TOWARDS HIGH-FREQUENCY TRACKING AND FAST EDGE-AWARE OPTIMIZATION

Akash Bapat

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2019

Approved by:

Jan-Michael Frahm

Enrique Dunn

Henry Fuchs

Alexander C. Berg

David Gallup

ABSTRACT

Akash Bapat: Towards High-Frequency Tracking and Fast Edge-Aware Optimization
(Under the direction of Jan-Michael Frahm)

Computer vision has seen tremendous success in refashioning cameras from mere recording equipments to devices which can measurement, understand and sense the surroundings. High-frequency and high-speed algorithms in computer vision have now become essential to process the vast amounts of image data generated by a multitude devices as well as enabling real-time applications like augmented reality/virtual reality (AR/VR). This dissertation focuses on increasing the frequency of tracking systems used for AR/VR and to parallelize the computationally intensive edge-aware optimization problem and make it even more efficient.

AR/VR is the most natural way of computing, where the physical and abstract worlds collide. We are sitting at the cusp of a radical change in how humans perform and interact with computing, all due to affordable, and accessible technology. This has been led by major technological advancements in hardware: sensors and high-performance processors, and in algorithms necessary to enable augmented and virtual reality: tracking, rendering and displays. However, the tracking frequency of the current machine vision systems is implicitly limited by the frame-rate of the camera. Humans are sensitive to small misalignments between real and the virtual world, and tracking at high frequencies becomes essential. This thesis presents a prototype system which can track at orders of magnitude higher than the state of the art methods using multiple commodity cameras. The proposed system exploits characteristics of the camera traditionally considered as flaws, namely rolling shutter and radial distortion. The experimental evaluation shows the effectiveness of the method for various degrees of motion.

Furthermore, edge-aware optimization is an indispensable tool in the computer vision arsenal for accurate filtering of depth-data and image-based rendering, which is increasingly being used

for content creation and geometry processing for AR/VR. This dissertation proposes such an edge-aware optimization framework which is efficient, accurate, and algorithmically scales well, all of which are much desirable traits not found jointly in the state of the art. The experiments show the effectiveness of the framework in a multitude of computer vision tasks like computational photography, stereo and super-resolution.

TABLE OF CONTENTS

LIST OF TABLES x

LIST OF FIGURES xi

LIST OF ABBREVIATIONS xvii

CHAPTER 1: INTRODUCTION 1

1.1 Dissertation Statement 3

1.2 Outline of Contributions 3

1.3 Thesis Outline 5

CHAPTER 2: TECHNICAL INTRODUCTION 6

2.1 Image capture and pinhole cameras 6

2.2 Lens and distortion 8

2.2.1 Radial Distortion 9

2.2.1.1 Inverse radial distortion model 10

2.3 Camera shutter 10

2.3.1 Global shutter 11

2.3.2 Rolling shutter 12

2.4 Optimization 14

2.4.1 Gradient descent 14

2.4.2 Heavy-Ball method 14

2.5 Solving a system of linear equations 15

2.5.1 Condition Number 16

CHAPTER 3: RELATED WORK 18

3.1 Rolling shutter 18
3.1.1 RS camera model 18
3.1.2 Motion models 19
3.1.3 Removing RS artifacts 20
3.1.4 Geometric methods for RS 21
3.1.5 RS calibration 24
3.2 Tracking 24
3.2.1 Sensor-based tracking 25
3.2.2 Hybrid tracking 25
3.2.3 Visual-based tracking 26
3.2.3.1 Feature-based systems 26
3.2.3.2 Direct/dense tracking 27
3.3 Edge-aware optimization 29
3.3.1 Bilateral filters 29
3.3.2 Superpixels 30
3.3.3 Machine learning for edge-awareness 31
3.3.4 The domain transform 31
3.3.5 Bilateral solvers 32
CHAPTER 4: ROLLING SHUTTER AND TRACKING 34
4.1 Introduction 34
4.2 Tracking with multiple rolling shutter cameras 36
4.3 The Approach 38
4.3.1 Background 38
4.3.2 Linear Model 39
4.3.3 Shift Estimation 42
4.3.4 Rotation Compensation 43

4.3.4.1 Adaptive Reference 44
4.3.5 Corrective Measures and Confidence Scores 45
4.4 Cluster setup 45
4.5 Experimental Results 47
4.5.1 Simulations 48
4.5.2 Experiments with real data 53
4.5.3 Effect of noise and scene homogeneity 54
4.6 Conclusion 58
CHAPTER 5: LENS DISTORTION AND TRACKING 59
5.1 Introduction 59
5.2 Improvements over RS tracking 61
5.3 Approach 63
5.3.1 Radial Distortion 65
5.3.1.1 Linear independent constraints 66
5.3.1.2 Homography-based warp 68
5.3.1.3 Robust shift estimation 68
5.3.2 Cluster configuration 69
5.4 Experiments 70
5.4.1 Synthetic data 71
5.4.2 Real data 75
5.4.3 Comparison with HoloLens 76
5.4.4 System conditioning and extent of distortion 78
5.4.5 Failure cases and degeneracies 78
5.5 Conclusion 79
CHAPTER 6: EDGE-AWARE OPTIMIZATION 80
6.1 Introduction 80

6.2 Approach 82
6.2.1 Optimization framework 82
6.2.2 Domain transform with the L_2 norm 84
6.3 Applications 86
6.3.1 Stereo optimization 86
6.3.2 Synthetic defocus from depth 88
6.3.3 Depth super-resolution 89
6.3.4 Colorization 90
6.4 Experiments 91
6.4.1 Stereo Optimization 91
6.4.2 Depth super-resolution 93
6.4.3 Colorization 94
6.4.4 Synthetic defocus from depth 99
6.4.5 High-resolution stereo 99
6.4.6 Benchmarking 104
6.5 Conclusion 106
CHAPTER 7: CONCLUSION 107
CHAPTER 8: LIMITATIONS AND FUTURE DIRECTIONS 109
8.1 High-frequency tracking 109
8.1.1 Row-Matching 110
8.1.2 Frame overlap and occlusion 110
8.1.3 Dense sampling of linear segments 111
8.1.4 Drift correction 111
8.1.5 Exposure length 112
8.1.6 Hardware implementation: A sketch 112
8.1.7 Tracking using multi-lens arrays 113

8.1.8 Tracker latency requirements and measurement 114
8.1.9 Generalizability and robustness 114
8.1.10 Social tracking 115
8.2 Edge aware optimization 116
8.2.1 Irregular domains 116
8.2.2 Alternating optimization 117
8.2.3 Semantics and CNN 117
8.2.4 Data compression 118
REFERENCES 119

LIST OF TABLES

Table 4.1 – RMSE in the estimation variables over 1500 row-samples for $v = 1.4$m/s and $\omega = 120$ deg/s with added Gaussian noise. 53

Table 5.1 – RMSE for tracking estimates using real imagery for the 4-camera rig, 6-camera rig, and HoloLens pose estimates compared against the ground-truth Hi-Ball tracking. For comparison with tracker from Chapter 4, the table shows RMSE computed for the same synthetic scene for the 4 and 6-camera cases, and the 20 camera (10 stereo-pairs) case from Chapter 4. The system from Chapter 4 incurs more errors, as it relies on only a single point per-row for pose estimation. 70

Table 6.1 – Performance comparison on images from the Middlebury dataset. The timing for FBS and my method DTS shows the additional time spent in processing MC-CNN (Zbontar and LeCun, 2016), and the total value in the parentheses. DTS takes a fraction of time as compared to FBS (Barron and Poole, 2016) to obtain a significant reduction in error versus MC-CNN. 92

Table 6.2 – Performance of DTS on the depth super-resolution task.DTS is 12x faster than FBS while having comparable performance in most images, especially images with higher upsampling factors. † marks results taken from Barron and Poole (2016). 93

Table 6.3 – The table lists the SSIM and PSNR scores for Cb anc Cr channels for the high-resolution images. HFBS (Mazumdar et al., 2017) can match the performance of DTS but, takes 3.64x more time. If the time taken by HFBS is restricted to that of DTS, HFBS has worse PSNR and SSIM scores. . 97

LIST OF FIGURES

Figure 1.1 – Milgram et al. (1995)'s taxonomy for AR/VR over a reality–virtuality continuum. 2

Figure 2.1 – Left: Image taken with a pinhole camera created from DSLR with 125 ms exposure time. Right: Image taken with a OnePlus5T phone equipped with lens with 0.5 ms exposure. 8

Figure 2.2 – Types of radial distortion. 9

Figure 2.3 – GS camera timing characteristics: All the rows are exposed for t_e time simultaneously, and the readout of consecutive rows is staggered by t_r. After the H rows of the frame are exposed and readout, the capture for the next frame starts after t_f time. 11

Figure 2.4 – Rolling shutter artifacts: As the rolling shutter sweeps in the horizontal direction, artifacts are induced. 12

Figure 2.5 – RS camera timing characteristics: Each row is exposed for t_e time, where the integration between consecutive rows is staggered by t_r. After the H rows of the frame are exposed and readout, the capture for the next frame starts after t_f time. 13

Figure 2.6 – Gradient descent algorithm gets stuck at local minimum while the Heavy-Ball algorithm crosses the hump and converges to global minima when starting at point 2. When starting at point 1, both the update schemes converge to global minima. 15

Figure 4.1 – *Top Left*: My prototype cluster built by mounting 10 Go-Pro cameras on a helmet. (1) through (5) denote stereo pairs pointing forward, right, backward, left, and up, relative to the head of the user. (H) marks a Hi-Ball tracking camera, which was used to obtain ground-truth 6-DoF motion for the device. *Bottom*: View of the room where the real-world experiments were captured. *Middle*: Example stereo pair images for one timepoint of capture from the real-world data. Stereo pair (5) captures a view of the ceiling. *Top right*: Illustration of the rolling shutter effect for a single image. Rows of the image are captured sequentially at high frequency. Motion of the camera during this capture induces distortion in the x-direction of the image plane. *Bottom right*: Tracking the pixel shift from row to row in multiple cameras allows the tracker to estimate 6-DoF device motion from one time point (blue) to the next (red) at a much higher frequency than the camera frame rate. 35

Figure 4.2 – Overview of the tracking system for simulated experiments: (a) Simulation provides image data for multiple virtual cameras. (b) The tracker algorithm estimates shifts and confidence scores for pose estimation. (c) Comparison of the tracking results with ground-truth Hi-Ball tracker motion data. 37

Figure 4.3 – *Virtuous Circle.* Each step reinforces the subsequent step. 39

Figure 4.4 – Measuring s_t and s_d using rotation compensation (best viewed in color). 42

Figure 4.5 – (a) Original rolling shutter image. (b) Visualization of pixel intensity values for the 200^{th} row of the image. (c) Double derivative of the smoothed row. (d) Representative binary descriptor. 43

Figure 4.6 – Rotation compensation: A block of rows from the previous frame are transformed using homography H to predict how the current row i will appear under the rotation of reference row j. 44

Figure 4.7 – Red channel image of a blinking red LED showing the rolling shutter effect. ... 46

Figure 4.8 – Scanned room, with small images showing the RS images captured by the left camera in each of the stereo pair from the virtual cluster. 49

Figure 4.9 – Comparison of pose estimates versus ground truth for the scanned room simulation, using Hi-Ball motion ground truth. (a) Translation. (b) Rotation. (c) Confidence score. (d) Pixel error in display. Note that y-axis scales are not the same (best viewed in color). 50

Figure 4.10 – (a) Translation, (b) Rotation, (c) Confidence score and (d) Pixel error in display, Estimated for Scanned Room, synthetic large motion, note scales not same (best viewed in color). 51

Figure 4.11 – (a) Translation, (b) Rotation, (c) Confidence score and (d) Pixel error in display, estimated for Scanned Room, synthetic extreme motion of $v = 1.4$ m/s and $\omega = 500$ deg/s, note scales not same (best viewed in color) 52

Figure 4.12 – (a) RS image without motion blur, (b) RS image with motion blur, blur is amplified to test for worst case scenarios. 53

Figure 4.13 – (a) Translation, (b) Rotation, (c) Confidence score. The results are noisy which when smoothed give the correct results. Note the axis scales are not same. (Best viewed in color.) 55

Figure 4.14 – (a) Translation, (b) Rotation, (c) Confidence score. Exponential smoothing is used to lower noise. Note the axis scales are not same. (Best viewed in color.) 56

Figure 4.15 – Tracking results for translation in x direction with different levels of noise, and (b) corresponding RMSE. 57

Figure 4.16 – Tracking results for translation in x direction with different levels of scene homogeneity, and (b) corresponding RMSE error. 57

Figure 5.1 – Due to radial distortion, rows of rolling shutter images correspond to rays which span a curve in space. Rolling shutter further provides a dense sampling of the scene in time. This combination of high frequency and curved sampling of the scene provides more information than traditional pinhole cameras. The highlighted region reflects per-row image region corresponding to the curve. 60

Figure 5.2 – Under radial distortion, an input row sample represents a curve in the undistorted image space. (a) Radially distorted image with one row highlighted by a dashed green line; (b) the same row in undistorted image space forms a curve. By approximating this curve with (for example) three linear line segments as shown in (c-e), multiple virtual cameras are obtained that each provide an independent linear constraint. In this sense, a single radially distorted line-image approximates the multiple pinhole cameras used in Chapter 4. For purposes of visualization, the distortion in (a) and (b) has been exaggerated; in practice, I use a larger number of local linear approximations that better fit the undistorted curve. 62

Figure 5.3 – For the same camera motion, the proposed tracker incurs far smaller rendering pixel errors using only 4 cameras as opposed to the 20-camera system of Chapter 4 for synthetic data. 71

Figure 5.4 – Tracking estimates of the proposed 4-camera configuration using synthetic imagery and Hi-Ball tracking data for ground truth: (a) Rotation estimates in degrees and (b) translation estimates in cm. 72

Figure 5.5 – Tracking estimates of the proposed 6-camera configuration using synthetic imagery and Hi-Ball tracking data for ground truth: (a) Rotation estimates in degrees and (b) translation estimates in cm. 73

Figure 5.6 – Tracking estimates of the 6-camera configuration using real imagery and Hi-Ball tracking data for ground truth: (a) Rotation estimates in degrees and (b) translation estimates in cm. Note that the scale of the y-axis is different for each figure. 74

Figure 5.7 – Tracking estimates of the 4-camera configuration using real imagery and Hi-Ball tracking data for ground truth: (a) Rotation estimates in degrees and (b) translation estimates in cm. Note that the scale of the y-axis is different for each figure. 75

Figure 5.8 – Render pixel estimates: The 4-camera configuration (black) has higher error than the 6-camera configuration (green). Blue dots show the rendering error obtained by the method of Kim et al. (2016), which maintains a pose estimate per keyframe, rather than per row. 76

Figure 5.9 – Tracking estimates of the HoloLens compared with the Hi-Ball. Note that the vertical axes are not the same across plots. The horizontal axis is expressed in terms of row-samples, instead of time, for easier comparison. . . . 77

Figure 5.10 – With higher radial distortion, the system stability increases (smaller condition number). Additional views increase stability (4- vs. 6-cameras). The plot shows the worst-case condition number (PinholeCam 4-camera result too large to show). 78

Figure 6.1 – The domain transform solver can tackle a variety of problems, including (a) colorization, (b) depth super-resolution using the color image as reference, and (c-e) depth map refinement. (c-e) shows a color image from the Middlebury dataset (Scharstein et al., 2014) with an initialization (target) obtained from MC-CNN (Zbontar and LeCun, 2016), which is then refined in an edge-aware sense to obtain DTS result (e). 81

Figure 6.2 – Stereo Optimization: The DTS result in (e) is computed using the color image (a) which is used to define color distance in the domain transform, and the target (c) disparity obtained from MC-CNN (Zbontar and LeCun, 2016). The confidence map (d) is used to weigh the target disparity in the optimization (Eq. (6.1)). Notice in the zoomed regions that DTS results are aligned to the edges of the color image. 87

Figure 6.3 – Render from defocus for the Middlebury scenes. The original image is all-in-focus. (a) Original vs. DTS, background in focus. (b) Original vs. DTS, foreground in focus. (c) Original vs. DTS, with the chairs in the front in focus. 88

Figure 6.4 – Depth super-resolution: (a) reference image, (b) confidence (c) ground-truth disparity, (d) bi-cubic interpolation (target), (e) the optimized disparity using DTS, and (f) results using FBS obtained from the author's website (Barron, 2008). The inset highlights the details and the amount of smoothness we obtain in homogeneous regions while being edge-aware. 90

Figure 6.5 – Colorization: (a) input grayscale image (the reference image), (b) user-annotated strokes (the target image), (c) result using DTS, and (d) result of Levin et al. (2004). Note that (c) and (d) look the same with only small differences in the hair. 91

Figure 6.6 – Stereo optimization closeup for Middlebury Pipes scene. (a) Color image. (b) ground-truth (GT) disparity. (c) Target obtained using MC-CNN (Zbontar and LeCun, 2016), and (d) DTS result. 92

Figure 6.7 – Colorization: (a) The grayscale image acts as the reference image and the user annotated strokes as the target, (b) DTS result and (c) Levin et al. (2004) result. Note that (b) and (c) are virtually identical. 94

Figure 6.8 – Propagation of colors: (a-h) shows how the colors from the user-annotated strokes propagate through the image while not crossing strong image edges using DTS. After iteration 50, the changes are small and can be ignored as a trade-off for speed. 95

Figure 6.9 – Colorization: (a) user-annotated color scribbles, (b) DTS results, and (c) results using Levin et al. (2004). 96

Figure 6.10 – Colorization for high-resolution images. (a) The target images retain color in only 20% of the pixels. (b) DTS result. (c) HFBS (Mazumdar et al., 2017) can match the performance of DTS but, takes 3.64x more time. (d) When HFBS is limited to the time of DTS, it has substantially worse PSNR and SSIM scores. 97

Figure 6.11 – Zoom-ins for Fig. 6.10 first row: Notice the speckles in (d) on the handle of the chest and red portion of the meat. 98

Figure 6.12 – Zoom-ins for Fig. 6.10 second row: Notice the speckles in (d) reddish spots on the orange, and red spots on the vase. 98

Figure 6.13 – PianoL scene: (a,b) DTS and MC-CNN's result where the stool is in focus, (c-e) shows a zoomed-in region. (f,g) The guitar is in focus. In (e) and (j), notice the rough edges at the right side of the guitar and on the left side of the leg of the table, which are reduced in the DTS results (d,i). . . 100

Figure 6.14 – PlaytableP scene: (a,b) DTS and MC-CNN's result where the front of the table is in focus, (c-e) shows a zoomed-in region. (f,g) The orange bucket is in focus. In (e) and (j), notice the jarring changes in blur due to incorrect depth, which is reduced in the DTS results (d,i). 101

Figure 6.15 – Shelves scene: (a,b) DTS and MC-CNN's result where the blue bag is in focus, (c-e) shows a zoomed-in region. (f,g) The hanger is in focus. In (e) and (j), notice the ringing near the boundary of the bag, which is removed in the DTS results (d,i). 102

Figure 6.16 – High-resolution stereo data: (a) Color images from the toy dataset, (b) point cloud visualization of results from DTS, and raw depth generated by COLMAP (Schönberger and Frahm, 2016) in (c). The point cloud views highlight that a significant amount of noise present in the raw depthmap is removed using DTS. 103

Figure 6.17 – High-resolution stereo data comparison: DTS is more accurate than HFBS for same time budget, and faster for equal accuracy demands. 104

Figure 6.18 – (a) DTS is independent of blur σ and remains more accurate than HFBS (Mazumdar et al., 2017). 105

Figure 6.19 – Dependence on image resolution for DTS algorithm. 105

Figure 6.20 – Dependence of MAE, RMSE and time on the number of iterations. (a-b) The errors reduce quickly for the first 300 iterations and then reduces gradually for further iterations. The average time taken increases linearly, as shown in (c). This provides a trade-off which can be chosen according to the application. 106

Figure 8.1 – The FPGA sends out control signal to master camera, which in turn drives the exposure in the n RS of the camera cluster. The cameras in turn send a row of the image as soon as the row is readout from the sensor which is processed by the FPGA to give high-frequency and low-latency 6 DoF pose updates. 113

Figure 8.2 – The multiple barrel lenses focus large parts of the scene on a subset of the rows increasing the descriptiveness of the rows as well as provides row-level sync for free across the lenses. 114

LIST OF ABBREVIATIONS

ADAS	adavanced driver assist systems
AR	augmented reality
AR/VR	augmented reality/virtual reality
ASIC	application-specific integrated circuit
BA	bundle adjustment
BGU	bilateral guided upsampling
BLS	BL-Stereo
BRISK	binary robust invariant scalable keypoint
CCD	charge-coupled device
CMOS	complementary metal oxide semiconductor
CNN	convolutional neural net
CRF	conditional random field
DoF	degree of freedom
DSLR	digital single-lens reflex
DT	domain transform
DTS	domain transform solver
EKF	extended Kalman filter

FAST features from accelerated segment test

FBS fast bilateral solver

FoV field of view

FPGA field-programmable gate array

fps frames-per-second

GPS global positioning system

GPU graphics processing unit

GS global shutter

GT ground-truth

HD high-definition

HEC hand-eye calibration

HFBS hardware-friendly bilateral solver

HMD head-mounted display

IMU inertial measurement unit

IR infra-red

JND just noticeable difference

KLT Kanade-Lucas-Tomasi

LED	light-emitting diode
LM	Levenberg–Marquardt
MAD	mean absolute difference
MAE	mean absolute error
MC-CNN	matching cost CNN
MPL	motion-to-photon latency
MVS	multi-view stereo
ODE	ordinary differential equation
ORB	oriented FAST and rotated BRIEF
P3P	perspective-3-point
PKR	peak ratio
PnP	perspective-n-point
PSNR	peak signal-to-noise ratio
PTAM	parallel tracking and mapping
RANSAC	random sampling and consensus
RGB	red-green-blue
RGB-D	red-green-blue-depth
RMSE	root mean square error
RoI	region of interest
RS	rolling shutter

SaM	structure-and-motion
SCAAT	single-constraint-at-a-time
SfM	structure-from-motion
SG-WLS	semi-global weighted least squares
SIFT	scale invariant feature transform
SLAM	simultaneous localization and mapping
SLERP	spherical linear interpolation
SLIC	simple linear iterative clustering
SNR	signal-to-noise ratio
SoC	system on a chip
SSIM	structural similarity
SURF	speeded-up robust features
SVD	singular value decomposition
SVM	support vector machine
VR	virtual reality
YCbCr	luma, chroma blue-difference, chroma red-difference

CHAPTER 1: INTRODUCTION

Human beings continuously measure and sense the surroundings to survive in this complex world. Our visual system performs a crucial function by letting us 'see' by capturing the light incident on the retina. The visual system processes the captured images to provide us with depth cues, motion estimates, relative locations of different objects, recognizes different types of objects and much more, just within seconds. The field of computer vision is set out to provide all of the aforementioned capabilities like sensing, measuring and scene understanding to computers using cameras.

Traditionally, cameras were used for recording the visual aspects of the surroundings like in photographs and videos. Computer vision has transformed cameras from just recording devices to devices which provide sensing and scene understanding capabilities. Computer vision has created new modes of capture which combines computation and imaging in the form of computational photography, developed novel measurement/sensing methods like simultaneous localization and mapping (SLAM) to reconstruct 3D geometry and estimate ego-motion and enabled understanding of higher-level semantics like vegetation, water-body and so on. This thesis focuses on two aspects of computer vision algorithms: efficient and high-frequency processing of image data. These two themes are applied to the the problems of high-frequency tracking for augmented reality/virtual reality (AR/VR) and edge-aware optimization.

The fields of AR/VR tackle the hard problem of fooling the human senses. While virtual reality (VR) presents the user an entirely virtual space in which the user can interact, navigate and feel 'immersed', augmented reality (AR) adds virtual content to the real world. Milgram et al. (1995) provided a taxonomy called 'reality–virtuality continuum' in their seminal work to organize AR and VR as shown in Fig. 1.1. In recent years, we have made great strides across this continuum with

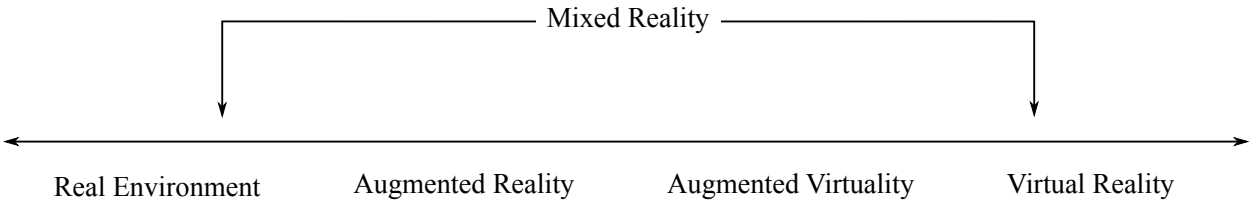


Figure 1.1: Milgram et al. (1995)'s taxonomy for AR/VR over a reality–virtuality continuum.

renewed interest by academicians and the industry alike, reinvigorating this field. This resurgence is in part fueled by AR/VR's great promise in a wide variety of fields such as medical treatment and surgery (Khor et al., 2016), training (Boud et al., 1999), manufacturing (Ong and Nee, 2013) and entertainment (Von Itzstein et al., 2017), as well as technological advances and improved computing capabilities.

Any AR/VR system continuously tracks the user's motion, renders the virtual content/objects according to the view-point, and then finally displays the content on a screen, so that the user believes that the virtual content is immersive. To maintain a reliable registration of the virtual world with the real world, AR/VR applications impose stringent requirements on the accuracy and speed of its core components: tracking, rendering and display (Lincoln et al., 2016). A convincing slight-of-hand for AR/VR requires that the core components be performed at faster rate than user perception (Sanchez-Vives and Slater, 2004). While all three of these present their own challenges, high-quality tracking is a significant roadblock for effective AR/VR, since errors and delay in tracking necessarily limit the performance of the subsequent steps (Welch and Bishop, 1996). This is especially true for AR scenarios, where users are able to notice even small misalignments of the virtual world with the external environment. To reduce misalignments and provide an immersive experience, there is a need of a high-frequency 6 DoF pose tracker. Although, there seems to be no agreement on the right way to do tracking (Welch and Foxlin, 2002), the general consensus is that 6 DoF tracking is better than 3 DoF orientation tracking since it provides parallax cues, affords the ability to move around and much more.

The frequency of camera-based tracking systems is implicitly limited by the frame-rate of the cameras. The frame-rate acts as an implicit barrier to the highest tracking frequency possible and

limits the usability of cameras for tracking, where cameras often play the secondary role of drift correction, while the high-frequency tracking is performed by other sensors like IMUs LaValle et al. (2014). One of the objectives of this dissertation is to address this frame-rate barrier enabling the AR/VR system of the future. I describe an approach in this thesis by leveraging rolling shutter (RS) cameras and estimating poses at frequencies as high as 80kHz. Furthermore, I utilize radial distortion to further constrain the ego-motion, improving the system stability while maintaining accuracy and tracking frequency.

Another objective of this dissertation is to improve efficiency of optimization algorithms that enforce edge-aware constraints. This is particularly useful for VR content creation where a fast optimization scheme is required to estimate depth which is consistent with the color image (Anderson et al., 2016). As such systems continue to improve, higher and higher video resolutions will be necessary to meet the desired quality for immersive VR experiences (Kanter, 2015). This objective is not only limited to content creation, but useful for 3D reconstruction and texturing (Waechter et al., 2014), computational photography (Barron et al., 2015) and real-time pointcloud processing (Valentin et al., 2018).

1.1 Dissertation Statement

The frequency of pose estimation in tracking can be significantly improved by leveraging rolling shutter and radial distortion, both of which are traditionally considered to be artifacts. High-speed edge-aware optimization can be parallelized by leveraging approximate distance-preserving 1D filtering techniques.

1.2 Outline of Contributions

This thesis makes the following contributions that advance the state of the art in high-frequency 6 degree of freedom (DoF) pose estimation using rolling shutter cameras and pushes the boundaries

of high-speed edge-aware optimization. The following research has been published in the support of the research statement (Bapat et al., 2016, 2018; Bapat and Frahm, 2019).

Rolling Shutter Based Tracking:

Traditionally, rolling shutter (RS) is considered to be an imaging artifact that has to be over-come and corrected (Forssén and Ringaby, 2010; Grundmann et al., 2012a). To turn this 'negative' phenomenon into a virtue, I propose to assess the feasibility of estimating a pose for each row of a rolling shutter camera in contrast to the traditional approach of estimating a pose per frame. To that effect, I investigate whether per-row poses can be estimated assuming past image frames and motion history is available. Additionally, I propose to develop a method for high-frequency tracking and design a multi-camera cluster of rolling shutter cameras (Bapat et al., 2016). I present evaluation to study the accuracy of the tracker for synthetic motion paths inside a simulator as well as human motion sequences captured using the Hi-Ball tracker (Welch et al., 1999).

Leveraging Distortion for Tracking:

In addition to rolling shutter, radial distortion is also considered to be an imaging artifact which needs to be corrected. To reduce the number of cameras in the multi-camera cluster, I will leverage radial distortion. Furthermore, I show that radial distortion in fact helps in improving the stability of the tracker (Bapat et al., 2018) while maintaining requisite accuracy.

Edge-Aware Optimization:

In edge-aware optimization, the goal is to optimize a set of variables, *e.g.* depth per pixel, while still respecting the edges in the guiding color image. This is useful in depth refinement, optical flow estimation (Revaud et al., 2015) and VR video stitching (Anderson et al., 2016). Traditionally, such optimization is conducted in a 5D space comprising of 2D pixels and 3D color. Due to the curse of dimensionality, increasing the channels of the image makes such an approach very costly. I propose a method which recognizes that the 5D space is sparsely populated since the image defines a function from 2D pixels to 3D color. Using this knowledge, I propose an edge-aware optimization framework which runs at high-speed and can be applied to a variety of computer vision tasks. I

study the speed-accuracy trade-off of this efficient and parallelizable algorithm (Bapat and Frahm, 2019).

1.3 Thesis Outline

In Chapter 2, I provide technical background and introduce notation used in this thesis. I describe the basics of imaging and camera shutter, camera models and optimization techniques used in the proposed methods.

In Chapter 3, I discuss relevant prior work with a particular focus on RS camera, visual odometry and efficient edge-aware optimization techniques. I provide an overview of RS camera model and the theory developed for removal of RS artifacts, RS camera-based geometric estimation and calibration. Then I describe computer vision based methods for odometry and high-frequency tracking. Finally, I review the edge-aware filters and optimization techniques and their computational complexity with a focus on high-speed and parallelizable methods.

In Chapter 4 and 5, I develop my high-frequency per-row 6 DoF tracking framework. I model the rolling shutter and describe how a linear motion model is useful for high-frequency tracking for AR/VR applications. Additionally, I model the radial distortion of the lens and describe how it is useful in decreasing the number of cameras in the multi-camera system, while maintaining the tracking accuracy.

In Chapter 6 I develop the highly efficient edge-aware optimization framework and show its effectiveness for multiple tasks such as colorization, depth super-resolution and stereo.

Finally I summarize the contributions of the thesis in Chapter 7, state the limitations and outline the future research directions building upon this thesis in Chapter 8.

CHAPTER 2: TECHNICAL INTRODUCTION

In this chapter I will provide an overview of concepts and tools useful for understanding this dissertation. In particular, I will introduce the traditional pinhole camera model, global shutter (GS), rolling shutter (RS) and highlight characteristics of RS camera. Furthermore, I will discuss the lens models used in this dissertation. This thesis uses a system linear of equations to estimate tracking, and gradient descent for edge-aware optimization. I will briefly introduce both of these tools for parameter estimation.

2.1 Image capture and pinhole cameras

A camera is a light capturing device, which can encode the sensed intensity either in the form of chemical changes in a film, or in the form of electric charge in charge-coupled device (CCD) or a complementary metal oxide semiconductor (CMOS) device. To understand the imaging process, let us follow the light during this process for a digital camera. The light first gets emitted from a light source, reflects from the surface of the scene, passes through a lens system (if present) and bends according to the lens refraction. When the camera opens the aperture using a shutter mechanism (physical or electronic), the light enters the camera aperture and falls on a light sensitive pixel like CMOS. This continues till the camera shuts the aperture, during which the camera is continuously integrating/accumulating the incident light. Finally, the camera 'reads out' the intensity value from the sensor by sequentially applying gain amplification and analog-to-digital conversion. This intensity value is stored onto a memory disk.

The imaging process for cameras in the most simple case is modeled by pinhole cameras. Pinhole cameras (or camera obscura) are 'ideal' cameras which have an infinitesimally small aperture and no lens system. Such a camera is represented mathematically via three geometric transforma-

tion applied consecutively: 1. pose, 2. perspective division and 3. scaling and re-centering according to intrinsic matrix. To understand these transformations, first let us define a world coordinate space. Let us assume that the world is referenced using a Cartesian coordinate system and the pinhole camera's location and orientation can be represented by a rigid transformation $\boldsymbol{V}$ also referred to as *pose* as follows:

$$\boldsymbol{V} = \left[\begin{array}{c|c} \boldsymbol{R}_{3x3} & T_{3x1} \\ \hline 0_{1x3} & 1 \end{array}\right], \tag{2.1}$$

where $\boldsymbol{R}$ is the rotation matrix and $T = -\boldsymbol{R}C$, where C is the location of the camera's center of projection. The first of the transformations defined by the pose of the camera maps a 3D point X_1 to another 3D point X_2 in the camera space so that the world is now rotated and centered such that $\boldsymbol{R} = \boldsymbol{I}$, the identity matrix, and $T = \overrightarrow{0}$. In other words, the camera is located at the origin with no rotation.

$$\left[\begin{array}{c} x_2 \\ y_2 \\ \hline z_2 \\ \hline 1 \end{array}\right] = \boldsymbol{V} \left[\begin{array}{c} x_1 \\ y_1 \\ \hline z_1 \\ \hline 1 \end{array}\right], \tag{2.2}$$

or $X_2 = \boldsymbol{V} X_1$ in short. In the following, I will use the homogeneous representation and the non-homogeneous one interchangeably. This is followed by the perspective division or z-division π of the point X_2 which maps the 3D point onto a 2D point in normalized image space.

$$\left[\begin{array}{c} \tilde{p}_x \\ \hline \tilde{p}_y \\ \hline 1 \end{array}\right] = \pi \left(\left[\begin{array}{c} x_2 \\ y_2 \\ \hline z_2 \\ \hline 1 \end{array}\right] \right), \tag{2.3}$$

where $[\tilde{p}_x, \tilde{p}_y] = [\frac{x_2}{z_2}, \frac{y_2}{z_2}]$. This is followed by scaling and centering of the normalized image space into image pixel location $[p_x, p_y]$. This last transformation is represented by the intrinsic matrix

Figure 2.1: Left: Image taken with a pinhole camera created from DSLR with 125 ms exposure time. Right: Image taken with a OnePlus5T phone equipped with lens with 0.5 ms exposure.

