stitching examples before and after color correction.

Method	Fig. 4.10	Fig. 4.12	Fig. 4.13	Fig 4.14	Fig. 4.15
No correction	0.2625	0.2484	0.2318	0.2550	0.2638
Gain comp.	0.2694	0.2479	0.2281	0.2559	0.2558
Xiong	0.2677	0.2505	0.2295	0.2562	0.2608
Proposed	0.2760	0.2647	0.2424	0.2635	0.2692

Table 4.6: DSS sharpness scores for the stitching examples without and with color correction.

For the stitching examples in Fig. 4.12 to 4.15, I compute the PSNR [36] and SSIM [141] scores over each overlap, and use the average scores over all overlaps as a measure of the effectiveness of the color correction approaches.



(a) Input images

(b) linear blending only. (c) Gain comp + linear blend.

(d) Xiong + linear blend. (e) My approach + linear blend.

Figure 4.13: Five-view stitching example. For (b)-(d), regions marked by red and green rectangles are enlarged on the right.

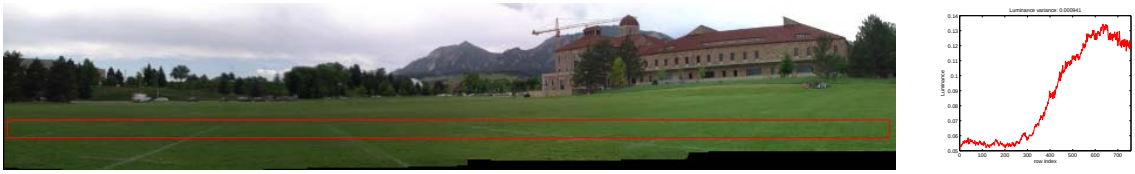
Also, I compute the DSS [160] score over the whole panorama as a sharpness measure. Tables 4.4 though 4.6 show the average scores. For these examples, my approach obtains the highest average PSNR scores compared to gain compensation [12] and Xiong's approach [149] for most of the examples, which means it is most effective at reducing the color difference between the images. Meanwhile, the lower average SSIM score shows my approach preserves the structure of the images most. Finally, my approach has consistently higher DSS, demonstrating that integration of structure makes the output panorama sharper.

4.5.6 Discussion and conclusions

In this section I proposed a hybrid color correction approach that performs color and contrast corrections simultaneously for general n-view image stitching. Compared to traditional color correction algorithms which only correct the color difference of the input images in the intensity/color domain, my approach also corrects the dynamic range difference by balancing the contrast in the gradient domain.

In my approach, contrast correction unifies the dynamic range of the input images and color correction balances the color difference. Adding reconstruction from the gradient domain can help increase the sharpness of the resulting images. In addition, my approach has a mathematical form providing good scalability which makes it suitable for

(a) Input images



(b) linear blending only.

(c) Gain comp + linear blend.

(d) Xiong + linear blend.

(e) My approach + linear blend.

Figure 4.14: A six-view stitching example. For rows 2 to 5, left is the stitched panorama, right is average brightness along selected rows marked by red rectangle in the left of row 2.

stitching image sets of large size (i.e. large n values). The major deficiency of my approach is its speed, and the bottleneck is my usage of an exact solver for the Poisson equations (Eq.(4.19)). A direction of future work is to explore approximate solvers to boost the speed of the approach.

4.6 Two-view video color correction

I have designed a new technique to solve the color correction problem for two-view videos. Previously in Sec. 4.5.2, I observed the affine-transform is a simple yet powerful transform for two-view color correction, and I

(a) Input images

(b) linear blending only. (c) Gain comp + linear blend.

(d) Xiong + linear blend. (e) My approach + linear blend.

Figure 4.15: Two-view stitching example. Contrast is enhanced in the stitched result for my approach.

make use of this observation and extend the transform to n-view color correction. Similarly, in this section I extend the affine-transform to take in multiple frame inputs which results in a color correction algorithm for two-view videos. There are two benefits of choosing the affine-transform as the core of my two-view video color correction algorithm. First, the transform is simple yet powerful (see Sec. 4.5.2). Secondly, the transform is parametric and is easily extended to take in multiple frame input.

Given a pair of images, the affine-transform-based color correction approach adjusts the color of the target image I_t to be close to that of the source I_s as follows:

$$C'_t(u) = \mu_s(\Omega_{st}) + \frac{\sigma_s(\Omega_{st})}{\sigma_t(\Omega_{st})}(C_t(u) - \mu_t(\Omega_{st})) \tag{4.22}$$

where $\Omega_{st} = \Omega(s,t)$ represents the overlap region between I_s and I_t. $(\mu_s(\Omega_{st}), \sigma_s(\Omega_{st}))$ and $(\mu_t(\Omega_{st}), \sigma_t(\Omega_{st}))$ are the mean and standard deviation of the colors of I_s and I_t in Ω_{st}.

However, the above formula only considers across-view spatial color adjustment if put in the context of two-view video color correction, i.e. it is only applicable to per-frame color correction. For panoramic two-view video color correction, one can also take the color continuity of the temporal frames into account. Let us suppose $V_s = \{I_s^{(1)}, \cdots, I_s^{(t-2)}, I_s^{(t-1)}, I_s^{(t)}, I_s^{(t+1)}, I_s^{(t+2)}, \cdots\}$ is the video frame sequence of the source view, $V_t = \{I_t^{(1)}, \cdots, I_t^{(t-2)}, I_t^{(t-1)}, I_t^{(t)}, I_t^{(t+1)}, I_t^{(t+2)}, \cdots\}$ the corresponding synchronized video of the target view. My two-view video color correction approach can be described as:

$$C'_t(u) = \bar{\mu}_s(\tilde{\Omega}_{st}) + \frac{\bar{\sigma}_s(\tilde{\Omega}_{st})}{\bar{\sigma}_t(\tilde{\Omega}_{st})}(C_t(u) - \bar{\mu}_t(\tilde{\Omega}_{st})) \tag{4.23}$$

In contrast to Eq.(4.22), here $\tilde{\Omega}_{st}$ represents union of the overlap regions between $2k+1$ succeeding frames temporally around frame $I_s^{(t)}$ in V_s and $2k+1$ succeeding frames temporally around frame $I_t^{(t)}$ in V_t:

$$\tilde{\Omega}_{st} = \{\Omega_{st}^{(t-k)}, \cdots, \Omega_{st}^{(t-1)}, \Omega_{st}^{(t)}, \Omega_{st}^{(t+1)}, \cdots, \Omega_{st}^{(t+k)}\} \tag{4.24}$$

Here $\Omega_{st}^{(i)}$ is the overlap region of frames $I_s^{(i)}$ and $I_s^{(i)}$ (for $i = t-k, t-k+1, \cdots, t+k-1, t+k$). k is a parameter that controls the size of the temporal frame neighborhood. $(\bar{\mu}_s(\tilde{\Omega}_{st}), \bar{\mu}_t(\tilde{\Omega}_{st}))$ are the weighted means of the colors defined on $\tilde{\Omega}_{st}$:

$$\bar{\mu}_s(\tilde{\Omega}_{st}) = \frac{\sum_{i=t-k}^{t+k} W(i) \sum_{p \in \Omega_{st}^{(i)}} I_s^{(i)}(p)}{\sum_{i=t-k}^{t+k} W(i) \sum_{p \in \Omega_{st}^{(i)}} 1} \tag{4.25}$$

$$\bar{\mu}_t(\tilde{\Omega}_{st}) = \frac{\sum_{i=t-k}^{t+k} W(i) \sum_{p \in \Omega_{st}^{(i)}} I_t^{(i)}(p)}{\sum_{i=t-k}^{t+k} W(i) \sum_{p \in \Omega_{st}^{(i)}} 1} \tag{4.26}$$

And $(\bar{\sigma}_s(\tilde{\Omega}_{st}), \bar{\sigma}_t(\tilde{\Omega}_{st}))$ are the weighed standard deviation of the colors defined on $\tilde{\Omega}_{st}$ [143]:

$$\bar{\sigma}_s(\tilde{\Omega}_{st})) = \sqrt{\frac{\sum_{i=t-k}^{t+k} W(i) \sum_{p \in \Omega_{st}^{(i)}} (I_s^{(i)}(p) - \bar{\mu}_s(\tilde{\Omega}_{st}))^2}{\sum_{i=t-k}^{t+k} W(i) \sum_{p \in \Omega_{st}^{(i)}} 1}} \tag{4.27}$$

$$\bar{\sigma}_t(\tilde{\Omega}_{st})) = \sqrt{\frac{\sum_{i=t-k}^{t+k} W(i) \sum_{p \in \Omega_{st}^{(i)}} (I_t^{(i)}(p) - \bar{\mu}_t(\tilde{\Omega}_{st}))^2}{\sum_{i=t-k}^{t+k} W(i) \sum_{p \in \Omega_{st}^{(i)}} 1}} \tag{4.28}$$

where $W(\cdot)$ is a normalized Gaussian weighting function for the $2k+1$ temporal frames centered at frame (t) .

For a three view camera system like the one shown in Fig. 3.8, I apply the above color correction approach to adjust the colors of the left and right views to that of the center view respectively (i.e. make the center view the source and the left and right views the target). Fig. 4.16 shows a comparison between panoramic videos stitched from a multi-view video captured by that camera system without and with my video color correction processing. Fig. 4.16(a) shows two example consecutive frames of the panoramic video stitched without any color correction but with Poisson blending [75]. The color differences between the center view and right/left views are obvious. Fig. 4.16(b) shows the same frames but with image color correction (Eq.(4.22)) applied to the frames individually. Because the temporal color transition from frame to frame is not considered, there is a "blinking" artifact when the two frames viewed in a

(a) No color correction.

(b) With individual image color correction.

(c) With proposed video color correction.

Figure 4.16: Comparison between panoramic video frames stitched without color correction, with individual image color correction and with proposed video color correction.

sequence although the correction of each frame looks good by itself. Fig. 4.16(c) shows the results of the proposed video correction technique — the the spatial color difference between the views are corrected and the transition from frame to frame is smoothed.

Although the color differences between the center view and the left/right views are obvious in Fig. 4.16(a), the difference between the approach of independent image color correction of each frame (corresponding to Fig. 4.16(b)) and the proposed video color correction approach (corresponding to Fig. 4.16(c)) are only observable during video playback. To quantify the temporal improvement in color consistency, it is helpful to examine actual metrics on the panoramic video differences. The statistical measure I adopt here is the mean of frame-to-frame mean square error (MSE) of RGB color channels which is defined as:

$$MSE_{video} = \frac{1}{K-1} \sum_{k=1}^{K-1} MSE_{RGB}(I^{(k+1)}, I^{(k)}) \tag{4.29}$$

where K is the total number of frames in the video. And MSE_{RGB} is the frame-to-frame MSE of RGB color channels which is defined as:

	individual image color correction	proposed video color correction
left view	43.4916	29.2843
right view	31.4745	29.4604

Table 4.7: Mean of frame-to-frame MSE of RGB color channels (MSE_{video}) of the left/right views of the panoramic videos processed by individual image color correction and by the proposed video correction.

$$MSE_{RGB}(I^{(k+1)}, I^{(k)}) = \frac{1}{3 * W * H} \sum_{c \in \{R,G,B\}} \sum_{(x,y)} (I_c^{(k+1)}(x,y) - I_c^{(k)}(x,y)) \tag{4.30}$$

where W and H are width and height of the video frames respectively. c is the color channel index and (x, y) is the pixel position index.

The compared videos are both composed of 96 frames (frame#: 2105 to 2200) and MSE_{RGB} is computed on the left/right views of the color corrected panoramic videos (either by individual image color correction or by the proposed video color correction). Table 4.7 shows the results. It can be seen from Table 4.7 that the panoramic video processed by the proposed video color correction approach has a lower MSE_{video} score than the one processed by individual frame color correction, on both the left/right views. A lower score means the video has a smoother color transition (i.e. less color difference) across the temporal frames. Thus the "blinking" artifact which arises due to dramatic color transition from one frame to the next is alleviated/removed.

4.7 Summary

In this Chapter I described my work on the color correction problem of multi-view image and video stitching which is the emphasis of this thesis. I first described my systematic review and extensive performance evaluation of nine selected representative color correction approaches. These performance evaluation results provide novel insights into the color correction problem which I have used to design two new color correction approaches, including a new hybrid and scalable color correction approach for multi-view image stitching and an affine-transform based color correction approach for two-view video stitching. The hybrid image color correction approach proposed was shown to be superior to previous scalable color correction approaches. The proposed video color correction approach can provide smoother temporal color transition from frame to frame in a color corrected panoramic video.

Chapter 5

Blurred Frame Detection and Repair

For mobile multi-view video capturing devices, motion blur is unavoidable in the captured videos, especially for videos of outdoor trails that have rough road surface or rapidly moving vehicles. Blur can severely bias the normal video alignment and panorama stitching process and thus should be removed.

The solution to the problem is comprised of two steps: first, one needs to detect which frames in a video are blurred and then one needs to repair these blurred frames. In the following sections I will discuss these two steps in detail.

5.1 Image partial blur detection and classification

5.1.1 Introduction

Blur due to motion or defocus is an imaging artifact that is common in video data. Automatic deblurring has already become an important function of current image and video processing systems. Given a blurred image, the system does not know in advance whether it is fully blurred or partially blur, or whether it is motion blurred or defocus blurred. Before the deblurring operation, the system needs to identify the type of the blur so that the most suitable deblurring algorithm can be used. It also needs to analyze the distribution of the blur in the image so that the exact, homogeneously blurred region can be extracted for estimating the blur kernel.

There has been a lot of work put onto image blur analysis and processing over the years. However, most previous work focuses on the estimation of the blur kernel and the corresponding image de-convolution procedure for a particular type of blur. General blur detection and recognition, especially partial blur detection and analysis, is relatively less explored. Among the few, Rugna et al. proposed a learning-based method to classify an input image

into blurry and non-blurry regions based on the observation that the blurry regions are more invariant to low-pass filtering [114]. Levin et al. explored the gradient statistics along different directions to segment an input image into blurred/nonblurred layers [74]. Later, Liu et al. combined four specially designed local blur measures under the Bayes classification framework for image partial blur detection and classification [81]. Dai et al. relied on the result of image matting to extract the motion blurred region from the input image [31]. Chakrabarti et al. combined "local frequency" components of an image, local gradient variation and color information under the MRF segmentation framework to extract the motion blurred regions [24]. Among the above work, Liu's analysis and discussion [81] is most general and involves local blur measures for both blur/nonblur classification and motion/defocus blur-type classification.

Unlike object detection in which a high-level template of the target object can be used to guide the organizing of low-level features, blur detection does not have any high-level "shape" template for reference but has to rely on low-level image analysis. Moreover, for partial blur detection, many existing low-level blurriness/sharpness metrics developed for measuring the quality of the whole image (e.g. for "auto-focus" applications [43]) are not applicable. Local blur measures relying on sparse image features (e.g. edge sharpness [49]) are not applicable either because the existence and density of these features cannot be guaranteed within every local patch.

Identifying blurriness in an given image is a long studied topic in image processing and photography although it is still not fully solved. The research began by developing different kinds of whole-image blurriness measures [133, 77], which are exploited by camera manufactures for camera auto-focusing and anti-shake functionalities. Later, the interest shifted from identifying whether an image is blurred, to identifying and recognizing the blurred regions and blur type in a partially blurred image. The latter task is more challenging, because it requires the blur measure to be very sensitive to a small amount of blurriness, to work locally, and at the same time to be robust to noise. Many previously proposed blur measures become "weak" under such requirements, thus people began to think about combining multiple local blur measures to achieve the goal.