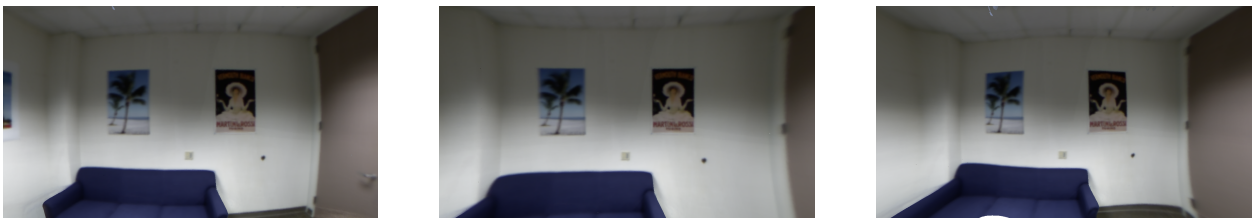
$\boldsymbol{K}_{3x3}$.

$$\begin{bmatrix} p_x \\ \hline p_y \\ \hline 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{K}} \begin{bmatrix} \tilde{p}_x \\ \hline \tilde{p}_y \\ \hline 1 \end{bmatrix}, \tag{2.4}$$

Ideally, (c_x, c_y) is the center of the image, also known as the principal point, and f_x and f_y define the focal length of the camera (zoom) in X and Y directions respectively. This representation of intrinsic matrix assumes no shearing, which is accurate for modern CMOS cameras. Often, the $\boldsymbol{K}$ matrix is subsumed in the π function as a shorthand.

2.2 Lens and distortion

The exposure length required for a pinhole camera is too long to be practical, and hence a lens system becomes necessary. Fig. 2.1 shows qualitatively similar images captured with pinhole camera created using black tape and a Nikon D5300 DSLR and OnePlus5T phone camera. The exposure for pinhole camera is 125 ms while the lens OnePlus5T phone focuses the light onto the sensor drastically reducing the required exposure to 0.5 ms. Although the lens is necessary in practical cameras, it can affect the image quality due to optical aberrations such as vignetting, radial distortion, tangential distortion and color aberrations to name a few. Out of these, I explore the radial distortion in more detail.



(a) Barrel distortion (b) Pincushion distortion (c) Moustache distortion

Figure 2.2: Types of radial distortion.

2.2.1 Radial Distortion

Radial distortion manifests due to uneven magnification by the lens. Fig. 2.2a depicts *barrel distortion* typically observed in wide-angle lenses which tend to have higher magnification at the center than the periphery, while the inverse effect describes the *pincushion distortion* as shown in Fig. 2.2b. If these distortions are simultaneously present, while less common, is called *mustache distortion* due to its similarity to a handlebar mustache, see Fig. 2.2c. I will refer to a class of such distortions as radial distortion.

As radial distortion is induced by the lens system, the easiest way to formulate it is to work in the normalized camera space. Radial distortion as characterised by Brown (1966), also known as Brown-Conrady model (Conrady, 1919), is described below. The distorted normalized image space pixel location $(\tilde{p}_{xd}, \tilde{p}_{yd})$ is modeled as a power series of the unobserved and undistorted normalized image pixel location $(\tilde{p}_{xu}, \tilde{p}_{yu})$ using the radius r as:

$$\tilde{p}_{xd} = \tilde{p}_{xu}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots), \tag{2.5}$$

$$\tilde{p}_{yd} = \tilde{p}_{yu}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots), \tag{2.6}$$

$$r^2 = \tilde{p}_{xu}^2 + \tilde{p}_{yu}^2. \tag{2.7}$$

The more elaborate model described by Brown (1966) also incorporates tangential distortion, but I forgo that here. The distortion parameters k_i along with intrinsic matrix $\boldsymbol{K}$ are estimated using camera calibration method by Zhang (2000).

2.2.1.1 Inverse radial distortion model

The less popular inverse radial distortion model expresses the undistorted pixel locations in terms of distorted locations.

$$\tilde{p}_{xu} = \tilde{p}_{xd}(1 + k'_1\rho^2 + k'_2\rho^4 + k'_3\rho^6 + \cdots), \tag{2.8}$$

$$\tilde{p}_{yu} = \tilde{p}_{yd}(1 + k'_1\rho^2 + k'_2\rho^4 + k'_3\rho^6 + \cdots), \tag{2.9}$$

$$\rho^2 = \tilde{p}_{xd}^2 + \tilde{p}_{yd}^2. \tag{2.10}$$

The above power series is related to the Brown's model via inversion (Tsai, 1987). Although, calibration methods exist which solve for intrinsic matrix $\boldsymbol{K}$ and k' parameters of the inverse model (Horn, 2000), in this thesis I use the Brown's model to describe radial distortion.

2.3 Camera shutter

While the lens system concentrates the incoming beam of light, the camera shutter controls the duration of integration of the light. This duration is called the *exposure time*, denoted by t_e. The camera keeps sensing the light throughout the exposure time, effectively integrating the irradiance E of light reaching a pixel location p. The digital intensity value captured by the camera is given by

$$I(p) = f_{cam}\left(\int_0^{t_e} E(p)\, dt\right) \tag{2.11}$$

The function $f_{cam}(\cdot)$ encapsulates the hardware and sensor characteristics particular to the camera, transforming the analog sensor response into a digital pixel intensity value.

One of the most primitive forms of shutter is a lens cap used to block light after a long exposure. This was later replaced by the use of mechanical shutters like a diaphragm or leaf shutter. A form of mechanical *rolling shutter* (RS) used one or more curtains that move across the aperture to uncover the sensor. The following describes the electronic shutters available in modern digital cameras.

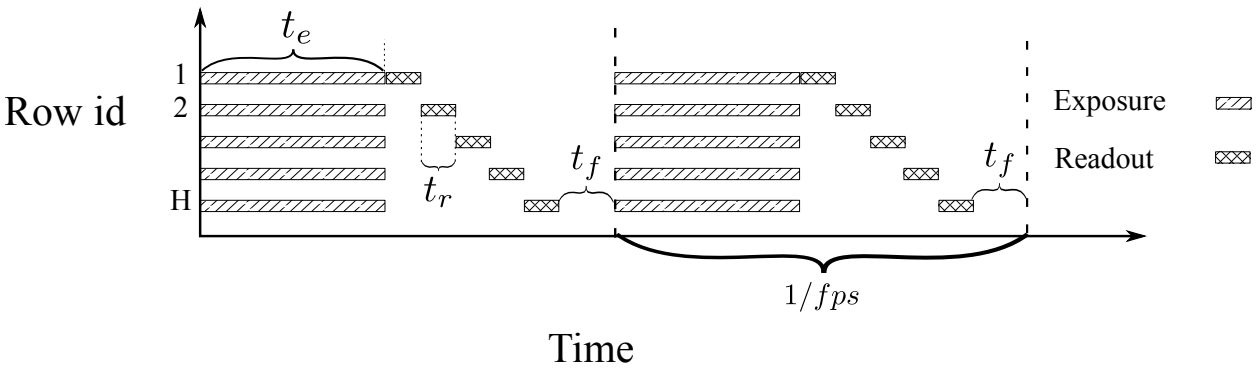
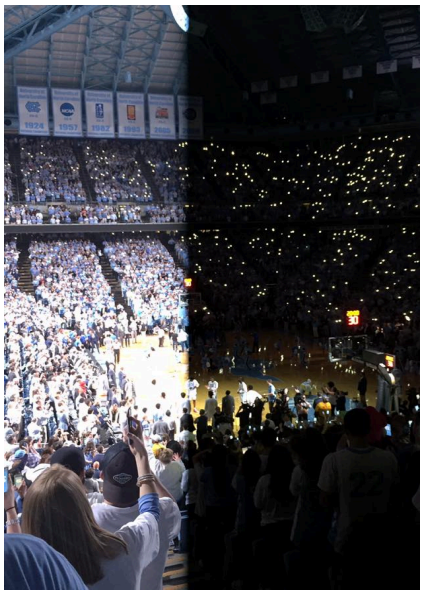


Figure 2.3: GS camera timing characteristics: All the rows are exposed for t_e time simultaneously, and the readout of consecutive rows is staggered by t_r. After the H rows of the frame are exposed and readout, the capture for the next frame starts after t_f time.

2.3.1 Global shutter

Traditional film cameras, as well as CCD arrays in early digital cameras, exposed the entire 2D image space during a common exposure period t_e; this type of exposure is called global shutter. As digital cameras became popular, electronic *global shutter* (GS) became the defacto standard. After a digital camera stops exposure, the readout hardware starts extracting the grayscale value from each row sequentially spending t_r time per row, see Fig. 2.3. After readout, the camera spends t_f time called frame-delay to reset the pixels and prepare for the next frame. For some GS cameras, t_f can be zero, or the readout is done simultaneously to support higher frames-per-second (fps).

To support video capture, such readout circuitry needs to be capable of reading the entire image in a very short amount if time. The most frequently used solution to enable fast readout is to use a dedicated readout circuits. This results into large amounts of the chip area being utilized for readout circuitry, increasing the cost and at the same time limiting the resolution of the sensor by physically limiting the packing density of the pixels (Ge, 2012). Affordable commodity cameras avoid this by time-multiplexing the readout circuitry by employing a *rolling shutter* (RS), which we will look at next.



(a) The lights switched off while the rolling shutter camera was exposing the frame creating a half dark and half bright image. Image courtesy of Chelsey McCotter.

(b) Static guitar strings.

(c) Rolling shutter causes the string to split after plucking.

Figure 2.4: Rolling shutter artifacts: As the rolling shutter sweeps in the horizontal direction, artifacts are induced.

2.3.2 Rolling shutter

With the advent of low-power CMOS camera sensors, rolling shutter (RS) cameras have quickly become ubiquitous for everyday camera-equipped devices. To save space on the silicon chip, the newer CMOS sensors perform sequential exposure of sensor rows, which only requires a single, simpler set of readout circuitry to read all rows and minimal buffer memory (Bradley et al., 2009). In this fashion, each individual row of the RS image is actually a snapshot of the scene at a different time instance. For image shots without significant scene or camera motion, rolling shutter is often an effective, low-cost imaging technique. However, when the scene is dynamic, *e.g.* strings of a guitar (Fig. 2.4c), or sudden changes in ambient light (Fig. 2.4a), or when the camera has fast motion relative to the frame rate of the camera, artifacts such as wobble, skew and other undesirable effects become apparent (Forssén and Ringaby, 2010).

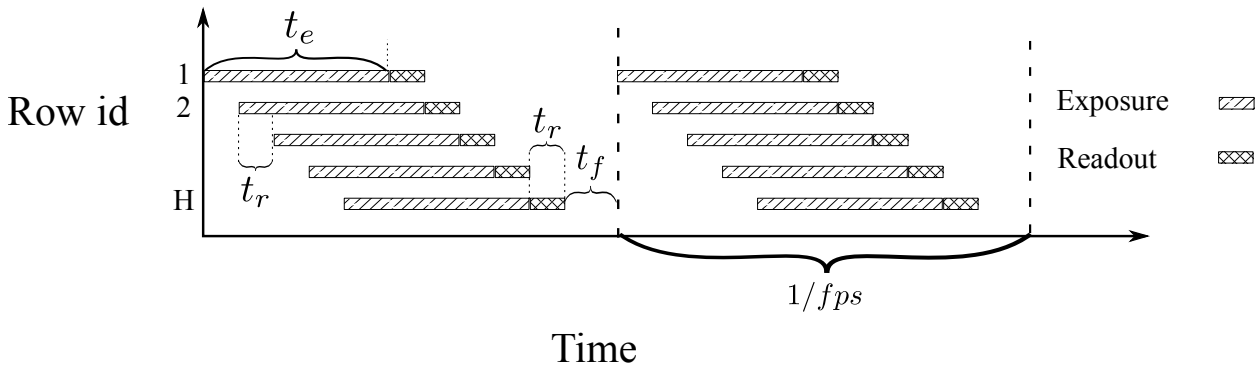


Figure 2.5: RS camera timing characteristics: Each row is exposed for t_e time, where the integration between consecutive rows is staggered by t_r. After the H rows of the frame are exposed and readout, the capture for the next frame starts after t_f time.

A RS camera can be characterized by the exposure time t_e, the row (line) time delay t_r between the start of exposure of consecutive rows and t_f, the frame delay between consecutive frames. The camera sensor exposes a row for t_e time and at the end of each row exposure, consumes t_r time to read it out, see Fig. 2.5 As this process time-multiplexes the readout circuitry across all rows, the start of consecutive row exposures is staggered by t_r. The time between the end of readout of last row and the start of exposure of first row is typically used for noise reduction and reset (Šmıd and Matas, 2017). In general, all of these are related to the fps of the camera (see Fig. 2.5):

$$\frac{1}{\text{fps}} = Ht_r + t_f + t_e, \tag{2.12}$$

where the rolling shutter is sweeping from top to bottom of the image and H is the image height. For a starting time τ_0, the y^{th} row of the camera of frame id f_i in Fig. 2.5 is exposed at time t given by:

$$t = \tau_0 + yt_r + f_i \frac{1}{\text{fps}}. \tag{2.13}$$

2.4 Optimization

Mathematical optimization is a technique of parameter estimation given a maximization or minimization problem. In the minimization setting, given a function $f_{opt}(\Theta, \mathcal{D})$ of observed data $\mathcal{D}$ and unknown parameters Θ, we want to estimate Θ such that $f_{opt}(\cdot)$ is minimized. This general framework of optimization encapsulates two methods we are interested here: 1. gradient descent, and 2. the heavy-ball method.

2.4.1 Gradient descent

Gradient descent, also known as steepest gradient descent, is an iterative algorithm which estimates the parameters Θ at every iteration by using the local gradient direction $\nabla f_{opt}(\Theta)$. To estimate the Θ_n at the n^{th} iteration, gradient descent defines the following update rule:

$$\Theta_n = \Theta_{n-1} - \gamma \nabla f_{opt}(\Theta), \tag{2.14}$$

where γ is the step size. For more details on estimating optimal γ and convergence rate of gradient descent, I refer the reader to Nocedal and Wright (2006). Gradient descent will give the optimal solution for a convex function, but converges to a locally optimal solution for general functions. For example, in Fig. 2.6, when the starting point Θ_0 is 'start point 1', gradient descent will converge to the global minima, but if Θ_0 is 'starting point 2', then it will get stuck at the local minima due to the bump ahead.

2.4.2 Heavy-Ball method

Heavy-Ball method by Polyak (1964) is a flavor of gradient descent which uses the past gradient updates by maintaining the *momentum*. It is inspired by the second order ordinary differential equation (ODE) which models the motion of a ball under friction f_{opt} and a potential field:

$$\ddot{\Theta} + a\dot{\Theta} + b\nabla f_{opt}(\Theta) = 0 \tag{2.15}$$

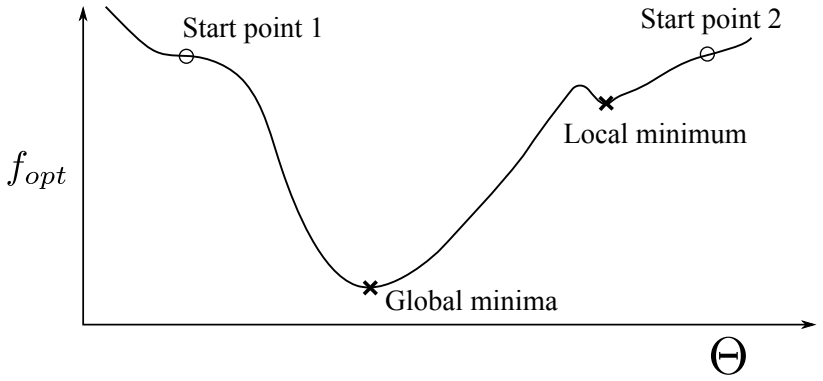


Figure 2.6: Gradient descent algorithm gets stuck at local minimum while the Heavy-Ball algorithm crosses the hump and converges to global minima when starting at point 2. When starting at point 1, both the update schemes converge to global minima.

The discrete version of this ODE is

$$\Theta_n = \Theta_{n-1} - \gamma \nabla f_{opt} + \beta \underbrace{(\Theta_{n-1} - \Theta_{n-2})}_{\text{momentum}} \tag{2.16}$$

When $\beta = 0$, this method is the same as gradient descent. The momentum term in Eq. 2.16 allows the optimizer to cross small bumps in the cost function as in Fig. 2.6 when Θ_0 is 'start point 2'. Hence the Heavy-Ball method in this particular case would converge to the global minima. Additionally, the momentum reduces jitter in the gradient by effectively smoothing it when traversing along the cost surface during minimization.

2.5 Solving a system of linear equations

A set of linear constraints on the unknown parameters defines a system of linear equations. Here, I will focus on the non-homogeneous case of system linear of equations defined by matrix $\boldsymbol{A}$ and vector B. If the matrix $\boldsymbol{A}$ is full-rank square matrix, then the Θ can be directly computed using the inverse.

$$\boldsymbol{A}\Theta = B \tag{2.17}$$

$$\Theta = \boldsymbol{A^{-1}} B$$

This thesis utilizes an over-constrained system of equations, *i.e.* more linear equations are available than the number of unknowns in Θ. Alternatively, $\boldsymbol{A}$ is full-rank non-square matrix. The solution to such an over-constrained system can be obtained using a number of methods such as Moore-Penrose pseudo-inverse, Cholesky decomposition and singular value decomposition (SVD). I will refer the interested reader to Press et al. (2007) for details about these methods. Now I will describe how the linear system's stability is characterized by analyzing $\boldsymbol{A}$ and B.

2.5.1 Condition Number

The estimation of Θ for a square matrix $\boldsymbol{A}$ can be reformulated as a minimization problem:

$$\min_{\Theta} ||\boldsymbol{A}\Theta - B||_2. \tag{2.18}$$

For an error δb in B, the relative ratio of change in solution Θ to the relative change in B is

$$\kappa(\boldsymbol{A}) = \max \left(\frac{\frac{||\boldsymbol{A}^{-1}\delta b||}{||\boldsymbol{A}^{-1}B||}}{\frac{||\delta b||}{||B||}} \right) = \max \left(\frac{||\boldsymbol{A}^{-1}\delta b||}{||\delta b||} \frac{||\boldsymbol{A}\Theta||}{||\Theta||} \right) = ||\boldsymbol{A}^{-1}||\,||\boldsymbol{A}|| \geq 1 \tag{2.19}$$

$\kappa(\boldsymbol{A})$ is also know as the condition number of the linear system, and for l_2 norm, it can be expressed as the ratio of singular values σ of $\boldsymbol{A}$:

$$\kappa(\boldsymbol{A}) = \frac{\sigma_{\max}(\boldsymbol{A})}{\sigma_{\min}(\boldsymbol{A})} \geq 1 \tag{2.20}$$

If the system is ill-conditioned, $\kappa(\boldsymbol{A}) \gg 1$. Analyzing the condition number is useful for checking the system stability, and I will use it to indirectly compare multi-camera tracking systems in Section 5.4.4. Weighted system of linear of equations places a weight per equation by using a diagonal matrix $\boldsymbol{W}$. The optimization reformulation can be rewritten as follows:

$$\min_{\Theta} ||\boldsymbol{W}\,(\boldsymbol{A}\Theta - B)\,||_2 \tag{2.21}$$

The weights $\boldsymbol{W}$ are carefully chosen such that the condition for the system in Eq. 2.21 $\kappa(\boldsymbol{W}\boldsymbol{A})$ decreases.

CHAPTER 3: RELATED WORK

Rolling shutter camera, visual tracking and edge-aware optimization each have been studied extensively. The following sections describes the relevant computer vision techniques and recent advances in these topics.

3.1 Rolling shutter

The ubiquitous presence of rolling shutter (RS) cameras in almost every cell-phone and low-cost camera has driven much research to 1. formulate its imaging model, 2. to remove RS induced artifacts, 3. to adapt traditional 3D methods for RS cameras, and 4. to estimate the timing parameters to calibrate the camera. Next, I will discuss each of these in detail.

3.1.1 RS camera model

In rolling shutter capture, each row (or column) is captured at a slightly different time. For simplicity, let us consider that the rolling shutter sweeps from top to bottom of the image frame. For correct modeling of rolling shutter, one has to consider a (potentially) different pose for each row in the image. To understand model rolling shutter, let us modify the pinhole camera model from Section 2.1 which is repeated here for clarity. In the following the intrinsic matrix is subsumed in the function π:

$$\left[\begin{array}{c} p_x \\ \hline p_y \\ \hline 1 \end{array}\right] = \pi\left(\boldsymbol{V} X\right) \tag{3.1}$$

This pinhole camera model projects the 3D point X according to the pose $\boldsymbol{V}$ onto the pixel $[p_x, p_y, 1]^T$ via the perspective projection function π. For RS camera, the pose $\boldsymbol{V}$ changes with time, and the

camera projection can be written similarly as:

$$\left[\begin{array}{c} p_x(t) \\ p_y(t) \\ \hline 1 \end{array}\right] = \pi\left(\boldsymbol{V}(t)\, X\right) \tag{3.2}$$

Remember that the Y coordinate $p_y(t)$ is analogous to the time according to Eqn. 2.13.

3.1.2 Motion models

Using this definition for RS camera projection, a variety of motion models have been explored to simplify the projection. Here, I will dive into more detail for a select few.

Translation only model

Translation only model assumes that the RS camera undergoes translation with a constant orientation during the image capture. This is especially true for adavanced driver assist systems (ADAS) (Saurer et al., 2015), or for systems with high linear (translational) velocity relative to angular (rotational) velocity. For such a motion model, the RS reprojection can be simplified to the following:

$$\left[\begin{array}{c} p_x(t) \\ p_y(t) \\ \hline 1 \end{array}\right] = \pi\left(\left[\begin{array}{c|c} \boldsymbol{R_0} & T(t) \\ \hline 0 & 1 \end{array}\right] X\right), \tag{3.3}$$

where $\boldsymbol{R_0}$ is the initial rotation of the camera, which is often assumed to be the identity matrix. Two models are used to put further constraints on the intra-frame motion: 1. uniform (constant) velocity, and 2. constant acceleration. These constraints further limit the complexity of the model and make the computation tractable (Purkait and Zach, 2018). Polynomial models which capture more complex motion are also used to model intra-frame translation (Liang et al., 2008).

Rotation only model

A rotation only model is effective to model hand-held device motion, and is often used to correct distortions induced by RS in smartphones. Information about angular acceleration and angular velocity can be obtained using external sensors like IMUs which provide priors on the motion, simplifying the motion estimation (Lee et al., 2018). This rotation only model is defined as follows:

$$\begin{bmatrix} p_x(t) \\ p_y(t) \\ \hline 1 \end{bmatrix} = \pi \left(\left[\begin{array}{c|c} \boldsymbol{R}(t) & T_0 \\ \hline 0 & 1 \end{array} \right] X \right), \tag{3.4}$$

where T_0 is the initial camera position, and is often assume as the zero vector ($\overrightarrow{0}$). Hanning et al. (2011) use an extended Kalman filter (EKF) based method to filter the IMU measurements to estimate rotation and perform rolling shutter compensation and video stabilization. These methods require calibration between the camera and the IMU. Karpenko et al. (2011) presented such a calibration method which uses a single video with a rotation only motion model to estimate the calibration and correct for RS distortions.

3.1.3 Removing RS artifacts

Traditionally rolling shutter is regarded as a 'distortion' introduced by the capture procedure of the cameras. Researchers have long striven to remove rolling shutter and the artifacts resulting from camera motion and the different exposure start times of the various rows of the camera have generally been regarded as nuisances of the imaging process. RS artifact removal has been studied extensively, and I will refer the interested reader to a rich set of approaches here (Liang et al., 2008; Heflin et al., 2010; Forssén and Ringaby, 2010; Grundmann et al., 2012b; Purkait et al., 2017; Rengarajan et al., 2016). Ringaby and Forssén (2012) parametrized intra-frame rotation with a linear spline and used this model for image rectification and video stabilization. In their approach, homographies between images rows are used to reconstruct the rectified image. In Chapter 4, I

utilize a similar approach and leverage such per-row homographies to compensate for rotation in the RS image. More recently, Rengarajan et al. (2017) proposed to correct the artifacts by utilizing the widely popular convolutional neural nets (CNNs) to predict polynomial models of rotation in Z direction and translation in X. They argue that these motion pattern induce the most noticeable RS artifacts.

In addition to RS induced artifacts, the combination of large exposure and strong motion induces motion blur aggravating the distortions. Su and Heidrich (2015) proposed a method to remove motion blur using a single rolling shutter image. Their method explicitly models for RS to estimate the motion during exposure by fitting polynomials to each degree of freedom of camera motion. Using this estimate, they invert the motion induced blur. Meilland et al. (2013) demonstrate a real-time structure-and-motion (SaM) system for RGB-D sensor while accounting for motion blur in a RS camera by assuming uniform velocity over the image.

3.1.4 Geometric methods for RS

The recent popularity of RS cameras in smartphones has led to a trend where traditional geometric algorithms are being adapted to model RS. The following explores efforts to address the absolute pose problem, the relative pose problem, stereo, structure-from-motion (SfM) and calibration algorithms for RS cameras.

Minimal Gröbner basis solvers for PnP

The perspective-n-point (PnP) problem seeks to estimate the pose of a calibrated camera given n correspondences between 3D points and their 2D pixel projections in the image. Minimal solvers sample the minimal number of correspondences required to solve for the pose from the n available correspondences. For robust pose estimation, this minimal solver is used inside a random sampling and consensus (RANSAC) loop. The PnP problem for a RS camera is explored by Albl et al. (2015). They show that the absolute pose problem (PnP) can be solved using six 2D-to-3D correspondences. They first use the traditional perspective-3-point (P3P) algorithm (Haralick et al., 1991) to get an

initial orientation estimate and then use the linearized motion model to account for rolling shutter. To solve for the motion, they use the Gröbner basis method. The Gröbner basis technique is useful to solve a polynomial system of equations which naturally arise in the PnP problem (Cox et al., 2006). When the vertical direction is available,*e.g.* via IMU in a smartphone, Albl et al. (2016a) show that the PnP problem can be solved using only five 2D-to-3D correspondences. Similar to the previous approach, they use Gröbner basis method and the linear motion model, but do not require an initial orientation estimate while being more efficient. Saurer et al. (2015) developed another minimal solver based on the Gröbner basis. They argue that when RS cameras are used in cars, the dominant motion is translation, and hence a translation only model with linear velocity is appropriate. They used five 2D-3D correspondences to estimate the translation and the unknown constant rotation.

Relative pose problem

Dai et al. (2016) developed epipolar constraints for RS cameras and showed that the constraints do not define a line but rather define epipolar curves. Analogous to the linear 8-point algorithm (Longuet-Higgins, 1981) for GS camera, they derive a 44-point linear algorithm for RS camera to solve for a 7×7 essential matrix assuming uniform rotation and translation. For a purely translating RS camera, they also propose a 20-point linear algorithm to solve for a 5×5 essential matrix. Using 44 points in a RANSAC loop is impractical, and Lee and Yoon (2017) alleviate this using IMU readings and propose a linear solver using 11 points for a camera undergoing constant translational and rotational velocities. When angular velocity dominates the linear velocity, only rotational model is useful and Lee et al. (2019) develop a Gröbner basis method to estimate an essential matrix using five correspondences, where the instantaneous rotation is obtained from IMU. Zhuang et al. (2017) estimate differential relative pose for constant translational and rotational acceleration for RS cameras. Using optical flow to establish correspondences, they can estimate the relative pose using 9 correspondences. For constant translational and rotational velocities, the same approach can be used to estimate the relative pose using only 8 correspondences.

Multi-view stereo structure-from-motion

Saurer et al. (2013) adapted the plane sweeping stereo algorithm (Yang and Pollefeys, 2003; Gallup et al., 2007) for RS and showed that the combination of RS artifacts and lens distortion leads to biased 3D reconstruction if global shutter is naïvely assumed. SfM traditionally assumes GS and was thus shown to be unreliable when used with images with large RS distortion (Liu et al., 2011; Hedborg et al., 2011). To resolve this, Hedborg et al. (2012) propose a RS aware bundle adjustment (BA) method to account for the RS effect. This effort improved the SfM pipeline, which had previously not worked well for RS images with significant motion (Liu et al., 2011). Later, Saurer et al. (2016) developed a full-fledged SfM pipeline in which they assumed constant intra-frame rotational and translational velocity (similar to Albl et al. (2015)) during BA and enforced temporal smoothness across frames using global positioning system (GPS) readings. Albl et al. (2016b) also show that RS cameras exhibit more degenerate configurations in SfM as compared to the traditional pinhole GS model. In such cases, BA prefers a trivial case where the scene is estimated to be planar. Ito and Okatani (2017) also discovered such degeneracies while investigating self-calibration for purely rotating RS camera when the aspect ratio of the focal length and skew parameter are unknown.

RS camera as a sensor

Instead of serving as a source of error, the following methods treat the RS effect as a source of information. I follow this spirit in my high-frequency tracking approach described in Chapters 4 and 5. Ait-Aider et al. (2006) use a RS camera to create a velocity sensor. For an object with known geometry and a stationary RS camera imaging it, they estimate the uniform translation and angular velocity of the moving object using sparse 2D-3D correspondences. They solve for the motion by modeling the distortion in the image induced by the combination of object motion and the RS effect. Magerand et al. (2012) used constrained global optimization to estimate uniform motion of a known object captured using a RS camera. Similar to the previous work, they treated the RS camera as a highly sensitive motion sensor and given the known object geometry and 2D-3D point

correspondences, they estimate the absolute pose of the camera with respect to the object. Ait-Aider et al. (2007) extend the above point-based methods to use line correspondences. These methods assume that the object geometry is known a-priori. To alleviate this, Ait-Aider and Berry (2009) study the problem of 3D point triangulation and velocity estimation of a moving 3D object with unknown geometry using a RS stereo rig. Since all the points in the image are captured at different time, they assume a constant velocity of the 3D object and solve a non-linear optimization problem to estimate 3D locations as well as object velocity by minimizing the reprojection error.

3.1.5 RS calibration

Rolling shutter exposes consecutive rows with a delay and hence the RS camera needs to undergo time-delay calibration. Geyer et al. (2005) propose a method to estimate the rate of rows exposed per second n_r and the frame delay t_f. If $n_r = \frac{H}{\text{fps}}$ then by definition $t_f = 0$. They estimate n_r by flashing light with known frequency using a pulse generator onto a RS camera without a lens. They process the captured pattern of light and estimate n_r using the Fourier analysis. Oth et al. (2013) use just a video and forgo the use of special hardware but achieve higher calibration accuracy. They use a weak motion prior with a continuous time trajectory model to estimate the line-delay t_r. They parameterize the pose using fourth order B-spline and iteratively update their model parameters while minimizing reprojection errors. I use this method for RS calibration.

3.2 Tracking

High-frequency throughput has long been a main research thrust in the three primary technologies, namely tracking, rendering, and display, of AR/VR (Azuma, 1997; Zhou et al., 2008). In this section, I will provide an overview of recent research efforts that have focused on addressing the tracking problem. On a high level, methods addressing tracking can be divided into three categories: 1. sensor-based, 2. a hybrid of sensor- and visual-based, 3. and purely visual-based.

3.2.1 Sensor-based tracking

High-frequency sensor-based tracking systems have been deployed with some degree of success for many decades; see Rolland et al. (2001) for a good review of early methods on this front. More recently, LaValle et al. (2014) have proposed to use only inertial sensors – such as accelerometers, magnetometers, and gyroscopes – combined with a predictive system for head pose tracking by assuming constant angular velocity or constant acceleration. Intriguingly, they also report limited success in positional tracking, in addition to rotational tracking, using only the inertial sensors. However, the inherent limitation of drift accumulation and the explosion due to integration of velocities to calculate the orientation renders the inertial sensors ineffective for full 6-DoF tracking. As such, most recent work has focused on using sensors and vision systems jointly to perform tracking.

3.2.2 Hybrid tracking

One approach toward hybrid sensor/vision tracking is to use inertial sensors as the primary tracking component and augment them with vision systems to mitigate drift (Leutenegger et al., 2015). Klein and Drummond (2004) used predictions from IMU sensors to account for motion blur and built a parametric edge detector on video input to prevent drift. Persa (2006) used IMUs and Kalman filtering for tracking, and to correct for drift, the author used GPS in outdoor environments and fiducial markers in indoor scenes. Mourikis and Roumeliotis (2007) use the EKF technique to incorporate geometric constraints for a single keypoint across multiple observations (images) whenever a new image is available, and in the meantime rely on the IMU for tracking. Li et al. (2013b) improve upon this EKF-based technique by modeling a RS camera by assuming a uniform translational and angular velocities. All of these systems use the IMU as the workhorse, and camera-based computer vision technique is the secondary tool, trying to augment and correct the high-frequency IMU readings.

3.2.3 Visual-based tracking

Visual trackers, which consider cameras as primary sensors, can be broadly classified into either feature-based or direct approaches. Whereas the former analyze a set of salient image keypoints, the latter rely on global image registrations. The following discusses research for tracking in general, as well as high-frequency tracking in particular for feature and direct methods.