5.1.2 Related work

A previous, significant work on developing and combining multiple local blur measure to identify and recognize partial image blurriness was published by Liu et al. [81]. In that approach, four local blur measures were proposed, namely Maximum Saturation (MS), Contrast-compensated Gradient Histogram Span (CGHS), Local Power Spectrum

Slope (LPSS) and Local Auto-correlation Congruency (LAC) using local color, gradient and spectral information. After Liu's work, Chen et al. [26] also proposed the Lowest Directional High-Frequency Energy (LDHFE) blurriness measure for motion blur detection.

Liu's Maximum Saturation (MS) measure is based on the observation that blurred pixels tend to have less vivid colors than unblurred pixels because of the smoothing effect of the blurring process. In [81], this color property of blur is measured by the following per-pixel color saturation metric:

$$S_p = 1 - \frac{3 \cdot \min(R, G, B)}{(R + G + B)} \tag{5.1}$$

The greater the value of S_p the more saturated the pixel is. The local blur measure is defined as:

$$MS = \frac{\max_{p \in P}(S_p) - \max_{p \in I}(S_p)}{\max_{p \in I}(S_p)} \tag{5.2}$$

One of the statistical properties of natural images is that their gradient magnitudes usually follow a heavy-tailed distribution of wide span [113]. However, blur can change the shape of this distribution and make it more peaked and narrower by smoothing out sharp edges [41]. For Liu's Contrast-compensated Gradient Histogram Span (CGHS) measure [81], the gradient distribution of an image patch is modeled by a two-component Gaussian mixture model:

$$G = \pi_0 G_0(x; u_0, \sigma_0) + \pi_1 G_1(x; u_1, \sigma_1) \tag{5.3}$$

with $\pi_0 + \pi_1 = 1$ and $\sigma_1 > \sigma_2$. After decomposition using the EM algorithm, the larger variant (i.e. σ_1) is treated as the gradient histogram span (GHS). This GHS measure is used with maximum local contrast C_p to measure the blurriness of the patch:

$$CGHS = \frac{\tau \cdot \sigma_1}{C_p + \epsilon^2} \tag{5.4}$$

Liu et al. proposed to decompose the gradient magnitude histogram as a 2-component Mixture of Gaussian (MoG) model using the EM algorithm [33] and measure image blurriness by the span of the major component of the MoG, q_2. Their measure has two problems in practice: first, it assumes that the gradient distribution can be well modeled by a 2-component mixture of Gaussian, but this assumption does not hold for very case. For example, at pretty uniform image regions (e.g. the blue sky) the measure is severally affected by noise and may be inaccurate. Second, dense EM decomposition at every pixel locations is not affordable to many higher-level applications due to its slow speed.

The slope of the power spectrum distribution of an image can be used to measure the blurriness of an image, which gives rise to the Local Power Spectrum Slope (LPSS) blur measure. Given an image I of size MxN, its squared magnitude in the frequency domain is:

$$S(u,v) = \frac{1}{M \cdot N}\mathcal{I}(u,v) \tag{5.5}$$

where $\mathcal{I}(u,v) = \mathcal{F}(I)$ is the Discrete Fourier transform (DFT) of I. Letting $S(f,\theta)$ be the corresponding polar representation of $S(u,v)$ with $u = f\cos\theta$ and $v = f\sin\theta$, the power spectrum S is then the summation of $S(f,\theta)$ over all directions θ [44, 17]:

$$S(f) = \sum_{\theta} S(f,\theta) \simeq \kappa f^{-\alpha} \tag{5.6}$$

where κ is an amplitude scaling factor for each orientation and α is the frequency exponent. Experiments show that for natural images the power spectra $S(f)$ fall off at $1/f^2$ with increasing f [44, 17]. This corresponds to a curve of a slope ≈ -2 in the log-log coordinates, that is, $\alpha \approx 2$.

In [81], Liu et al. claimed that direct use of the power spectrum slope α_P for a patch P is inaccurate at measuring local blurriness. They then suggested a relative measure of blurriness that compensates each local α_P with the global α_I of the image I which is called Local Power Spectrum Slope (LPSS):

$$LPSS = \frac{\alpha_P - \alpha_I}{\alpha_I} \tag{5.7}$$

Given an image I, the image gradient at a point (x,y) at a direction θ can be represented as:

$$\Delta I(x,y)\cdot\theta = [I_x(x,y)I_y(x,y)][\cos\theta\,\sin\theta]^T \tag{5.8}$$

Then, within a local path P one can build a directional energy measurement of squared directional gradients [26]:

$$E(P) = \sum_{W}([I_x(x,y) \quad I_y(x,y)][\cos\theta \quad \sin\theta]^T)^2 = \begin{bmatrix}\cos(\theta) & \sin(\theta)\end{bmatrix} D(x,y) \begin{bmatrix}\cos(\theta)\\ \sin(\theta)\end{bmatrix}$$

where

$$D = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22}\end{bmatrix} = \sum_{W}\begin{bmatrix} I_x^2(x_i,y_i) & I_x(x_i,y_i)I_y(x_i,y_i) \\ I_x(x_i,y_i)I_y(x_i,y_i) & I_y^2(x_i,y_i)\end{bmatrix} \tag{5.9}$$

is the local autocorrelation matrix.

Liu et al. first eigen-decompose D and get eigenvalues $\lambda_1(x,y)$ and $\lambda_2(x,y)$ and corresponding eigenvectors $V_1(x,y)$ and $V_2(x,y)$ where $\lambda_1(x,y) > \lambda_2(x,y)$. Then, they construct a directional response histogram hist(θ) for each patch P, where each bin represents one direction θ (corresponding to the direction of $V_2(x,y)$) and the value of each bin is the number of pixels with eigenvector V_2 along direction θ in patch P, weighted by their ellipse axis ratio $\sqrt{\lambda_1/\lambda_2}$. Finally, their Local Auto-correlation Congruency (LAC) measure is computed as the variance of the normalized bin values:

$$LAC = Var(\text{hist}(\theta)) \tag{5.10}$$

A discriminative feature of motion blur is that the corresponding local gradient orientation distribution is more peaky than normal with the peak at the direction perpendicular to the motion direction. Chen et al. explore this feature and use the gradient energy along the estimated local motion direction for motion blur detection [26]. To this end, they represent the directional energy measurement of squared directional gradients $E(P)$ (Eq.(5.9)) as a function of θ:

$$f(\theta) = E(P) = \frac{1}{2}(d_{12} + d_{21})\sin(2\theta) + (d_{22} - d_{11})\sin^2(\theta) \tag{5.11}$$

Its minimum value can be obtained by setting $\frac{\partial f(\theta)}{\partial \theta} = 0$, which yields $\hat{\theta} = \theta_{base} + \frac{n\pi}{2}$, where $\theta_{base} = \frac{1}{2}\tan^{-1}\left(\frac{d_{12}+d_{21}}{d_{11}-d_{22}}\right)$. In [26], the local motion (blur) direction is estimated as $\hat{\theta}_{motion} = \text{argmin}\{f(\theta_{base}), f(\theta_{base} + \frac{\pi}{2})\}$, and the corresponding energy $f(\hat{\theta}_{motion})$ is used to measure motion blur:

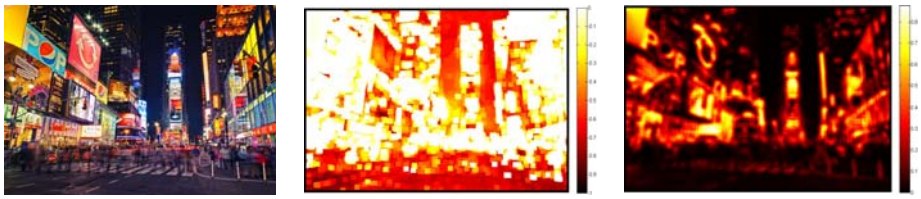
$$LDHFE = f(\hat{\theta}_{motion}) \tag{5.12}$$

5.1.3 My approach

Liu's state-of-the-art approach to blur detection leaves much room for improvement both in the local blur measures used and at the classifier level. There are a number of classifiers that may be a better choice than naive Bayes in practice. Starting with the idea of classifying image regions as blurred/nonblurred and motion or defocus blurred, I make the following contributions:

- I have developed a set of new or enhanced local blur measures with stronger discriminative power, better across-image stability or higher computational efficiency.

- I combined these blur measures in an SVM-based learning framework and evaluated the resulting approach



(a) Input Image (b) Liu's MS measure (c) Proposed b_1 measure

Figure 5.1: Local color saturation map for blur measurement. (a) is the input image. (b) is the Maximum Saturation (MS) measure proposed by Liu et al. (see Eq.(5.2)). It has two problems: 1) abnormal values at some pixels of almost a single color, and 2) the max$(\cdot)$ operation extend the saturation to nearby unsaturated pixels. (c) is the proposed measure b_1. Comparing to the MS measure, b_1 shows more natural and accurate modeling of local color saturation.

using a large size dataset created with a self-developed graphic tool. Experiments show my approach can achieve higher detection accuracy rate and faster speed than existing systems.

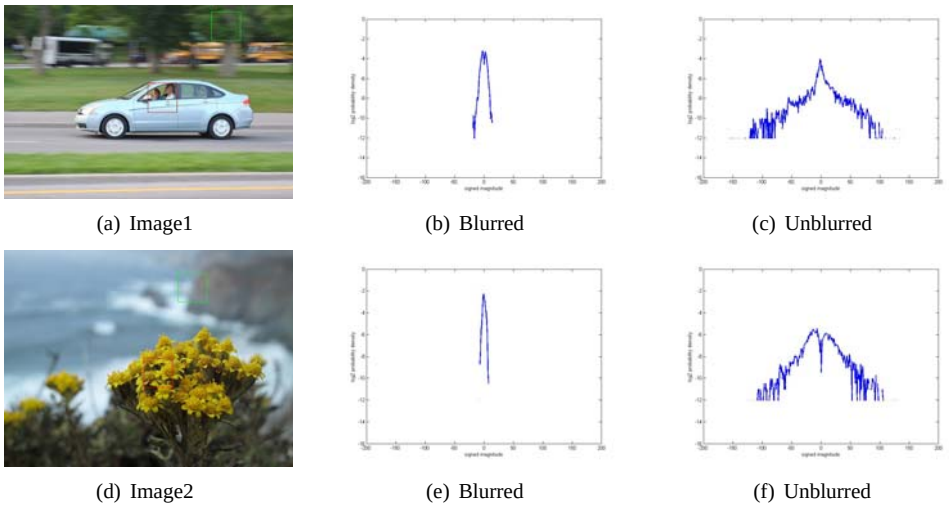
The emphasis of my work is on developing better local blur measures to serve this solution. Particularly, describe the deficiencies and incorrect usage of some current local blur measures, and develop a set of new or enhanced local blur measures with stronger discriminative power, better across-image stability or higher computational efficiency. Next, I will introduce my local blur measures according to their usage: either for blur/nonblur classification or for motion/defocus blur classification.

5.1.4 Blur measures for blur/nonblur classification

5.1.4.1 Measure from local saturation

Blurred pixels tend to have less vivid colors than unblurred pixels because of the smoothing effect of the blurring process. However, the maximum saturation measure S_p developed by Liu's has a severe problem: it only considers the minimum of the three color channels, but does not model the difference between the R, G and B values. Thus it is very inaccurate at measuring the color saturation of single-color images. For example, pixels of the deep blue sky captured at evening usually have very low R and G values and mid-range B values, which makes the resulting S_p value extraordinarily high (see Fig. 5.1). To overcome this shortcoming, I propose a new more comprehensive saturation measure [55]:

$$S'_p = \frac{1}{2}(\tanh(\delta \cdot ((L_p - L_T) + (C_T - \|C_p\|_2)) + 1) \tag{5.13}$$



(a) Image1 (b) Blurred (c) Unblurred

(d) Image2 (e) Blurred (f) Unblurred

Figure 5.2: Gradient magnitude distribution of blurred and unblurred regions. (a) and (d) are partially blurred images with selected blurred and unblurred regions marked by green and red rectangles respectively. (b) and (e) are gradient distribution of the blurred regions in (a) and (d) respectively, and (c) and (f) are those of the unblurred regions.

Here L and $C = (a, b)^T$ represent the lightness and the color components in the CIELAB color space respectively. L_T and C_T are thresholds for L and C, and δ controls the growing speed of the measure. S'_p is only effective when correct values of these parameters are used, but the parameter determination procedure is pretty empirical. After a lot of experiments I have found $L_T = 100$, $C_T = 25$ and $\delta = 1/20$ are good settings for my task. With these settings, my blur measure from local saturation is defined as the mean value of S'_p in a local neighborhood P:

$$b_1 = \frac{1}{N_P} \sum_{p \in P} S'_p \tag{5.14}$$

5.1.4.2 Shape of the gradient magnitude histogram

I propose to measure image blurriness by the whole shape of the local gradient magnitude distribution without any assumption of the parametric format of the distribution or dense decomposition:

$$b_2 = \frac{f_1(H)}{f_2(H) + \epsilon} \tag{5.15}$$

where $H = [h_1, \cdots, h_N]$ is the normalized gradient magnitude histogram with N bins of patch P. $f_1(H) = \text{Var}(H)$ measures the vertical variation of H, which is an indirect measure of horizontal span because H is normalized. $f_2(H) = \sum_i^N |h_{i+1} - h_i|$ measures the smoothness of the histogram across bins. ϵ is a small constant to avoid divide-by-zero errors. Fig. 5.2 illustrate this measurement. By comparing Fig. 5.2(b)/ 5.2(c) with Fig. 5.2(e)/ 5.2(f),

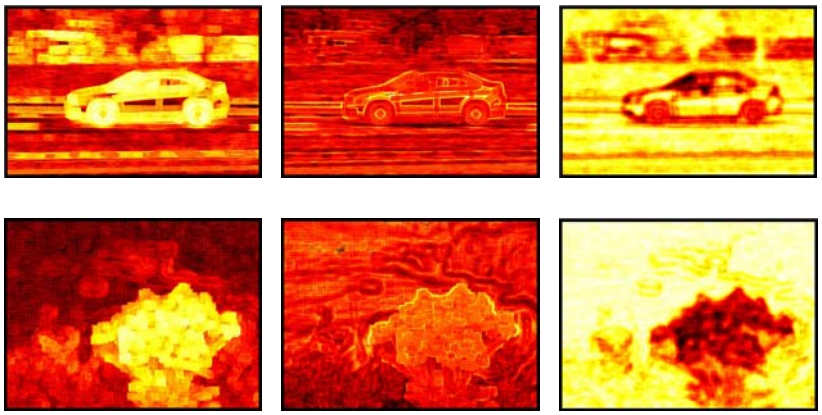


Figure 5.3: Comparison of different blur measures from gradient magnitude distribution. The top row is for the image shown in Fig.5.2(a) (image size:640x480) and bottom row is for the image in Fig.5.2(d) (image size: 640x430). On each row, from left to right are the gradient histogram span (GHS) map, contrast-compensated gradient histogram span (CGHS) map (see Eq.(5.4)), and the map of the proposed measure b_2. (Maps are displayed in log space for better visualization.)

one can observe that in additional to the decomposed horizontal span, histogram vertical variation (peakedness) and curve smoothness can also be used to discriminate between blurred and unblurred regions.

Fig. 5.3 illustrates the difference between the proposed measure and that of Liu et al. [81]. Close examination of the GHS and CGHS maps reveal that they are actually pretty noisy due to the invalidation of the assumption they make. Also, GHS maps and CGHS maps took about 1.5 hours to compute, while the b_2 maps took < 200 seconds, demonstrating that my proposed local blurriness measure b_2 is much more efficient to compute.