3.2.3.1 Feature-based systems

The early successful trackers like the KLT tracker (Tomasi and Detection, 1991) and Shi-Tomasi tracker (Shi et al., 1993) established criteria for quality of 'ideal' feature points which results in robust tracking under a brightness constancy assumption among consecutive images. With the innovations of more descriptive features like SIFT (Lowe et al., 1999) and SURF (Bay et al., 2006), and efficient descriptors like ORB (Rublee et al., 2011), FAST (Rosten and Drummond, 2006), and BRISK (Leutenegger et al., 2011), tracking with sparse features in real-time became viable. Many feature-based tracking systems are subsystems of the larger problem of simultaneous localization and mapping (SLAM) and here, I will highlight just the tracking subsystem. A typical procedure used by the tracking subsystem in a SLAM framework is as follows: For each new frame, 1. extract features or keypoints, 2. associate the keypoints (or a subset of) with the keypoints from the previous frame (or keyframe), and 3. estimate the camera motion by minimizing reprojection error. The extracted features and *keyframes* are arranged in a bipartite graph, where the edges indicate that a feature projects into a keyframe. Klein and Murray (2007) developed such a system called parallel tracking and mapping (PTAM) which used separate threads for tracking and mapping to solve the SLAM problem. For each new frame they predicted a pose prior using a motion model. Using the prediction, they track the features using a coarse-to-fine approach by first estimating a rough pose and then refining it using high-resolution features in an image pyramid scheme. Ventura et al. (2014) used a client-server model, where the server has a 3D model of the environment constructed offline. A mobile phone acts as the client and runs a SLAM system in local frame of reference, using the server for global registration and bundle adjustment. Forster et al. (2014) demonstrated

a ‘semi-direct’ approach running at 300 fps on a consumer laptop. This method uses photometric error between projections of 3D points in consecutive frames for motion estimation and employs FAST (Rosten et al., 2010) features for the mapping stage.

Other feature-based hybrid tracking systems use strategically placed fiducial markers for tracking distinct points in the environment (Zhang et al., 2002; Naimark and Foxlin, 2005). Typically, light-emitting diodes (LEDs), beacons, or unique texture markers are used in this framework. The Hi-Ball system (Welch et al., 2001) is one such construction, in which blinking LEDs are placed on the ceiling of a capture environment. A cluster of infrared cameras (the “Hi-Ball”) observes the blinking pattern of LEDs, and the system uses strong triangulation constraints combined with motion prediction to provide highly accurate 6-DoF pose. The major drawback of using such fiducial markers is that it becomes cumbersome as well as costly to place them throughout the environment, and, moreover, the tracking system is limited to the confines of the area in which the markers have been installed. I use the Hi-Ball system to obtain ground truth for the real-world experiments and to validate the accuracy of my approach in Chapters 4 and 5. Concurrent to this thesis, Blate et al. (2019) developed a low-latency tracker using a stereo-pair of duo-lateral photodiodes and custom hardware. The HMD is equipped with IR LEDs which are observed by the externally stationed stereo-pair. The duo-lateral photodiodes provide independent measurements in X and Y direction in the imaging plane, providing accurate positions of three LEDs, which enables pose estimatiomn at 50kHz and 28μs.

Feature-based methods fundamentally limit themselves in utilizing image information which conforms to the definition of the feature. In contrast, dense methods utilize the complete image for estimating the relative pose, but this comes at a much larger computational complexity, as described next.

3.2.3.2 Direct/dense tracking

Direct SLAM approaches have also succeeded for both visual tracking and scene reconstruction in recent years. Newcombe et al. (2011) used dense image registration to a 3D model to localize the

camera, and used the registered image to update the 3D model in a loop. The LSD–SLAM algorithm of Engel et al. (2014a) and its derivatives (Caruso et al., 2015; Engel et al., 2015) use direct, semi-dense image alignment to reconstruct 3D models at nearly the rate of frame input. For tracking, they minimize dense photometric error between the keyframe and the current frame. To achieve this, they keep a rolling buffer of depth per keyframe which they refine using the adjacent frames. In addition to the depth, they use error propagation to keep track of variance in depth estimates, which they utilize to normalize the dense photometric error. This helps in giving importance to highly accurate 3D points over others. Schöps et al. (2014) further introduced a direct approach on a mobile phone using semi-dense depth maps for mesh-based geometry representation. Their method is hybrid in that they find the ground plane using data from the built-in accelerometer on the mobile device.

Overall, direct SLAM methods have been demonstrated to work on a large scale with relatively low computational requirements, and these systems have been leveraged for pose estimation in AR systems. However, the key limiting factor preventing these visual tracking methods from general AR/VR use is they require full camera video frame inputs, which is a substantial bottleneck for overall system latency. Effort by Dahmouche et al. (2008) towards this end was to predict and capture only the region of interest (RoI) around as opposed to capturing complete image and then finding the edges/features. They predicted the RoIs based on previous data under a constant velocity assumption to simultaneously track pose and velocity at 333Hz using a high speed camera.

Recent direct methods incorporate RS camera model in their motion estimatiion framework. Kim et al. (2016) introduced a monocular SLAM system specifically for RS cameras by adapting LSD–SLAM Engel et al. (2014b). They parametrized the intra-frame motion via a k-control point B-spline. In contrast to our method, they assumed no radial distortion and estimate single pose for the entire frame. More recently, they incorporated radial distortion in (Kim et al., 2017) by using generalized epipolar curves instead of epipolar lines. Unlike SLAM systems, my camera tracking system does not build an environmental map. Instead it relies on scan-line stereo depth variations across time to estimate 6 DoF motion, see Chapter 4 for details.

In Chapters 4 and 5, I will introduce a tracking system (without mapping) which essentially turns the rolling shutter camera into a *computational sensor*. I demonstrate the effectiveness of RS as a high-frequency measurement device. This is possible because I explicitly estimate intra-frame motion. Furthermore, I show that treating the radial distortion as a virtue also benefits the tracker, and hence methods like (Kim et al., 2016; Kerl et al., 2015; Patron-Perez et al., 2015) which parametrize the intra-frame motion using splines can benefit from my tracking method.

3.3 Edge-aware optimization

I will review the prior work related to edge-based filtering and optimization, namely: 1. bilateral filters and their variants, 2. optimizations leveraging superpixels, 3. machine learning for edge-aware filtering, 4. the domain transform and its filtering applications, 5. and bilateral solvers. In the following I will refer an algorithm as a *filtering technique* when a filter is applied on the input image to produce an output image. On the other hand, I will refer to an algorithm as a *solver* when it uses one or more input images and optimizes towards a goal defined by a cost or a loss function.

3.3.1 Bilateral filters

The bilateral filter was introduced by Tomasi and Manduchi Tomasi and Manduchi (1998) and is one of the initial edge-aware blurring techniques. The major bottleneck for bilateral filtering is that it is costly to compute, especially for large blur windows. For N pixels in an image, filter radius r in each dimension, and d dimensions, its complexity is $\mathcal{O}(Nr^d)$ (Pham and Van Vliet, 2005). Since its invention, there have been multiple approaches proposed to speed up the bilateral filter (Weiss, 2006; Adams et al., 2010; Chen et al., 2007; Yang et al., 2015; Elad, 2002): Durand and Dorsey (2002) approximate the bilateral filter by a piece-wise linear function. Pham and Van Vliet (2005) proposed to use two 1-D bilateral kernels, reducing the complexity to $\mathcal{O}(Nrd)$. Paris and Durand (2006) treat the image as a 5-D function of color and pixel space and then apply 1-D blur kernels in this high-dimensional space, leading to complexity of $\mathcal{O}(N + N/r^2 \times \prod_{i=3}^{d}(D_i/r))$, where the N/r^2 term is due to the 2-D pixel space and the remainder is the contribution of dimen-

sions $i = 3 \ldots d$, each with size D_i, *e.g.* $D = 255$ for an 8-bit grayscale image (Paris et al., 2009). Note that this is still exponential in the dimensionality (Adams et al., 2010). These approximations to the bilateral filter decouple the 2-D adaptive bilateral kernel into a 1-D kernel, reducing the computational cost significantly.

When used as a post-processing step, the bilateral filter removes noise in homogeneous regions but is sensitive to artifacts such as salt and pepper noise (Zhang and Gunturk, 2008). The method introduced in Chapter 6 emphasizes edge-aware concepts in the same spirit as bilateral filters, and the formulation yields a generalized optimization framework that is efficient and accurate. When using robust cost functions, my method remains resistant to outliers.

3.3.2 Superpixels

To combat issues of computational complexity during bilateral optimizations, several approaches leverage superpixels, *i.e.*, they group pixels together based on appearance and location. These approaches treat the superpixels as atomic unit which are far fewer in number in contrast to pixels, reducing the processing time. Superpixel extraction algorithms like simple linear iterative clustering (SLIC) (Achanta et al., 2012) are often used in optimization problems for two major reasons: 1) They reduce the number of variables in the optimization (Cadena and Košecká, 2014), and 2) they adhere to color and (implicitly) object boundaries. In one application, Bódis-Szomorú et al. (2015) use sparse SfM data, image gradients, and superpixels for surface reconstruction to ensure that the edges of triangles are aligned to the edges in the image. Lu et al. (2013) use SLIC superpixels to enforce spatially consistent depths in a PatchMatch-based matching framework (Barnes et al., 2009) to estimate stereo. As superpixels may cover arbitrarily large regions with widely varying depths, unfortunately, using them leads to a loss in resolution, as an entire superpixel is assigned one estimate; *e.g.*, in stereo, this leads to fronto-parallel depths. Additionally, using superpixels inherently assumes perfect segmentation, and these assumptions often do not hold in practice.

3.3.3 Machine learning for edge-awareness

Porikli (2008) achieve independence to kernel sizes, but is inaccurate for low color variances. To remedy this, Yang et al. (2010) use support vector machines (SVMs) to mimic a bilateral filter by using the exponential of spatial and color distances as feature vectors to represent each pixel. Their approach supports varying intra-image color variance while remaining efficient. Traditionally, conditional random fields (CRFs) are used for enforcing pair-wise pixel smoothness via the Potts potential. Alternatively, Krähenbühl and Koltun (2011) proposed to use the permutohedral lattice data structure (Adams et al., 2010), which is typically used in fast bilateral implementations, to accelerate inference in a fully connected CRF by using Gaussian distances in space and color.

With the recent explosion of compute capacity and convolutional models in the vision community, there are also deep-learning methods that attempt to achieve edge-aware filtering. Chen et al. (2016c) presented DeepLab to perform semantic segmentation; there, they use the fully connected CRF from Krähenbühl and Koltun (2011) on top of their CNN to improve the localization of object boundaries. Xu et al. (2015) learn edge-aware operators from the data to mimic various traditional handcrafted filters like the bilateral, weighted median, and weighted least squares filters (Farbman et al., 2008). More recently, deep learning methods have been introduced to learn optimal bilateral weights (Gharbi et al., 2017; Liu et al., 2017a; Wu et al., 2018) instead of using the traditional Gaussian color weights. However, machine learning approaches require large amounts of training data specific to a task, plus significant compute power, while my approach works without any task-dependent training and runs efficiently on a single graphics processing unit (GPU), see Chapter 6 for details.

3.3.4 The domain transform

Gastal and Oliveira (2011) introduced the domain transform, a novel and efficient method for edge-aware filtering that is akin to bilateral filters. The domain transform is defined as a 1-D isometric transformation of a multi-valued 1-D function such that the distances in the range and domain are preserved. When applied to a 1-D image with multiple color channels, the transformation maps

the distances in color and pixel space into a 1-D distance in the transformed space. When the *scalar* distance is measured in the transformed space, it is equivalent to measuring the *vector* distance in [R, G, B, X] space. This has the benefit of dimensionality reduction, leading to a fast edge-aware filtering technique which respects edges in color while blurring similar pixels. To apply the domain transform to a 2-D image, the authors apply two passes, once in the X direction and once in Y. This has a complexity of $\mathcal{O}(Nd)$, and as a result scales well with dimensionality. Chen et al. (2016b) proposed to perform edge-aware semantic segmentation using deep learning and use a domain transform filter in their end-to-end training of their deep-learning framework. They also alter the definition of what is considered as 'edge' by learning an edge prediction network, and they then use the learned edge-map in the domain transform. Their application of the domain transform is in the form of a *filter*.

Applying the domain transform to an image results in a filtering effect, in contrast to my method which optimizes according to an objective function and hence is a solver. I use the domain transform in my method in an iterative fashion within the optimization framework because it provides an efficient way to compute the local edge-aware mean. The application of the domain transform as a filter similar to Chen et al. (2016b) produces less accurate results than my framework. See Chapter 6 for more details.

3.3.5 Bilateral solvers

Recently, Barron et al. (2015) suggested to view a color image as a function of the 5-D space [Y,U,V,X,Y], which they call the 'bilateral space', to estimate stereo for rendering defocus blur in a mobile phone. They proposed to transform the stereo optimization problem by expressing the problem variables in the bilateral space and then optimizing in this new space. I will refer to this method as BL-Stereo (BLS). Barron and Poole (2016)'s fast bilateral solver (FBS), on the other hand, solves a linear optimization problem in the bilateral space, which is different from BLS. In this setting, they require a target map to enhance, as well as a confidence map for the target. The linearization of the problem allows them to converge to the solution faster. Both of

these approaches quantize the 5-D space into a grid, where the grid size is governed by the blur kernel size. This reduces the number of optimization variables and hence the complexity, leading to low runtimes. Mazumdar et al. (2017) use a dense grayscale-space 3-D grid (instead of a 5-D color-space grid) and the Heavy-Ball algorithm to make FBS faster, but at the cost of accuracy. I compare my method with the above solvers in Chapter 6, and show that my method is more efficient than existing techniques.

CHAPTER 4: ROLLING SHUTTER AND TRACKING

4.1 Introduction

The fields of augmented reality/virtual reality (AR/VR) tackle the hard problem of fooling the human senses. End-to-end system latency has long been the fundamental challenge in AR/VR applications. Any AR/VR system attempts to continuously track the user's motion, render the virtual content/objects according to the view-point, and then finally display the content on a screen, so that the user believes that the virtual content is part of a realistic scene. To maintain a reliable registration of the virtual world with the real world, AR/VR applications impose stringent requirements on the accuracy and speed of tracking, rendering and display (Lincoln et al., 2016).

Tracking solutions in VR devices, too, are still underdeveloped, and to a large extent there exists no high-quality, low-cost VR tracking system that works in unrestricted environments. With the increasing ubiquity of inexpensive VR displays, and with the continual development of AR displays, obtaining accurate, low-latency 6-DoF tracking is one of the most crucial immediate problems for the AR and VR communities to solve.

In commercially available VR HMDs like the Oculus Rift DK2 and Samsung GearVR, IMUs are used to estimate the relative 3-DoF rotation of the device. IMU equipped with gyroscopes and accelerometers provide fairly reliable rotational measurements with nearly 1000 Hz tracking frequency. In addition, an external camera might be used to get low frequency positional tracking, as in the Oculus Rift. The recent research has sought to provide full 6-DoF tracking missing from inertial sensors by relying on computer vision techniques (visual tracking) that allow for either *inward* or *outward* tracking of the device in relation to the surrounding environment. For inward tracking, one or more "inward-looking cameras" (*i.e.* cameras situated in the environment and pointed at the user) track distinct markers on the HMD. Such markers are typically light sources, *e.g.* infra-

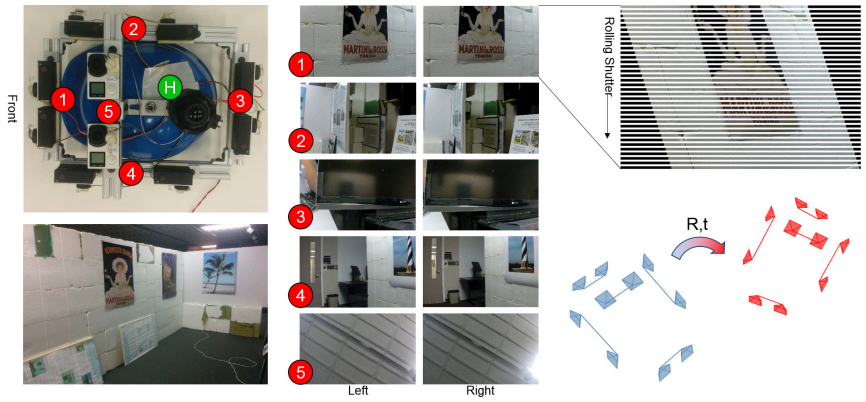


Figure 4.1: *Top Left*: My prototype cluster built by mounting 10 Go-Pro cameras on a helmet. (1) through (5) denote stereo pairs pointing forward, right, backward, left, and up, relative to the head of the user. (H) marks a Hi-Ball tracking camera, which was used to obtain ground-truth 6-DoF motion for the device. *Bottom*: View of the room where the real-world experiments were captured. *Middle*: Example stereo pair images for one timepoint of capture from the real-world data. Stereo pair (5) captures a view of the ceiling. *Top right*: Illustration of the rolling shutter effect for a single image. Rows of the image are captured sequentially at high frequency. Motion of the camera during this capture induces distortion in the x-direction of the image plane. *Bottom right*: Tracking the pixel shift from row to row in multiple cameras allows the tracker to estimate 6-DoF device motion from one time point (blue) to the next (red) at a much higher frequency than the camera frame rate.

red (IR) light-emitting diodes (LEDs) on the Oculus Rift, or large ping-pong-size balls, as on the Sony Move. In an inverse configuration, outward tracking techniques use one or more cameras on the AR/VR device to detect markers placed in the surrounding environment. The main drawback to these tracking approaches is they are typically high-latency, high-cost, or both. Moreover, the requirement of cameras or markers placed in the external environment limits the generality of these systems – that is, 6-DoF tracking of the AR/VR device is only possible within the small area containing the cameras/markers. Newer systems based on simultaneous localization and mapping (SLAM), refer Section 3.2 are being developed to perform markerless tracking to mitigate this.

Two basic challenges for high-frequency visual tracking are: 1) the basic assumption of a single pose for each captured image, and 2) the requirement of multiple frames for reliable pose estimation. Accordingly, multi-frame approaches inject high latency and over-smoothing of the estimated 6-DoF motion. Moreover, the required tracking frequency for AR/VR is commonly much higher than the frame rate of the camera. For instance, the human neck can support rotational speeds of up

to 700 deg/s at peak rates and up to 70 deg/s at normal rates (Bishop and Fuchs, 1984), and normal human walking speed is 1.39 m/s (5 km/hr). Extremely fast tracking is essential to precisely match the motion of a user's head.

In this chapter, I describe a new markerless approach for outward visual tracking of AR/VR devices that achieves high accuracy and high-frequency without requiring any off-device tracking elements. The key insight is that extremely high-frequency image sampling is attainable using rolling shutter cameras. I treat a rolling shutter camera as a high frequency 1D sensor capturing H 1D row-images sequentially in time, rather than a 2D sensor capturing a single H$\times$W 2D frame at each time-point. This enables the proposed tracker to process the video frame per-row rather than waiting for the complete image formation. Using a rig of multiple rolling-shutter stereo pairs, the tracker estimates small-motion poses at very high sampling rates *entirely* from cameras located on the HMD, see Fig. 4.1. This approach can obtain tracking rates as high as 57 kHz on synthetic data simulating actual user head motion. I also provide a proof-of-concept real data experiment using a prototype model of the rolling shutter tracking system with 5 rolling-shutter stereo pairs capturing at 120 frame rate. Using this cluster, the system is able to track at frequencies as large as 80.4kHz on real data. Fig. 4.2 shows an overview of our approach.

The rest of this chapter is laid out as follows: In Section 4.2, I motivate why RS cameras are benefical for high-frequency tracking, and how they unlock immense potential for high-framerate pose estimates solely based on cameras. In Section 4.3, I will describe the approach to row-wise rolling-shutter tracking, and I will detail the camera cluster set-up in Section 4.4. In Section 4.5, I provide thorough experimental evaluation of the method on both simulated and real-world data. Finally, I will conclude in Section 4.6.

4.2 Tracking with multiple rolling shutter cameras

Current state-of-the-art visual tracking systems, such as the Oculus Rift, perform rotational and positional tracking of the AR/VR device using an inward-facing camera focused on the user. Because this approach involves tracking the user based on fiducial points in the 2D image plane, the

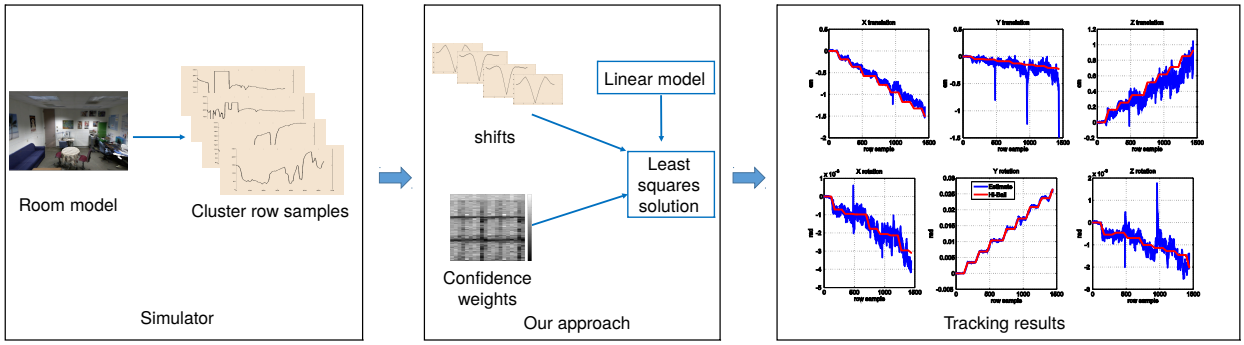


Figure 4.2: Overview of the tracking system for simulated experiments: (a) Simulation provides image data for multiple virtual cameras. (b) The tracker algorithm estimates shifts and confidence scores for pose estimation. (c) Comparison of the tracking results with ground-truth Hi-Ball tracker motion data.

system has a strict lower bound on latency that is determined by the frame rate of the camera (*e.g.* 60 Hz). Moreover, the capture set-up inherently confines the tracking area to the field of view of the camera. To solve both of these problems, I propose to instead perform visual tracking using an *outward-view*, *rolling-shutter* approach, where multiple rolling-shutter camera stereo pairs located on the device itself are used to recover the motion of the user, without any specific design for the surrounding environment.

I propose to exploit the rolling shutter effect to our advantage to enable high-frequency tracking. I leverage the fact that each row of a rolling shutter image is shifted in time by a very small offset. I propose to treat a rolling shutter camera as a high-frequency line sensor where each of the rows constitutes an individual sample. With this definition, even an inexpensive smartphone camera with 1000 rows and 30 fps has an equivalent 1D sampling frequency of 30 kHz. The natural barrier of camera frame rate, which is the limiting factor for visual tracking approaches using 2D images, is surpassed by several orders of magnitude using this approach. The trade-off is that this increased sampling comes with a substantial reduction in the vertical field of view.

4.3 The Approach

In the following, I describe how rolling shutter can be leveraged for high-frequency tracking. I will then introduce the linear model used for rolling-shutter-based tracking, explain how to search for correspondences between captured rows, and address how to estimate rotational motion via homographic mappings between video frames. At the end of this section, I describe corrective measures and confidence scores that help to maintain the stable tracking.

4.3.1 Background

Bishop and Fuchs (1984) introduced an inside-out tracker using a cluster of custom-made 1D sensors capturing line-images at 1000 fps. The proposed tracker utilized a computationally efficient linear motion estimation framework under a small-motion assumption. Assuming small motion results in a simplified model, which reduces the computation time. In turn, the lower computational burden allows higher sampling frequencies, reinforcing the original small motion assumption. This 'virtuous' circle, see Fig. 4.3, is one of the key enablers of the approach.

In this Chapter, I generalize Bishop's model to use rolling shutter cameras, with each row-image treated as an individual 1D sensor. This provides an array of line sensors per RS camera, each sampling at the frame rate of the camera (*e.g.* 120 Hz). Because the row sampling occurs sequentially, the effective sampling rate of these line sensors is in the kHz range for a typical high-definition (HD) frame sensor. Moreover, these line sensors are in a fixed configuration with respect to each other, meaning change in relative pose between camera rows is entirely governed by the motion of the entire device.

To achieve this high-rate capture, it is necessary to model the set of line sensors in a manner similar to a single high-frequency sensor, for which the small motion assumption is trivially valid. For the single-sensor case, sequential captures are directly comparable because the same physical sensor array is used for both timepoints. For finding such correspondences in RS imagery, the key challenge is to hypothesize the current row-image n but with a time of capture before the

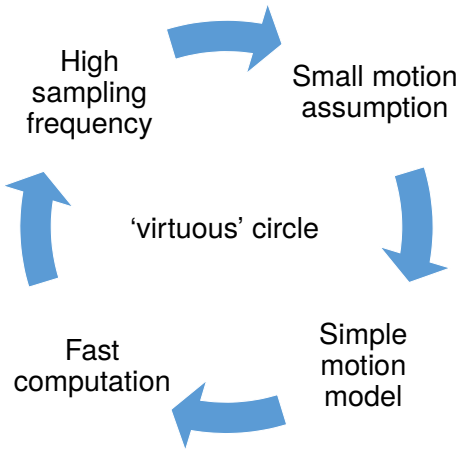


Figure 4.3: *Virtuous Circle.* Each step reinforces the subsequent step.

current time.To this end, I use an inter-frame homography to predict a line n in the current frame by transforming lines from the previous frame. I follow Bishop and Fuchs (1984) and measure shifts between the current row-image and the recreated row-image to quantify the small motion. The shift between these rows separated in time is denoted as s_t.

Given that Bishop's model requires estimation of depth, cameras are arranged in stereo pairs in the cluster. A row-wise depth value, s_d, is then estimated by measuring the shift within each stereo pair. A rigid cluster configuration is assumeed with known intrinsic and extrinsic camera calibrations, which can be achieved in practice using calibration methods such as (Kumar et al., 2008; Li et al., 2013a). Due to the relatively small baseline of the stereo pairs, which leads to poor depth estimates in large, open spaces, I also assume a static indoor scene for the capture environment. In the simulation experiments, I use 10 stereo-pairs of RS cameras looking in all directions, and in the real-world data experiments, I use 5 stereo pairs.

4.3.2 Linear Model

Next, I introduce the linear model used in this work. Let $\boldsymbol{V}_{cam} \in SE(3)$ represent the fixed relative pose (cluster-to-camera) of a given camera relative to the coordinate frame of the cluster center. Consider a 3D point X^t_{cam} whose coordinates are expressed relative to the reference frame of camera cam at time t. If $\boldsymbol{M}^t$ represents the change in camera cluster pose between time-steps t

and $t+1$, we can express the change in a 3D point's position relative to a given camera as

$$X_{cam}^{t+1} = \boldsymbol{V}_{cam}\boldsymbol{M}^{t}\boldsymbol{V}_{cam}^{-1}X_{cam}^{t}, \tag{4.1}$$

where X_{cam}^{t+1} is the new 3D position of X_{cam}^{t} relative to the given camera at time $t+1$.

The camera model is a 1D rolling-shutter sensor row, with a sampling rate equal to the frame rate of the imaging device. Let $\hat{X}_{cam}^{t}$ denote the 3D point that projects onto the center pixel of this row. For a given time point, we can apply Eq. (4.1) to $\hat{X}_{cam}^{t}$ for each camera. This gives us one independent linear equation for each camera.

Now, the shift from time-point t to $t+1$ can be expressed in terms of stereo and temporal disparities. As mentioned previously, let s_d represent the distance, in pixels, between $\hat{X}_{cam}^{t}$ and its corresponding point in the other stereo camera at time t. Let s_t represent the detected pixel shift between the $\hat{X}_{cam}^{t}$ and $\hat{X}_{cam}^{t+1}$ when the points are projected onto the 1D image plane of the camera. (The method for obtaining this correspondence is explained in the next section.) Then, the homogeneous points $\hat{X}_{cam}^{t}$ and $\hat{X}_{cam}^{t+1}$ can be expressed in terms of the measured shifts s_t and s_d as follows:

$$\hat{X}_{cam}^{t} = \begin{bmatrix} 0 & f_r(r) & -f_d(s_d) & 1 \end{bmatrix}^T \tag{4.2}$$

$$\hat{X}_{cam}^{t+1} = \begin{bmatrix} f_s(s_t) & \boxtimes & \boxtimes & 1 \end{bmatrix}^T, \tag{4.3}$$

where the $\boxtimes$ denote unknown values that require future row-images for estimation. As the $\boxtimes$ values are only determined by future observations, I do not make use of them in the motion estimation. Here, f_d converts the disparity s_d into camera-space distance according to the focal lengths and baseline b of the stereo pair. Similarly, f_r and f_s convert the row index r and the shift due to motion s_t into camera space.

Motion model

Any general motion $\boldsymbol{M}$, can be expressed in terms of translation and rotation in each direction. At high sampling frequencies, the small motion assumption allows a differential rotational model, that is, as a small perturbation from identity. The differential motion $d\boldsymbol{M}$ takes the following form:

$$d\boldsymbol{M} = \begin{bmatrix} 1 & \theta_z & -\theta_y & -t_x \\ -\theta_z & 1 & \theta_x & -t_y \\ \theta_y & -\theta_x & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.4}$$

Where the values θ_* are the (differential) Euler angles while t_* are small translations in each direction. On combining Eq. (4.1) with Eqs. (4.2) and (4.3), then expanding to express in terms of the six unknowns, an over-determined weighted linear system can be formed, which is expressed as follows:

$$\boldsymbol{CA}Y = \boldsymbol{C}B. \tag{4.5}$$

Vector $Y = [\theta_x\ \theta_y\ \theta_z\ t_x\ t_y\ t_z]^T$ represents the unknowns, and $\boldsymbol{C}$ captures the confidence in the shifts s_t; see Sec. 4.3.5 for a description of $\boldsymbol{C}$. $\boldsymbol{A}$ is a function of the cluster configuration, row index r and shifts s_d which can be computed from previous data. Since the vector B is the only part of Eq. (4.5) that depends upon the shifts s_t obtained using current data, the matrix $\boldsymbol{A}$ can be pre-computed.

This is an incremental system which inherently drifts over time. In the system of Bishop and Fuchs (1984), the use of beacons placed strategically in the scene was suggested as a solution. In the case of 1D sensors, the necessity for this constraint arises because the sensors cannot be used for global drift correction; hence, beacons are needed to serve as fixed external references. As my system uses rolling shutter cameras, it does not have this limitation, and I can use the 2-D images and the per-row poses as an input for feature-based systems, which can run at a lower frequency

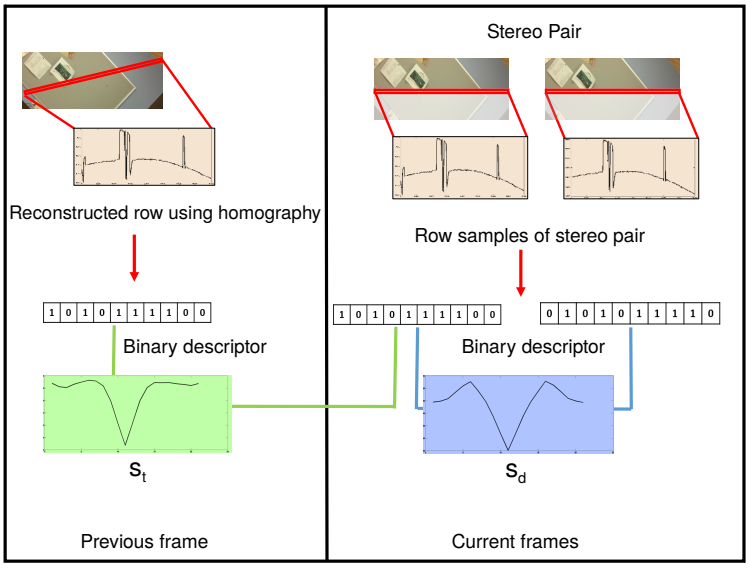


Figure 4.4: Measuring s_t and s_d using rotation compensation (best viewed in color).

for global drift correction. I leave it as a future work to employ a SLAM-type system which would run in parallel to my high-frequency tracking approach.

4.3.3 Shift Estimation

Row Descriptor

So far, I have assumed that I am able to accurately measure the shifts s_t and s_d. Now, I describe the method used to estimate the shifts described previously. Bishop and Fuchs (1984) highlighted the importance of a binary descriptor for rows, but the approach was not robust to camera noise patterns. I adapt Bishop's representation and convolve each row sample with a double derivative of the Gaussian to obtain a smoothed version of the edge response (see Fig. 4.5 as an example). The smoothing provides robustness against noise, and the double derivative helps in localizing the shifts to sub-pixel accuracy. After this convolution, the resulting values are thresholded at zero to provide a binary descriptor for each row. This binary descriptor is used to measure both the shifts s_t and s_d. Fig. 4.4 shows the measurement of s_d by using corresponding rows in a stero-pair. The shift s_t is measured by comparing the binray descriptors across different frames. The following describes this matching in more detail.

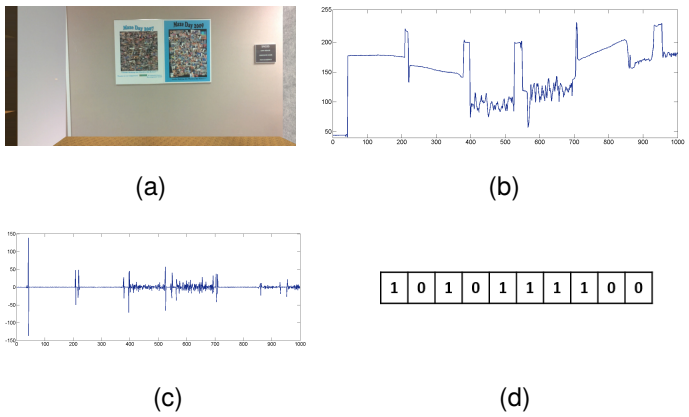


Figure 4.5: (a) Original rolling shutter image. (b) Visualization of pixel intensity values for the 200^{th} row of the image. (c) Double derivative of the smoothed row. (d) Representative binary descriptor.

Descriptor Matching

The binary descriptor is used for matching the rows to estimate shifts s_t and s_d. The Hamming distance is used for fast computation of the matching cost. The evaluated pixel shifts s_t are limited to the range of $[-20, 20]$ pixels. A parabola is fitted at the minima to recover a subpixel shift estimate (Scharstein and Szeliski, 2002). To estimate the depth disparity s_d, the descriptors are matched for a larger range of $[10, 70]$ pixel shifts. A lower bound is used to avoid inaccuracies which occur at large depths, and an upper bound is used because a very high disparity indicates that the camera is too close to a surface and hence the stereo-pair cameras might not observe sufficient structure to produce an accurate shift.

4.3.4 Rotation Compensation

To estimate the motion-induced shift of a single camera across time, the pixel content of the currently sensed row-image is compared against the corresponding content observed on the previous frame. In practice, the motion between successive frames is dominated by the rotational component. Hence, in order to justify the use of the small motion assumption, the rotation is compensated through an homographic warping between consecutive frames. A block of rows is extracted from the previous frame and transformed to match the rotation of reference time j occurring some time

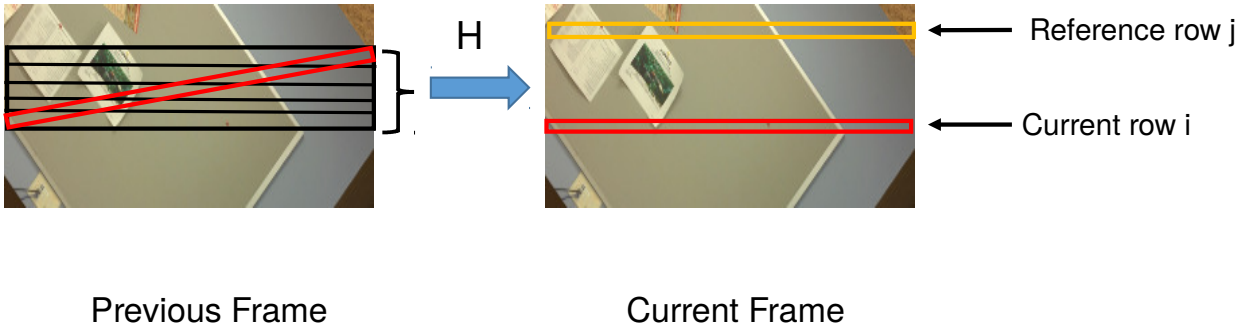


Figure 4.6: Rotation compensation: A block of rows from the previous frame are transformed using homography H to predict how the current row i will appear under the rotation of reference row j.

before the current row i (see Fig. 4.6). This rotation compensation is performed using an adaptively specified reference time j so that the rotation between row i and the rotation of row exposed at time time j does not fall in the magnitude of noise and hence can be estimated reliably. Let us denote the rows which are to be transformed using a homography as l_q, where q is the row index in the image. If the current frame index is k, row-image i is in k^{th} frame, while row l_q is from the $(k-1)^{th}$ frame. Let the rotation at row l_q be R_q and the rotation of the row exposed at time j be R_j. The adaptive reference scheme (see Sec. 4.3.4.1) decides whether time j corresponds to a row in frame k or in frame $k-1$. The homography between time j and row l_q, given the camera intrinsic matrix K, can be computed as

$$H_{j,q} = K R_j R_q^T K^{-1}. \tag{4.6}$$

R_j and R_q are known quantities which were previously estimated, hence it is trivial to compute $H_{j,q}$. Elimination of the dominant image motion, *i.e.* rotation, ensures that a large range of motions can be covered by the high-frequency tracking system. Fig. 4.11 shows that the proposed system can handle motion as large as 500 deg/s using this compensation technique.

4.3.4.1 Adaptive Reference

As implied above, the reference time j need not necessarily be equal to the time corresponding to row $i-1$. The reference time is adaptively changed according to the rotation velocities. The pixel shift values between two consecutive rows of same frame at peak rotational velocities (*e.g.*

500 deg/s) fall between the range of -0.5 to 0.5 pixels. For normal motion, they are in the range of 0.1-pixel shifts. Such a small shift is at the magnitude of noise and thus cannot be measured accurately. Instead, I introduce an adaptive technique that enables the tracker to work at reasonable shifts without violating the small motion assumption. To change the reference time j adaptively, a manually chosen maximum allowable rotation between rows is used, denoted as $d\theta$. Whenever this threshold is crossed, the reference time j is updated to the timestamp of a new row where the shifts would not fall below the magnitude of noise.

4.3.5 Corrective Measures and Confidence Scores

I employ the widely used disparity confidence measure peak ratio (PKR) (Hu and Mordohai, 2012) to form the diagonal matrix $\boldsymbol{C}$ to weigh each equation in the linear system according to the confidence in its shift s_t. For robust outlier rejection of shifts, a single exponential smoothing (Kalekar, 2004) is used to predict the shift s_{pred} at the current time using previous data. If the difference between measured shift s_{meas} and s_{pred} is high, the predicted shifts are used instead of the observations, but assigned a lower confidence score.

4.4 Cluster setup

Synchronization

Previously, I assumed that the cameras in the cluster are capturing in synchronization. In reality, this is difficult to achieve without a hardware genlock (Wilburn, 2004). As a proof of concept, I used GoPro cameras in the experiments which do not expose a hardware genlock. Here, I describe my approach to achieve synchronization using a few post-processing steps. Additional specific details are provided in the experiments section.

The GoPro cameras have a vertical rolling shutter. I attached one red LED per camera so that it occupies the entire vertical FoV. For synchronization, a square wave signal is fed to all the LEDs. The peaks of the mean red channel value in the frames enable frame-level synchronization for each

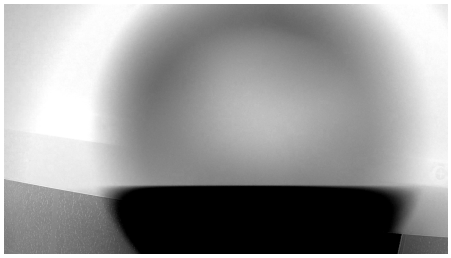


Figure 4.7: Red channel image of a blinking red LED showing the rolling shutter effect.

camera. Since the LEDs are blinking at know frequency, the rolling shutter effect becomes apparent (Fig. 4.7). As the turn-off and turn-on time of a LED is on the order of nanoseconds, the index of the row which is at the temporal edge of the drastic change in light can be precisely detected. I detect such rows over a number of blinks and regress to get accurate row indices. As the period of the square wave is known, we can match the slopes and extrapolate to any frames captured after the synchronizing signal is stopped. With this, it is precisely known across all cameras in the cluster, which rows were exposed at the same instant.

Calibration

While calibration of intrinsic camera parameters and extrinsic stereo pair parameters is nowadays a relatively simple task, it is inherently difficult to calibrate cameras with non-overlapping views. The method introduced to calibrate non-overlapping view cameras by Li et al. (2013a) proved to be insufficiently accurate for my system. Moreover, to compare the tracker performance against the ground-truth data captured from the Hi-Ball tracker (Welch et al., 2001), the cluster to the Hi-Ball calibration is also required. The following explains the estimation of the relative camera calibration w.r.t the Hi-Ball. Let $\boldsymbol{T}_{c2H}$ denote the transformation from a camera in the cluster to the Hi-Ball coordinate system. To calibrate the entire cluster, I first calibrated the stereo pairs independently using the traditional checkerboard pattern and Zhang (2000)'s method. Then, the left camera of each stereo pair is calibrated with the Hi-Ball. The transformation from the cameras to the Hi-Ball $\boldsymbol{T}_{c2H}$ follows the hand-eye calibration (HEC) problem. Shah et al. (2012) provide a good survey of solving the HEC problem, and I follow the method described by Park and Martin

(1994) that casts the HEC problem in a least-squares framework. In HEC, there are two known dynamic poses that change over time but can be estimated, and there are two unknown static poses which need to be estimated. To solve for $\boldsymbol{T}_{c2H}$, I simultaneously capture images of a checkerboard pattern, as well as the Hi-Ball tracker data, at various positions and orientations. The two known poses at each time step are the pose of left camera w.r.t. the fixed checkerboard pattern and the pose of the Hi-Ball w.r.t. to the ceiling. The two unknown static poses are the unknown $\boldsymbol{T}_{c2H}$ and the pose of the checkerboard pattern w.r.t. ceiling.

Number of camera pairs and their layout

As noted in Sec. 4.3.2, only one linear equation of the form Eqn. 4.1 is obtained for each camera. To complete the equation, we need depth, hence we need cameras in stereo-pairs. For a fully in-step genlocked camera cluster, where each camera captures a row with the same index at the same time, atleast 6 stereo-pairs are required. In the following experiments, although we know when each row was captured for each camera using the synchronization method described above, there is a stagger between the rows, *e.g.* Cam1 captured row 310, Cam2 captured row 331,…at the same instant. This row staggering provides one additional constraint for each cameras, to give 10 equations in total from 10 GoPros. I have placed the 5 stereo-pairs heuristically to minimize overlap between their fields of view. This ensures that each stereo-pair provides newer information, and reduces the chance of multiple cameras capturing homogeneous scenes.

4.5 Experimental Results

To evaluate the proposed approach, I will first present rigorous experiments on RS image data generated using a simulator, which provides complete flexibility over camera parameters, their relative positioning, size of the room in which data is captured, and the ground-truth motion.

Hi-Ball tracker

For testing, synthetic ground-truth motion is the most suitable in terms of flexibility, but it may not capture the distinctiveness of real human motion data. Therefore, I utilize human motion

captured using Hi-Ball tracker system (Welch et al., 2001). According to the authors Welch et al. (2001), this system can record over 2000 pose estimates per second with less than 1ms latency and less than 0.5mm and 0.03 degrees of absolute error. This is one of the fastest tracker available for capturing groundtruth. As I show in Section 5.4.3, the Hi-Ball system performs better than the Microsoft HoloLens tracking system. To simulate this captured human motion at more than 2000 Hz, the simulator linearly interpolate the translation and uses spherical linear interpolation (SLERP) for rotation. This is valid as long as the motion captured is casual human movement. To compare the proposed tracker at larger velocities, I rely on using synthetic data and velocities, see Fig. 4.11 for an example of extreme motion.

Rendering pixel error

I characterize the errors in the tracking system in terms of pixel errors in display of synthetic objects at a distance of 1m away from the user, as done by (Bishop and Fuchs, 1984). I consider an error of 1 pixel display error as permissible. To quantify these errors, the display resolution of 1080×1200 pixels (per eye) is used. This corresponds to the resolution of HTC Vive VR device[1]. In all experiments, I assume that the motion for the first frame is zero, and hence all rows in this frame share the same pose. As a result, the first image is essentially a global shutter image, while the rest of the frames are captured using rolling shutter. I use this assumption because the proposed system is incremental and requires a reference pose; having a known pose for each row of the initial rolling shutter frame would also be acceptable. The hyperparameter $d\theta = 0.06$ radian is used as the threshold for rotation compensation.

4.5.1 Simulations

The simulator is written in OpenGL and QT to simulate synthetic ground truth motion and capture RS images at very high frequencies. To simulate a real indoor scene, I use a room mesh obtained from system similar to Dou et al. (2013), which is referred to as the *scanned* room, see

[1] `http://www.htcvive.com/us/product/`

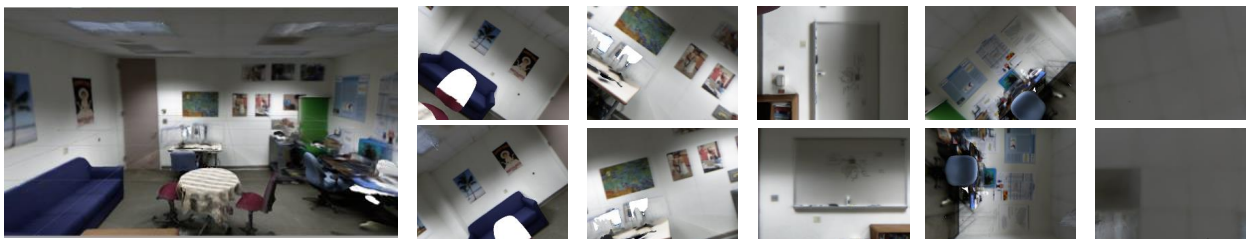


Figure 4.8: Scanned room, with small images showing the RS images captured by the left camera in each of the stereo pair from the virtual cluster.

Fig. 4.8. For all simulator experiments, I use 10 RS stereo pairs with a baseline 10cm. Each camera has 640×480 resolution, 60 degree vertical FoV, and captures at a frame rate of 120 Hz. The effective RS sampling frequency is 57600 rows/s.

Experiment 1

In this experiment, I examine the general characteristics shown by the tracker. Fig. 4.9 shows the tracking results for Hi-Ball motion tracker data simulated inside the *scanned* room, where the X axis is the row-sample index (analogous to time) and the Y-axis is the estimated variable. The red graph is the ground-truth data obtained from the Hi-Ball tracker, while the blue graph is the tracking result. Note that in (a) and (b), the scale of the Y axis for each variable is different. The plots show that the relative error increases when the variable to be estimated is within the order of noise. Fig. 4.9 (c) depicts the confidence as a heat map, where the X axis depicts the camera number and the Y axis is the row-sample index (time). The color signifies the confidence level of each camera for a particular row. The confidence scores indicate that cameras 9, 10, 19, and 20 have low confidence in measured shifts. Upon further examination, it was found that these cameras were observing a region with homogeneous color. Fig. 4.9(d) highlights the problem of drift, where an error in the estimate is carried over to the next frame and periodically increases. The red marks at the bottom signify image boundaries.

Experiment 2: Synthetic and large motion

In this experiment, synthetic motion data with added Gaussian noise is used to simulate moderate and extremely large motions. For the moderate motion, I consider a translation velocity (v)

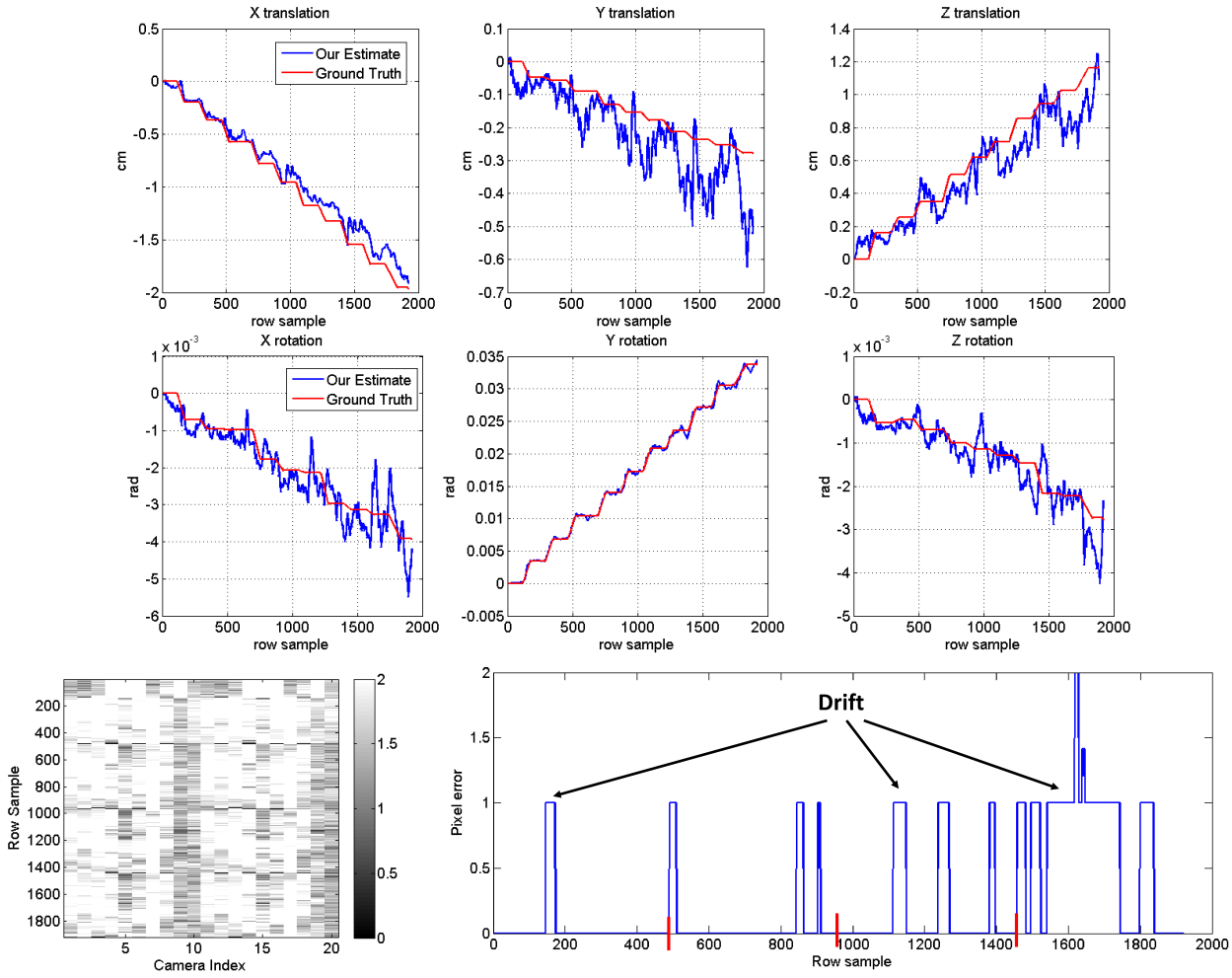


Figure 4.9: Comparison of pose estimates versus ground truth for the scanned room simulation, using Hi-Ball motion ground truth. (a) Translation. (b) Rotation. (c) Confidence score. (d) Pixel error in display. Note that y-axis scales are not the same (best viewed in color).

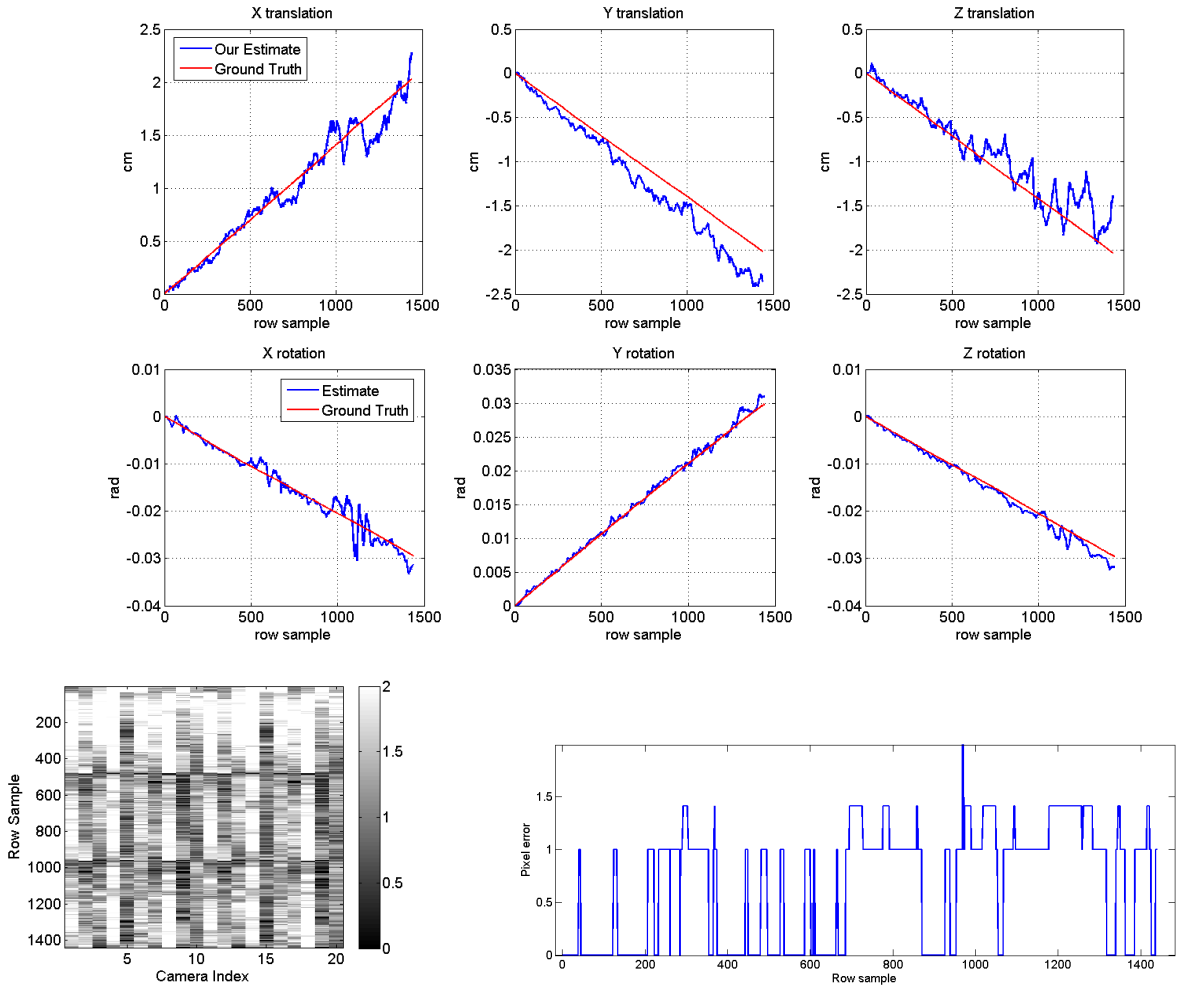


Figure 4.10: (a) Translation, (b) Rotation, (c) Confidence score and (d) Pixel error in display, Estimated for Scanned Room, synthetic large motion, note scales not same (best viewed in color).

of 1.4 m/s (which is the average human walking speed) and rotational velocity (ω) as 120 deg/s. As we can see from the tracking plots in Fig. 4.10, jumps occur at image boundaries (480, 960,…) which is also reflected in the confidence score. If all the confidence scores are less than 1, it implies that the measured shifts are highly unreliable and hence the estimated motion should be ignored. This occurs because the row reconstruction step is based on a homography, but at the boundary of the image, an insufficient number of rows are available for accurate prediction.

For extreme motion (Fig. 4.11), I simulated $v = 1.4$ m/s and $\omega = 500$ deg/s. The error plot shows that even with sustained extreme motion, the pixel error incurred hovers around the acceptable limit of 1 pixel but increases at the image boundary. The proposed rotation compensation

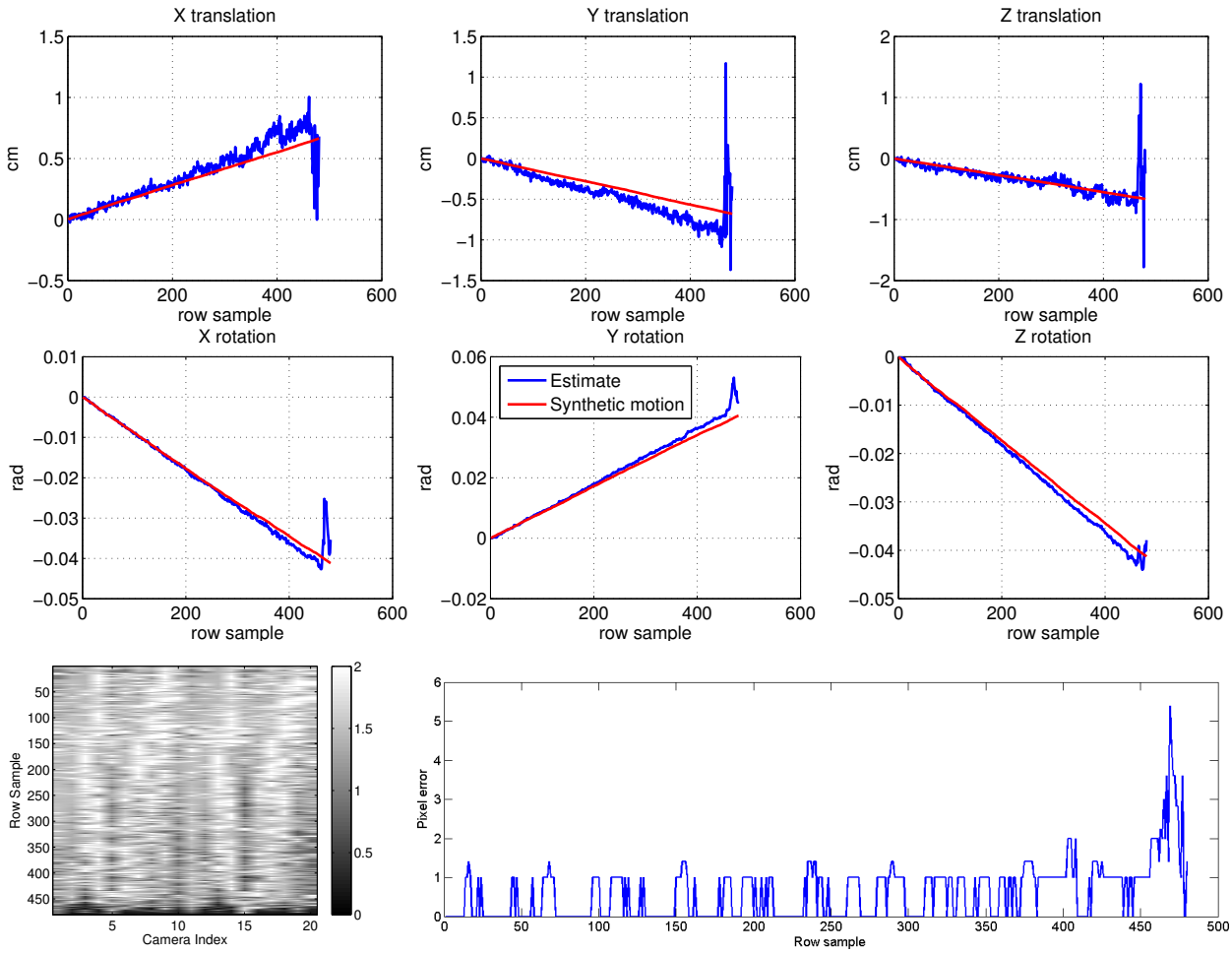


Figure 4.11: (a) Translation, (b) Rotation, (c) Confidence score and (d) Pixel error in display, estimated for Scanned Room, synthetic extreme motion of $v = 1.4$ m/s and $\omega = 500$ deg/s, note scales not same (best viewed in color)

technique from Section 4.3.4, enables the tracker to function at these extreme speeds. Note that these extreme motions cannot be tracked for a long period of time since there is limited overlap between consecutive frames.

Experiment 3: Non-linear solution

Here I show the effect of the linear approximation for motion. The fundamental assumption in the proposed approach is that between two rows of RS image, the head motion is linear. I verify if this assumption holds and check how much it differs from the non-linear solution here. I use the Levenberg–Marquardt (LM) algorithm to solve the non-linear Eqn. 4.1 with the approximate

Exp #	RMSE	T_x (cm)	T_y (cm)	T_z (cm)	θ_x (rad)	θ_y (rad)	θ_z (rad)
Experiment 3	Linear approximation	0.0841	0.079	0.11	0.000332	0.000253	0.000295
	Non-Linear solution	0.07799	0.0898	0.1089	0.000909	0.000499	0.000384
Experiment 4	Motion without blur	0.09573	0.2476	0.2002	0.001269	0.001308	0.001433
	Motion with blur	0.10535	0.2429	0.2243	0.001615	0.0016	0.0023

Table 4.1: RMSE in the estimation variables over 1500 row-samples for $v = 1.4$m/s and $\omega = 120$ deg/s with added Gaussian noise.

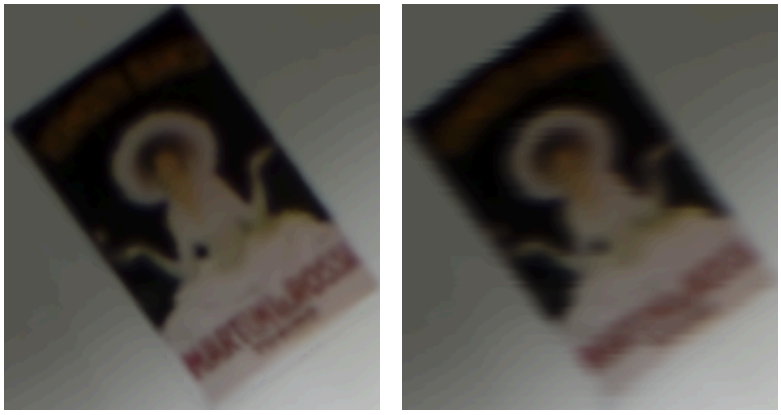


Figure 4.12: (a) RS image without motion blur, (b) RS image with motion blur, blur is amplified to test for worst case scenarios.

linear solution as a starting point for the optimization. As the Table 4.1 shows, the difference in the non-linear versus the linear approximation solution is not discernible.

Experiment 4: Motion Blur

Here, I show the effect of motion blur. In general, a high-fps GS camera has a shorter exposure and hence lower motion blur. Since I use commodity RS cameras, a long exposure time is maintained even under the fast sampling of consecutive lines. The motion blur is simulated in software and the blurred rows are used as input to the proposed system, see Fig. 4.12 for an example of RS image with motion blur. I exaggerated the motion blur so that I can test for the worst possible conditions. As highlighted by Table 4.1 the approach is robust to motion blur.

4.5.2 Experiments with real data

For real experiments, the camera rig shown in Fig. 4.1 is used with 10 Go-Pro cameras arranged in 5 stereo-pairs. The rig is moved in a room of size 2.6m$\times$6.4m$\times$2.7m with the Hi-Ball tracking

LEDs on the ceiling to record precise ground truth. The camera cluster has eight Go-Pro Hero 3+ silver edition and two Go-Pro Hero 4 black edition cameras. The cameras are set at a narrow FoV with a resolution of 1280×720 capturing at 120 fps. For synchronization, each camera is attached with one red LED so that it will occupy the entire vertical FoV of the camera. A commodity micro-controller (like Arduino or Raspberry Pi) is used to generate a square wave. A current of about 10mA is supplied to each LED so that it can saturate the camera. After the cameras record enough LED blinks, the synchronization signal is turned off and the LED is removed without stopping the video recording. This initial part of the video is used to synchronize the cameras. For the real data experiments, I process one row-image per RS camera at a time and hence work at the highest sampling frequency, which is h×fps. Given that the cameras need to be rectified, the effective height of the RS image captured by the camera is reduced to a minimum of 670 pixels. Thus, effective tracking frequency comes out to be 80.4 kHz at 120 fps. Fig. 4.13 shows the tracking results using the proposed camera cluster; note that the results are noisy. To mitigate this, an exponential smoothing can be employed for stable tracking at the cost of increased latency, see Fig. 4.14 for smoothed results. The error plots for the errors in terms of display pixel errors are not shown in both the cases because they are less than 0.5 pixels and quantize to zero error.

4.5.3 Effect of noise and scene homogeneity

Noise and homogeneity of scene adversely affects the tracking accuracy. A combination of both is seen in real data. Fig. 4.15 shows tracking results for translation in X direction for different Gaussian noise with $\sigma = 3, 5, 10$ added to the simulation images; I excluded other variables as they follow the same trend. While noise can be mitigated by better lighting conditions or using quality sensors, it is difficult to overcome scene homogeneity. I simulated scene homogeneity by blurring the simulation images with a Gaussian kernel, $\sigma = 3, 5, 10$. Fig. 4.16 shows the degradation in performance due to scene homogeneity, which is characteristically different from noise. While noise is random, scene homogeneity's effect on tracking has more structure, where blocks of rows might not provide correct measurements. At extreme noise levels or homogeneous scenes the system can-

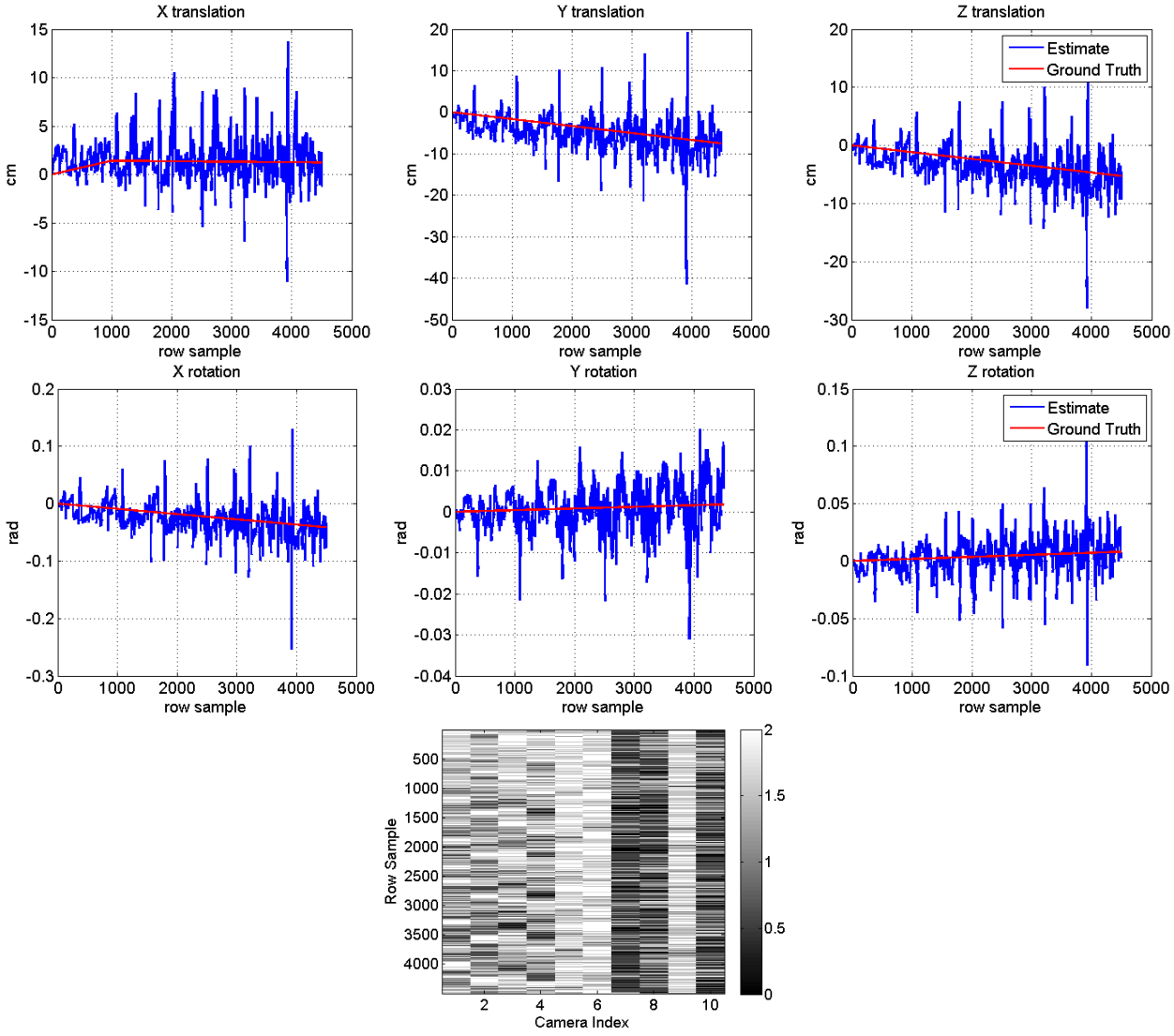


Figure 4.13: (a) Translation, (b) Rotation, (c) Confidence score. The results are noisy which when smoothed give the correct results. Note the axis scales are not same. (Best viewed in color.)