5.1.4.3 The power spectrum slope

The last measure I use to evaluate blur is the power spectrum slope which measures the decay of signals across frequencies in the Fourier domain. In Sec. 5.1.2, I have already described in detail the definitions of the power spectrum slope and Liu's local power spectrum slope. Due to Liu's claim that the power spectrum slope is less stable than his measure as a local blurriness measure, I specifically performed experiments to compare the two measures with real images. Surprisingly, after testing both measures on a lot of images, I have obtained a different conclusion refuting Liu's claim: My experiments show the power spectrum slope has a stable range across a wide variety of images, while Liu's relative measure $\frac{\alpha_P - \alpha_I}{\alpha_I}$ is on the contrary rather unstable. Fig. 5.4 uses a few examples to show my findings.

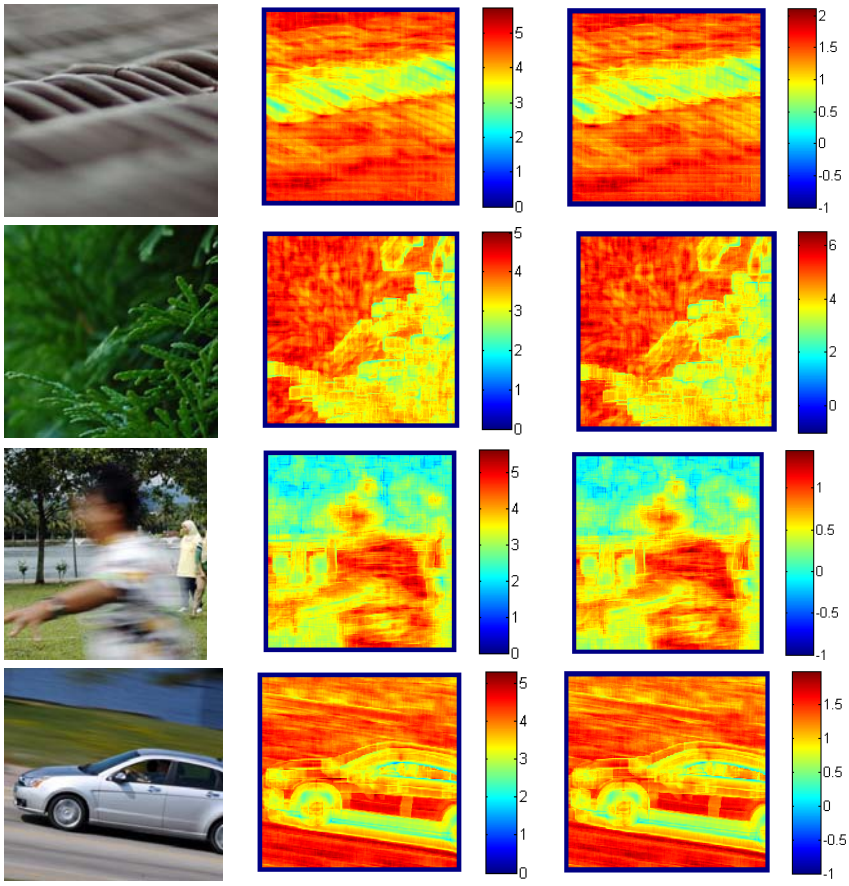


Figure 5.4: Four partially blurred images and their power spectrum slope (α_P) the local power spectrum slope (LPSS) maps (see Eq.(5.7)). On each row, left is the image, central is α_P map, and right is LPSS map. The colorbars on the right of the maps show unlike what was claimed in [81], the value range of the α_P map is pretty stable for different images, while the range of the LPSS map is much less stable and varies dramatically across the images.

Thus I propose a third local blur measure

$$b_3 = \alpha_P \tag{5.16}$$

See Eq.(5.1.2) for the definition of α_P.

5.1.5 Blur measures for motion/defocus blur classification

Chen's lowest directional high-frequency energy (LDHFE) measure only uses the energy along the direction of the motion to measure motion blur (see Sec. 5.1.2). I think the energy along the direction perpendicular to the motion direction also contains useful information and can be used together with LDHFE to discriminate motion blur. I propose a measure which includes both the energies along the estimated motion direction and the ratio of energies at

extreme directions:

$$m_1 = \log(f(\hat{\theta}_{motion})) \tag{5.17}$$

and

$$m_2 = \log\left(\frac{f(\hat{\theta}_{motion} + \frac{\pi}{2})}{f(\hat{\theta}_{motion}) + \epsilon}\right) \tag{5.18}$$

where $\log(\cdot)$ is used to bring the measures to an easier-to-manage range. One good thing of m_2 is that it is a relative measure which is not affected by the absolute intensity of a patch.

In addition to these two measures, Liu's local auto-correlation congruency (LAC) measure (Sec. 5.1.2) can encode the gradient directional distribution of a larger neighborhood, so I use it as my third measure of motion blur:

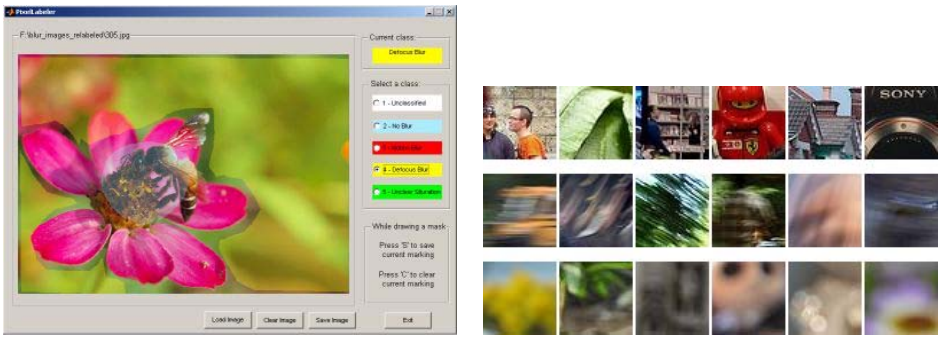
$$m_3 = \log(\text{LAC}(P)) \tag{5.19}$$

5.1.6 Experimental results

I implement partial blur detection and recognition via patch-based classification. I define three blur labels for classification: sharp, motion blur or defocus blur. Given an input image patch, I perform blur/nonblur classification on it first. If it is identified as blurred, I then perform motion/defocus blur classification to identify the detailed blur type. I use a linear SVM classifier with the blur measures b_1, b_2 and b_3 for blur/nonblur classification, and use another linear SVM classifier with the motion blur measures m_1, m_2 and m_3 for motion/defocus blur classification.

The dataset used in my experiments was created from 200 partially blurred images downloaded from `www.flickr.com`. Images are rescaled so that their maximum dimension is 640 pixels. I have developed a graphic tool for convenient selection and labeling of the sharp/blurred segments in these images. Fig. 5.5(a)) illustrates the interface of this tool. The user selects a region and indicates whether it is sharp, motion blurred or defocus blurred by drawing indicating polygons of different colors. For selecting large regions or regions of irregular shape, the tool allows the user to finish the drawing in several steps and automatically merges overlapping input polygons of the same color. The tool also supports converting a finer pixel-level segment map to a coarser patch-label segment map by analyzing the composition of pixel labels within each block. With this tool the labeling of my dataset became much easier and faster. I have made this tool available for download by other researchers.

I use 5-fold cross validation to find the optimal hyper-parameters for the linear SVM classifiers. I use ROC



(a) The labeler software (b) Image patch examples

Figure 5.5: Dataset creation and example image patches. In (b), the rows from top to bottom are respectively sharp patches, motion blur patches and defocus blur patches.

curves (true positive rate (TPR) v.s. false positive rate (FPR)) and accuracy rate (AR) for the labeled image patches in the dataset to measure the performance of my approach. Let N be the number of patches to be classified, L_i be the label for patch i, and t_i be the ground truth label for patch i, the TPR, FPR and AR measurements are defined as:

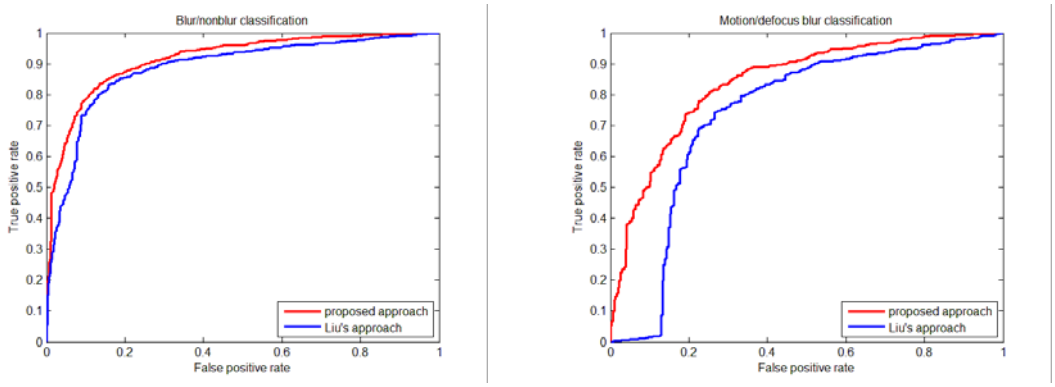
$$\text{TPR} = \frac{|\{i|L_i = true \& t_i = true\}|}{|\{i|t_i = true\}|} \tag{5.20}$$

$$\text{FPR} = \frac{|\{i|L_i = true \& t_i = false\}|}{|\{i|t_i = false\}|} \tag{5.21}$$

$$\text{AR} = \frac{|\{i|L_i = t_i\}|}{N} \tag{5.22}$$

With the learned SVM classifiers, my approach has achieved 84.6% and 80.2% accuracy rates for blur/nonblur and motion/defocus blur classifications on the test dataset. In comparison, Liu's approach can only achieve 80.4% and 28.3% accuracy rates respectively. The ROC curves for the classifiers for the 30% test set are given in Figure 5.6. Note also that my approach achieves this performance gain over Liu's approach with much lower computational cost.

Example results for image partial blur segmentation are shown in Figure 5.7. Although the ROC curves show that Liu's approach and my approach do roughly equally well for blur/nonblur classifying, one can tell from these examples that the sharp regions (marked in blue) identified by Liu's approach are less tight around the true object than ours. This is shown by the flower in Fig.5.7(g) and by the body of the bee in Fig.5.7(j). Another severe problem of Liu's approach is that it prefers to classify a region as motion blur (marked in red) and mis-classifies many sharp and defocus blur region to motion blur. This was confirmed by the very low (28.3%) motion/defocus blur classification accuracy rate, and is demonstrated here again by the segmentation examples. For example, in Figs.5.7(b), 5.7(e) and



(a) Blur/nonblur classification (b) Motion/defocus blur classification

Figure 5.6: ROC curves of blur classifications and image partial blur segmentation used the trained classifiers.

5.7(f), much of the tree areas, which are pretty clear and sharp, are mis-classified as motion blur by Liu's approach. Also, Liu's classification of the defocus blurred areas in Figs.5.7(a), 5.7(d), 5.7(g) and 5.7(j) are completely wrong. I think this is because Liu's approach uses only one motion blur feature and a naive Bayes classifier for motion/defocus blur classification, both of which are not very reliable and prone to errors. In contrast, my approach uses three motion blur features and SVM-based classification and thus demonstrates more robust performance.

Figs.5.7(h) and 5.7(i) are complicated examples because they contain all three kinds of regions: sharp, motion blurred and defocus blurred. My approach works very well for these complicated cases, while Liu's approach always mis-classifies the defocus blurred regions as motion blur. For three other examples in Figs.5.7(b), 5.7(c) and 5.7(f), my approach produces competitive results to those of Liu's approach, while for the mis-classified regions Liu's approach prefers to classify them as motion blurred while ours prefers to classify them as defocus blurred.

5.1.7 Discussion and conclusions

In this section, I proposed a set of new or enhanced local blur measures based on my study and analysis of the relationship between image blurriness and the variation of different types of image information. These blur measures were tested on patch-based blur classification using linear SVM classifiers, and a performance gain over the state-of-the-art approach [81] was achieved. Part of my future work would be exploring more advanced classifiers and kernels to further improve the overall performance. Also, because the emphasis of this work is on local blur measures I did not use any contextual information between classified image patches in image partial blur segmentation (Fig. 5.6(b)).

(a) example 1 (b) example 2

(c) example 3 (d) example 4

(e) example 5 (f) example 6

(g) example 7 (h) example 8

(i) example 9 (j) example 10

Figure 5.7: Image partial blur segmentation using the trained classifiers. For each example, from left to right are: input image, segmentation output of Liu's approach and segmentation output of my approach. Unblurred regions are marked in blue, motion blurred region in red, and defocus blurred regions in yellow.

Exploring that information would also be an interesting topic for future work.

5.2 Blurred frame detection and repair

5.2.1 Related work

In the image processing and computer vision literature, there are three categories of metrics for measuring the blurriness/sharpness of an image: 1) full-reference, 2) partial-reference and 3) no-reference (blind). A full-reference metric means given a blurred image, there exists an example sharp image (or a set of example sharp images) that can be fully used as a reference to build a metric for the degree of blurriness/sharpness of the blurred image. A no-reference

metric means there is no reference image that can be used, and the metric has to be built (blindly) from the blurred image itself only. Partial-reference is in the position between full-reference and no-reference, and usually means there does exist a reference image (or a set of reference images) to use, but blurriness/sharpness of the reference image is also unknown so that it can only be partially used to build the metric.

Full-frame blur detection falls into the category of using a no-reference (blind) metric to measure the blurriness/sharpness of a given frame. This section presents a review of existing popular no-reference blurriness/sharpness metrics, which can be roughly categorized as follows: 1) pixel-based metrics, 2) transform-based metrics, 3) gradient and edge-based metrics, and 4) histogram-based metrics. Most of the metrics that will be introduced were developed for "auto-focus" applications, where the primary requirement is to ensure the monotonicity of the perceived blur with the metric for a single image.

5.2.1.1 Pixel-based metrics

Pixel-based techniques include analysis of statistical properties and correlation between pixels. In [38], Erasmus et al. simply used the variance of the input image as a blurriness metric. This is because as the blurriness of the image increases, the edges are smoothed and the transitions between the gray-scale levels in the image decrease, and thus the variance decreases. The deficiency of this simple measure is that it can be biased by the number of edges present in the input image.

The autocorrelation-based metric [6] is similar in mechanism to the variance metric. This metric is derived from the auto-correlation function which uses the difference between auto-correlation values at two different distances along the horizontal and vertical directions, respectively. If the image is blurred or the edges are smoothed, the correlation between neighboring pixels becomes high. Consequently, the autocorrelation will increase, and thus the sum of the difference metric will decrease.

5.2.1.2 Transform-based metrics

Transform-based approaches take advantage of the fact that sharper edges increase high frequency components. Firestone et al. [45] computed the summation of all frequency component magnitudes above a certain threshold, and used it to measure image sharpness. The threshold is chosen experimentally and it is usually set between $[\pi/4, \pi/2]$.

Increasing the threshold may catch more edges but at the same time the metric will be more susceptible to noise.

Nill and Bouzas [100] proposed the Image Quality Measure (IQM) which calculates the normalized image power spectrum weighted by a modulation transfer function (MTF). The MTF is derived empirically taking into account the response of the Human Visual System to different image frequencies (i.e., cycles/deg) [34, 119]. The higher the metric, the sharper the image.