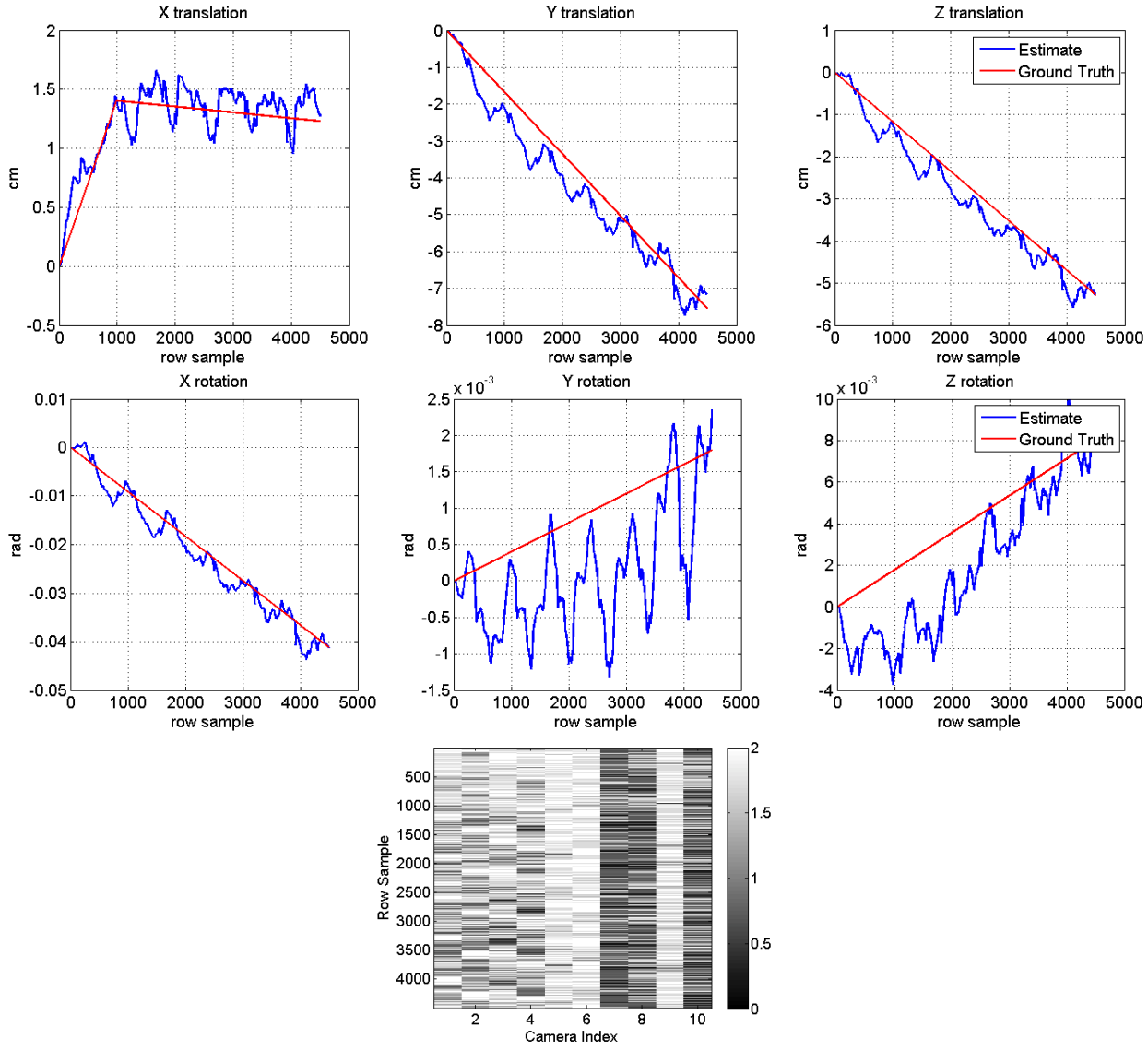


Figure 4.14: (a) Translation, (b) Rotation, (c) Confidence score. Exponential smoothing is used to lower noise. Note the axis scales are not same. (Best viewed in color.)

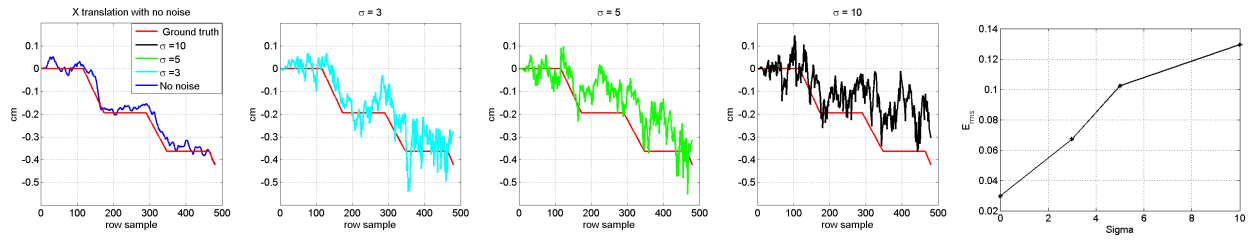


Figure 4.15: Tracking results for translation in x direction with different levels of noise, and (b) corresponding RMSE.

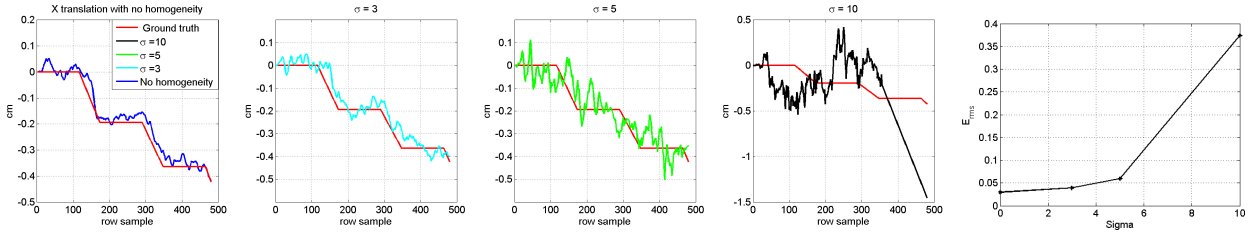


Figure 4.16: Tracking results for translation in x direction with different levels of scene homogeneity, and (b) corresponding RMSE error.

not track *any* motion. Possible future directions to address these issues will be to process blocks of rows rather than individual rows to average out noise, and to use prediction techniques (Kalekar, 2004; Welch et al., 2001) to mitigate scene homogeneity.

Preconditioning

Now, let us examine the stability issues of the linear system. Eqn. 4.5 describes a weighted linear system where $\boldsymbol{A}$ is $N \times 6$, N is the number of cameras in the cluster. On simplifying Eqn. 4.1 to construct $\boldsymbol{A}$, the three elements of each row of $\boldsymbol{A}$ are V_{11},V_{12} and V_{13} from camera pose matrix $\boldsymbol{V}$, which constitute the first row of rotation matrix of the pose. Hence, the scale of the unknown variables becomes important as V_{11}, V_{12}, and V_{13} are guaranteed to be between -1 and 1. Hence, pre-scaling is important where the translation should be in meters and the rotations should be measured in radians.

4.6 Conclusion

I have presented a markerless, multi-camera and egocentric visual tracker that breaks through the frames-per-second sampling barrier, by leveraging commodity rolling shutter cameras as dynamic 1D (*e.g.* line scan) sensors. Towards this end, the spatio-temporal relationships of a rigid camera cluster are modeled in terms of linear motion approximation and efficient image representations, which are based on the small-motion assumptions enabled by the kHz sampling frequencies. Moreover, I have developed a prototype system achieving upwards of 80kHz visual tracking. I have validated and characterized the performance and limitations of the tracking components on representative synthetic data. Finally, I have discussed integration and implementation details critical to system deployment, such as temporal synchronization and spatial calibration of the multi-camera cluster system.

The importance of the proposed approach lies in bridging the sampling frequency gap between inertial (*i.e.* IMU) and visual sensors, by enabling visual tracking at frequencies that are orders of magnitude greater than the RS camera native fps. Along these lines, the long-term potential for a more homogenized sensing frequency landscape opens exciting opportunities for novel sensor integration mechanisms.

CHAPTER 5: LENS DISTORTION AND TRACKING

5.1 Introduction

The recent resurgence of augmented reality/virtual reality has provided challenging computer vision problems of drastically higher requirements. For example, recent state-of-the-art low-latency rendering techniques (Zheng et al., 2014; Lincoln et al., 2016) assume that very high-frequency ($>$30 kHz) tracking is available. However, commercial AR/VR systems like Google's Daydream and Cardboard products and Samsung VR, provide only rotational tracking, which limits the immersive experience. The Oculus Rift and the HTC Vive provide full 6-DoF (*i.e.* rotation and translation) tracking at $\sim$1 kHz frequencies. This full 6-DoF tracking comes at a cost by requiring external lighthouses or IR cameras. Microsoft's HoloLens provides inside-out tracking without requiring any external apparatus, but still incurs tracking errors; see Sec. 5.4.3 for a more detailed analysis. It is well understood that high tracking rates are critical for immersive experiences in AR/VR systems (Sanchez-Vives and Slater, 2004). In this chapter, I present a full 6-DoF tracking system using multiple cameras that can track at extremely high frequencies up to 86.4kHz using commodity rolling shutter cameras. In comparison to the previous tracker, this chapter introduces a method to reduce the number of cameras required in the cluster.

Many commercial AR/VR systems rely on camera-based tracking, but for cost reasons use standard cameras with frames rates of a few tens of frames per second: *e.g.*, the Oculus Rift DK2 uses a 60 Hz external IR camera. In order to achieve the higher frame rates required for tracking, they rely on gyroscope data to improve the tracking frame rate. However, positional tracking still remains difficult to achieve, as IMUs drift (LaValle et al., 2014). Even using high frame-rate global shutter cameras is not the solution, as the high frame rates required lead to a decrease in the maximum possible exposure time (equal to the inverse of the kHz frame rate), making capturing

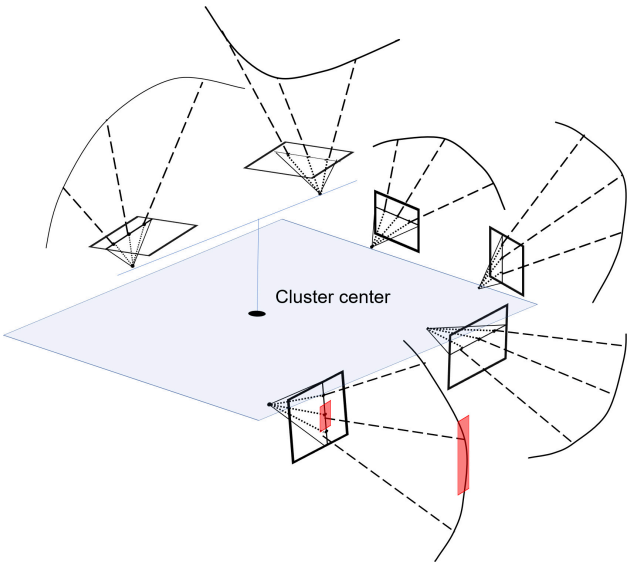


Figure 5.1: Due to radial distortion, rows of rolling shutter images correspond to rays which span a curve in space. Rolling shutter further provides a dense sampling of the scene in time. This combination of high frequency and curved sampling of the scene provides more information than traditional pinhole cameras. The highlighted region reflects per-row image region corresponding to the curve.

sufficient light impractical, especially indoors (Handa et al., 2012), where AR/VR systems are most frequently used. Also, most of these systems are head-worn and require small-form-factor cameras, further limiting the amount of light captured.

The proposed camera-based approach uses standard low-cost commodity cameras and leverages two unlikely camera '*features*': rolling shutter and radial distortion, both of which are usually considered to be nuisances in computer vision applications. In Chapter 4, I demonstrated a proof-of-concept inside-out tracking system using a cluster of 10 rolling shutter cameras that operated at frequencies as high as 80kHz. This chapter builds on that system and overcomes the limitations of the large number of required cameras as well as the need for stereo rectification and radial undistortion. A comparable tracking stability is achieved while only requiring *four* cameras instead of ten, making the system significantly more practical for use in a headset. As a result, the proposed system is simpler and more flexible in terms of camera placement in the multi-camera rig. The novel insight is that radial distortion in the camera lens induces constraints on the tracking system, which in turn reduces the number of cameras required, see Fig. 5.1. Additionally, I forgo the

need to compensate the image for radial undistortion and directly use the radially distorted rolling shutter image as captured by the camera yielding more efficient computation, again of significant importance for high-frequency tracking.

The rest of the chapter is organized as follows. In Section 5.2 I will briefly describe the innovations over the previous tracker. In Section 5.3, I will revise the high-frequency tracking system introduced in Chapter 4 and describe how radial distortion is in fact beneficial for tracking. In Section 5.4, I describe the experimental evaluation of the method and finally conclude in Section 5.5.

5.2 Improvements over RS tracking

In many computer vision applications, rolling shutter artifacts and radial distortion are considered negative aspects of the capture process that must be corrected for after the image formation. However, just as Chapter 4 and the associated journal paper (Bapat et al., 2016) converted rolling shutter "distortion" into an enabling feature for high frequency tracking, I argue that radial distortion can serve to greatly reduce a tracking system's complexity. It is in fact a feature, not an artifact.

This chapter extends the high frequency 6-DoF tracking method introduced in Chapter 4 that used a cluster of RS cameras arranged in stereo-pairs. In that system, each row of the RS camera is treated as a line-camera capturing the scene at high frequency. Each row is processed to measure pixel disparities for the stereo camera to obtain depth of a single point per camera. That system also measures the pixel shift of this point across time to obtain a constraint on the cluster motion. Such a formulation provides a single constraint per stereo-pair in a system of linear equations, requiring at least 12 cameras. In the simulation experiments, I assumed the 10 stereo-pairs that have their epipolar lines exactly parallel to the rows. I also assume in the synthetic experiments that the left-right rows expose at the same time. This is hard to achieve in reality at kHz capture rates, particularly as the 5 GoPro stereo-pairs are not genlocked. This leads to staggering of row exposures in time, which enables tracking points with different y coordinates reducing the number

(a) (b) (c) (d) (e)

Figure 5.2: Under radial distortion, an input row sample represents a curve in the undistorted image space. (a) Radially distorted image with one row highlighted by a dashed green line; (b) the same row in undistorted image space forms a curve. By approximating this curve with (for example) three linear line segments as shown in (c-e), multiple virtual cameras are obtained that each provide an independent linear constraint. In this sense, a single radially distorted line-image approximates the multiple pinhole cameras used in Chapter 4. For purposes of visualization, the distortion in (a) and (b) has been exaggerated; in practice, I use a larger number of local linear approximations that better fit the undistorted curve.

of cameras from 6 to 5 stereo-pairs. I use the spirit of this earlier work to exploit the different rows of a RS camera as independent high-frequency line cameras. Given this high-frequency sampling, I propose a novel system that leverages the naturally occurring radial distortion of the lens as a feature for motion estimation to obtain a complementary set of constraints, without the need of row-level in-step capture. The tracking also benefits from a wider field of view where radial distortion is more pronounced. Such cameras are commonly used in tracking research, *e.g.*, using a 130° FoV camera is recommended for LSD-SLAM (Engel and Schöps, 2014).

Ideal pinhole rolling shutter cameras only allow for a single linear constraint per row-image. The key insight leveraged in this chapter is that the process of undistorting a radially distorted row yields a curve in the image space that can be approximated by multiple line segments, as shown in Fig. 5.2. Linear shifts can thus be computed for each segment independently, affording a linear constraint per segment, substantially reducing the number of cameras by 60%: from ten cameras in Chapter 4 to four cameras in the new system. In tetherless AR/VR devices, reducing system complexity is critical, and the proposed approach substantially lowers the video bandwidth and headset weight. The proposed extension has the following advantages over the previous system while maintaining its benefits of kHz tracking frequencies: 1. Instead of ten cameras, I show high-frequency tracking can be performed using just four cameras. They are arranged so that every camera has overlap with at least one other camera. 2. The proposed approach does not need guarantees of

physically in-plane stereo camera sensors for depth estimation, as I propagate a depthmap for each camera across time. Rather, the tracker benefits from the rotation between cameras, as each view provides additional scene coverage. 3. I exploit the lens' radial distortion and use it to track multiple points per row; see Section 5.3.1 for a detailed description.

5.3 Approach

Here, first I will briefly revisit the method introduced in Chapter 4 and later build on it to add constraints obtained by leveraging radial distortion. The time step Δt is defined from t_1 to t_2, where $t_2 > t_1$, and magnitude of $t_2 - t_1$ may vary. Bold capital letters define a matrix, *e.g.* $\boldsymbol{M}$. A relative transformation from space w to space c is defined as ${}^{c}\boldsymbol{T}_{w}$, which transforms points in space w to space c, and a transform within a space w to be $\boldsymbol{T}_{w}$.

The tracking method of Chapter 4 uses a cluster of cameras to track the head-pose starting from the identity pose. There are N cameras in the cluster, and the relative pose of camera $n \in N$ in the cluster space is denoted by ${}^{n}\boldsymbol{T}_{cl}$, which can be estimated by performing an extrinsic calibration. The 6 DoF head-pose tracking is performed by measuring small pixel shifts between the projection of a 3D world point $X_w = [x_w\ y_w\ z_w\ 1]^T$ in camera n at time instants t_1 and t_2, and transforming the shifts into the 3D space using the depth of the point. These operations can be expressed for a camera n as follows:

$$\begin{aligned} X(n,t_1) &= {}^{n}\boldsymbol{T}_{cl}\,{}^{cl}\boldsymbol{T}_{w}(t_1)X_w \\ X(n,t_2) &= {}^{n}\boldsymbol{T}_{cl}\,{}^{cl}\boldsymbol{T}_{w}(t_2)X_w \end{aligned} \tag{5.1}$$

where $X(n,t_1) = [x_1\ y_1\ z_1\ 1]^T$ and $X(n,t_2) = [x_2\ y_2\ z_2\ 1]^T$ correspond to the same 3D world point X_w, and are expressed relative to camera n at time t_1 and t_2 respectively. The transformation ${}^{cl}\boldsymbol{T}_{w}$ transforms the 3D point X_w from world space to the cluster space.

The 3D points $X(n,t_1)$ and $X(n,t_2)$ are related to the corresponding pixels $\boldsymbol{x}(t_1)$ and $\boldsymbol{x}(t_2)$, $\boldsymbol{x}(t_1) = [p_x(t_1)\ p_y(t_1)\ 1]^T$, via the intrinsic matrix $\boldsymbol{K}_n$. Depths $z(t_1)$, $z(t_2)$ as defined by the

following relations:

$$\begin{aligned} X(n,t_1) &= z(t_1)\ \boldsymbol{K}_n^{-1}\boldsymbol{x}(t_1) \\ X(n,t_2) &= z(t_2)\ \boldsymbol{K}_n^{-1}\boldsymbol{x}(t_2) \end{aligned} \tag{5.2}$$

Tracking at high frequencies allows a small relative motion assumption at each time step. This small motion approximation for the combined transform ${}^{cl}\boldsymbol{T}_w(t_2){}^{cl}\boldsymbol{T}_w^{-1}(t_1)$ is denoted by $\delta\boldsymbol{M}_{cl}$. The cluster's approximate small motion $\delta\boldsymbol{M}_{cl}$ can be expressed in matrix form as

$$\delta\mathbf{M}_{cl} = \begin{bmatrix} 1 & \theta_z & -\theta_y & -t_x \\ -\theta_z & 1 & \theta_x & -t_y \\ \theta_y & -\theta_x & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

Using the small motion assumption, Eq.(5.1) can be simplified to

$$X(n,t_2) = {}^{n}\mathbf{T}_{cl}\ \delta\ \boldsymbol{M}_{cl}\ {}^{cl}\mathbf{T}_n\ X(n,t_1) \tag{5.4}$$

The depth $z(t_1)$ is estimated using the rows captured at time t_1 from the left and right cameras in the stereo-pair by measuring pixel disparity. To estimate motion, temporal stereo is used to measure the shift in pixels $p_x(t_2) - p_x(t_1)$ across the time step Δt.

Using Eqn.(5.2), the image points are substituted for 3D points since the measurements are in terms of pixels. Subsequently, a linear system is formulated by rearranging Eq.(5.4) to the form $\boldsymbol{C}\ \boldsymbol{A}\ Y = \boldsymbol{C}\ B$, where $\boldsymbol{C}$ is a diagonal matrix of weighting factors estimated according to the confidence in the measured shifts and $Y = [\theta_x\ \theta_y\ \theta_z\ t_x\ t_y\ t_z]^T$ represents the unknown 6-DoF motion vector of the camera. Matrix $\boldsymbol{A}$ captures the cluster configuration and the tracked 3D points, while vector B captures the shift measurements.

However, the proof-of-concept high frequency tracking method from Chapter 4 assumes ideal conditions regarding row-level in-step exposure and exactly in-plane stereo-pairs. Also a perfect lens is assumed, *i.e.* the radial distortion is treated as an artifact and is corrected for during the pre-

processing in the experiments. Due to this, I was forced to perform rectification and undistortion of the images to make the input images more amenable to their model. This pre-processing scheme however blends neighboring rows captured at different times during the radial undistortion and rectification steps and reduces image size. In contrast, the proposed method leverages the full breadth of the original image information to obtain tracking with fewer cameras.

5.3.1 Radial Distortion

Significant radial distortion is observed in wide angle cameras and activity cameras such as GoPros. Such wide-angular cameras are preferred for more stable tracking. Radial distortion transforms image rows to curves, and hence rays corresponding to each row span a curved surface in 3D space, rather than a plane as assumed in the pinhole model (Fig. 5.1). In this chapter, I model the radial distortion in the system and explicitly leverage the curved mapping introduced by it to measure points at different rows in the undistorted image space. I use the first two parameters, k_1 and k_2, of Brown's radial distortion model (Brown, 1966). These parameters transform an undistorted normalized pixel $(\tilde{p}_{xu}, \tilde{p}_{yu})$ into a distorted normalized pixel $(\tilde{p}_{xd}, \tilde{p}_{yd})$ as

$$\begin{aligned} \tilde{p}_{xd} &= \tilde{p}_{xu} \left(1 + k_1 r^2 + k_2 r^4\right), \\ \tilde{p}_{yd} &= \tilde{p}_{yu} \left(1 + k_1 r^2 + k_2 r^4\right) \\ r^2 &= \tilde{p}_{xu}^2 + \tilde{p}_{yu}^2. \end{aligned} \tag{5.5}$$

When such a distorted row is captured by a rolling shutter camera, the mapping from the 3D world point to the image is dependent on the time at which a row is exposed, as well as the distortion transformation.

In the rest of the chapter, I omit the intrinsic camera parameters for brevity and assume normalized coordinates for the image points. The radial distortion is leveraged to relax two restrictions: 1) radial distortion bends the row into a curve in the image plane, which means that shifts measured at different $\tilde{p}_{xd}$ positions at the same row $\tilde{p}_{yd}$ have different $\tilde{p}_{yu}$ locations. This has a similar effect

as the time staggering previously observed in real cameras in Chapter 4. 2) Multiple points per row are tracked, as opposed to a single point, providing more constraints, enabling the method to track using fewer cameras.

Rolling shutter provides row-wise capture, and the method measures how the current row shifts across time in the horizontal direction. In a given row, we can express the 3D X-coordinate in terms of $\tilde{p}_{xu}$ pixel positions at two time instants t_1 and t_2 using the pixel shift s_t as follows:

$$x(t_2) = \tilde{p}_{xu}(t_2)z(t_2) \approx (\tilde{p}_{xu}(t_1) + s_t)z(t_1) = x(t_1) + s_t z(t_1). \tag{5.6}$$

5.3.1.1 Linear independent constraints

Now, I show the linear independence of the constraints obtained for different points on the row whose rays span a 3D curve. Consider Eqn.(5.4), with ${}^{n}\boldsymbol{T}_{cl}$ as identity for simplicity (corresponding to the anchor camera in the cluster). The horizontal shift s_t corresponds to the first row of this vector equation. Combining Eqn.(5.4) with Eqn.(5.6), we obtain

$$\begin{aligned} x(t_2) &= x(t_1) - \theta_z \tilde{p}_{yu} z(t_1) + \theta_y z(t_1) + t_x \\ \Rightarrow s_t &= -\theta_z \tilde{p}_{yu} + \theta_y + \frac{t_x}{z(t_1)}. \end{aligned} \tag{5.7}$$

In Eqn.(5.7), for a given fixed $\tilde{p}_{yu}$ and small motion $\delta\boldsymbol{M}_w$, the only way the observed pixel shift s_t can change is due to the depth of the point $z(t_1)$, which is scene-dependent. That is why in Chapter 4, I had to rely on more cameras to obtain sufficiently many constraints. In contrast, the proposed method overcomes this inherent limitation by recognizing that, due to the radial distortion, each distorted row spans a curve in undistorted space. Hence, now I can extract an independent constraint from a piece-wise local linear segment in undistorted space as depicted in Fig. 5.2, providing multiple independent constraints per radially distorted row. The additional cameras in the cluster provide even more constraints. This is imperative for reducing the required number of cameras in the new system, as each camera in the cluster provides multiple constraints at each point in time instead of

just one. Similar analysis can be done for ${}^{n}\boldsymbol{T}_{cl}$ not equal to identity matrix ($\boldsymbol{I}$), which is the case for the rest of the cameras in the cluster.

Derivation of constraints for ${}^{n}\boldsymbol{T}_{cl} \neq \boldsymbol{I}$

Here, I derive the additional constraints that are obtained from the cameras in the cluster with non-identity pose ${}^{n}\boldsymbol{T}_{cl}$. For non-identity pose ${}^{n}\boldsymbol{T}_{cl}$, 3D point $X(n, t_1)$ can be expressed in cluster space as:

$$X_{cl}(n, t_1) = {}^{cl}\boldsymbol{T}_{n}\, X(n, t_1) = [x\ y\ z\ 1]^T \tag{5.8}$$

If we expand the matrix multiplication of $\delta\ \boldsymbol{M}_{cl} X_{cl}$ in Eqn.(5.4) using Eqn.(5.8), such that the unknowns are in a column vector, we can rewrite Eqn.(5.4) as follows:

$$\delta \boldsymbol{M}_{cl} X_{cl}(n, t_1) = \begin{bmatrix} -1 & 0 & 0 & 0 & -z & y \\ 0 & -1 & 0 & z & 0 & -x \\ 0 & 0 & -1 & -y & x & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \\ \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} + X_{cl}(n, t_1)$$

$$= \qquad \boldsymbol{Q}_n \qquad\qquad Y \quad + X_{cl}(n, t_1).$$

Considering just the first row ρ_1 of Eqn.(5.4), we can write a constraint for any camera n as follows:

$$X(n, t_2)(1) - {}^{n}\boldsymbol{T}_{cl}.\rho_1 X_{cl}(n, t_1) = {}^{n}\boldsymbol{T}_{cl}.\rho_1 \boldsymbol{Q}_n Y \tag{5.9}$$

Note that $X(n, t_2)(1) - {}^{n}\boldsymbol{T}_{cl}.\rho_1 X_{cl}(n, t_1)$ is a single scalar and is an element of column vector B from the system of linear equations $\boldsymbol{C}\ \boldsymbol{A}\ Y = \boldsymbol{C}\ B$ described in Section 5.3. Similarly, ${}^{n}\boldsymbol{T}_{cl}.\rho_1 \boldsymbol{Q}_n$ forms a row of matrix $\boldsymbol{A}$. Hence, multiple points observed in the same camera will share ${}^{n}\boldsymbol{T}_{cl}.\rho_1$ but will have different $\boldsymbol{Q}_n$, providing different constraints for each point. The linear independence

of the constraints still persists as the choice of coordinate system to express poses ${}^{n}\boldsymbol{T}_{cl}$ for the cameras is arbitrary and the constraints are obtained from a single row of each camera.

The stability of such an over-determined linear system can be examined using the condition number (Belsley et al., 2005). I found that the condition number of our linear system using a 4- or 6-camera cluster is of the same order as that of the 10-camera system of Chapter 4. Note that the number of constraints depends upon the extent of radial distortion present in the image. Higher radial distortion would give more varying $\tilde{p}_{yu}$ locations for the sampled $\tilde{p}_{yd}$ locations, making the system more stable. I assume that significant radial distortion exists due to the lens in all of our experiments. Without radial distortion, the linear system reduces to the case of Chapter 4. Small distortion, in the range of one pixel, is also not sufficient, as the constraints obtained from a single row are similar to each other forming an ill-conditioned system.

5.3.1.2 Homography-based warp

I directly process the radially distorted rolling shutter rows as soon as they become available. To achieve this, for each incoming distorted row, I synthesize a reconstructed row from the older frames for measuring shifts. This reconstructed row is created by sampling from a global shutter undistorted image frame F_{GS} with absolute pose $\boldsymbol{V}_{ref}$. This frame F_{GS} is synthesized from previous row images. To create F_{GS}, I split the rotational homography warp of $\boldsymbol{H} = \boldsymbol{K}\boldsymbol{R}_{ref}\boldsymbol{R}_{row}^{-1}\boldsymbol{K}^{-1}$, which maps each rolling shutter row to the reference pose by creating two lookup tables: 1) a lookup table to undistort previous frames and adjust their rotation by application of $\boldsymbol{R}_{row}^{-1}\boldsymbol{K}^{-1}$, and 2) a similar table for $\boldsymbol{K}\boldsymbol{R}_{ref}$. This helps to avoid redundant computation when the reference pose $\boldsymbol{V}_{ref}$ changes. The reference pose is chosen adaptively such that the motion between the reference and the current row is within a predefined threshold.

5.3.1.3 Robust shift estimation

As I directly compare distorted rows, small errors in the pixel shift measurements can yield a pose estimate with significant errors depending upon their corresponding depth and position in the

distorted space. To mitigate this, I use a robust double exponential Holt-Winters smoothing filter to remove outlier shift estimates (Gelper et al., 2010). For a time series m_t, this filter with trend B_t is defined as follows:

$$\begin{aligned} \tilde{m}_t &= \alpha m_t^* + (1-\alpha)(\tilde{m}_{t-1} + B_{t-1}) \\ \tilde{B}_t &= \beta(\tilde{m}_t - \tilde{m}_{t-1}) + (1-\beta)B_{t-1}, \end{aligned} \tag{5.10}$$

where α and β are filter parameters and "tilde" denotes the filtered data. m_t^* is a "cleaned" version of m_t using Huber loss Ψ to penalize large differences between noisy estimates m_t and one-step forecasts $m_{t|t-1}$. The cleaned version m_t^* is

$$m_t^* = \Psi\left(\frac{m_t - m_{t|t-1}}{\hat{\sigma}_t}\right)\hat{\sigma}_t + m_{t|t-1}. \tag{5.11}$$

The scale of this difference is estimated by a slowly varying τ^2-scale estimate $\hat{\sigma}_t$, which is highly robust to outliers. Alg. 1 gives a pseudocode for this robust filtering. I refer the reader to Gelper et al. (2010) for more details.

Algorithm 1 Robust shift smoothing

function RobustFilter(Input : raw value m_t)
  Compute one-step forecast $m_{t|t-1} = \tilde{m}_{t-1} + B_{t-1}$
  Estimate τ^2-scale estimate $\hat{\sigma}_t$
  Refine $m_{t|t-1}$ to obtain m_t^* ▷ Eq. 5.11
  Apply Holt-Winters smoothing on m_t^* to obtain $\tilde{m}_t$ ▷ Eq. 5.10
  return Filtered output $\tilde{m}_t$

5.3.2 Cluster configuration

The placement of the cameras in the cluster affects the tracking estimates according to Eqn.(5.4). I use two cluster configurations in our experiments: 1) a 4-camera configuration, and 2) a 6-camera configuration. The even-numbered cameras are rotated by 90° in the image plane and have translations and small rotations in all three directions w.r.t the odd-numbered cameras. See Fig. 5.1 for a visualization of the 6-camera cluster. Although using a 2-camera cluster is possible as the problem is well-posed, in the current cluster, the first two cameras have principal axes mostly along the Z

RMSE	T_x (cm)	T_y (cm)	T_z (cm)	θ_x (degree)	θ_y (degree)	θ_z (degree)	Rendering error (px)
4-cam real data	0.0648	0.0974	0.1064	0.0541	0.0776	0.0720	0.6804
6-cam real data	0.1783	0.0890	0.1139	0.0326	0.0499	0.0959	0.7507
HoloLens Fig.(5.9)	0.21	0.04	0.05	0.05	0.03	0.02	0.8249
4-cam synthetic data	0.43	0.18	0.23	0.14	0.23	0.06	2.55
6-cam synthetic data	0.38	0.27	0.22	0.13	0.22	0.08	2.39
20 cam system from Chapter 4	0.67	0.83	1.34	0.21	0.47	0.29	4.63

Table 5.1: RMSE for tracking estimates using real imagery for the 4-camera rig, 6-camera rig, and HoloLens pose estimates compared against the ground-truth Hi-Ball tracking. For comparison with tracker from Chapter 4, the table shows RMSE computed for the same synthetic scene for the 4 and 6-camera cases, and the 20 camera (10 stereo-pairs) case from Chapter 4. The system from Chapter 4 incurs more errors, as it relies on only a single point per-row for pose estimation.

direction (forward) creating an *ill-conditioned* linear system. Thus measurements constraining the Z direction are sensitive to noise and the accuracy is unsatisfactory.

The overlap between adjacent cameras can be decreased to extend the overall field of view of the tracking system. The large overlap between cameras is primarily used to bootstrap the method by estimating a depthmap per camera. A possible reduction in overlap can be achieved if bootstrapping is done using stereo over multiple baselines across time as in LSD-SLAM (Engel et al., 2014b).

5.4 Experiments

The experiments to validate the performance of the proposed high-frequency tracker are conducted using synthetic and real world imagery. For the synthetic experiments, I created a simulator using OpenGL and Qt, which is capable of simulating rolling shutter capture under radial distortion. For the real data experiment, I used tracking information from a 2000Hz Hi-Ball tracking system (Welch et al., 1999) as ground truth. The Hi-Ball system is a low-latency wide-area tracking system operating with absolute errors of less than 0.03° and 0.5mm.