The kurtosis is a statistical measure of the peakedness or flatness of a distribution: a narrow distribution has high kurtosis and vice versa. It was used in the frequency domain for measuring sharpness [162, 21]. Zhang et al. [162] show that the spectral density function can be considered as a 2-D probability density function of a bivariate random vector. Increasing the image sharpness will decrease the kurtosis; blurring the image will increase the kurtosis, so the kurtosis is inversely proportional to the sharpness.

Shaked and Tastl [120] proposed a metric based on the high-pass to band-pass frequency ratio applied to local features that are extracted by thresholding the bandpass filter output. The sharper the image the higher the ratio since the number of high frequency components in the spectrum increases resulting in higher energy at the output of the high-pass filter.

Ferzli and Karam [42] proposed the Noise Immune Sharpness (NIS) metric. It uses the Lipschitz regularity properties to separate the signal singularities from the noise singularities, applies the dyadic wavelet transform and then measures the sharpness using the perceptual blur metric [89]. Compared to other sharpness metrics, this metric will perform well under low to moderate SNR since the noise will be reduced across wavelet scales.

5.2.1.3 Gradient and edge-based metrics

Gradient(derivative)-based metrics work by detecting the slope of the edges in an image. They include the first-order (gradient) and second-order (Laplacian) derivative metrics [6]. These metrics act as a high-pass filter in the frequency domain. While the Laplacian-based method has good accuracy, it is highly sensitive to noise.

Ong et al. [104] used the intensity difference between the two sides of image edges to measure blurriness. They first find the edge pixels using the Canny edge detector. For each edge pixel, the local extrema (i.e., minimum and maximum), along the gradient direction and closest to the edge pixel, are located and the edge width is calculated by counting the number of pixels with increasing gray-scale values from one side and the number of pixels with

decreasing gray-scale values from the other side. The average edge widths over all the pixels are then found and used in an exponential model to find the quality metric. Note that the parameters of the exponential model need to be estimated by training over a large data set of images with available subjective ratings. The sharper the image, the lower the metric is.

Marziliano et al. [89] proposed a perceptual metric to measure image blurriness. Edge detection is first performed on the input image. Then, the start and end positions of the edge are defined as the locations of local extrema closest to edge. The edge width is calculated as the distance between the end and start positions. The overall metric is calculated as the average of the edge widths of the local blur values over all edges found.

Ferzli and Karam [43] developed the notion of Just noticeable blur(JNB) which is determined as a function of the local contrast. JNB was then used to derive an edge-based sharpness metric based on a Human Vision System (HVS) model that makes use of a probability summation over space. Liang et al. [76] studied the image gradients along local image structures and proposed a new perceptual blur metric based on the gradient profile sharpness of image edges along the horizontal or vertical direction. The sharpness distribution histogram is rectified by a just noticeable distortion (JND) threshold to evaluate the blurring artifacts and assess the image quality.

5.2.1.4 Histogram-based approaches

Firestone et al. [45] proposed a sharpness measure based on thresholding the intensity histogram of the input image. The metric is defined as the weighted sum of the histogram bin values above a certain threshold. The threshold is usually selected to be near the mean of the image. It is assumed that a sharper image contains a higher number of gray-scale levels and thus the histogram will be wider containing a higher number of bins. The sharper the image, the higher the metric is.

Chern et al. proposed the histogram entropy-based metric [28]. The entropy is a measure of the information content of an image; if the probability of occurrence of each gray level is low, the entropy is high and vice versa. The probabilities are calculated by normalizing the obtained histogram. Sharper images contain a larger number of gray levels, meaning a lower probability and, thus, higher entropy.

The metric proposed by Marichal et al. [87] is based on the occurrence histogram of nonzero Discrete Cosine Transform (DCT) coefficients throughout all 8x8 blocks of the image (more weight is given to DCT coefficients on

the diagonal). The blurriness metric is estimated by examining the number of coefficients that are almost always zero by counting the number of zeros in the histogram. The metric is lower for sharper images.

5.2.2 Blurred frame detection and repair

5.2.2.1 Blurred frame detection

My blurred frame detection is based on the judgment of whether or not more than a certain percentage of all of the image patches composing the image are motion blurred. Given an input frame I, suppose it is split into a total of N non-overlapping blocks and the partial blur detection and classification approach described in Sec. 5.1 is used to classify these N blocks into sharp/defocus-blurred/motion-blurred classes and giving M estimated motion blurred blocks. Then the simplest strategy for judging whether or not the frame I is motion blurred is:

$$\frac{M}{N} > \tau \tag{5.23}$$

where τ is a threshold set by the user to determine whether the frame is blurred. This is the simplest scheme to determine a blurred frame. More advanced schemes that consider the spatial distribution of motion blurred blocks can also be developed.

5.2.2.2 Blurred frame repair

After a frame is determined to be motion blurred, the next step is to repair it. My strategy to accomplish the task is to treat the blurred frame as a missing frame, and to interpolate the frame out from its temporal neighboring frames given that those neighboring frames are not themselves motion blurred. This is carried out in two steps: 1) Inferring the global motion parameters of the blurred frame from the global motion chain of its temporal local neighborhood (not including the blurred frame itself), and 2) Interpolating the blurred frame from the frames immediately before and after it using backward interpolation, and merging the two interpolation results.

The frame to frame transformation is assumed to be similarity transform as in Sec. 2.2.2. I used polynomial data fitting to estimate the global motion parameters of the blurred frame (including the rotation angle θ, the scaling factor s and the translation vector (t_x, t_y) (see Eq.(2.9))) from the accumulative arrays of these parameters of the temporal local neighborhood of the blurred frame (no including the blurred frame itself). I denote the blurred frame as

I_t, and its temporal local neighborhood as $\{I_{t-n}, \cdots, I_{t-2}, I_{t-1}, I_t, I_{t+1}, I_{t+2}, \cdots, I_{t+n}\}$. Here n is the order of the polynomial function used for data fitting. In theory, the minimum neighborhood size required for order-n polynomial data fitting is $n+1$ (not including I_t itself). In my implementation, I set the neighborhood size to be $2n$ to make the neighborhood to be minimum symmetric with respect to I_t (i.e., including n frames immediately before I_t and n frames immediately after it). Note $2n \geq n+1$ for $n \geq 1$ and thus the neighborhood of $2n$ size provides sufficient data items for order-n polynomial data fitting. I set $n = 2$ in all my experiments. Fig. 5.8 shows an example of using polynomial data fitting to estimate the missing global motion parameters.

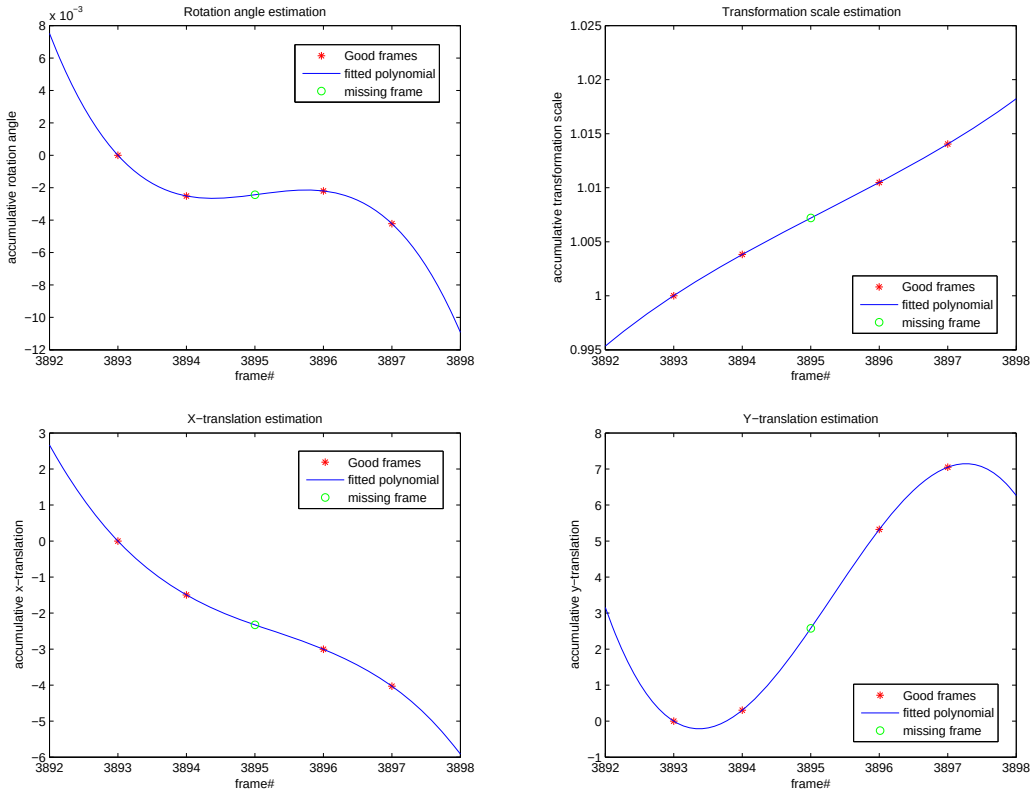


Figure 5.8: Using polynomial data fitting to estimate the global motion parameters of a blurred frame of a video (frame#3895).

Note before the above polynomial data fitting process, the accumulative array of the rotation angle parameter needs to be pre-processed using Algorithm 1 to avoid the generation of abnormal interpolation values. After all these global motion parameters of frame I_t are estimated, the estimated global motion transform matrix of I_t, $\hat{M}_t$, is composed from them following Eq.(2.10).

Given the estimated global motion parameters of the missing (blurred) frame, I can interpolate the frame out from both the frame immediately before it and the frame immediately after it. The transformation matrix used to interpolate the blurred frame I_t out from frame I_{t-1} is computed as:

$$S_{t-1,t} = \hat{M}_t M_{t-1}^T \tag{5.24}$$

where $S_{t-1,t}$ is the similarity transform matrix from frame I_{t-1} to frame I_t. $\hat{M}_t$ is the estimated global motion matrix of I_t, and M_{t-1} is the global motion matrix of I_{t-1}.

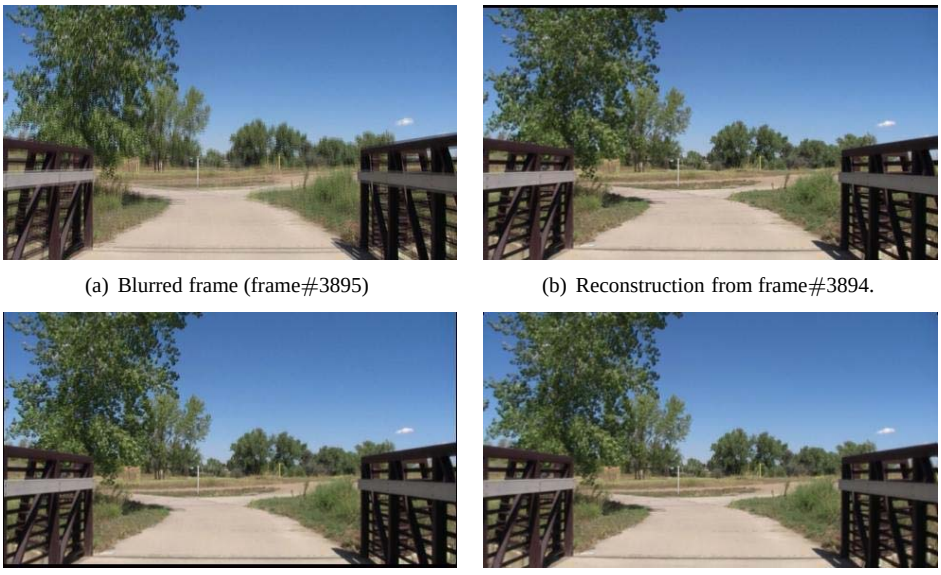
Similarly, the transformation matrix used to interpolate the blurred frame I_t out from frame I_{t+1} is computed as:

$$S_{t+1,t} = \hat{M}_t M_{t+1}^T \tag{5.25}$$

where $S_{t+1,t}$ is the similarity transform matrix from frame I_{t+1} to frame I_t. $\hat{M}_t$ is the estimated global motion matrix of I_t, and M_{t+1} is the global motion matrix of I_{t+1}.

Backward interpolation is used for these frame interpolation operations. This results in two reconstructions of the blurred frame I_t, one transformed from frame I_{t-1} and one transformed from frame I_{t+1}. Linear blending (Eq.(3.30)) is then used to merge them into the final reconstruction. Fig. 5.9 illustrates the process. The image in Fig. 5.9(d) is obviously much clearer than that in Fig. 5.9(a), demonstrating the success of my approach.

I have also compared the results of my approach based on frame interpolation to those of one of the state-of-the-art single image deblurring algorithms [150]. Fig. 5.10 shows the comparison on three blurred frame repair examples. One of the problems of the single image deblurring algorithm of [150] is that because image deblurring is an ill-posed problem [63] it relies on fine parameter tuning to achieve good results. On other words, this algorithm cannot always provide good or even satisfactory results if run automatically. Figs. 5.10(b), 5.10(c) and 5.10(d) show the outputs of this algorithm at different blur scales which is a user parameter of the algorithm. It can be seen that when the user selects the right scale the algorithm may be able to provide a satisfactory result, otherwise its deblurring is either insufficient (e.g. Fig. 5.10(b)-center) or extravagant (e.g. causing the “ringing” artifact in Fig. 5.10(d)-center and Fig. 5.10(d)-right). Another problem of this algorithm is that it assumes the blur is uniformly distributed over



(a) Blurred frame (frame#3895) (b) Reconstruction from frame#3894.

(c) Reconstruction from frame#3896. (d) Final reconstruction

Figure 5.9: Blurred frame repair. (a) is the blurred frame. (b) and (c) are reconstructions of the blurred frame from frames immediately before and after it respectively. (d) is the final reconstruction obtained by merging (b) and (c).

the image and thus uses the same blur kernel to deblur different parts of the whole image, which might not be true in practice. If real motion blur is not uniformly distributed over the image but a single blur kernel is used for deblurring it, the "ringing" artifact may occur anywhere real and estimated blur kernels do not match. This can be seen in Fig. 5.10(d)-right — while the trees in the right-side far-field are properly deblurred, the edge of the road in the near-field and the cloud in the sky are over-deblurred and get "ringing" artifacts around them. The last problem of this deblurring algorithm, which is actually a common problem to all of regular deblurring algorithms, is that because a blurred image may have already lost image information at some spatial frequencies, it is not possible to recover it post-hoc to full extent, depending on how much information is lost due to blur during the capturing phase [63, 111, 152]. This can be seen in Fig. 5.10(b)-center, Fig. 5.10(c)-center and Fig. 5.10(d)-center — none of the scales is able to provide satisfactory results for this blurred frame. Note the above three problems are faced by not only the deblurring algorithm of [150] but also many other deblurring algorithms.

In comparison, my solution of frame interpolation is not limited by the above problems faced by a single image deblurring algorithm like [150]. No matter how much information from the blurred image is lost, my approach can still recover it from neighboring frames automatically. Of course my approach also has limitations: First, it needs the

frame#	CS	SS	frame#	CS	SS
2100	36.9901	0.9574	4100	34.6941	0.9528
2500	36.6847	0.9576	4500	37.5271	0.9609
2900	36.9905	0.9523	4900	36.8760	0.9517
3300	32.8938	0.9252	5300	35.9391	0.9433
3700	33.0787	0.9442	5700	37.0416	0.9558

Table 5.1: Color similarity (CS) scores and structural similarity (SS) scores between the selected frames and the outputs of my approach. The frames are randomly selected from a video composed of 4001 frames (frame#: 2000 to 6000).

temporal neighborhood of the blurred frame, which should not be a problem for any video processing applications. Second, the neighboring frames themselves should not be blurred, otherwise the quality of the reconstructed frame is affected (e.g., the results shown in Fig. 5.10(e) are not completely blur free although they are already much better than the inputs). In this respect, how to combine my frame interpolation approach with regular single image deblurring operations to obtain better video deblurring results is an interesting topic for future research.

It is difficult to quantitatively measure the quality of a deblurred frame since information has been lost in the blur process and we do not know what the true unblurred frame should look like. One could blur a sharp frame and then evaluate the difference between the original and deblurred versions, however my algorithm does not use the blurred frame itself, only its sharper neighbors. To evaluate the quality of my frame reconstruction it should be informative to examine a sharp sequence of frames and measure how well a particular frame is recreated from its neighbors. I perform the following experiments to quantitatively evaluate the quality of my frame interpolation based blurred frame repair approach. I randomly select a set of clear (unblurred) frames and interpolate them out from their temporal neighbors using my approach. Then, I compare the interpolation results of my approach to these selected frames which are regarded as ground truth. I register these two and compute the color similarity (CS) and structural similarity (SS) scores (i.e. PSNR and SSIM scores, see Sec. 4.4.3 for detailed definitions) between them. Table 5.1 shows these scores.

It can be seen from Table 5.1 that my approach achieves very high CS and SS scores for these randomly selected frames. The CS scores are all above 32 and the SS scores are all above 0.92 (note the theoretical maximum of SS scores is 1), which proves the interpolation quality of my approach is high.

5.3 Summary

In this section I described my work on the problem of blurred frame detection and repair. I first proposed an image partial blur detection and classification approach which was shown to be superior to the state-of-the-art. My full-frame blur detection scheme is built upon the output of this approach. The blurred frame repair task is accomplished by global motion chain fixing and interpolation from neighboring video frames.

(a) Three blurred frame inputs

(b) Output of [150] at small blur scale.

(c) Output of [150] at medium blur scale.

(d) Output of [150] at large blur scale.

(e) Output of my approach.

Figure 5.10: Three blurred frame repair examples. Each column of this figure shows an example.

Chapter 6

Conclusion and Future Work

Multi-view video stitching is an important topic in the joint domain of computer vision, image and video processing, multimedia and computational photography. In this thesis I addressed several important sub-problems of multi-view video stitching, including: video stabilization, efficient alignment and stitching of high-resolution and long-duration panoramic videos, color correction for panoramic image/video stitching, image partial blur detection and classification, and blurred frame detection and repair.

In Chapter 1, I first introduced the concepts and the standard procedure for panoramic video stitching (see Fig. 1.2). Then, I described my contributions to individual stages of this stitching procedure in the following chapters. The "video pre-processing" stage aims at removing motion jitter and repairing blurred frames existing in the individual mono-view videos because they not only decrease the quality of the videos but also may distort image information and cause later computer vision algorithms to fail. For motion jitter removal, the standard Kalman filtering based smoothing technique may generate abnormal values in the global motion chain because it does not consider the physical continuity of rotation angle values when successive values cross the boundary of Cartesian quadrants II and III. I thus proposed an approach called continuity aware Kalman filtering of rotation angles to solve this problem.

For repairing the blurred frames in a video, the first task is to detected those frames. My blurred frame detection scheme is based on evaluating the percentage of motion blurred blocks in an input frame. I developed a learning-based image partial blur detection and classification approach for this purpose which combines various kinds of local blur features in an SVM-based learning framework. My blurred frame repair approach treats the blurred frame as a frame with "missing" global motion parameters in its temporal neighborhood, and tries to interpolate this frame out from its immediate neighboring frames after first completing the global motion chain using polynomial curve fitting. In

developing my blur classification technique I have created an easy to use labeling tool for acquiring human-labeled instances of the three blur classes (sharp/motion/defocus). In addition I will release my labeled blur examples as a public dataset for evaluation of blur detection methods.

With the popularization of commodity HDTV cameras, the second stage "multi-view video alignment" and the fourth stage "panoramic video stitching" in the stitching procedure face huge input data and demand processing efficiency. Traditional alignment and stitching algorithms are incapable processing HDTV-level (e.g., 1920x1080 pixels) and long-duration (e.g. one hour long) multi-view videos on standard workstations in a timely manner. I thus proposed three approaches to solve this problem including constrained and multi-grid SIFT matching schemes, concatenated image projection and warping and min-space feathering of high-resolution images. These three approaches together can greatly reduce the computational time and memory requirements for panoramic video stitching, which makes it feasible to stitch long, high-resolution panoramic videos using standard workstations. In addition to the efficiency issues, I also proposed to use multiple frames instead of a single frame for multi-view video alignment to achieve better robustness to the uneven spatial distribution of SIFT features.

My work on the "color correction" stage a primary emphasis of this thesis. I first performed an extensive survey and performance evaluation of representative color correction approaches in the context of two-view image stitching. One contribution of this evaluation work is that it unifies color balancing approaches, color transfer approaches and color correction approaches, which were originally proposed in different research domains including computer vision, video processing, computer graphics and color and imaging sciences. Analyzing these approaches under the single framework of panoramic image stitching, broadens the horizon of the research. The other contributions include giving useful insights and conclusions about the relative performance of the selected approaches, and pointing out the remaining challenges and possible directions for future color correction research. As a result of this worked I have created and made public a standard evaluation dataset for evaluation of color correction algorithms. Based on the conclusions drawn by this evaluation work, I proposed a hybrid and scalable color correction approach for general n-view image stitching and a two-view video color correction approach for panoramic video stitching.

In conclusion, in this thesis I described a sequence of approaches I proposed to address the technical or practical problems existing in the different stages of panoramic image/video stitching. The proposed approaches are novel, effective and efficient solutions to the problems they address. The approaches have either been tested in a real multi-

view video stitching system, the Virtual Exercise Environment (VEE) system (see [151] and Appendix C) or have been compared to the state-of-the-art approaches with real image and video stitching examples.

6.1 Open research issues and future work

This thesis has addressed several basic and critical problems in panoramic video stitching. In additional to these problems, there are many interesting and advanced topics remaining to be explored in future research. Representatives of the open research issues on the topic of panoramic video stitching include:

Efficient SIFT matching schemes In my current work I have developed two locality constrained schemes for efficient SIFT matching. However, other locality constrained schemes such as region constraints [153] can also be explored.

Automatic video synchronization There are two open issues to explore under this topic: First, my thesis assumes the synchronization of the input multi-view video is already done before stitching, and in practice the video synchronization in the VEE system is done manually. It would be better to develop automatic key frame selection and matching schemes for the purpose of automatic video synchronization. This is an open research topic in video processing and there are many new ideas and techniques which could be developed to solve the problem. Secondly, the estimated synchronization point of multi-view fames may occur at sub-frame accuracy (i.e. at a point in between two successive temporal frames), thus appropriate multi-view video interpolation techniques need to be developed to generate a new frame at the synchronization point.

Video stabilization In this thesis a Kalman filtering-based smoothing technique is used to stabilize the videos captured by the mobile capture device. However, video stabilization by itself is a long-studied but not-yet-solved research problem. There is space for exploring other video stabilization techniques or even to develop new ones for producing better quality videos. Also, with the development of automatic video synchronization techniques, multi-view video co-stabilization might be an interesting topic to explore.

Algorithm parallelization All of the approaches proposed in this thesis are implemented using standard MATLAB routines which may or may not be parallelized. The current implementation can process 45-minute to one-hour long HDTV-level panoramic videos in a few days using standard workstations (see Appendix C), which

satisfies the requirement of most virtual reality applications. But there is still room for speed improvement by parallelizing the algorithms or using GPU acceleration, which may be desired by real-time applications such as environment surveillance.

Better blurred frame repair approach It was discussed in Sec. 5.2.2.2 that both frame interpolation and single image deblurring techniques are not perfect at repairing the blurred frames and there is room for improvement by combining the mechanisms of these two kinds of solutions.

The above only lists a few topics on panoramic video stitching I think worthy future work based on my knowledge and experience. Generally speaking, the research on panoramic video stitching is still at its early stage and there is much room for exploration on almost every aspect of it.

Bibliography

[1] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen2. Interactive digital photomontage. ACM Transactions on Graphics, 23:292–300, 2004.

[2] Kebin An, Jun Sun, and Lei Zhou. A linear color correction method for compressed images and videos. IEICE Transactions on Information and Systems, E89-D(10):2686–2689, 2006.

[3] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stephane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. Google street view: Capturing the world at street level. Computer, 43:32–38, 2010.

[4] F. Badra, A. Qumsieh, and G. Dudek. Rotation and zooming in image mosaicing. In Proc. IEEE Workshop on Applications of Computer Vision, pages 50–55, 1998.

[5] A. Bartoli, M. Coquerelle, and P. Sturm. A framework for pencil-of-points structure from motion. In Proc. 8th European Conference on Computer Vision (ECCV'04), pages 28–40, 2004.

[6] C. F. Batten. Autofocusing and Astigmatism Correction in the Scanning Electron Microscope. PhD thesis, University of Cambridge, 2000.

[7] J. Beis and D.G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In Proc. 1997 IEEE Conference on Computer Vision and Pattern Recognition, pages 1000–1006, 1997.

[8] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In Proc. Second European Conference on Computer Vision, pages 237–252, 1992.

[9] Pravin Bhat, L. Zitnick, Michael Cohen, and Brian Curless. Gradientshop: A gradient-domain optimization framework for image and video filtering. ACM Trans. on Graphics, 29, 2010.

[10] M. Brown. Autostitch. `http://www.cs.bath.ac.uk/brown/autostitch/autostitch.html`.

[11] Matthew Brown and David Lowe. Recognising panoramas. In Proc. International Conference on Computer Vision (ICCV'03), volume 2, pages 1218–1225, 2003.

[12] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. International Journal of Computer Vision, 74(1):59–73, 2007.

[13] Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 510–517, 2005.

[14] Matthew A. Brown. Multi-Image Matching using Invariant Features. PhD thesis, University of British Columbia, 2006.

[15] C. Buehler, M. Bosse, and L. McMillan. Non-metric image-based rendering for video stabilization. In Proc. 1994 IEEE Conference on Computer Vision and Pattern Recognition, pages 609–614, 2001.

[16] Peter J. Burt and Edward H. Adelson. A multiresolution spline with application to image mosaics. ACM Transactions on Graphics, 2:217–236, 1983.

[17] G. J. Burton and I. R. Moorhead. Color and spatial structure in natural scenes. Applied Optics, 26(1):157–170, 1987.

[18] Frank M. Candocia. Simultaneous homographic and comparametric alignment of multiple exposure-adjusted pictures of the same scene. IEEE Transaction on Image Processing, 12:1485–1494, 2003.

[19] D. Capel and A. Zisserman. Automated mosaicing with super-resolution zoom. In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 885–891, 1998.

[20] G. Carneiro and A. Jepson. Multi-scale local phase-based features. In Proc. 2003 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03), volume 1, pages 736–743, 2003.

[21] J. Caviedes and F. Oberti. A new sharpness metric based on local kurtosis, edge and energy information. Signal Process.: Image Commun., 19:147–161, 2004.

[22] editor C.C. Slama. Manual of Photogrammetry (4th Ed.). American Society of Photogrammetry, 1980.

[23] A. Cenci, A. Fusiello, and V. Roberto. Image stabilization by feature tracking. In Proc. 10th International Conference on Image Analysis and Processing, pages 665–670, 1999.

[24] Ayan Chakrabarti, Todd Zickler, and William T. Freeman. Correcting over-exposure in photographs. In Proc. 2010 IEEE Conference on Computer Vision and Pattern Recognition, pages 2512–2519, 2010.

[25] S.E. Chen. Quicktime vr c an image-based approach to virtual environment navigation. In Proc. SIGGRAPH'95, page 29C38, 1995.

[26] Xiaogang Chen, Jie Yang, Qiang Wu, and Jiajia Zhao. Motion blur detection based on lowest directional high-frequency energy. In Proc. 17th IEEE Int. Conf. on Image Processing, pages 2533 – 2536, 2010.

[27] Yuanhang Cheng, Xiaowei Han, and Dingyu Xue. A mosaic approach for remote sensing images based on wavelet transform. In Proc. 4th International Conference on Wireless Communications, Networking and Mobile Computing, pages 1–4, 2008.

[28] N. K. Chern, N. P. A. Neow, and Jr M. H. Ang. Practical issues in pixel-based autofocusing for machine vision. In Proc. 2001 IEEE International Conference on Robotics and Automation, pages 2791–2796, 2001.

[29] V. Cheung, B.J. Frey, and N. Jojic. Video epitomes. In Proc. 2005 IEEE Conference on Computer Vision and Pattern Recognition, pages 42–49, 2005.

[30] Sunglok Choi, Taemin Kim, and Wonpil Yu. Robust video stabilization to outlier motion using adaptive ransac. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1897–1902, 2009.

[31] Shengyang Dai and Ying Wu. Estimating space-variant motion blur without deblurring. In Proc. 15th IEEE Int. Conf. on Image Processing, pages 661–664, 2008.

[32] J. Davis. Mosaics of scenes with moving objects. In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 354–360, 1998.

[33] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society, Series B (Methodological), 39:1–38, 1977.

[34] J.J. DePalma and E.M. Lowry. Sine wave response of the visual system ii: Sine wave and square wave contrast sensitivity. J. Opt. Soc. Amer., 52:328–335, 1962.

[35] Z. Duric and A. Resenfeld. Shooting a smooth video with a shaky camera. Journal of Machine Vision and Applications, 13:303–313, 2003.

[36] Yuval Fisher (ed.). Fractal Image Compression: Theory and Application. Springer Verlag, New York, 1995.

[37] Ashley Eden, Matthew Uyttendaele, and Richard Szeliski. Seamless image stitching of scenes with large motions and exposure differences. In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), volume 2, pages 2498–2505, 2006.

[38] S. Erasmus and K. Smith. An automatic focusing and astigmatism correction system for the sem and ctem. Journal of Microscopy, 127:185–199, 1982.

[39] Raanan Fattal, Dani Lischinski, and Michael Werman. Gradient domain high dynamic range compression. ACM Trans. on Graphics, 21, 2002.

[40] Ulrich Fecker, Marcus Barkowsky, and André Kaup. Histogram-based prefiltering for luminance and chrominance compensation of multiview video. IEEE Transactions on Circuits and Systems for Video Technology, 18(9):1258–1267, 2008.

[41] R. Fergus, B. Singh, A. Hertzmann, S. Roweis, and W. Freeman. Removing camera shake from a single photograph. ACM Transaction on Graphics, 25(3):787–794, 2006.

[42] R. Ferzli and L. J. Karam. No-reference objective wavelet based noise immune image sharpness metric. In Proc. 2005 International Conference on Image Processing, volume 1, pages 405–408, 2005.

[43] Rony Ferzli and Lina J. Karam. A no-reference objective image sharpness metric based on the notion of just noticeable blur (jnb). IEEE Transaction on Image Processing, 18(4):717–728, 2009.

[44] David J. Field. Relations between the statistics of natural images and the response properties of cortical cells. Journal of Optical Society of America, 4(12):2379–2394, 1987.

[45] L. Firestone, K. Cook, N. Talsania, and K. Preston. Comparison of autofocus methods for automated microscopy. Cytometry, 12:195–206, 1991.

[46] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Comm. of the ACM, 24(6):381C395, 1981.

[47] J. Foote and D. Kimber. Enhancing distance learning with panoramic video. In Proc. 34th Annual Hawaii International Conference on System Sciences, pages 1–7, 2001.

[48] David A. Forsyth and Jean Ponce. Computer Vision: A Modern Approach. Prentice Hall, 2002.