The metric I use for evaluating our approach is the rendering pixel error, that is, the error in the rendering of the virtual scene caused by tracking errors as seen in a VR headset, similar to Chapter 4. This is crucial for AR/VR, as the rendering pixel errors directly affects user experience. For

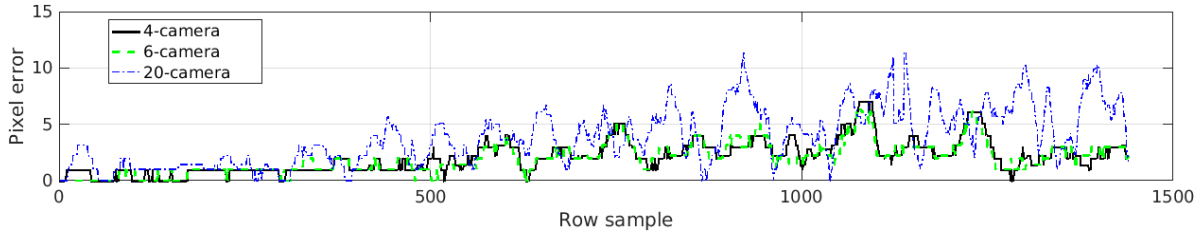
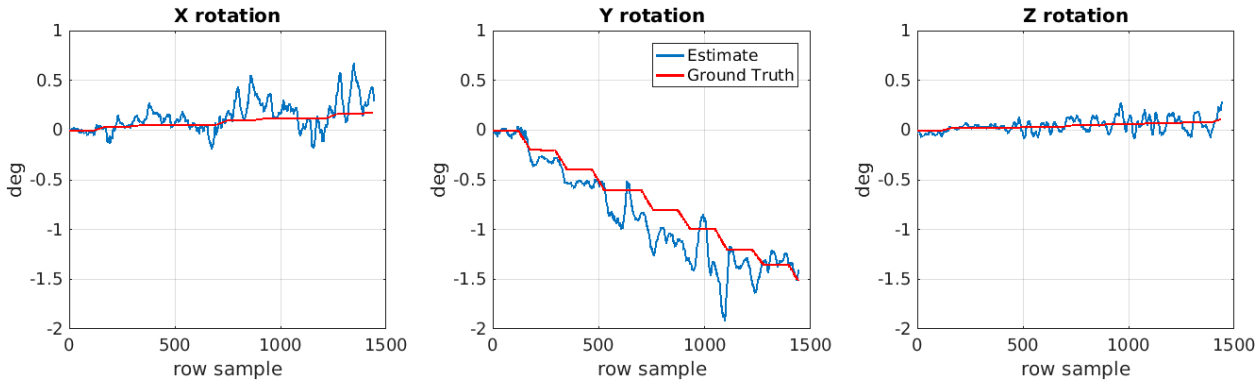


Figure 5.3: For the same camera motion, the proposed tracker incurs far smaller rendering pixel errors using only 4 cameras as opposed to the 20-camera system of Chapter 4 for synthetic data.

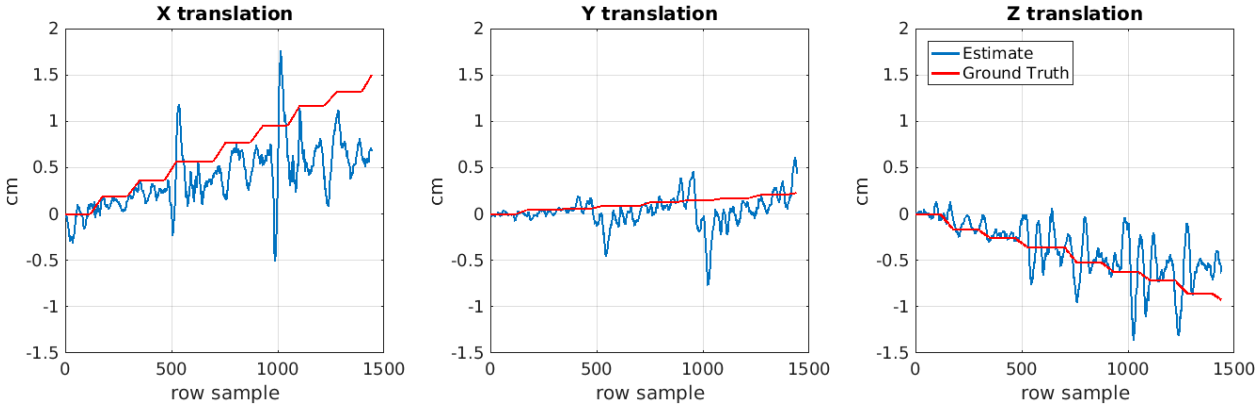
this metric, I estimate the rendering pixel error for a point 1m in front of the user for a resolution of 1080×1200 per eye, which is widely used by standalone headgears like the HTC Vive, Oculus Rift and Razer OSVR HDK2. I also present the root mean square error of the tracking with respect to the ground truth provided by the Hi-Ball. To compare the results with the tracker introduced in Chapter 4, I use synthetic imagery as it is easy to capture images of the same room along the same motion track using a simulator. I present the tracking results of the proposed approach and compare against the rolling-shutter-aware SLAM method of Kim et al. (2016) by capturing real imagery using the custom cluster. Additionally, I compare the performance of Microsoft's HoloLens with the Hi-Ball to provide a perspective on how the errors of my tracker compare against tracking with current commercial AR systems, which work at much lower tracking frequencies.

5.4.1 Synthetic data

Now I present experiments using synthetic data, which was captured with camera parameters designed to be similar to real cameras. To compare directly with tracker from Chapter 4, all of the cluster cameras use the same vertical FoV of 60° and the images were rendered at 120Hz frame rate with a resolution of 480×640, making the tracking frequency 57.6kHz. As the proposed approach needs radial distortion, I used distortion parameters $k_1 = -0.27$ and $k_2 = 0.11$ to reflect real cameras. Real human motion data captured using the Hi-Ball is used to render this synthetic imagery. As the Hi-Ball tracking frequency does not match our tracking frequency, I interpolated it to match the tracker's estimation frequency of 57.6kHz. The room used to simulate motion was captured using a handheld Kinect (Dou et al., 2012).



(a)



(b)

Figure 5.4: Tracking estimates of the proposed 4-camera configuration using synthetic imagery and Hi-Ball tracking data for ground truth: (a) Rotation estimates in degrees and (b) translation estimates in cm.

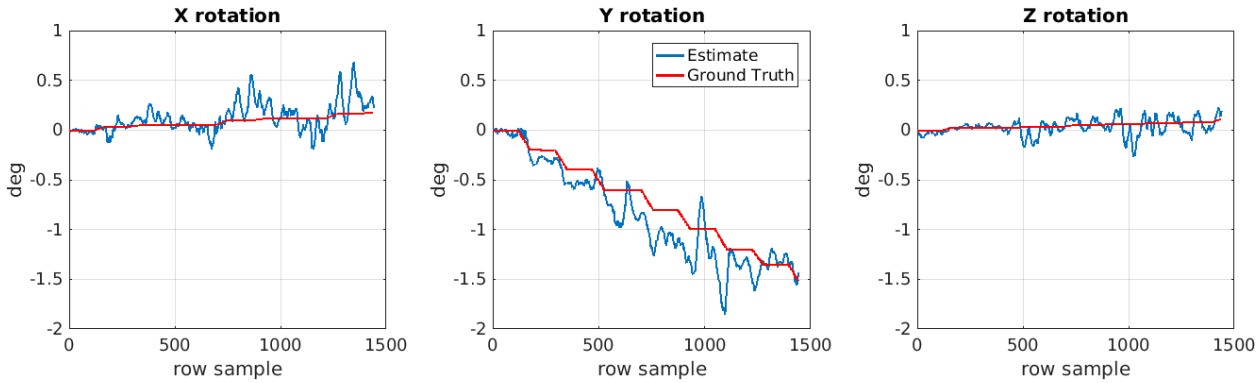


Figure 5.5: Tracking estimates of the proposed 6-camera configuration using synthetic imagery and Hi-Ball tracking data for ground truth: (a) Rotation estimates in degrees and (b) translation estimates in cm.

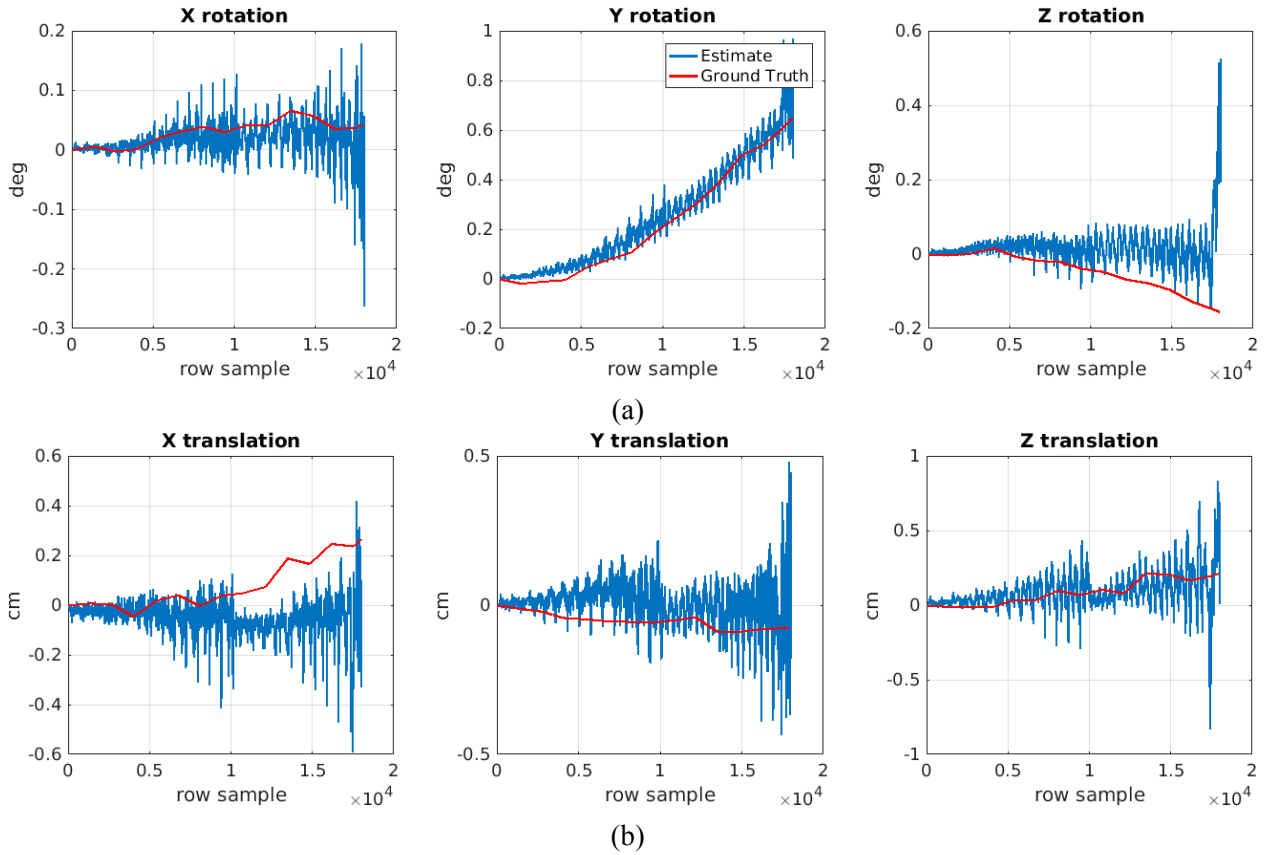


Figure 5.6: Tracking estimates of the 6-camera configuration using real imagery and Hi-Ball tracking data for ground truth: (a) Rotation estimates in degrees and (b) translation estimates in cm. Note that the scale of the y-axis is different for each figure.

The 6-camera case is accurate for real motions, see Table 5.1, while the 4-camera case shows more errors in tracking estimates, as it is inherently more sensitive to noise in pixel shifts. The 20-camera system of Chapter 4 estimates pose from a single point in a row, and hence is not robust. On the other hand, the proposed system leverages multiple points in each row for pose estimation and employs robust filters, leading to 2.55px RMS error for 4-camera case. Fig. 5.3 shows the rendering pixel error incurred across time for the 4- and 6-camera systems, and also the 20 camera cluster of Chapter 4. The rendering pixel errors for the 20-camera system deviate and start to accumulate quickly, while our system shows gradual drift. Fig. 5.4 shows the motion plot for the 6-DoF pose estimates against Hi-Ball ground truth. As the 4-camera cluster has fewer of redundant constraints, it exhibits higher noise sensitivity as compared to the 6-camera case; see Fig. 5.5. The rendering pixel errors for these motion plots are in Fig. 5.3 of the paper.

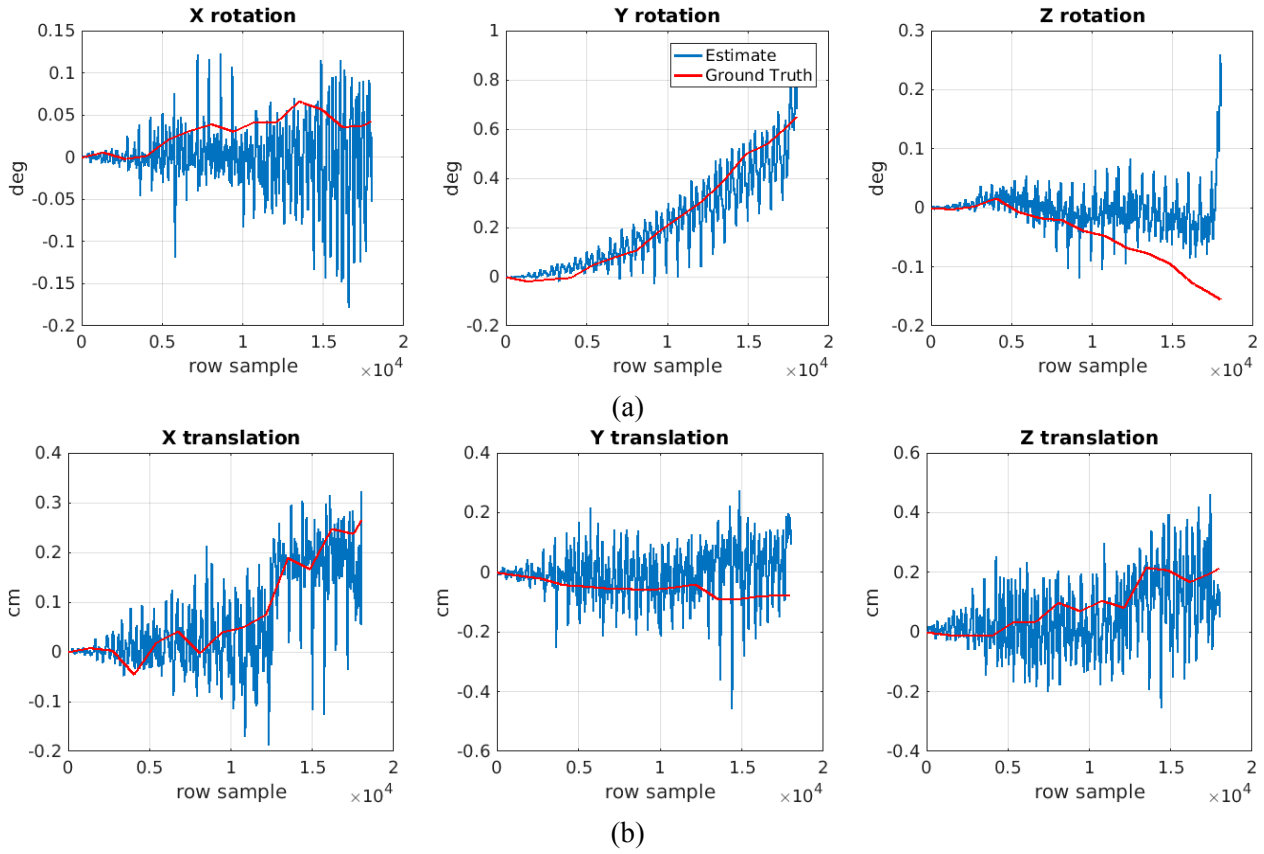


Figure 5.7: Tracking estimates of the 4-camera configuration using real imagery and Hi-Ball tracking data for ground truth: (a) Rotation estimates in degrees and (b) translation estimates in cm. Note that the scale of the y-axis is different for each figure.

5.4.2 Real data

For experiments on real data, I created a rig of six RS cameras mounted on top of a hardhat. All the cameras were GoPro Hero3+ silver edition and were kept at a narrow FoV, capturing at 720×1280 resolution at 120Hz. The line delay for each camera was calibrated using the method introduced by Oth et al. (2013) using a checkerboard pattern. The overlap in the cameras helped in calibrating adjacent cameras in pairs using Zhang (2000)'s method. All cameras were synchronized in time using a blinking LED similar to the method described in Chapter 4.

Now, I present experiments using captured data at 120Hz, amounting to 86.4kHz tracking frequency. The tracker can estimate poses orders of magnitude faster than the current commercial trackers for a fraction of the cost. The proposed method supports high-frequency tracking using the 4-camera and 6-camera cluster configurations. With the 6-camera setup, less smoothing is

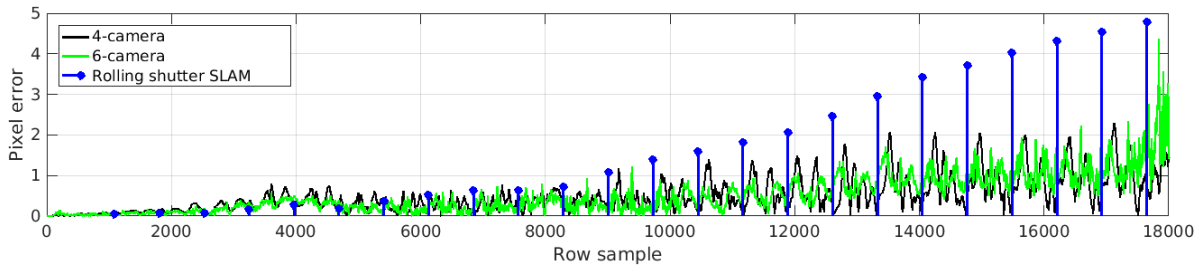


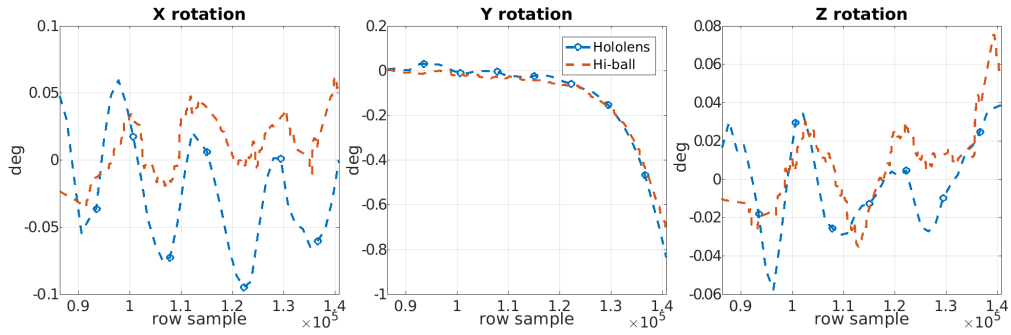
Figure 5.8: Render pixel estimates: The 4-camera configuration (black) has higher error than the 6-camera configuration (green). Blue dots show the rendering error obtained by the method of Kim et al. (2016), which maintains a pose estimate per keyframe, rather than per row.

required as the linear system has more redundant constraints; see Fig. 5.6, as compared to the 4-camera setup, see Fig. 5.7. Fig. 5.8 shows the rendering pixel error of the 4- and 6-camera setup against the RS–aware SLAM method of Kim et al. (2016). As Kim et al. (2016) estimate a pose per keyframe, the pixel error is shown at the center of each frame. The rendering pixel errors incurred for the proposed approach are generally around 1 pixel and increase to 3 pixels as the tracking drifts with some outlier spikes in errors. On the other hand, the RS–aware SLAM system shows stronger deviation up to 5 pixels.

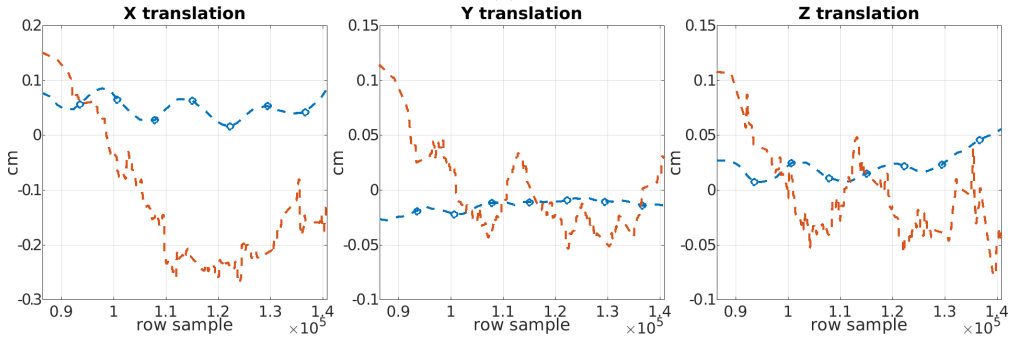
Compared to the tracker from Chapter 4, the proposed system is able to track for much longer, estimating poses for more than twice the row-samples while still maintaining accuracy. If we compare similar track length tracking estimates as in Chapter 4, the proposed system incurred only 1 pixel error for real imagery. Table 5.1 shows the RMSE for translation, rotation, and rendering pixel error. For real imagery, the 4-camera case and 6-camera cases perform equally well, incurring 0.68px and 0.75px rendering error, respectively.

5.4.3 Comparison with HoloLens

Now, I evaluate how the tracking errors for the proposed tracker compare to traditional systems like Microsoft's HoloLens for small-scale motion. It is impractical to wear the HoloLens, Hi-Ball and the camera cluster simultaneously, but we can attach the Hi-Ball to either HoloLens or the cluster. Hence, I compare the HoloLens with the Hi-Ball and my cluster with the Hi-Ball to provide an indirect comparison. Using the front color camera of the HoloLens, a hand-eye



(a)



(b)

Figure 5.9: Tracking estimates of the HoloLens compared with the Hi-Ball. Note that the vertical axes are not the same across plots. The horizontal axis is expressed in terms of row-samples, instead of time, for easier comparison.

calibration is estimated using Tsai and Lenz (1989) to register the HoloLens tracking data with the Hi-Ball. Fig. 5.9 shows the HoloLens tracking against the Hi-Ball tracking and demonstrates that the HoloLens suffers from translational drift but remains accurate in rotation. The motion range in this plot is similar to the scale of motion in the experiment with real imagery. The HoloLens is particularly good at tracking large translations, and at correcting for drift and incorrect translation estimates, due to its SLAM system. Fig. 5.9 highlights that the dominant motion in the Y-direction is tracked well by the HoloLens, but the rest of the motion axes have significant errors. Table 5.1 shows the RMSE for the HoloLens tracking. HoloLens incurs 0.82 rendering pixel RMSE; on the other hand, for a similar scale of motion, our tracking system incurs 0.68px for our 4-camera cluster. The proposed approach thus maintains high accuracy in tracking 6 DoF poses, just using commodity rolling shutter cameras.

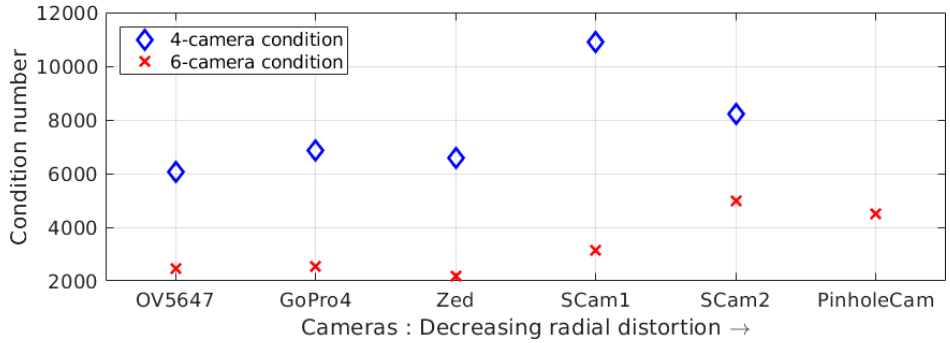


Figure 5.10: With higher radial distortion, the system stability increases (smaller condition number). Additional views increase stability (4- vs. 6-cameras). The plot shows the worst-case condition number (PinholeCam 4-camera result too large to show).

5.4.4 System conditioning and extent of distortion

System conditioning depends upon the amount of radial distortion present in the image. I rendered synthetic images for real, widely used cameras like the Pi OV5647 (k_1 = -0.31, k_2 = 0.083), GoPro 4 (-0.27, 0.11), and Zed (-0.17, 0.023). For additional data points, I also rendered images for simulated cameras: SCam1 (0.15, 0.05), SCam2 (0.05, 0.05), and PinholeCam (0, 0). For each of these camera types, I generated synthetic images for our 4- and 6-camera configurations. I report the worst-case condition number across all time points and compared this across cameras (Fig. 5.10). The condition number decreases with higher distortion due to the availability of more constraints per row, and it also decreases when more cameras are present. Average condition numbers (not shown) are substantially lower.

5.4.5 Failure cases and degeneracies

The proposed approach depends upon densely matching the row-image pixels, and the system fails when there is a complete absence of texture or small texture gradient. In practice, however, this is remedied by using cameras looking in different directions. Degeneracies arise if the cluster consists of all cameras looking in the same direction with the same orientation, which makes the constraints linearly dependent. The proposed cluster design ensures that at least one row can sense motion in each direction. It uses pairs of cameras looking in orthogonal directions to give

constraints in X, Y, and Z (see Section 5.3.2). The cameras in the pairs themselves are at 90° rotations, giving constraints in the local X and Y directions.

5.5 Conclusion

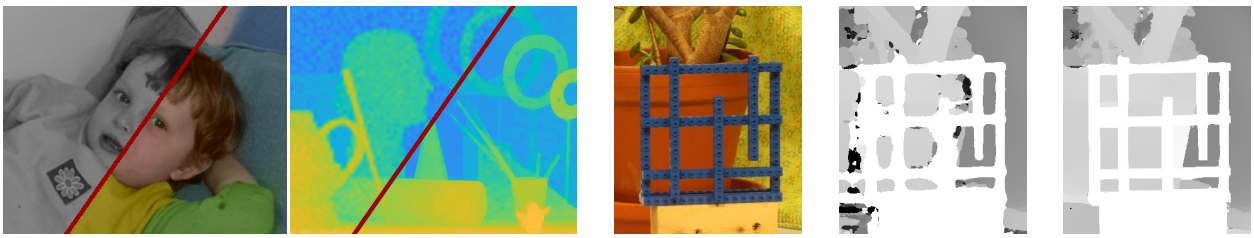
In this chapter, I have introduced a simple and flexible high frequency head-pose tracking system that explicitly leverages rolling shutter exposure and radial distortion without using any external markers. The proposed method achieves tracking frame rates that are orders of magnitude larger than current commercial systems like NDI's Optotrak Certus (NDI, 2018) for a fraction of the cost. In order to achieve this, every row of a rolling shutter image is processed to estimate a pose per-row. This high-frequency sampling of rows allows a linear intra-frame motion assumption. By splitting a radially distorted rolling shutter row into multiple locally linear line segments, the system extracts multiple linear constraints on 6-DoF head motion and accordingly requires only 4-6 cameras for stable 86.4 kHz tracking. Through rigorous experiments, I show the viability of the proposed approach using synthetic data and validate using real imagery captured using the prototype camera cluster headgear. Moreover, the approach allows more flexibility in designing the cluster and simplifies the physical alignment constraints on the system. Most importantly, I have introduced the first approach to leverage both rolling shutter and radial distortion to improve high-frequency tracking.

CHAPTER 6: EDGE-AWARE OPTIMIZATION

6.1 Introduction

Edge-aware optimization is a widely utilized tool in computer vision. It has been applied to a large variety of tasks, including semantic segmentation (Krähenbühl and Koltun, 2011), stereo (Bleyer et al., 2011), colorization (Levin et al., 2004), and optical flow (Revaud et al., 2015). Edge-awareness is motivated by the intuition that similar-looking pixels often have similar properties. For this reason, a wide variety of edge-aware filtering algorithms have been developed, including the bilateral filter (Tomasi and Manduchi, 1998), anisotropic diffusion (Perona and Malik, 1990), and edge-avoiding wavelets (Fattal, 2009), all of which identify similar-looking pixels. However, using such filters in optimization frameworks typically leads to computationally expensive algorithms. While high-level groupings like super-pixels can be used to compensate for this sluggishness (Lu et al., 2013), the color-space clusterings of such approaches are not guaranteed to respect the semantics of the underlying domain, often leading to artifacts.

In this chapter, I propose a general optimization framework called domain transform solver (DTS) that directly operates in the pixel space while maintaining distances in the combined color and pixel space with an edge-aware regularizer. The framework can be applied for a variety of optimization problems, as demonstrated in Fig. 6.1 and Section 6.3. The method achieves competitive accuracy in applications like stereo optimization (Section 6.3.1), rendering from defocus (Section 6.3.2), depth super-resolution (Section 6.3.3), and high resolution depth filtering for multi-view stereo (Section 6.4.5). While being extremely fast, my framework excels in two aspects that are not jointly achieved by any prior edge-aware optimization algorithms: 1. with increasing image resolution, as well as a growing number of image dimensions or channels, the method scales linearly and is more than twice as fast as existing parallel methods, and 2. the approach is independent



(a) Colorization result

(b) Noisy depthmap vs. our 16x upsampled smooth result

(c) Color image

(d) Target image

(e) DTS result

Figure 6.1: The domain transform solver can tackle a variety of problems, including (a) colorization, (b) depth super-resolution using the color image as reference, and (c-e) depth map refinement. (c-e) shows a color image from the Middlebury dataset (Scharstein et al., 2014) with an initialization (target) obtained from MC-CNN (Zbontar and LeCun, 2016), which is then refined in an edge-aware sense to obtain DTS result (e).

of blur kernel sizes. Considering the increasingly high resolutions of cameras in consumer devices such as smartphones, and the desire for on-device computing for tasks like automatic photo enhancement, the contributions of this chapter are poised to enable crucial advances for upcoming technologies. DTS is closely related to fast bilateral solver (FBS) (Barron et al., 2015; Barron and Poole, 2016) in that I target the same goal of developing general solvers that are edge-aware and efficient. The gridding strategy of (Barron et al., 2015; Barron and Poole, 2016) scales well with higher blur amounts and larger spatial windows. However, using higher blur windows is not always a viable option, especially in high-resolution imagery where it is important to maintain fine details, such as multi-camera capture for virtual reality and satellite imagery. In contrast, DTS does not require large blur kernels to be efficient. It scales well with higher image resolutions regardless of blur kernel size. My DTS approach is inherently parallelizable, and hence can greatly benefit from GPU processing. To foster broad use of my framework, the source code is available at `https://github.com/akashbapat/domain_transform_solver`.

The remainder of this chapter is organized as follows: Section 6.2 derives the DTS optimization framework and highlights its similarities as well as dissimilarities with previous work. I also illustrate how my framework can be leveraged for various vision tasks like stereo, depth super-resolution, colorization, high-resolution depth filtering, and synthetic defocus, in Section 6.3. In

Section 6.4, I provide a quantitative evaluation to show the competitive accuracy of DTS, as well as validation of the timing performance which shows the significant speed-up that DTS can deliver. Finally, in Section 6.5 I conclude and provide some future directions for expanding the DTS framework.

6.2 Approach

Edge-aware filtering techniques smooth similar-looking regions of the image while preserving crisp edges. Filtering techniques can often be formulated as single iterations in a solver. For instance, the bilateral filter is equivalent to a single step in minimizing a weighted least squares energy function (Elad, 2002). In the following, I will introduce the domain transform solver (DTS), which leverages repeated applications of the domain transform (DT) filter. Firstly, I will describe an optimization framework that is analogous to the existing bilateral solver formulations and then explain how the DT significantly boosts the framework's speed and scalability.

6.2.1 Optimization framework

The DTS solves the following optimization problem:

$$\min_{z} \lambda \sum_{i} \underbrace{\left(z_i - \bar{z}_{N_i}\right)^2}_{=e(z_i, N_i)} + \omega_i c_i \left(z_i - t_i\right)^2 + \sum_{m} \lambda_m \Phi_m \left(z\right) \tag{6.1}$$

Here, the z_i are the values we want to estimate, *e.g.* disparity in stereo, at the i^{th} pixel of an image. The initial target estimate t_i with a confidence c_i is also given for the i^{th} pixel; ω_i is an edge-aware normalization term described below. This optimization objective has an edge-aware regularizer $e\left(z_i, N_i\right)$, which forces the z_i to be similar to the mean $\bar{z}_{N_i}$ of its neighborhood N_i, computed in an edge-aware sense. This edge-aware mean is $\bar{z}_{N_i} = \left(\sum_j W_{i,j} * z_j\right) / \sum_j W_{i,j}$, where $W_{i,j}$ takes into account the pixel color similarity as well as the distance between pixels i and j. I will derive $W_{i,j}$ in Section 6.2.2. The term $\omega_i = 1/\sum_j W_{i,j}$ scales the confidence of each pixel, such that

pixels with similar neighbors are influenced less by their target value, and vice versa. The $\bar{z}_{N_i}$ is computed using the domain transform, which enables DTS to evaluate the pair-wise regularizer for any blur kernel size and dimensionality, faster than traditional approaches (Bleyer et al., 2011; Tomasi and Manduchi, 1998; Paris et al., 2009; Fattal et al., 2007; Paris and Durand, 2006). $\Phi_m(z)$ is an application-dependent term with a weighting factor of λ_m. For example, $\Phi_m(z)$ could be the photometric matching cost for the left-right image pair for stereo.

In all applications, the DTS framework aims to solve Eq. (6.1). The minimum at the point of the solution necessarily has a zero derivative. Next I will characterize this minimum in order to leverage it later in the proposed approach. For simplicity, let us first investigate a simplified version of Eq. (6.1) that does not contain the problem-specific term $\Phi_m(z)$. This simplified version can be written as :

$$\min_z F(z) = \min_z \lambda \sum_i \underbrace{\left(z_i - \bar{z}_{N_i}\right)^2}_{=e(z_i,N_i)} + \sum_i \omega_i c_i \left(z_i - t_i\right)^2. \tag{6.2}$$

Taking the gradient of Eq. (6.2) with respect to z_i and setting it to zero, at the minima of Eq. (6.2) we have

$$z_i = \frac{\lambda \bar{z}_{N_i} + \omega_i c_i t_i}{\lambda + \omega_i c_i}. \tag{6.3}$$

On the other hand, FBS's optimization objective is

$$\min_z \frac{\lambda_{FBS}}{2} \sum_{i,j} W_{i,j} \left(z_i - z_j\right)^2 + \sum_i c_i \left(z_i - t_i\right)^2. \tag{6.4}$$

Inspecting the derivative of Eq. (6.4) at the minimum, we obtain:

$$z_i = \frac{\lambda_{FBS}\ \bar{z}_{N_i} + \frac{c_i t_i}{\sum_j W_{i,j}}}{\lambda_{FBS} + \frac{c_i}{\sum_j W_{i,j}}}. \tag{6.5}$$

This optimal solution is, in fact, the same as that of the DTS, see Eqn. (6.3). However, DTS uses a different solving scheme that only approximates the solution. The key distinction for my method lies in the explicit computation of $\bar{z}_{N_i}$ via $W_{i,j}$. Whereas the FBS algorithmically depends on a

kernel-size-dependent domain gridding or tiling in the computation of $W_{i,j}$, the DTS instead maps the z_i values into a 1-D space using the domain transform, allowing pixel affinities to be calculated with a runtime independent of blur kernel size.

The domain transform is isometric only when the domain is 1D (Gastal and Oliveira, 2011), but I use the 2D pixel space as the domain. Hence, DTS must iteratively solve for the ideal solution in the X and Y directions independently. On any given pass, the lateral neighbors of each swept pixel are not considered, and thus only an approximate solution is obtained. In addition, Eq.(6.3) assumes $\frac{\partial \bar{z}_{N_i}}{\partial z_i} = 0$, which is exact when $W_{ii} = 0$. While filtering using moving average in the domain transform space, however, this is only approximately maintained for pixels with neighborhoods of sufficient weight.

6.2.2 Domain transform with the L_2 norm

Gastal and Oliveira (2011) define an isometric transformation, which they call the domain transform (DT), for a 1-D multi-valued function $I : \Omega \rightarrow \mathbb{R}^c$, $\Omega = [0, \infty)$ by treating $C = (x, I(x))$ as a curve in $\mathbb{R}^{c+1}$. The domain transform $DT : \mathbb{R}^{c+1} \rightarrow \mathbb{R}$ is such that it preserves distances between two points on the curve C under a given norm. In contrast to Gastal and Oliveira (2011), I use the L_2 norm to define the distances. This results in higher accuracy as shown in Table 6.2. Hence, I will derive the domain transform here, which satisfies the constraint $\|DT(x_i, I(x_i)) - DT(x_j, I(x_j))\|_2 = \|(x_i, I(x_i)) - (x_j, I(x_j))\|_2$ for the nearest neighbors x_i and x_{i+1}. Here, only x_i and x_{i+1} are considered because the following derivation is exact only when we are close to x_i. Using a shorthand notation $DT(x) = DT(x_i, I(x_i))$ and assuming a small shift h in x, we can express the distance in pixels and color equal to the distance of the transform as follows:

$$(DT(x+h) - DT(x))^2 \stackrel{\text{def}}{=} h^2 + \sum_{k=1}^{c} (I(x+h) - I(x))^2$$

Rearranging the above,

$$\left(\frac{DT(x+h)-DT(x)}{h}\right)^2 = 1+\sum_{k=1}^{c}\left(\frac{I(x+h)-I(x)}{h}\right)^2$$

taking limit, $h \to 0$

$$\left(DT^{'}(x)\right)^2 = 1+\sum_{k=1}^{c}\left(I^{'}(x)\right)^2$$

$$DT^{'}(x) = \sqrt{1+\sum_{k=1}^{c}\left(I^{'}(x)\right)^2}$$

Integrating and assuming $DT(0)=0$

$$DT(u) = \int_0^u \sqrt{1+\sum_{k=1}^{c}\left(I^{'}(x)\right)^2}dx \tag{6.6}$$

Using this definition of the domain transform of the 4-D space $[X, R, G, B]$ with the curve C defined by RGB color and the spatial domain X, the edge-aware weights are expressed as follows:

$$W_{i,j} = \delta_r\left(\,|DT(z_i)-DT(z_j)|\,\right) \tag{6.7}$$

with indicator function $\delta_r(d) = (d \leq r)$.

The relation with the simple domain transform blurring (Gastal and Oliveira, 2011) can be seen by setting the confidence scores c_i to zero in Eq. (6.1). This will lead to the same solution as the domain transform filtering. Similarly, setting c_i to zero and $W_{i.j}$ to Gaussian weights in color and space will lead to bilateral filtering. Note that the above derivation is isometric since the function I is multi-valued but with a *1-dimensional* domain Ω. By extending the domain to 2-D, the exact isometry is not valid, and following Gastal and Oliveira (2011), I use alternating passes by separately considering the image as a function of X and then Y.

The confidence scores c_i are computed by estimating the variance of the refined variable z in an edge-aware sense using the domain transform as suggested by Barron and Poole (2016), and normalizing the variance into [0,1] range by taking negative exponential scaled by σ_c.

$$c_i = \exp\left(-\frac{V_i}{2\sigma_c^2}\right),\ V_i = DT\left(z_i^2\right) - DT\left(z_i\right)^2. \tag{6.8}$$

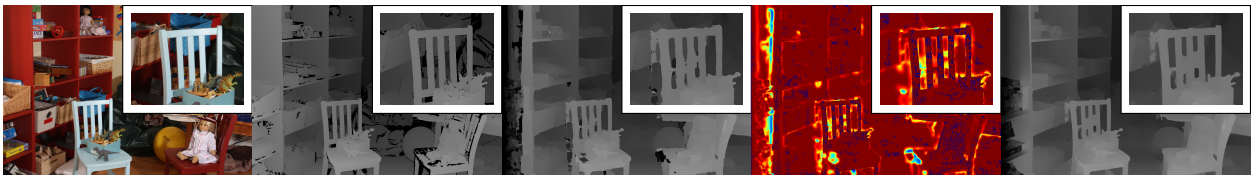
In this formulation, the domain transform is treated as a local estimate of the mean in an edge-aware sense, while σ_c scales the variance to get confidence scores. The confidence c_i is weighed during gradient descent updates by $\sum_j W_{i,j}$ to mimic its effect in Eq.(6.5), which provides less weight to target t_i when there is a large support via the similarities expressed by $W_{i,j}$.

6.3 Applications

Now, all the terms except the application-specific terms in Eq. (6.1) are defined. In the following, I describe the application of the proposed framework to a variety of scenarios where I will adapt Eq. (6.1) by changing function Φ_m.

6.3.1 Stereo optimization

Stereo depth estimation is a well-studied problem (Scharstein and Szeliski, 2002; Hirschmuller and Scharstein, 2007) in which the task is to estimate a matching correspondence of pixels in the left image to the pixels in the right image. This matching correspondence defines the disparity of the pixels and in turn the depth, and when done for each pixel provides us with a disparity map. Typically, a dense search is done along the row of a rectified pair by matching the pixel color similarity, *i.e.*, applying a photometric matching cost. In the following, disparity map is refined. The initial disparity map is obtained from matching cost CNN (MC-CNN) (Zbontar and LeCun, 2016), which acts as the target (Fig. 6.2c) for which confidence score is calculated (Fig. 6.2d). The left color image is used to compute the weights $W_{i,j}$ required to estimate the edge-aware mean. Then the disparities are optimized to obtain a disparity map that is smooth at homogeneous regions



(a) Color image (b) GT disparity (c) Target image (d) Confidence (e) DTS result

Figure 6.2: Stereo Optimization: The DTS result in (e) is computed using the color image (a) which is used to define color distance in the domain transform, and the target (c) disparity obtained from MC-CNN (Zbontar and LeCun, 2016). The confidence map (d) is used to weigh the target disparity in the optimization (Eq. (6.1)). Notice in the zoomed regions that DTS results are aligned to the edges of the color image.

but has sharp edges (Fig. 6.2e). Similar to the proposed solver, Barron and Poole (2016) show that the FBS works well for a wide variety of optimization problems including stereo. When they apply the FBS to the stereo problem, they achieve faster convergence compared to BL-Stereo because they neglect the physical implication of changing the disparity. In other words, if an optimizer changes the estimate of disparity at a point in the left image, this results in a change in the matching pixel in the right image and accordingly carries the potential of a change in the corresponding color. Here, I present a method for solving for the disparity in an edge-aware sense while having a photometric penalty for the left-right matching. The loss for stereo optimization is

$$\begin{aligned}\min_{z} \lambda \sum_{i} \left(z_i - \bar{z}_{N_i}\right)^2 + \sum_{i} \omega_i c_i \underbrace{\rho\left(z_i - t_i\right)}_{target\ term} \\ + \sum_{i} \underbrace{\gamma \left(I_L(i) - I_R(i - z_i)\right)^2}_{=\lambda_m \Phi_m(z_i)},\end{aligned} \tag{6.9}$$

where I_L and I_R are the left and right image of the stereo pair respectively. As disparity can be viewed as 1-D flow, for robust optimization, I use a Charbonnier loss $\rho\left(r\right) = \sqrt{r^2 + \varepsilon^2}$ with $\varepsilon = 0.001$ on the target term, which has been shown to be effective for optical flow (Sun et al., 2010).

Next, I will describe how rendering-defocus-from-depth problem which heavily relies on accurate depth edges benefits from DTS.

(a) Jadeplant: back carton in focus (b) Jadeplant: blue block in focus (c) Playroom: chairs in focus

Figure 6.3: Render from defocus for the Middlebury scenes. The original image is all-in-focus. (a) Original vs. DTS, background in focus. (b) Original vs. DTS, foreground in focus. (c) Original vs. DTS, with the chairs in the front in focus.

6.3.2 Synthetic defocus from depth

Interest in creating synthetic defocus from depth is growing, with phones like the Google Pixel 3 and the OnePlus 6 providing a portrait mode where the shallow depth of field effect is mimicked through the estimation of depth. In fact, BL-Stereo's synthetic defocus method is used as part of the Lens Blur feature on Google's phones (Barron et al., 2015). I use our stereo optimization from Section 6.3.1 to estimate depth maps, which retain sharp discontinuities at color edges. Fig. 6.3 shows the original color image and the defocus rendering produced by using our estimated depthmaps for scenes in the Middlebury dataset (Scharstein et al., 2014). As the stereo optimization is edge-aware, the defocus rendering maintains high quality even at the edges. In the Jadeplant scene shown in Fig. 6.3a, the background is in focus, and for the same scene the blue block in the front is kept in focus in Fig. 6.3b. In the Playroom scene in Fig. 6.3c, the front chairs are chosen to be in focus.

To render the synthetic defocus, I used the algorithm described in Section 6 of the supplementary material of Barron et al. (2015), which is reproduced here for convenience. Given the left image I_L and the optimized disparity Z according to the method described in Section 6.3.1, and a disparity which we want in focus d_f, the magnitude of shallow depth-of-field effect m, and a function RadialBlur$(\cdot, r)$ which applies blur of radius r, the method is described in Algorithm 2.

Algorithm 2 Render from defocus algorithm reproduced from Barron et al. (2015).

function RenderFromDefocus(I_L, Z, m,d_f)

$I_{\text{num}} \leftarrow 0$

$I_{\text{denom}} \leftarrow 0$

for $z = \min(Z) : \frac{1}{m} : \max(Z)$ **do**

$A \leftarrow |Z - z| \leq \frac{1}{m}$

$B \leftarrow I_L \,.*\, A$

$A_b \leftarrow$RadialBlur$(A, m|z - d_f|)$

$B_b \leftarrow$RadialBlur$(B, m|z - d_f|)$

$I_{\text{num}} \leftarrow I_{\text{num}} \,.*\, (1 - A_b) + B_b$

$I_{\text{denom}} \leftarrow I_{\text{denom}} \,.*\, (1 - A_b) + A_b$

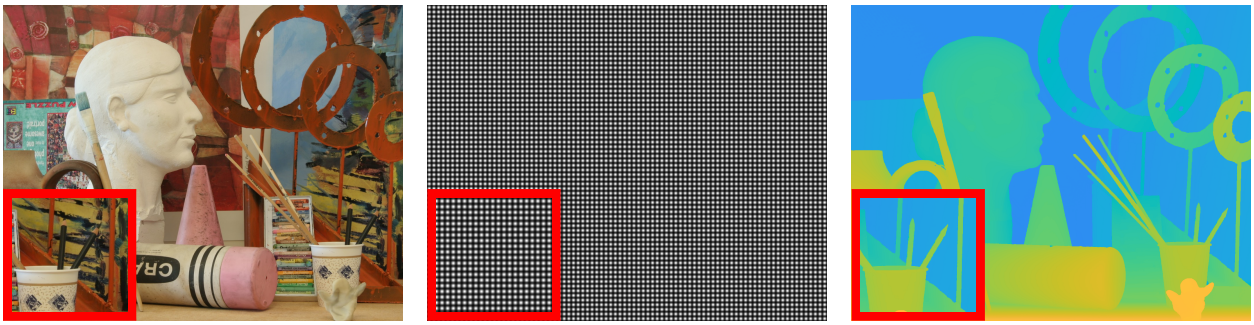
$I_{\text{defocus}} \leftarrow I_{\text{num}} \,./ I_{\text{denom}}$

return I_{defocus}

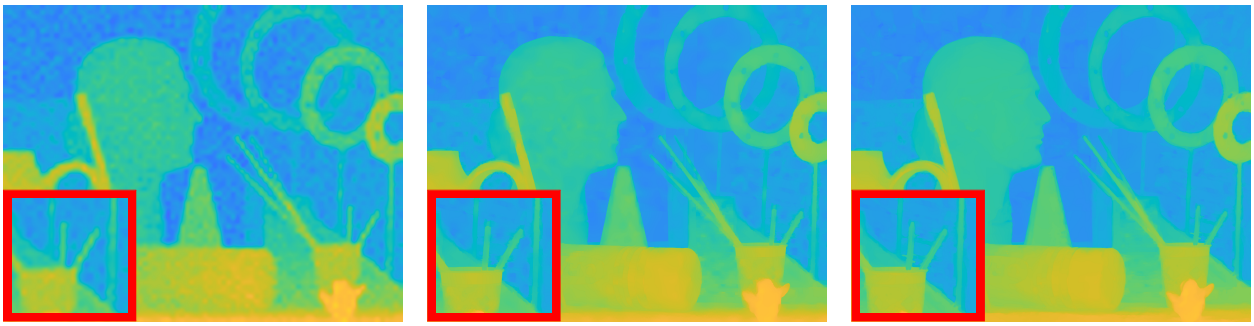
where $.*$ and $./$ are pixel-wise multiply and divide.

6.3.3 Depth super-resolution

The availability of cheap commodity depth sensors like the Microsoft's Kinect, Asus Xtion, and Intel RealSense has spurred many avenues of research, including depth super-resolution. Depth super-resolution is important for sensors like the above because, often, the color camera is of high resolution, but the depth camera or projector has low-resolution, which leads to crude depth maps (Khoshelham and Elberink, 2012). Ferstl et al. (2013) adapted the Middlebury dataset for the depth super-resolution task to create a benchmark, which we use to evaluate our method. For this task, I use a simple bicubic interpolation to upsample the low-resolution depth map and use this map as a target in the optimization; then I use the high-resolution color image to compute the domain transform based on an edge-aware mean and obtain the optimized result (Fig. 6.4e). I follow Barron and Poole (2016) by setting the confidence scores using a Gaussian bump model to represent the



(a) Color image. (b) Gaussian bump model for c. (c) GT disparity



(d) Bicubic interpolation (e) DTS (f) Barron and Poole (2016).

Figure 6.4: Depth super-resolution: (a) reference image, (b) confidence (c) ground-truth disparity, (d) bi-cubic interpolation (target), (e) the optimized disparity using DTS, and (f) results using FBS obtained from the author's website (Barron, 2008). The inset highlights the details and the amount of smoothness we obtain in homogeneous regions while being edge-aware.

contribution of each pixel to the nearby upsampled pixels. I do not use additional penalties in Eq. (6.1) for this task in the form of Φ_m.

6.3.4 Colorization

Levin et al. (2004) presented a method to convert a grayscale image into a color image using a few color strokes as input; see Fig. 6.1a for the result obtained using DTS. Instead of an approach specifically developed for this task (Yatziv and Sapiro, 2006), I show the generalizability and efficiency of DTS by applying it to the colorization task achieving high quality results. Fig. 6.5a shows the grayscale image that I convert into the YCbCr color space (luma, chroma blue-difference, chroma red-difference) to extract the Y channel. I use the Y channel as our guide image to compute the bilateral weights $W_{i,j}$. Fig. 6.5b shows the images with user annotated color strokes, which I convert to YCbCr to extract their Cb and Cr channels. The input strokes act as the target with a

(a) Grayscale image (b) Marked color strokes (c) DTS (d) Levin et al. (2004)

Figure 6.5: Colorization: (a) input grayscale image (the reference image), (b) user-annotated strokes (the target image), (c) result using DTS, and (d) result of Levin et al. (2004). Note that (c) and (d) look the same with only small differences in the hair.

confidence of one, and all other confidences are zero. I then apply the DTS twice with each Cb and Cr as target to estimate the edge-aware and completely filled Cb/Cr channels.

6.4 Experiments

In the following, I will detail the quantitative evaluation of the DTS framework as well as its run time performance. All the comparisons were conducted on the same machine with a 1080 Ti GPU. When I did not have access to the code of a particular method, I have reported the numbers from the respective papers as indicated in the tables. In the following, the number of iterations for a dataset is selected as the maximum of iterations required for any image in the dataset.

6.4.1 Stereo Optimization

I use the Middlebury dataset (Scharstein et al., 2014) for quantitative evaluation for the stereo problem. Barron and Poole (2016) used MC-CNN (Zbontar and LeCun, 2016) as their initialization. For a fair comparison, I also use it as the input target disparity map. Table 6.1 shows the results for the training and testing set in the form of the mean absolute error (MAE), root mean square error (RMSE), time per megapixel (sec/MP), and time normalized by number of disparity hypotheses (sec/GD) for non-occluded regions and for all pixels. All of these values were determined by the Middlebury evaluation website, and all of the reported times include the time to calculate the MC-CNN target disparity maps. The timings for FBS and my method DTS show the additional

	Algorithm	MAE (px) no occ | all	RMSE (px) no occ | all	sec/MP	sec/GD
Training	MC-CNN	3.81 | 11.8	18.0 | 36.6	83.3	259
	MC-CNN + FBS	2.60 | 6.66	10.2 | 20.9	42.7 (126)	153 (412)
	MC-CNN + DTS (ours)	3.02 | 9.12	10.8 | 27.4	5.9 (89.2)	19 (278)
Testing	MC-CNN	3.82 | 17.9	21.3 | 55.0	112	254
	MC-CNN + FBS	2.67 | 8.19	15.0 | 29.9	28 (140)	91 (345)
	MC-CNN + DTS (ours)	3.78 | 14.6	17.6 | 43.4	10 (122)	23 (277)

Table 6.1: Performance comparison on images from the Middlebury dataset. The timing for FBS and my method DTS shows the additional time spent in processing MC-CNN (Zbontar and LeCun, 2016), and the total value in the parentheses. DTS takes a fraction of time as compared to FBS (Barron and Poole, 2016) to obtain a significant reduction in error versus MC-CNN.

(a) Color image (b) GT disparity (c) Target image (d) DTS result

Figure 6.6: Stereo optimization closeup for Middlebury Pipes scene. (a) Color image. (b) ground-truth (GT) disparity. (c) Target obtained using MC-CNN (Zbontar and LeCun, 2016), and (d) DTS result.

time spent in processing MC-CNN, with the total value in parentheses. Note that DTS has a large performance boost compared to MC-CNN with a marginal overhead in time. DTS compares favorably to FBS, especially for non-occluded pixels, while having significant computational savings. The hyperparameter values used are $\sigma_x = \sigma_y = 64px$, $\sigma_r = 0.25$ with RGB (red-green-blue) colors normalized to a range of [0,1], $\lambda = 0.99$, and $\gamma = 0.001$. These parameters were found to work best via a grid search strategy on the Middlebury training data. I ran the gradient descent algorithm for 3000 iterations in this experiment with a step size of 0.99 times the gradient to avoid numerical precision errors. Fig. 6.1 (c-e) and Fig. 6.6 show zoomed regions from the Jadeplant and Pipes scenes to highlight that the DTS method improves the target disparity maps from MC-CNN (Zbontar and LeCun, 2016) to estimate sharp depth edges.

	Art				Books				Moebius				RMSE	Time
	2x	4x	8x	16x	2x	4x	8x	16x	2x	4x	8x	16x		(ms)
Bicubic	5.32	6.07	7.27	9.59	5.00	5.15	5.45	5.97	5.34	5.51	5.68	6.11	5.94	7†
BGU (Chen et al., 2016a)	4.77	6.63	9.39	14.17	2.35	3.57	5.93	10.41	2.19	3.19	5.38	9.00	5.48	-
HFBS-w (Chen et al., 2016a)	5.06	6.94	9.42	14.21	2.57	3.83	5.95	10.41	2.36	3.39	5.40	8.96	5.67	-
DT (Gastal and Oliveira, 2011)	3.95	4.91	6.33	8.78	1.80	2.40	3.23	4.43	1.83	2.44	3.41	4.70	3.60	21†
FBS (Barron and Poole, 2016)	3.02	3.91	5.14	7.47	1.41	1.86	2.42	3.34	1.39	1.82	2.40	3.26	2.75	234†
HFBS (Mazumdar et al., 2017)	4.73	5.56	6.38	8.32	2.14	2.60	3.08	4.04	2.25	2.67	3.18	4.11	3.74	49.86
DTS $_{L1}$	3.27	4.01	5.18	7.93	1.85	2.30	3.10	4.34	1.92	2.40	3.37	4.71	3.38	18.88
DTS	3.77	4.29	5.15	7.56	1.78	2.18	2.81	3.91	1.78	2.20	2.96	4.23	3.24	19.29

Table 6.2: Performance of DTS on the depth super-resolution task.DTS is 12x faster than FBS while having comparable performance in most images, especially images with higher upsampling factors. † marks results taken from Barron and Poole (2016).

6.4.2 Depth super-resolution

I use the dataset introduced by Ferstl et al. (2013) to evaluate DTS for the depth super-resolution task. This dataset consists of three scenes (Art, Books, and Moebius) with added noise at 2, 4, 8, and 16x levels of upsampling. I used the following hyperparameter values : $\sigma_x = \sigma_y = 8 \times f \; px$ where f is the level of upsampling. I used $\sigma_r = 0.1$ with YCbCr colors normalized to a range of $[0, 1]$, $\lambda = 0.99$, and 10 iterations of the gradient descent with a step size of 0.99. In Table 6.2, I present the RMSE for each scene and mean geometric error for the dataset. The bicubic, DT and FBS results were produced by using the data and code provided by Barron (Barron, 2008) (marked with † in Table 6.2). I also used the same code to evaluate my method. Both DTS and the FBS use the bicubic upsampling as the target image. The time is computed as the average over all images over 10 trials and include the 7 ms required for bicubic upsampling. DTS is 12 times faster than FBS (Barron and Poole, 2016) while achieving comparable performance on most images, especially for higher upsampling factors. The DTS is also more accurate and 2x faster than hardware-friendly bilateral solver (HFBS) (Mazumdar et al., 2017), which is one of the fastest parallel edge-aware optimizers.

I also compare against the available results for Liu et al. (2017b), who use the mean absolute difference (MAD) metric and report on 2×, 4×, and 8× upsampling. For $(r = 1, \tau = 1)$, their semi-global weighted least squares (SG-WLS) results in a MAD of 1.812, and for $(r = 4, \tau = 4)$, they report a MAD of 1.51. In comparison, the DTS has a MAD of 1.48.

(a) Grayscale image with color strokes.

(b) DTS result.

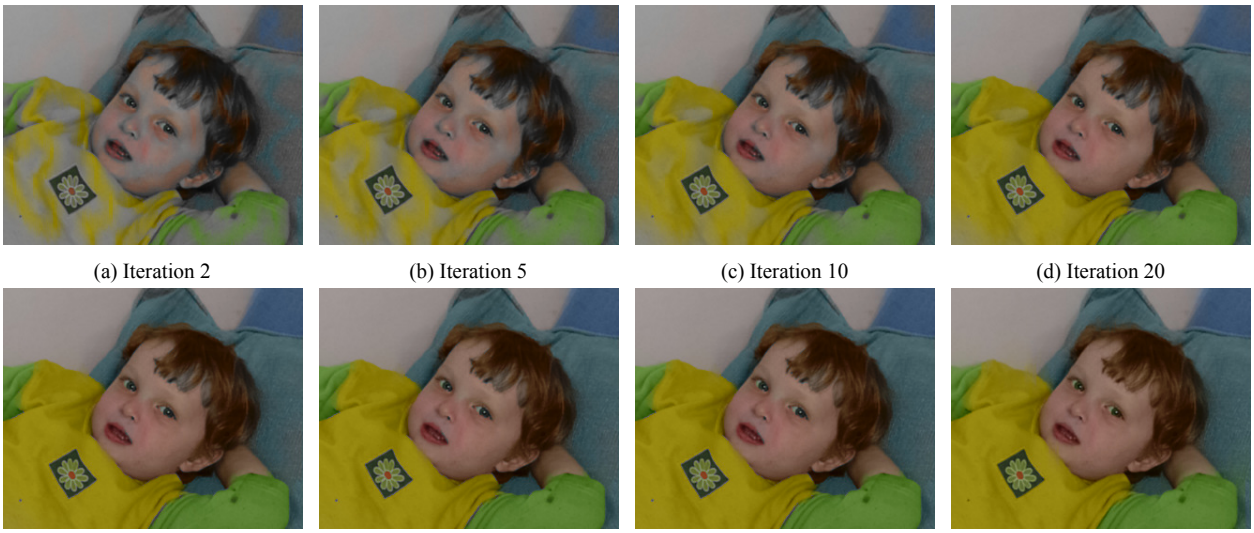
(c) Levin et al. (2004)

Figure 6.7: Colorization: (a) The grayscale image acts as the reference image and the user annotated strokes as the target, (b) DTS result and (c) Levin et al. (2004) result. Note that (b) and (c) are virtually identical.

6.4.3 Colorization

For the colorization experiment, I use the dataset introduced by Levin et al. (2004), see Fig. 6.7 for an example of grayscale image marked with color strokes which undergoes colorization. To simulate infinite confidence in the input strokes, I overwrite the solution with the strokes at each iteration of the gradient descent. I used the following parameters for this experiment: $\sigma_x = \sigma_y = 64$, $\sigma_r = 0.25$, and 100 iterations in the gradient descent scheme. Fig. 6.8 illustrates how the color propagates as a function of the iterations of gradient descent: (a-g) depicts illustrates the colorization result at 2, 5, 10, 20, 50, 70, 90 iterations, and (h) shows the final result using DTS. Note that (e-h) look the same, and further iterations produce only slight changes. This shows that one can adjust the accuracy/time trade-off and stop at 50-70 iterations if time is important. DTS performs the computation at 0.267s/MP, which is a more than 3x speedup in comparison to Barron and Poole (2016). Fig. 6.9 shows examples taken from (Levin et al., 2004) with the user annotations, our result, and results from Levin et al. (2004). All of the input images were obtained from the author's website (Yair, 2004). Note that the DTS colorization results are visually indistinguishable from theirs, yet computed significantly faster.

These colorization images are less than 1k$\times$1k in resolution and do not fully exploit the highly parallel nature of the DTS algorithm. To highlight the parallelism of DTS, 3k$\times$2k high-resolution images were captured using an iPhone 6 Plus. I sampled 20% of the pixels at random to create



(a) Iteration 2 (b) Iteration 5 (c) Iteration 10 (d) Iteration 20

(e) Iteration 50 (f) Iteration 70 (g) Iteration 90 (h) DTS result

Figure 6.8: Propagation of colors: (a-h) shows how the colors from the user-annotated strokes propagate through the image while not crossing strong image edges using DTS. After iteration 50, the changes are small and can be ignored as a trade-off for speed.

a target image, (Fig. 6.10a). Since these images are of high resolution, in addition to DTS result (Fig. 6.10b), I also show results using a parallel implementation of HFBS (Mazumdar et al., 2017) which matches the DTS's peak signal-to-noise ratio (PSNR) but is 3.64x slower than DTS (Fig. 6.10c). I will refer to this case as HFBS(equal quality). To explore the quality/speed trade-off for HFBS I lowered the number of iterations to match DTS's run time and then examine the quality of the output of HFBS result (Fig. 6.10d). I will refer to this as HFBS(equal runtime). This results in visually noticeable artifacts, and this deterioration in quality is also reflected by lower PSNR and structural similarity (SSIM) (Wang et al., 2004) scores for the Cb and Cr channels, as seen in Table 6.3. In the table, I report the SSIM and PSNR scores for DTS for the above three cases. The DTS results and HFBS (equal quality) have similar SSIM and PSNR results. This is also reflected in visually indistinguishable results in (Fig. 6.10c), (Fig. 6.10b), Fig.(6.11 b,c) and Fig.(6.12 b,c), but HFBS is much slower. On comparing the DTS results with HFBS (equal runtime) in Fig.(6.10d), one can notice speckles and spots, zooms of which are shown in Fig.(6.11 b,d) and Fig.(6.12 b,d).

(a) User annotated scribbles (b) DTS (ours) (c) Levin et al. (2004)

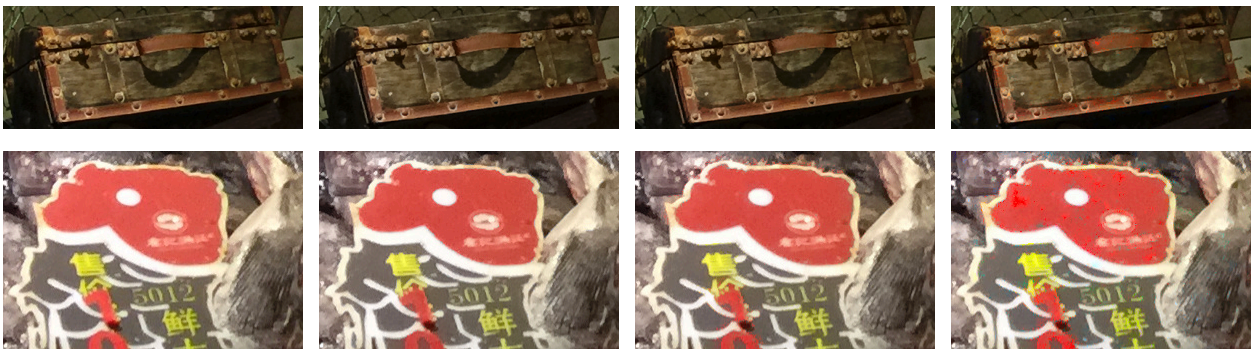
Figure 6.9: Colorization: (a) user-annotated color scribbles, (b) DTS results, and (c) results using Levin et al. (2004).

	SSIM (Cb, Cr)		PSNR (Cb, Cr) dB		Time ms/megapixels
	Fig.(6.11)	Fig.(6.12)	Fig.(6.11)	Fig.(6.12)	
DTS	0.981, 0.983	0.979, 0.975	45.41, 45.60	44.65, 44.00	19.10
HFBS equal quality	0.979, 0.976	0.973, 0.965	45.79, 45.01	45.10, 44.20	69.54
HFBS equal runtime	0.948, 0.944	0.915, 0.884	43.60, 42.91	42.30, 41.18	19.09

Table 6.3: The table lists the SSIM and PSNR scores for Cb anc Cr channels for the high-resolution images. HFBS (Mazumdar et al., 2017) can match the performance of DTS but, takes 3.64x more time. If the time taken by HFBS is restricted to that of DTS, HFBS has worse PSNR and SSIM scores.

(a) 20% randomly sampled color. (b) DTS (c) HFBS, equal PSNR. (d) HFBS, equal time budget.

Figure 6.10: Colorization for high-resolution images. (a) The target images retain color in only 20% of the pixels. (b) DTS result. (c) HFBS (Mazumdar et al., 2017) can match the performance of DTS but, takes 3.64x more time. (d) When HFBS is limited to the time of DTS, it has substantially worse PSNR and SSIM scores.



(a) Ground truth. (b) DTS (c) HFBS equal quality (d) HFBS equal time

Figure 6.11: Zoom-ins for Fig. 6.10 first row: Notice the speckles in (d) on the handle of the chest and red portion of the meat.

(a) Ground truth. (b) DTS (c) HFBS equal quality (d) HFBS equal time

Figure 6.12: Zoom-ins for Fig. 6.10 second row: Notice the speckles in (d) reddish spots on the orange, and red spots on the vase.

6.4.4 Synthetic defocus from depth

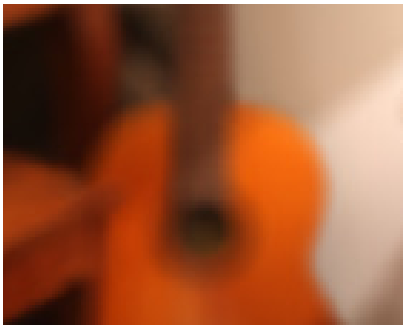
A good synthetic defocus effect requires the estimated depth to align well with the color edges. Figs. 6.13–6.15 show near- and far-focus renderings using depthmaps refined using DTS by processing predictions from MC-CNN (Zbontar and LeCun, 2016) for scenes from the Middlebury dataset (Scharstein et al., 2014). The ground-truth defocus rendering was created using the ground-truth depth. The Figs. 6.13–6.15 show that the DTS optimization framework removes any jarring artifacts and ringing and creates visually pleasing synthetic defocus. These artifacts are the most noticeable at object boundaries and occur due to inaccurate depth estimates in the MC-CNN defocus results.