[49] Pina Marziliano Frederic, Frederic Dufaux, Stefan Winkler, Touradj Ebrahimi, and Genimedia Sa. A no-reference perceptual blur metric. In Proc. 2002 IEEE International Conference on Image Processing, pages 57–60, 2002.

[50] Brian V. Funt and Benjamin C. Lewis. Diagonal versus affine transformations for color correction. Journal of the Optical Society of America A, 17(11):2108–2112, 2000.

[51] D.B. Goldman and J.H. Chen. Vignette and exposure calibration and compensation. In Proc. International Conference of Computer Vision (ICCV'05), pages 899–906, 2005.

[52] Google. Google street view. `www.google.com/streetview`.

[53] A. Ardeshir Goshtasby. 2-D and 3-D Image Registration. John Wiley & Sons, Hoboken, NJ, 2005.

[54] Point Grey. Ladybug2 camera. `http://www.ptgrey.com/products/ladybug2/ladybug2_360_video_camera.asp`.

[55] Dong Guo, Yuan Cheng, Shaojie Zhuo, and Terence Sim. Correcting over-exposure in photographs. In Proc. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'10), pages 515–521, 2010.

[56] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P. Burt. Real-time scene stabilization and mosaic construction. In Proc. 1994 IEEE Conference on Computer Vision and Pattern Recognition, pages 54–62, 1994.

[57] C. Harris and M. Stephens. A combined corner and edge detector. In Proc. 1998 Alvey Vision Conference, pages 147–151, 1998.

[58] Richard Hartley and Andrew Zisserman. Multiple View Geometry in Computer Vision (2nd Ed.). Cambridge University Press, 2004.

[59] David Hasler and Sabine Süsstrunk. Mapping colour in image stitching applications. Journal of Visual Communication and Image Representation, 15:65–90, 2004.

[60] E.H. Helmer and B. Ruefenacht. Cloud-free satellite image mosaics with regression trees and histogram matching. Photogrammetric Engineering & Remote Sensing, 71:1079–1089, 2005.

[61] Rong Hu, Rongjie Shi, I fan Shen, and Wenbin Chen. Video stabilization using scale-invariant features. In Proc. 11th International Conference on Information Visualization, pages 871–877, 2007.

[62] Adrian Ilie and Greg Welch. Ensuring color consistency across multiple cameras. In Proc. International Conference of Computer Vision (ICCV'05), pages 1268–1275, 2005.

[63] P. Jansson. Deconvolution of Image and Spectra, 2nd ed. Academic Press, 1997.

[64] J. Jia, T. Wu, Y. Tai, and C. Tang. Video repairing: Inference of foreground and background under severe occlusion. In Proc. 2004 IEEE Conference on Computer Vision and Pattern Recognition, pages 364–371, 2004.

[65] Jiaya Jia, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Bayesian correction of image intensity with spatial consideration. In Proc. 8th European Conference on Computer Vision (ECCV'04), volume 3, pages 342–354, 2004.

[66] Jiaya Jia and Chi-Keung Tang. Image registration with global and local luminance alignment. In Proc. ICCV'03, volume 1, pages 156–163, 2003.

[67] Jiaya Jia and Chi-Keung Tang. Eliminating structure and intensity misalignment in image stitching. In Proc. 10th IEEE International Conference on Computer Vision, volume 2, pages 1651–1658, 2005.

[68] Jiaya Jia and Chi-Keung Tang. Tensor voting for image correction by global and local intensity alignment. IEEE Transaction on Pattern Analysis and Machine Intelligence, 27(1):36–50, 2005.

[69] X. Jing, Y.X. Hong, and S.X. Xin. A new medical image mosaic algorithm based on fuzzy sets recognition method. In Proc. 2010 International Conference on Bioinformatics and Biomedical Technology (ICBBT), pages 180–184, 2010.

[70] Luo Juan and Oubong Gwun. Surf applied in panorama image stitching. In Proc. 2nd International Conference on Image Processing Theory Tools and Applications, pages 495–499, 2010.

[71] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In Proc. 8th European Conference on Computer Vision (ECCV'04), pages 228–241, 2004.

[72] Seon Joo Kim and Marc Pollefeys. Robust radiometric calibration and vignetting correction. IEEE TPAMI, 30(4):562–576, 2008.

[73] Hsien-Che Lee. Introduction to Color Imaging Science. Cambridge University Press, 2005.

[74] Anat Levin. Blind motion deblurring using image statistics. In 2006 Advances in Neural Information Processing Systems, pages 841–848, 2006.

[75] Anat Levin, Assaf Zomet, Shmuel Peleg, and Yair Weiss. Seamless image stitching in the gradient domain. In ECCV'04, pages 377–389, 2004.

[76] Luhong Liang, Jianhua Chen, Siwei Ma, Debin Zhao, and Wen Gao. A no-reference perceptual blur metric using histogram of gradient profile sharpness. In Proc. 16th International Conference on Image Processing, pages 4369–4372, 2009.

[77] Suk Hwan Lim, Jonathan Yen, and Peng Wu. Detection of out-of-focus digital photographs. Technical Report HPL-2005-14, HP Laboratories Palo Alto, 2005.

[78] Wen-Yan Lin, Siying Liu, Yasuyuki Matsushita, Tian-Tsong Ng, and Loong Fah Cheong. Smoothly varying affine stitching. In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 345–352, 2011.

[79] A. Litvin, J. Konrad, and W. Karl. Probabilistic video stabilization using kalman filtering and mosaicking. In Proc. IS&T/SPIE Symp. Electronic Imaging, Image, and Video Comm., pages 663–674, 2003.

[80] A. Litvinov and Y.Y. Schechner. Addressing radiometric nonidealities: A unified framework. In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 2, pages 52–59, 2005.

[81] Renting Liu, Zhaorong Li, and Jiaya Jia. Image partial blur detection and classification. In Proc. 2008 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, 2008.

[82] David Lowe. Object recognition from local scale-invariant features. In Proc. International Conference on Computer Vision (ICCV'99), volume 2, pages 1150–1157, 1999.

[83] David G. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91–110, 2004.

[84] B.D. Lucas and T. Kanade. An iterative image registration technique with an application in stereo vision. In Proc. Seventh International Joint Conference on Artificial Intelligence (IJCAI-81), pages 674–679, 1981.

[85] S. Mann and R.W. Picard. Virtual bellows: Constructing high-quality images from video. In Proc. First IEEE International Conference on Image Processing, pages 363–367, 1994.

[86] Rafal Mantiuk, Karol Myszkowski, and Hans-Peter Seidel. A perceptual framework for contrast processing of high dynamic range images. ACM Trans. on Applied Perception, 3:286–308, 2006.

[87] X. Marichal, W. Ma, and H. J. Zhang. Blur determination in the compressed domain using dct information. In Proc. 1999 International Conference on Image Processing, volume 2, pages 386–390, 1999.

[88] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In Proc. International Conference of Computer Vision (ICCV'01), volume 2, pages 416–423, 2001.

[89] P. Marziliano, F. Dufaux, S. Winkler, and T. Ebrahimi. Perceptual blur and ringing metrics: Applications to jpeg2000. Signal Process.: Image Commun., 19:163–172, 2004.

[90] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In Proc. 13th British Machine Vision Conference (BMVC'02), volume 1, pages 384–393, 2002.

[91] J Matas, O. Chuma, M. Urbana, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. Image and Vision Computing, 22(10):761–767, 2004.

[92] MathWorks. Matlab. http://www.mathworks.com/products/matlab/.

[93] 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, 7:1150–1163, 2006.

[94] P.F. McLauchlan and A. Jaenicke. Image mosaicing using sequential bundle adjustment. Image and Vision Computing, 20:751C759, 2002.

[95] Microsoft. Photosynth. `photosynth.net`.

[96] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In Proc. 2003 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03), volume 2, pages 257–263, 2003.

[97] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. International Journal of Computer Vision, 60(1):63–86, 2004.

[98] Alec Mills and Gregory Dudek. Image stitching with dynamic elements. Image and Vision Computing, 27(10):1593–1602, 2009.

[99] Harsh Nanda and Ross Cutler. Practical calibrations for a real-time digital omnidirectional camera. In Technical Sketches, CVPR'01, 2001.

[100] N. B. Nill and B. H. Bouzas. Objective image quality measure derived from digital image power spectra. Opt. Eng., 31:813C825, 1992.

[101] Oscar Dalmau-Cede no, Mariano Rivera, and Pedro P. Mayorga. Computing the αcchannel with probabilistic segmentation for image colorization. In Proc. International Conference of Computer Vision (ICCV'07), pages 1–7, 2007.

[102] Nokia. Nokia panorama. `http://store.ovi.com/content/112209`.

[103] Miguel Oliveira, Angel D. Sappa, and Vitor Santos. Unsupervised local color correction for coarsely registered images. In Proc. 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'11), pages 201–208, 2011.

[104] E. P. Ong, W. S. Lin, Z. K. Lu, S. S. Yao, X. K. Yang, and L. F. Jiang. No-reference quality metric for measuring image blur. In Proc. 2003 International Conference on Image Processing, pages 469–472, 2003.

[105] Henryk Palus. Colour spaces. Chapmann and Hall, 1998.

[106] M. Pilu. Video stabilization as a variational problem and numerical solution with the viterbi method. In Proc. 2004 IEEE Conference on Computer Vision and Pattern Recognition, pages 625–630, 2004.

[107] Francçois Pitiť, Anil C. Kokaram, and Rozenn Dahyot. N-dimensional probability density function transfer and its application to color transfer. In Proc. ICCV'05, volume 2, pages 1434–1439, 2005.

[108] Francçois Pitiť, Anil C. Kokaram, and Rozenn Dahyot. Automated colour grading using colour distribution transfer. Computer Vision and Image Understanding, 107(1):123–137, 2007.

[109] Tania Pouli and Erik Reinhard. Progressive color transfer for images of arbitrary dynamic range. Computers & Graphics, 35:67–80, 2011.

[110] Kari Pulli, Marius Tico, and Yingen Xiong. Mobile panoramic imaging system. In Proc. 6th IEEE Workshop on Embedded Computer Vision, in Conjunction with CVPR'2010 (ECVW2010, pages 108–115, 2010.

[111] R. Raskar, A. Agrawal, and J. Tumblin. Coded exposure photography: motion deblurring using fluttered shutter. ACM Transaction on Graphics, 25(3):795–804, 2006.

[112] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. Color transfer between images. IEEE Computer Graphics and Applications, 21(5):34–41, 2001.

[113] Stefan Roth and Michael J. Black. Fields of experts: a framework for learning image priors. In Proc. 2005 IEEE Conference on Computer Vision and Pattern Recognition, volume 2, pages 860–867, 2005.

[114] Jerome Da Rugna and Hubert Konik. Automatic blur detection for meta-data extraction in content-based retrieval context. In Proc. SPIE, volume 5304, pages 285–294, 2003.

[115] H.S. Sawhney and R. Kumar. True multi-image alignment and its application to mosaicing and lens distortion correction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21:235–243, 1999.

[116] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or how do i organise my holiday snaps?. In Proc. 7th European Conference on Computer Vision (ECCV'02), volume 1, pages 414–431, 2002.

[117] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. International Journal of Computer Vision, 37(2):151–172, 2000.

[118] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio. A theory of object recognition: Computations and circuits in the feedforward path of the ventral stream in primate visual cortex. Technical Report MIT-CSAIL-TR-2005-082, MIT Computer Science and Artificial Intelligence Laboratory, 2005.

[119] J. A. Shagri, P. S. Cheatham, and A. Habibi. Image quality based on human visual system model. Opt. Eng., 28:813–818, 1986.

[120] D. Shaked and I. Tastl. Sharpness measure: Towards automatic image enhancement. In Proc. 2005 International Conference on Image Processing, volume 1, pages 937–940, 2005.

[121] Jianbo Shi and Carlo Tomasi. Good features to track. In Proc. 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03), pages 593–600, 1994.

[122] H.Y. Shum and R. Szeliski. Construction of panoramic mosaics with global and local alignment. International Journal of Computer Vision, 36:101–130, 2000.

[123] Dan Simon. Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches. John Wiley & Sons, Inc., 2006.

[124] Irfan Skiljan. Irfanview (ver. 4.23). http://www.irfanview.com.

[125] R. Szeliski. Image mosaicing for tele-reality applications. In Proc. IEEE Workshop on Applications of Computer Vision, pages 44–53, 1994.

[126] R. Szeliski. Video mosaics for virtual environments. IEEE Computer Graphics and Applications, 16:22–30, 2002.

[127] R. Szeliski. Image alignment and stitching: A tutorial. Technical Report MSR-TR-2004-92, Microsoft Research, One Microsoft Way, Redmond, WA, 2004.

[128] Richard Szeliski. Video mosaics for virtual environments. Computer Graphics and Applications, 16(2):22–30, 1996.

[129] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and texture-mapped models. In Proc. SIGGRAPH'97, pages 251–258, 1997.

[130] Yu-Wing Tai, Jiaya Jia, and Chi-Keung Tang. Local color transfer via probabilistic segmentation by expectation-maximization. In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 1, pages 747–754, 2005.

[131] Guozhi Tao, Renjie He, Sushmita Datta, and Ponnada A. Narayana. Symmetric inverse consistent nonlinear registration driven by mutual information. Computer Methods and Programs in Biomedicine, 95(2):105–115, 2009.

[132] Gui Yun Tian, Duke Gledhill, Dave Taylor, and David Clarke. Colour correction for panoramic imaging. In Proc. 6th International Conference on Information Visualisation, pages 483–488, 2002.

[133] Hanghang Tong, Mingjing Li, Hongjiang Zhang, and Changshui Zhang. Blur detection for digital images using wavelet transform. In Proc. 2004 IEEE International Conference on Multimedia and Expo, pages 17–20, 2004.

[134] B. Triggs. Detecting keypoints with stable position, orientation, and scale under illumination changes. In Proc. 8th European Conference on Computer Vision (ECCV'04), volume 4, pages 100–113, 2004.

[135] T. Tuytelaars and L. Van Gool. Matching widely separated views based on affine invariant regions. International Journal of Computer Vision, 59(1):61–85, 2004.

[136] International Telecommunication Union. Recommendation itu-r bt.601-5, 1982-1995.

[137] International Telecommunication Union. Recommendation itu-r bt.709-5, 1990-2002.

[138] Unknown. Open tensor voting framework (OpenTVF). http://sourceforge.net/projects/opentvf/.

[139] Matthew Uyttendaele, Ashley Eden, and Richard Szeliski. Eliminating ghosting and exposure artifacts in image mosaics. In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01), volume 2, pages 509–516, 2001.

[140] J.M. Wang, H.P. Chou, S.W. Chen, and C.S. Fuh. Video stabilization for a hand-held camera based on 3d motion model. In Proc. 16th IEEE International Conference on Image Processing, pages 3477–3480, 2009.

[141] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4):600–612, 2004.

[142] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. In Proc. 2004 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04), volume 1, pages 120–127, 2004.

[143] Wikipedia. Mean square weighted deviation. `http://en.wikipedia.org/wiki/Mean_square_weighted_deviation`.

[144] Gunther Wyszecki and W.S. Stiles. Color Science Concepts and Methods, Quantitative Data and Formulae. John Wiley and Sons, Inc., 2000.

[145] Yao Xiang, Beiji Zou, and Hong Li. Selective color transfer with multi-source images. Pattern Recognition Letters, 30(7):682–689, 2009.

[146] Xuezhong Xiao and Lizhuang Ma. Color transfer in correlated color space. In Proc. 2006 ACM international conference on Virtual reality continuum and its applications, pages 305–309, 2006.

[147] Xuezhong Xiao and Lizhuang Ma. Gradient-preserving color transfer. Computer Graphics Forum, 28:1879–1886, 2009.

[148] Yingen Xiong and Kari Pulli. Color matching for high-quality panoramic images on mobile phones. IEEE Transactions on Consumer Electronics, 56(4):2592–2600, 2010.

[149] Yingen Xiong and Kari Pulli. Color matching of image sequences with combined gamma and linear corrections. In Proc. 2010 ACM Int. Conf. on Multimedia, 2010.

[150] Li Xu and Jiaya Jia. Two-phase kernel estimation for robust motion deblurring. In Proc. 11th European Conference on Computer Vision (ECCV'10), volume 1, pages 157–170, 2010.

[151] Wei Xu, Jaeheon Jeong, and Jane Mulligan. Augmenting exercise systems with virtual exercise environment. In Advances in Visual Computing, LNCS 5875, pages 490–499, 2009.

[152] Wei Xu and Scott McCloskey. 2d barcode localization and motion deblurring using a flutter shutter camera. In Proc. 2011 IEEE Workshop on Applications of Computer Vision (WACV'11), pages 159–165, 2011.

[153] Wei Xu and Jane Mulligan. Feature matching under region-based constraints for robust epipolar geometry estimation. In Advances in Visual Computing, LNCS 5876, pages 264–273, 2009.

[154] Wei Xu and Jane Mulligan. Performance evaluation of color correction approaches for automatic multi-view image and video stitching. In Proc. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'10), pages 263–270, 2010.

[155] Yuchi Xu and Shiyin Qin. A new approach to video stabilization with iterative smoothing. In Proc. 10th IEEE International Conference on Signal Processing, pages 1224–1227, 2010.

[156] Kenji Yamamoto and Ryutaro Oi. Color correction for multi-view video using energy minimization of view networks. International Journal of Automation and Computing, 5(3):234–245, 2008.

[157] Kenji Yamamoto, Tomohiro Yendo, Toshiaki Fujii, Masayuki Tanimoto, and David Suter. Color correction for multi-camera system by using correspondences. In Research posters, SIGGRAPH'06, 2006.

[158] Sai-Kit Yeung, Chi-Keung Tang, Pengcheng Shi, Josien P.W. Pluim, Max A. Viergever, Albert C.S. Chung, and Helen C. Shen. Enforcing stochastic inverse consistency in non-rigid image registration and matching. In Proc. 2008 IEEE Conference on Computer Vision and Pattern Recognition, volume 2, pages 1–8, 2008.

[159] Yun Zeng, Wei Chen, and Qun-Sheng Peng. A novel variational image model: Towards a unified approach to image editing. Journal of Computer Science and Technology, 21:224–231, 2006.

[160] Buyue Zhang, Jan P. Allebach, and Zygmunt Pizlo. An investigation of perceived sharpness and sharpness metrics. In Proc. SPIE 5668, pages 98–110, 2005.

[161] Maojun Zhang and Nicolas D. Georganas. Fast color correction using principal regions mapping in different color spaces. Real-Time Imaging, 10(1):23–30, 2004.

[162] N. Zhang, A. Vladar, M. Postek, and B. Larrabee. A kurtosis-based statistitcal measure for two-dimensional processes and its application to image sharpness. In Proc. Section of Physical and Engineering Sciences of American Statistical Society, pages 4730–4736, 2003.

[163] Y.J. Zheng, S. Lin, and S.B. Kang. Single-image vignetting correction. In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), volume 1, pages 461–468, 2006.

[164] Y.J. Zheng, J.Y. Yu, S.B. Kang, S. Lin, and C. Kambhamettu. Sigle-image vignetting correction using radial gradient symmetry. In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'08), volume 2, pages 1–8, 2008.

[165] Z. Zhu, G. Xu, Y. Yang, and J. Jin. Camera stabilization based on 2.5d motion estimation and inertial motion filtering. In Proc. IEEE Int'l Conf. Intelligent Vehicles, pages 329–334, 1998.

[166] I. Zoghlami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2d mosaic from a set of images. In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 420–425, 1997.

Appendix A

Popular Color Spaces

A.1 RGB color space

The trichromatic theory states that there are three types of photo receptors, approximately sensitive to the red, green and blue region of the visible light spectrum, which respectively represent the long, middle and short wave length sensitivity [144]. Following this theory, the RGB color space describes a color with three components : red (R), green (G) and blur (B). The value of these components is the sum of the respective sensitivity functions and the incoming light:

$$R = \int_{300}^{830} S(\lambda)R(\lambda)d\lambda \tag{A.1}$$

$$G = \int_{300}^{830} S(\lambda)G(\lambda)d\lambda \tag{A.2}$$

$$B = \int_{300}^{830} S(\lambda)B(\lambda)d\lambda \tag{A.3}$$

where $S(\lambda)$ is the light spectrum, $R(\lambda)$, $G(\lambda)$ and $B(\lambda)$ are the sensitivity functions for the R, G and B sensors respectively.

The RGB values depend on the specific sensitivity function of the capturing device. This makes the RGB color space a device-dependent color space. However, there exist methods for the calibration of devices and one can transform the RGB space into a linear, perceptually uniform color space. The RGB color space is the basic color space, which can be (providing calibration data) transformed into other color spaces if needed.

A.2 XYZ color space

There is a linear transformation from the RGB space to the XYZ space. The transformation is as follows [112]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.5141 & 0.3239 & 0.1604 \\ 0.2651 & 0.6702 & 0.0641 \\ 0.0241 & 0.1228 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Note this transformation takes a device-independent conversion that maps white in the chromaticity diagram (CIE xy) to white in RGB, rather than taking the standard D65 white point assumption.

A.3 $l\alpha\beta$ color space

$L\alpha\beta$ color space can be obtained by converting from the XYZ color space in three steps. First, one needs to the image from the XYZ color space to the LMS color space using the following transformation [112]:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3897 & 0.6890 & -0.0787 \\ -0.2298 & 1.1834 & 0.0464 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Next, the data in the LMS color space shows a great deal of skew, which one can largely eliminate by converting the data to logarithmic space **LMS**:

$$\begin{aligned} \mathbf{L} &= logL \\ \mathbf{M} &= logM \\ \mathbf{S} &= logS \end{aligned} \tag{A.4}$$

Finally, one converts from **LMS** to $l\alpha\beta$ using the following transformation

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{L} \\ \mathbf{M} \\ \mathbf{S} \end{bmatrix}$$

The l axis represents an achromatic channel, while the α and β channels are chromatic yellow-blue and red-green opponent channels. The data in this space are symmetrical and compact, while we achieve decorrelation to

higher than second order for the set of natural images tested [112].

A.4 CIECAM97s color space

CIECAM97s color space because it closely relates to the $l\alpha\beta$ color space. The transformation matrix to convert from LMS to CIECAM97s is [112]:

$$\begin{bmatrix} A \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 2.00 & 1.00 & 0.05 \\ 1.00 & -1.09 & 0.09 \\ 0.11 & 0.11 & -0.22 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

Here A, C_1 and C_2 are the three color channels of CIECAM97s. This equation shows that the two chromatic channels C_1 and C_2 resemble the chromatic channels in the $l\alpha\beta$ space except a scaling of the axes, while the achromatic channel is different. Another difference is that CIECAM97s operates in linear space, but $l\alpha\beta$ is defined in log space.

A.5 CIELAB color space

CIE L*a*b* (CIELAB) is the most complete color space specified by the International Commission on Illumination (French Commission internationale de l'clairage, hence its CIE acronym). It describes all the colors visible to the human eye and was created to serve as a device independent model to be used as a reference.

The three coordinates of CIELAB represent the lightness of the color (L* = 0 yields black and L* = 100 indicates diffuse white; specular white may be higher), its position between red/magenta and green (a*, negative values indicate green while positive values indicate magenta) and its position between yellow and blue (b*, negative values indicate blue and positive values indicate yellow).

The CIELAB color space can be transformed from the CIEXYZ color space using the following formulas:

$$L^* = 116 f(Y/Y_n) - 16 \tag{A.5}$$

$$a^* = 500[f(X/X_n) - f(Y/Y_n)] \tag{A.6}$$

$$b^* = 200[f(Y/Y_n) - f(Z/Z_n)] \tag{A.7}$$

where

$$f(t) = \begin{cases} t^{1/3} & \text{if } t > (\frac{6}{29})^3 \\ \frac{1}{3}(\frac{29}{6})^2 t + \frac{4}{29} & \text{otherwise} \end{cases} \tag{A.8}$$

Here X_n, Y_n and Z_n are the CIEXYZ tristimulus values of the reference white point.

A.6 YUV color space

In the European PAL standard the RGB signals are encoded in the YUV space with the following equations [105]:

$$Y = 0.299R + 0.587G + 0.114B \tag{A.9}$$

$$U = -0.147R - 0.289G + 0.437B = 0.493(B - Y) \tag{A.10}$$

$$V = 0.615R - 0.515G - 0.1B = 0.877(R - Y) \tag{A.11}$$

A.7 YCbCr color space

YCbCr color space is defined in the ITU-R BT.601-5 [136] and ITU-R BT.709-5 [137] standards of ITU (International Telecommunication Union). These documents define YCbCr as a color space for digital television systems. Specifically, ITU601 standard is defined for standard-definition television use, and ITU709 standard is defined primarily for HDTV use. Individual color components of YCbCr color space are luma Y, chroma C_b and chroma C_r. Chroma Cb corresponds to the U color component, and chroma Cr corresponds to the V component of a general YUV color space. According to [136, 137], formulae for the direct conversion from RGB to YCbCr are the following:

$$\begin{aligned} Y &= K_{ry}R + K_{gy}G + K_{by}B \\ C_b &= B - Y \\ C_r &= R - Y \\ K_{ry} + K_{gy} + K_{by} &= 1 \end{aligned} \tag{A.12}$$

All coefficients used in the above formulae can be calculated from the just two main coefficients K_{ry} and K_{by}. These two coefficients define a relative contribution of red and blue color components in the total value of luma Y.

These coefficients reflect the relative sensitivity of human vision to the red and blue portions of a visible light. At the same time, values of the 2 main coefficients take into account colorimetric parameters of image reproducing devices, such as the gamma function of displays. In the ITU601 standard, $K_{ry} = 0.299$ and $K_{gy} = 0.114$. In the ITU709 standard, $K_{ry} = 0.2126$ and $K_{gy} = 0.0722$.

Appendix B

Solution of The Hybrid Color Correction Equation (Eq.(4.19))

In Sec. 4.5, the linear system represented by of Eq.(4.19) is used to reconstruct an output image for each input image which obeys the required color and contrast conditions. Specifically, given an input image I, suppose its affine-transform color corrected resulting image is I', its contrast-adjusted gradient fields is $({G_x}', {G_y}')$, and the corresponding unknown output image is O, then by putting Eqs.(4.20) and (4.21) into Eq.(4.19), I get the following equation at each internal pixel position $O(x, y) \in O \setminus \partial O$ (where ∂O denotes the boundary of image O):

$$\lambda_1 O(x,y) + \lambda_2[O(x+1,y) + O(x-1,y) + O(x,y+1) + O(x,y-1) - 4O(x,y)] \\ = \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) - G'_x(x-1,y) + G'_y(x,y) - G'_y(x,y-1)] \quad \text{(B.1)}$$

That is,

$$\lambda_2 O(x-1,y) + \lambda_2 O(x,y-1) + (\lambda_1 - 4\lambda_2)O(x,y) + \lambda_2 O(x,y+1) + \lambda_2 O(x+1,y) \\ = \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) - G'_x(x-1,y) + G'_y(x,y) - G'_y(x,y-1)] \quad \text{(B.2)}$$

That is,

$$\begin{bmatrix} \cdots & \lambda_2 & \cdots & \lambda_2 & (\lambda_1 - 4\lambda_2) & \lambda_2 & \cdots & \lambda_2 & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ O(x-1,y) \\ \vdots \\ O(x,y-1) \\ O(x,y) \\ O(x,y+1) \\ \vdots \\ O(x+1,y) \\ \vdots \end{bmatrix}$$

$$= \lambda_1 I'(x,y) + \lambda_2 [G'_x(x,y) - G'_x(x-1,y) + G'_y(x,y) - G'_y(x,y-1)] \quad \text{(B.3)}$$

This is one line of a linear system $Ax = b$, where

$$x = \begin{bmatrix} \cdots & O(x-1,y) & \cdots & O(x,y-1) & O(x,y) & O(x,y+1) & \cdots & O(x+1,y) & \cdots \end{bmatrix}^T \quad \text{(B.4)}$$

is the unknown variable vector, which is a re-organization of the image O by concatenating its columns one after one (the MATLAB convention).

For each pixel position on the boundary of the image O, Eq.(B.1) needs to be modified according to 1) which side of the boundary (top, bottom, left or right side) the pixel resides, and 2) the Neumann boundary condition (i.e. image gradients perpendicular to the boundary are zero) enforced at that pixel position. There are eight situations: pixel on top-side boundary, pixel on bottom-side boundary, pixel on left-side boundary and pixel on right-side boundary, the top-left corner, the top-right corner, the bottom-left corner and the bottom-right corner, each of which has different modified equations.

Top-side boundary If the pixel $O(x,y)$ is on the top-side boundary of image O, then the Laplacian and the divergence are de-generated into:

$$\triangle O = O(x+1,y) + O(x-1,y) + O(x,y+1) - 3O(x,y) \quad \text{(B.5)}$$

$$div({G_x}', {G_y}') = G'_x(x,y) - G'_x(x-1,y) + G'_y(x,y) \tag{B.6}$$

By putting Eqs.(B.5) and (B.6) into Eq.(4.19), I get

$$\lambda_1 O(x,y) + \lambda_2[O(x+1,y) + O(x-1,y) + O(x,y+1) - 3O(x,y)]$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) - G'_x(x-1,y) + G'_y(x,y)] \tag{B.7}$$

That is,

$$\lambda_2 O(x-1,y) + (\lambda_1 - 3\lambda_2)O(x,y) + \lambda_2 O(x,y+1) + \lambda_2 O(x+1,y)$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) - G'_x(x-1,y) + G'_y(x,y)] \tag{B.8}$$

That is,

$$\begin{bmatrix} \cdots & \lambda_2 & \cdots & (\lambda_1 - 3\lambda_2) & \lambda_2 & \cdots & \lambda_2 & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ O(x-1,y) \\ \vdots \\ O(x,y) \\ O(x,y+1) \\ \vdots \\ O(x+1,y) \\ \vdots \end{bmatrix}$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) - G'_x(x-1,y) + G'_y(x,y)] \tag{B.9}$$

Bottom-side boundary If the pixel $O(x,y)$ is on the bottom-side boundary of image O, then the Laplacian and the divergence are de-generated into:

$$\triangle O = O(x+1,y) + O(x-1,y) + O(x,y-1) - 3O(x,y) \tag{B.10}$$

$$div({G_x}', {G_y}') = G'_x(x,y) - G'_x(x-1,y) - G'_y(x,y-1) \tag{B.11}$$

By putting Eqs.(B.10) and (B.11) into Eq.(4.19), I get

$$\lambda_1 O(x,y) + \lambda_2[O(x+1,y) + O(x-1,y) + O(x,y-1) - 3O(x,y)]$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) - G'_x(x-1,y) - G'_y(x,y-1)] \tag{B.12}$$

That is,

$$\lambda_2 O(x-1,y) + \lambda_2 O(x,y-1) + (\lambda_1 - 3\lambda_2)O(x,y) + \lambda_2 O(x+1,y)$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) - G'_x(x-1,y) - G'_y(x,y-1)] \tag{B.13}$$

That is,

$$\begin{bmatrix} \cdots & \lambda_2 & \cdots & \lambda_2 & (\lambda_1 - 3\lambda_2) & \cdots & \lambda_2 & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ O(x-1,y) \\ \vdots \\ O(x,y+1) \\ O(x,y) \\ \vdots \\ O(x+1,y) \\ \vdots \end{bmatrix}$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) - G'_x(x-1,y) - G'_y(x,y-1)] \tag{B.14}$$

Left-side boundary If the pixel $O(x,y)$ is on the left-side boundary of image O, then the Laplacian and the divergence are de-generated into:

$$\triangle O = O(x+1,y) + O(x,y+1) + O(x,y-1) - 3O(x,y) \tag{B.15}$$

$$div({G_x}', {G_y}') = G'_x(x,y) + G'_y(x,y) - G'_y(x,y-1) \tag{B.16}$$

By putting Eqs.(B.15) and (B.16) into Eq.(4.19), I get

$$\lambda_1 O(x,y) + \lambda_2[O(x+1,y) + O(x,y+1) + O(x,y-1) - 3O(x,y)]$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) + G'_y(x,y) - G'_y(x,y-1)] \tag{B.17}$$

That is,

$$\lambda_2 O(x,y-1) + (\lambda_1 - 3\lambda_2)O(x,y) + \lambda_2 O(x,y+1) + \lambda_2 O(x+1,y)$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) + G'_y(x,y) - G'_y(x,y-1)] \tag{B.18}$$

That is,

$$\begin{bmatrix} \cdots & \cdots & \lambda_2 & (\lambda_1 - 3\lambda_2) & \lambda_2 & \cdots & \lambda_2 & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ O(x,y-1) \\ O(x,y) \\ O(x,y+1) \\ \vdots \\ O(x+1,y) \\ \vdots \end{bmatrix}$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) + G'_y(x,y) - G'_y(x,y-1)] \tag{B.19}$$

Right-side boundary If the pixel $O(x,y)$ is on the right-side boundary of image O, then the Laplacian and the divergence are de-generated into:

$$\triangle O = O(x-1,y) + O(x,y+1) + O(x,y-1) - 3O(x,y) \tag{B.20}$$

$$div({G_x}', {G_y}') = -G'_x(x-1,y) + G'_y(x,y) - G'_y(x,y-1) \tag{B.21}$$

By putting Eqs.(B.20) and (B.21) into Eq.(4.19), I get

$$\lambda_1 O(x,y) + \lambda_2[O(x-1,y) + O(x,y+1) + O(x,y-1) - 3O(x,y)]$$
$$= \lambda_1 I'(x,y) + \lambda_2[-G'_x(x-1,y) + G'_y(x,y) - G'_y(x,y-1)] \tag{B.22}$$

That is,

$$\lambda_2 O(x-1,y) + \lambda_2 O(x,y-1) + (\lambda_1 - 3\lambda_2)O(x,y) + \lambda_2 O(x,y+1)$$
$$= \lambda_1 I'(x,y) + \lambda_2[-G'_x(x-1,y) + G'_y(x,y) - G'_y(x,y-1)] \tag{B.23}$$

That is,

$$\begin{bmatrix} \cdots & \lambda_2 & \cdots & \lambda_2 & (\lambda_1 - 3\lambda_2) & \lambda_2 & \cdots & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ O(x-1,y) \\ \vdots \\ O(x,y-1) \\ O(x,y) \\ O(x,y+1) \\ \vdots \\ \vdots \end{bmatrix}$$
$$= \lambda_1 I'(x,y) + \lambda_2[-G'_x(x-1,y) + G'_y(x,y) - G'_y(x,y-1)] \tag{B.24}$$

Top-left corner If the pixel $O(x,y)$ is the top-left corner of image O, then the Laplacian and the divergence are de-generated into:

$$\triangle O = O(x+1,y) + O(x,y+1) - 2O(x,y) \tag{B.25}$$

$$div({G_x}', {G_y}') = G'_x(x,y) + G'_y(x,y) \tag{B.26}$$

By putting Eqs.(B.25) and (B.26) into Eq.(4.19), I get

$$\lambda_1 O(x,y) + \lambda_2[O(x+1,y) + O(x,y+1) - 2O(x,y)]$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) + G'_y(x,y)] \tag{B.27}$$

That is,

$$(\lambda_1 - 2\lambda_2)O(x,y) + \lambda_2 O(x,y+1) + \lambda_2 O(x+1,y)$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) + G'_y(x,y)] \tag{B.28}$$

That is,

$$\begin{bmatrix} \cdots & \cdots & (\lambda_1 - 2\lambda_2) & \lambda_2 & \cdots & \lambda_2 & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ O(x,y) \\ O(x,y+1) \\ \vdots \\ O(x+1,y) \\ \vdots \end{bmatrix}$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) + G'_y(x,y)] \tag{B.29}$$

Top-right corner If the pixel $O(x,y)$ is the top-right corner of image O, then the Laplacian and the divergence are de-generated into:

$$\triangle O = O(x-1,y) + O(x,y+1) - 2O(x,y) \tag{B.30}$$

$$div({G_x}', {G_y}') = -G'_x(x-1,y) + G'_y(x,y) \tag{B.31}$$

By putting Eqs.(B.30) and (B.31) into Eq.(4.19), I get

$$\lambda_1 O(x,y) + \lambda_2[O(x-1,y) + O(x,y+1) - 2O(x,y)]$$
$$= \lambda_1 I'(x,y) + \lambda_2[-G'_x(x-1,y) + G'_y(x,y)] \tag{B.32}$$

That is,

$$\lambda_2 O(x-1,y) + (\lambda_1 - 2\lambda_2)O(x,y) + \lambda_2 O(x,y+1)$$
$$= \lambda_1 I'(x,y) + \lambda_2[-G'_x(x-1,y) + G'_y(x,y)] \tag{B.33}$$

That is,

$$\begin{bmatrix} \cdots & \lambda_2 & \cdots & (\lambda_1 - 2\lambda_2) & \lambda_2 & \cdots & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ O(x-1,y) \\ \vdots \\ O(x,y) \\ O(x,y+1) \\ \vdots \\ \vdots \end{bmatrix}$$
$$= \lambda_1 I'(x,y) + \lambda_2[-G'_x(x-1,y) + G'_y(x,y)] \tag{B.34}$$

Bottom-left corner If the pixel $O(x,y)$ is the bottom-left corner of image O, then the Laplacian and the divergence are de-generated into:

$$\triangle O = O(x+1,y) + O(x,y-1) - 2O(x,y) \tag{B.35}$$

$$div({G_x}', {G_y}') = G'_x(x,y) - G'_y(x,y-1) \tag{B.36}$$

By putting Eqs.(B.35) and (B.36) into Eq.(4.19), I get

$$\lambda_1 O(x,y) + \lambda_2[O(x+1,y) + O(x,y-1) - 2O(x,y)]$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) - G'_y(x,y-1)] \tag{B.37}$$

That is,

$$\lambda_2 O(x,y-1) + (\lambda_1 - 2\lambda_2)O(x,y) + \lambda_2 O(x+1,y)$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) - G'_y(x,y-1)] \tag{B.38}$$

That is,

$$\begin{bmatrix} \cdots & \cdots & \lambda_2 & (\lambda_1 - 2\lambda_2) & \cdots & \lambda_2 & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ O(x,y+1) \\ O(x,y) \\ \vdots \\ O(x+1,y) \\ \vdots \end{bmatrix}$$
$$= \lambda_1 I'(x,y) + \lambda_2[G'_x(x,y) - G'_y(x,y-1)] \tag{B.39}$$

Bottom-right corner If the pixel $O(x,y)$ is the bottom-right corner of image O, then the Laplacian and the divergence are de-generated into:

$$\triangle O = O(x-1,y) + O(x,y-1) - 2O(x,y) \tag{B.40}$$

$$div({G_x}', {G_y}') = -G'_x(x-1,y) - G'_y(x,y-1) \tag{B.41}$$

By putting Eqs.(B.40) and (B.41) into Eq.(4.19), I get

$$\lambda_1 O(x,y) + \lambda_2[O(x-1,y) + O(x,y-1) - 2O(x,y)]$$
$$= \lambda_1 I'(x,y) + \lambda_2[-G'_x(x-1,y) - G'_y(x,y-1)] \tag{B.42}$$

That is,

$$\lambda_2 O(x-1,y) + \lambda_2 O(x,y-1) + (\lambda_1 - 3\lambda_2)O(x,y)$$
$$= \lambda_1 I'(x,y) + \lambda_2[-G'_x(x-1,y) - G'_y(x,y-1)] \tag{B.43}$$

That is,

$$\begin{bmatrix} \cdots & \lambda_2 & \cdots & \lambda_2 & (\lambda_1 - 2\lambda_2) & \cdots & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ O(x-1,y) \\ \vdots \\ O(x,y+1) \\ O(x,y) \\ \vdots \\ \vdots \end{bmatrix}$$
$$= \lambda_1 I'(x,y) + \lambda_2[-G'_x(x-1,y) - G'_y(x,y-1)] \tag{B.44}$$

Eqs.(B.3), (B.9),(B.14),(B.19), (B.24), (B.29),(B.34),(B.39) and (B.44) form a linear system for all of the pixels in the image O: $Ax = b$. In this system, x is the unknown variable vector where each unknown variable is a pixel in the output image O. If the size of the image O is $M \times N$, then the size of this unknown vector x is $MN \times 1$. A is the coefficient matrix composed properly from the coefficient vectors of Eqs.(B.3), (B.9),(B.14),(B.19),

(B.24), (B.29),(B.34),(B.39) and (B.44). A is a sparse matrix (each row has at most five nonzero elements) of size $MN \times MN$. b is the right side vector of the linear equation system composed from the right side terms of Eqs.(B.3), (B.9),(B.14),(B.19) and (B.24) properly. ('Properly' here means according to whether a pixel $O(x, y)$ is in the internal of the image O, or is on the top, bottom, left or right boundary of O, or is the top-left, top-right, bottom-left, or bottom right corner of O). The size of B is $MN \times 1$. This linear system is solved directly using the MATLAB command $x = A\backslash b$. The output image O is a reshape of the solved x vector.

Appendix C

The Virtual Exercise Environment (VEE) system

I have implemented and integrated the panoramic video stitching techniques described by this thesis into a real virtual reality system called Virtual Exercise Environment (VEE), which is part of the joint project Rec-Tech between the University of Illinois at Chicago and the University of Colorado at Boulder. The goal of the VEE system is to incorporates various computer vision and multimedia techniques to augment standard exercise machines to help people, especially people with disabilities, stay with indoor exercise programs. Specifically, it develops and integrates various computer vision and multimedia techniques to generate large-size virtual environment of outdoor exercise trails, and designs and implements a common interface to various kinds of standard indoor exercise machines including stationary exercise bike, arm ergometer, treadmill and etc. so that the user can easily drive the generated virtual exercise environment.

As Fig. C.1 shows, the VEE system operates in three phases: 1) outdoor exercise trail data collection, 2) data processing and integration, and 3) virtual exercise trail playback. During the outdoor exercise trail data collection phase, not only multi-view videos of the outdoor exercise trail are captured by a multi-view camera set mounted on a mobile platform, and at the same time physical data of the terrain along the trail including accumulated travel distance and road surface height changes are collected by a GPS device mounted on the mobile platform. The data processing and integration phase first stitches the individual videos captured by the three commodity HDTV cameras into a high-definition panoramic video, then it generates trail data units by associating each frame of this panoramic video with the interpolated physical data of the terrain based on the time-stamp generated at data collection time. The generated trail data units are saved into a database indexed by the accumulated travel distance. The last virtual exercise trail playback phase is driven by the user exercising on the exercise machine. Most of the commodity exercise machines record

the total travel distance of the user, which can be used to fetch the corresponding trail data unit for playback. Data playback includes video playback and physical data playback two parts to simulate the real outdoor exercise trails: Frames of the panoramic videos are played back onto a array of high-definition TV screens or a head mounted display (HMD) device, the corresponding road surface height change are physically played back onto the exercise machine the user is using (e.g. to change the slope of the running surface of a treadmill or to change the resistance of an arm ergometer).

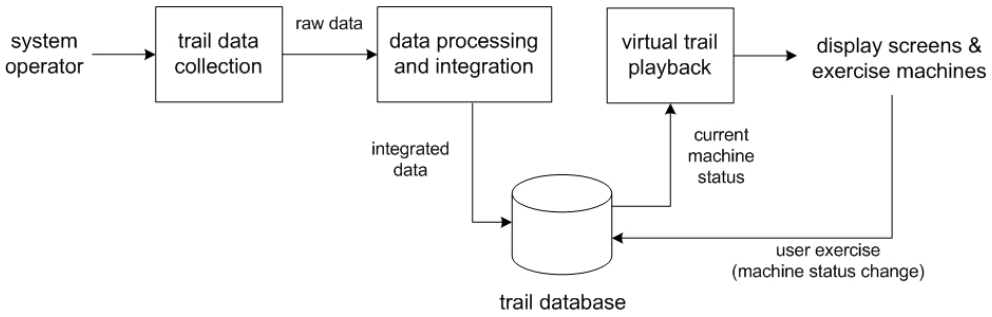


Figure C.1: VEE system flow chart.

The capture hardware used in the VEE system was already shown in Fig.3.8. The VEE system uses three commodity HDTV cameras to capture multi-view videos of outdoor exercise trails. What each camera captures is a mono-view colorful video of 1920x1080 resolution. Every pair of neighboring cameras has a overlap area of small (Horizontal of View) HoV angle so that the captured images can be stitched together in later processing. The panoramic video stitching approaches proposed in this paper is used to seamlessly stitching three (partially overlapped) individual HDTV videos into a high-definition panoramic video of 5760x1080 resolution (see Fig. C.2 for an example frame). The display system illustrated in Fig. C.3(a) is used to play back the generated high-definition panoramic video and the associated physical data of the trail, and is driven by the user exercising on the exercise machines. Furthermore, the VEE system provides a common interface to most of the exercise machines in the market based on the CSAFE (Communications SpecificAtion for Fitness Equipment) protocol. Fig. C.3(b) shows part of the exercise machines that has been successfully applied to the VEE system.

The VEE system requests high-definition (5760x1080 pixels) long-length (typically 45 minutes to one hour long) panoramic videos of outdoor exercise trails which are difficulty or impossible for traditional image and video stitching algorithms to create on standard workstations in a timely manner. In comparison, with the panoramic video

Figure C.2: An example high-resolution panoramic frame (with black padding area extracted) used in the VEE system .

(a) Display system.

(b) Exercise machines used in VEE.

Figure C.3: Hardware of the VEE system.

stitching approaches proposed in this paper, I was able to create panoramic videos at such specification using a standard workstation in a few days. Specifically, the workstation I used to do the task is a regular desktop computer (HP workstation xw4400 base unit with Windows XP SP3 OS, Intel Core 2 Duo CPU at 2.4GHz and 3.25GB RAM). It took traditional stitching algorithms more than 50 seconds on average to generate a panoramic frame of 5760x1080 pixel resolution on this HP desktop computer, while only took about 10 seconds if using the proposed algorithms. In practice, it took about five days to finish the stitching of a 45 minutes long panoramic video of such resolution, while it would had taken roughly one month to do so if using traditional stitching algorithms by my estimation.

An online demo of the VEE system can be found at Youtube: `http://www.youtube.com/watch?v=4gEAfuAbntk`. More details about the VEE system including an online video depository can be found at `http://ml.cs.colorado.edu/RecTechWebpage//Home.html`.