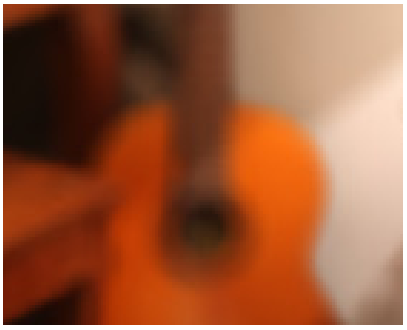
6.4.5 High-resolution stereo

When working with high-resolution data, one can truly exploit the high level of parallelism of DTS optimization framework. To demonstrate the parallelism of our approach, I tested DTS on high-resolution depthmaps generated using the COLMAP 3D reconstruction software (Schönberger and Frahm, 2016). For this task, I created a dataset of 6000×4000 px images using a Nikon D5300 DSLR. During COLMAP's processing, I had to downsample the image sizes to 4500×2945 px after radial undistortion due to memory limits on the GPU. I used COLMAP to generate a dense depthmap per image of the dataset, which acted as target for DTS. The confidence scores were calculated according to the variance in target depth, see Section 6.2.1 for details. The raw noisy depthmaps are refined to obtain filtered depthmaps, which are visualized in Fig. 6.16 in the form of point clouds. As DTS can optimize the depths in parallel, it can achieve a high throughput. In particular, DTS requires 0.0098s/MP to process the high-resolution imagery of this dataset. In contrast, FBS is difficult to parallelize due to the use of a sparse color grid which introduces non-coherent memory access and a hierarchical preconditioner that has interdependent gradients across multiple optimization variables (Mazumdar et al., 2017). I use the following hyper-parameters for this experiment: $\sigma_x = \sigma_y = 64$, $\sigma_r = 0.25$, $\sigma_c = 32$ and used 10 iterations for the gradient descent scheme.

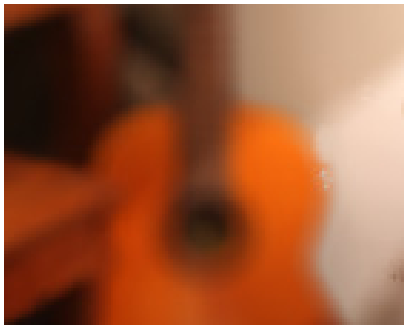
(a) DTS

(b) MC-CNN





(c) Ground truth, near focus.

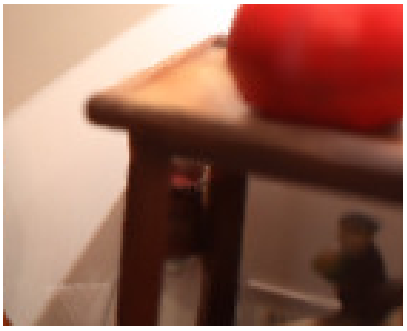


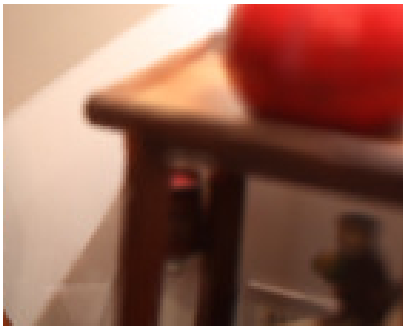
(d) DTS

(e) MC-CNN

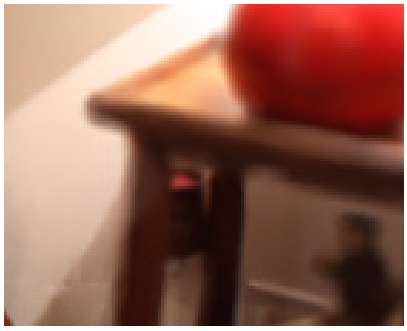
(f) DTS

(g) MC-CNN





(h) Ground truth, far focus.



(i) DTS

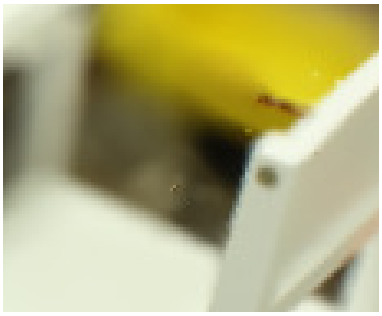
(j) MC-CNN

Figure 6.13: PianoL scene: (a,b) DTS and MC-CNN's result where the stool is in focus, (c-e) shows a zoomed-in region. (f,g) The guitar is in focus. In (e) and (j), notice the rough edges at the right side of the guitar and on the left side of the leg of the table, which are reduced in the DTS results (d,i).

(a) DTS

(b) MC-CNN

(c) Ground truth, near focus.

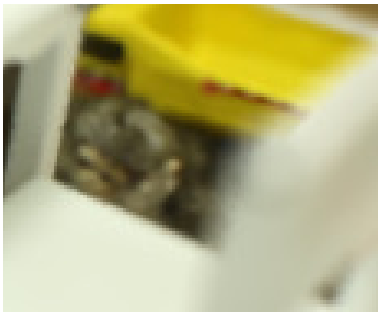


(d) DTS

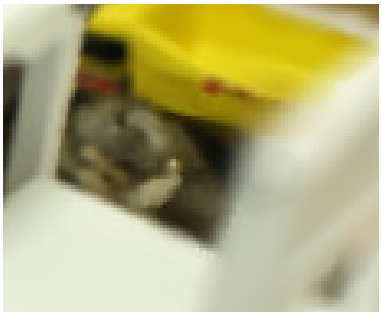
(e) MC-CNN

(f) DTS

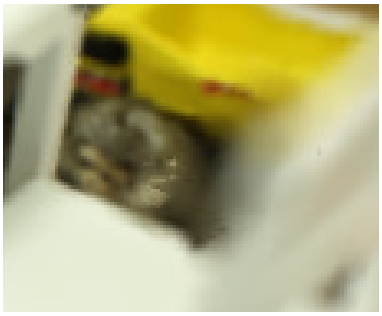
(g) MC-CNN



(h) Ground truth, far focus.

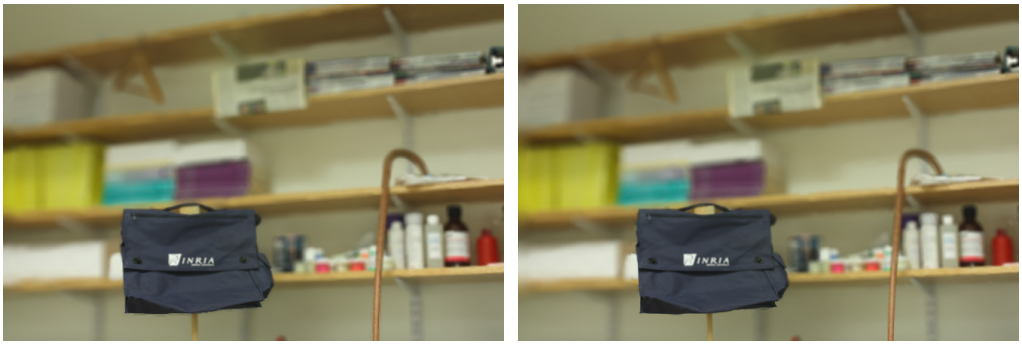


(i) DTS



(j) MC-CNN

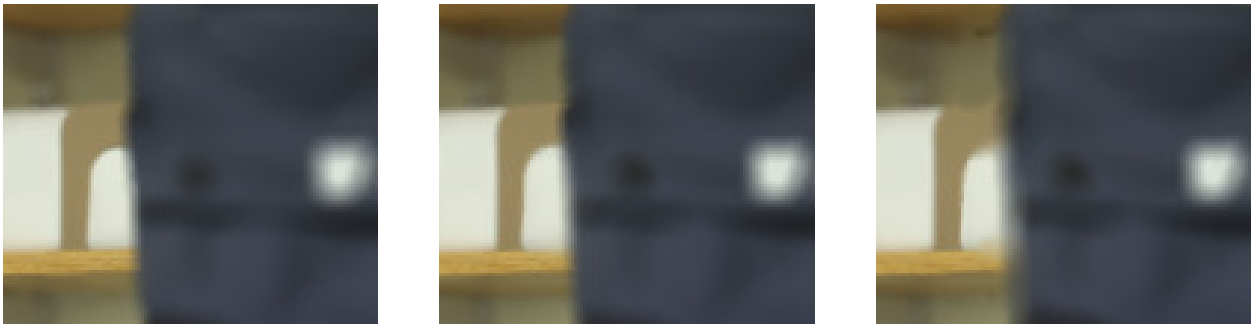
Figure 6.14: PlaytableP scene: (a,b) DTS and MC-CNN's result where the front of the table is in focus, (c-e) shows a zoomed-in region. (f,g) The orange bucket is in focus. In (e) and (j), notice the jarring changes in blur due to incorrect depth, which is reduced in the DTS results (d,i).



(a) DTS (b) MC-CNN

(c) Ground truth, near focus. (d) DTS (e) MC-CNN

(f) DTS (g) MC-CNN



(h) Ground truth, far focus. (i) DTS (j) MC-CNN

Figure 6.15: Shelves scene: (a,b) DTS and MC-CNN's result where the blue bag is in focus, (c-e) shows a zoomed-in region. (f,g) The hanger is in focus. In (e) and (j), notice the ringing near the boundary of the bag, which is removed in the DTS results (d,i).

(a) Color images.

(b) DTS

(c) COLMAP raw geometric depthmap visualization.

Figure 6.16: High-resolution stereo data: (a) Color images from the toy dataset, (b) point cloud visualization of results from DTS, and raw depth generated by COLMAP (Schönberger and Frahm, 2016) in (c). The point cloud views highlight that a significant amount of noise present in the raw depthmap is removed using DTS.

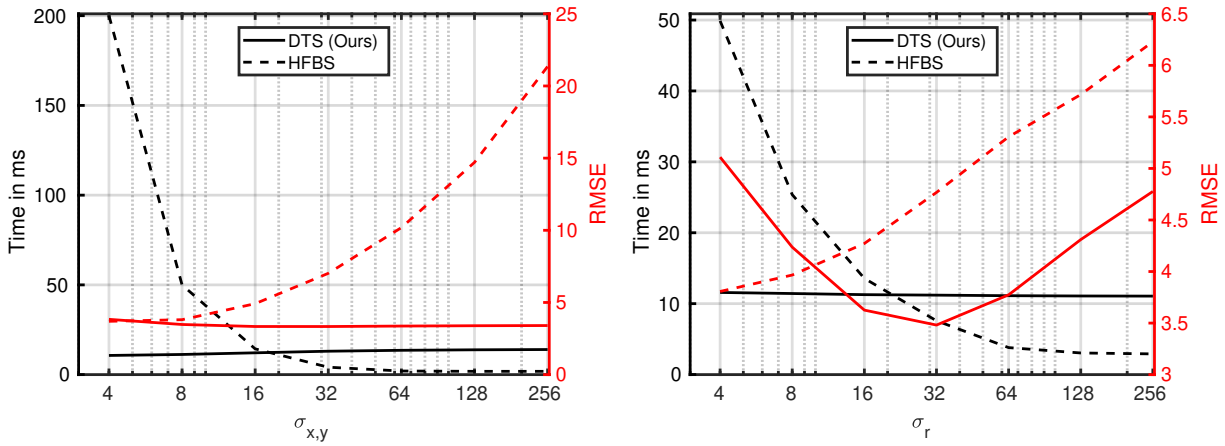
(a) DTS result: 9.8ms/MP (b) HFBS: 10.3ms/MP (c) HFBS: 36ms/MP

Figure 6.17: High-resolution stereo data comparison: DTS is more accurate than HFBS for same time budget, and faster for equal accuracy demands.

On comparing with HFBS for the high-resolution MVS experiment, one can observe the same pattern of DTS being more accurate for the same time budget, and faster for equal accuracy demands, as observed in the colorization experiment. Fig. 6.17(b) shows HFBS result with same time budget but is still very noisy compared to DTS result in Fig. 6.17(a), while Fig. 6.17(c) is equally accurate as DTS, but is 3.6× slower.

6.4.6 Benchmarking

I benchmark my parallel approach DTS using the depth super-resolution dataset and compare against a parallel implementation of HFBS. As HFBS relies on a grid, its speed is dependent on the grid size and, in turn, the blur kernel size used to determine the voxel sizes. For an image with N pixels and D_i possible values in each dimension i, HFBS, like any other grid based method (Paris and Durand, 2006; Durand and Dorsey, 2002; Barron and Poole, 2016), partitions the d-dimensional space into a grid using the blur radii $\{r_i\}$. For HFBS, $d = 3$, but in general, it spends $\mathcal{O}(\prod_i^d (D_i/r_i))$ time alone to create the grid. Because of this, one can observe an exponential increase in runtime as the $\sigma_{x,y} = r_i$ decrease, see Fig. 6.18a. On the other hand, run-time for DTS is independent of r_i, and hence the time is constant. At the same time, DTS is more accurate than HFBS. I evaluate the runtime and RMSE for $\sigma_{x,y} = 8, 16, 32, 64, 128, 256$, and keep the remainder of the parameters the same as in the earlier experiments. The time is averaged over 10 trials. A similar pattern emerges when I change σ_r with a fixed $\sigma_{x,y} = 8$ (Fig. 6.18b).



(a) Dependence of time (black) and RMSE (red) on $\sigma_{x,y}$.

(b) Dependence of time (black) and RMSE (red) on σ_r.

Figure 6.18: (a) DTS is independent of blur σ and remains more accurate than HFBS (Mazumdar et al., 2017).

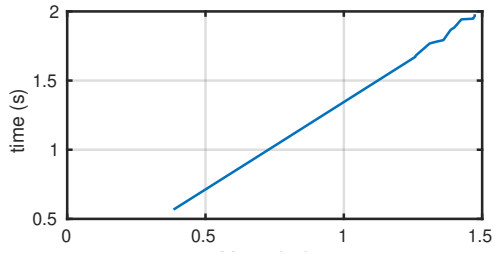


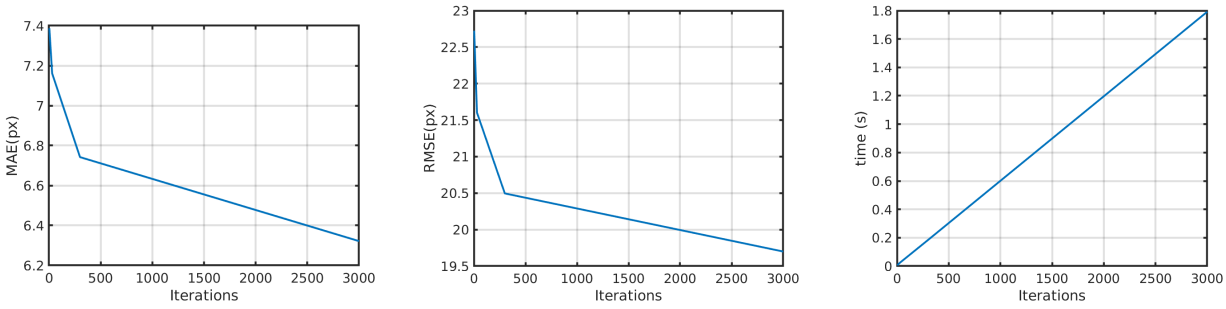
Figure 6.19: Dependence on image resolution for DTS algorithm.

Scale

Next I show how DTS scales with increasing image resolution. In practice, DTS scales linearly with the number of pixels in the image. Fig. 6.19(a) shows the dependence of time in seconds on the number of pixels in the image. I use the training images from the Middlebury dataset and only show the time consumed by DTS for the stereo task at 3000 iterations. Due to this linear dependence, the DTS optimization framework is also suitable for high-resolution imagery. Not that this experiment was carried on a GTX 980 GPU.

Iterations

The number of iterations in the gradient descent scheme affects the accuracy. Here, I show how changing the iterations affects the accuracy for the training images of the Middlebury dataset.



(a) MAE pixel error (b) RMSE pixel error (c) Avg. time(s) vs iterations

Figure 6.20: Dependence of MAE, RMSE and time on the number of iterations. (a-b) The errors reduce quickly for the first 300 iterations and then reduces gradually for further iterations. The average time taken increases linearly, as shown in (c). This provides a trade-off which can be chosen according to the application.

Fig. 6.20 (a) and (b) show the MAE and RMSE, which I report for the training set for all the pixels in the dataset. The 3000 iterations result is the same as the shown in Table 6.1, but the accuracy numbers are different because the Middlebury evaluation internally uses a non-public weighting scheme, which is not used here. Both MAE and RMSE reduce very quickly at approximately 300 iterations, and after that the gains are smaller. The time taken for iterations is linear – see Fig. 6.20(c). This allows us to easily trade off resolution, quality, and run time depending on the application.

6.5 Conclusion

I have presented a novel edge-aware solver called DTS that achieves scalable performance across a variety of applicable tasks. The proposed method is faster by an order of magnitude compared to the state of the art while performing at comparable accuracy. The approach is highly parallelizable and scales well w.r.t image resolution and outshines at high image resolutions. Furthermore, unlike existing methods, it is independent of blurring kernel size. The framework is faster and accurate than previous parallel edge-aware alglorithms. To extend the DTS, a future step is to move to multi-GPU setting, as well as use advanced optimization methods like conjugate gradient descent to obtain faster convergence.

CHAPTER 7: CONCLUSION

This dissertation focused on high-frequency and efficient computer vision algorithms useful for AR/VR. I presented a prototype high-frequency tracking system for AR/VR and a parallelizable and efficient edge-aware optimization algorithm useful for AR/VR content creation, 3D data filtering and many other applications.

A high-frequency tracking system is essential for the success of AR/VR. The proposed high-frequency tracker exploits two natural camera characteristics traditionally viewed as artifacts, rolling shutter and radial distortion. Chapter 4 shows how rolling shutter can be refashioned as a virtue for tracking. The dense temporal sampling due to the rolling nature of the shutter grants us the ability to estimate a pose per-row rather than a pose per-frame. This breaks the implicit frame-rate barrier of any camera-based tracking system, and enables intra-frame motion estimation. The high frequency of the tracker enables a linear-motion model assumption simplifying the model and hence reducing the computation, and in turn allows the high frequency of tracking. The proposed tracker uses multiple RS cameras to track the head-pose at frequencies orders of magnitude faster than state of the art, further bridging the gap between the existing technologies and the holy grail of tracking. Chapter 5 describes that radial distortion of the lens is yet another 'flaw' of the camera that can be re-purposed to the advantage of tracking. Leveraging the radial distortion, I describe a method to drastically reduce the number of cameras required for tracking in the multi-camera cluster.

Finally, Chapter 6 introduced a highly parallelizable edge-aware optimization framework called DTS and shows its effectiveness on multiple tasks like stereo, colorization, depth-super resolution and render from defocus. DTS incorporates priors and optimizes an objective function while respecting color edges, unlike plain filtering techniques. DTS is efficient, accurate, deployable on

GPUs, scales linearly with number of pixels and is independent of blur sigmas, all of which are desirable qualities not found together in a single solver in the previous work.

CHAPTER 8: LIMITATIONS AND FUTURE DIRECTIONS

Through extensive experimentation, I have shown the effectiveness of the high-frequency tracking methods in Chapters 4 and 5 and that of the efficient edge-aware optimization algorithm in Chapter 6. The work in this dissertation has limitations which can be improved upon in various ways for future research, some of which are outlined here. Accompanying the promising research direction, I have also provided the biggest hindrances that I think might hamper the progress.

8.1 High-frequency tracking

This thesis outlines a prototype method to address high-frequency tracking. Although, this method is quite successful in tracking at high-frequencies, following are the limitations and directions of research we can build upon it in the future. I will first address the limitations of the proposed approaches and then describe the future research directions one can explore specific to my system. The high-frequency tracker proposed in this thesis should be realized using using RS cameras providing row-level synchronization. Towards, that end I have provided a sketch design assuming such a RS synchronization is available, and I discuss promising camera candidates and hardware useful for building such a system. If such a RS synchronization is not possible, another direction is to explore more complex lenses on a sing RS camera. Using a multi-lens on a RS camera effectively creates virtual cameras which are implicitly synchronized, and I will describe this envisaged tracker.

There are general issues in the space of high-frequency tracking, which still remain unsolved. An ideal tracking system functions well in all environments, has low-latency and high-frequency, is robust and is ergonomically suitable for daily wear. Welch and Foxlin (2002) argue that there is no silver bullet to realizing such a system. To fulfill the qualities of the ideal tracking system, we need

a better understanding of how much latency and high-frequency is required for comfortable daily wear. Additionally, we need to design robust systems which work outdoors as well indoors, and are robust to changing lighting conditions and dynamic elements in the scenes. Moving forward, we need to also recognize that multiple users are going to use AR/VR in the same space. Following the old adage '*The whole is greater than the sum of its parts*', an interesting research direction is how to complement and help multiple trackers, where instead of interfering, the trackers help each other. I describe all of these themes in more detail in the following.

8.1.1 Row-Matching

The high-frequency tracking algorithms depend upon estimation of pixel shifts using the binary descriptors. Since the binary row-descriptor uses the image gradient, it is implicitly assumed that the scene has sufficient texture. Hence, the tracker will fail in a completely textureless scene. If a user is situated in a completely textureless room, *e.g.* a completely white room, an interesting question to ask is 'Is tracking even necessary in such a case since the user also cannot localize visually?'. While the user has no discriminating visual signal, the vestibular system which provides humans with sense of position and orientation (proprioception), other signals like the sense of position due to sound reverberations and the self-awareness about the movement of the head/neck before we tell the brain to move, all should indicate that the head has (will) move. A simple example of such human sense is evident in the children's game 'Blind man's buff' where a blind-folded person is rotated to confound their sense of direction and is asked to catch the other players. In that game, although we lose the sense of absolute position and orientation, we do retain relative sense of position and orientation. This intuitively warrants head-pose tracking for AR/VR, although the precise requirements in terms of accuracy need to be studied by conducting user studies.

8.1.2 Frame overlap and occlusion

Additionally, the success of row matching assumes that the same 3D point is observed in the two row-images. This assumption is violated when the 3D point is occluded by a foreground ob-

ject. Occlusion depends upon the combination of the camera motion and scene geometry and is difficult to avoid. There are no certifiable guarantees that the matching is correct and occlusion has not happened. An incorrect match due to occlusion cannot be disambiguated by reconstructing the surrounds and then ray-casting since we need to know the motion. Additionally, the motion compensation technique based on homography-based warps (Section 4.3.4 and Section 5.3.1.2) rely on the fact that the current row is previously observed in the past image data. The defense used in the proposed methods against such incorrect matches is to modulate the confidence scores according to the smoothness of the estimated shift and the as well as to used robust filters. Additionally, the hope is that the redundant cameras observe the motion, even if some of the cameras in the cluster face strong occlusion or textureless regions. For robustness, one direction to explore is to use a RANSAC framework to estimate the minimal solution by randomly selecting the constraints. Furthermore, a bunch of scan-line scan be used instead of a single row for increases robustness at the cost of some latency.

8.1.3 Dense sampling of linear segments

The virtual RS cameras induced by the radial distortion correspond to a linear line-segment in the undistorted space (refer Section 5.3.1). Using the constraints induced by the virtual cameras, a over-determined linear system is solved to estimate the head motion. The natural extension is the sample the line-segments densely, effectively creating a segment per pixel of the row. The use of such approach is akin to dense variants of SLAM.

8.1.4 Drift correction

The proposed approaches drift due to the absence of any drift correction mechanism. The SLAM systems use a keyframe-based framework to reduce such drift where the relative poses between keyframes are optimized (Kümmerle et al., 2011). The presented methods will benefit by graph-based optimization scheme to reduce the accrued drift. It needs to investigated how often such drift correction is required and what is a ‘good’ keyframe in the presence of RS.

8.1.5 Exposure length

The exposure of the row effectively blurs the intra-row motion for the t_e time for which the row is exposed. This is the same as running a low-pass filter of width t_e on the observations and in turn the observed motion. For a 120 Hz camera, the maximum exposure length is 8.33 ms which is too long for the latency requirements for AR/VR. The requirements for tracker latency is given by the motion-to-pose latency which is much shorter than the motion-to-photon latency (MPL). The MPL is in the range of a few milliseconds (Zheng et al., 2014). Hence, the RS camera should be exposed for much smaller exposure times to reduce the use of stale pose information. This implies that the tracker better in well-lit scenes. This motivates further study into the effect of exposure length on the tracker latency.

8.1.6 Hardware implementation: A sketch

The next step to build upon this thesis is to implement the system in hardware. The following provides a sketch to achieve this objective. First we need a RS camera which provides hardware row-level sync as well as access to row-level pixel data with minimal latency. Suggested cameras are OV9740, OV6946, OV6948 and OV6922 all by OmniVision, with the most promising being OV6930 and the Iris 15 by Teledyne Photometrics. Additionally, we need a field-programmable gate array (FPGA) or an application-specific integrated circuit (ASIC) board capable of receiving multiple video streams from multiple cameras and logiADAK Zynq-7000 SoC by Xilinx seems to be promising for this purpose. With this hardware in mind, an implementation as shown in Fig. 8.1 is possible. The n cameras in the cluster are arranged in a master-slave fashion where the master camera is triggered by the FPGA. All the cameras synchronously capture a row and send out the row-pixel data as soon as it is read from the sensor to the FPGA for further processing. The FPGA deploys the high-frequency tracking algorithm to produce pose updates.

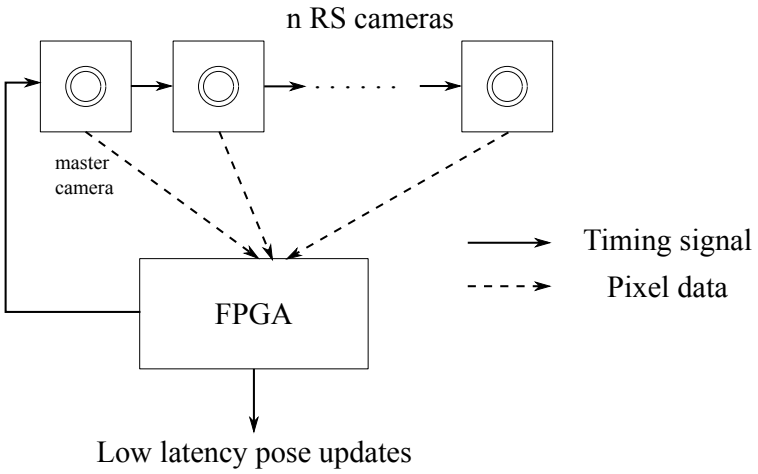


Figure 8.1: The FPGA sends out control signal to master camera, which in turn drives the exposure in the n RS of the camera cluster. The cameras in turn send a row of the image as soon as the row is readout from the sensor which is processed by the FPGA to give high-frequency and low-latency 6 DoF pose updates.

8.1.7 Tracking using multi-lens arrays

Bishop and Fuchs (1984) used a barrel lens to project the entire scene onto a line-camera in their tracker. Motivated by this use of a barrel lens, the work described in Chapter 5 and micro-lens arrays typical of light-field cameras, an interesting direction to pursue is to use a more complex lens on a single RS camera. As noted before, one of the main challenges to multi-camera tracker is obtaining row-level sync. This constraint can be overcome by using a multi-lens array which focuses large overlapping chunks of the scene onto individual rows of the camera using barrel distortion. One such design is shown in Fig. 8.2. Each smaller rectangle is a barrel lens with large FoV along the vertical direction, but small FoV along horizontal direction. This in effect is an array of virtual cameras which have synchronous row-exposures. Additionally, the barrel distortion increases the signal-to-noise ratio (SNR) for the matching of the row-descriptors. Hence, a single camera can provide synchronous observations, albeit with small baseline. The error in depth is inversely proportional to the baseline (Gallup et al., 2008), and the small baseline in the observations can be a cause of concern. Moreover, this method can be aided by using an IR projector system similar to Microsoft's Kinect so that the tracker works in texture-less environments as well.

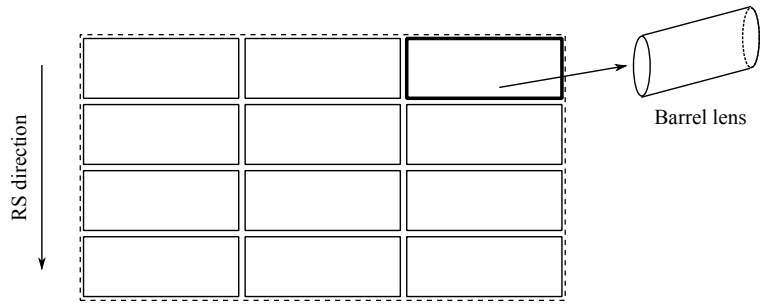


Figure 8.2: The multiple barrel lenses focus large parts of the scene on a subset of the rows increasing the descriptiveness of the rows as well as provides row-level sync for free across the lenses.

8.1.8 Tracker latency requirements and measurement

There has been a lot of research to quantify the typical daily human motion (Bussone, 2005; List, 1983) as well as extreme motion, *e.g.* in an American football game (Rowson et al., 2009), what are the latency requirements for tracking such range of motions (Bailey et al., 2004; Jerald, 2010; Jerald and Whitton, 2009) and how to measure accuracy of trackers (Allen, 2007; Vorozcovs et al., 2006; Wiersma et al., 2013; Nafis et al., 2006; Pustka et al., 2010). Although it is clear that higher tracking frequency and lower latency is required (Pausch et al., 1992), there is no definitive answer as to how much frequency is enough. This in part is due to the fact that we need high-frequency trackers to measure physiological response at such low latency and high motion in the first place. The tracker by Blate et al. (2019), although limited in tracking volume and free movement, is quite useful for obtaining such measurements. There is a need to measure how much MPL is required, and in particular how much latency is acceptable for tracking systems. One measure to quantify this is the just noticeable difference (JND) for visual acuity in static as well as dynamic environments.

8.1.9 Generalizability and robustness

The holy grail of tracking system has at least the following qualities: 1. works inside closed spaces as well as outdoors, *i.e.* is generalizable, 2. low-latency and high-frequency, 3. robust to sudden changes and to dynamic objects in the environment, and 4. ergonomics: is low-power, tether-less and has form-factor of eye-glasses. Achieving all of these criteria in a single tracker is often treated as a dream (Bishop et al., 1995; Welch and Foxlin, 2002), and rather we are always

in a multi-dimensional trade-off space. Out of these, the limited generalizability and robustness available in the current systems are the major hindrances to widespread use of AR/VR devices. Multiple issues affect the generalizability and robustness of the tracker, *e.g.* high-dynamic range of the scene outdoors vs. indoors, depth variability and complexity of the scene, range of possible human motion and dynamic objects in the environment, to name a few. Improving on these issues is a promising direction of research.

8.1.10 Social tracking

Social tracking is the natural extension to high-frequency tracker, where multiple users of AR/VR are situated in a common physical space and share data to improve tracking. When multiple users share a space simultaneously, robustness against dynamic objects and occlusions becomes crucial. Not only do we need to resolve interference among trackers, *e.g.* HTC Vive lighthouse tracking interferes with the Hi-Ball tracker, but we need to design trackers which can collaborate, resulting in better overall robustness. The most straight-forward way to achieve this is to jointly share and update the 3D model of the shared space for a SLAM system (Kim et al., 2010; Andersson and Nygards, 2008). Such a sharing of maps is also beneficial even if the users are not co-located at the same time. As a new user explores the 3D space, they can reuse previously captured data by some other user. The major challenge in this envisaged use case is how to fuse observations from different sensors and modalities which are captures asynchronously. This asynchronicity can be helpful to get even higher tracking frequencies via the use of the single-constraint-at-a-time (SCAAT) algorithm, which allows the individual sensors to capture at lower rates without compromising on tracking rates. Additionally, the trackers should be able to model any errors caused by the presence of the other users and interference caused by their tracking devices.

One example of tracking systems helping each other is when the users are co-located at the same time, and if the tracking systems have active components, *e.g.* lasers or IR LEDs, these can be used as additional external beacons to aid in tracking. The ultimate case of social tracking is where most people are using a AR/VR device, and some of the space is equipped with beacons.

Even if the individual trackers might be insufficient to track by themselves, all the trackers present in the shared space together enable tracking for all. Communication overhead introduced by the data sharing required for enabling the above is a another separate challenge.

8.2 Edge aware optimization

The edge-aware optimization problem described in Chapter 6 utilizes the reference (guide map) which is defined on a regular pixel grid, to define edges in N-dimensional space. Using the edges, the solver minimizes a cost function, where the input target is an initial solution. The work on efficient edge-aware optimization can be advanced further by generalizing each of these characteristics of the problem. The method can be extended to 1. different domains on which the reference (guide map) is defined, 2. what is being optimized/estimated, 3. how to characterize or define an 'edge', and 4. how to apply edge-aware optimization to other domains. Each of these characteristics of the problem is considered next.

8.2.1 Irregular domains

The domain for 2-D images is a regular pixel grid, limiting the reference to be dense and regular. Often, the reference is irregular, sparse or is a surface/manifold in a higher dimensional space (Gastal and Oliveira, 2012). 3D modeling and reconstruction is such an example of a surface, where most of the space is empty. One can extend the work presented in Chapter 6 by adding additional *sparse* dimension to model depth, effectively creating a hybrid domain with sparse and dense dimensions. This can be further extended to general graphs, where neighborhood of vertices (pixels) are defined by adjacency lists.

Additionally, an irregular grid is useful for real-time applications like autonomous driving where we need to trade-off speed vs. time. In real-time scenarios, we can preferentially process different regions of an image at different resolutions to save time, *e.g.* the road is processed at higher resolution and sky at lower resolution, by modeling the image as an irregular grid. Another

example of irregular grid is the equi-rectangular image used to represent 360° images. Tateno et al. (2018) highlight the need to model the unequal projection of the pixels in equi-rectangular image.

The above examples have irregular grids which are static across time. Often, it is useful to apply a variable blur across time, *e.g.* in a video, we might want to aggressively blur static regions than dynamic regions, resulting into a dynamic irregular grid.

8.2.2 Alternating optimization

The initial target is the result another optimization function, *e.g.* depth in stereo problem. One interesting direction of research is when the reference (guide map) itself is the product of another optimization. For such a scenario, a solution is to alternate between four optimization problems by interchanging the role of target and reference: 1. estimate target, 2. edge-aware refine target with constant reference, 3. estimate reference, and 4. edge-aware refine reference with constant target. This alternating approach simultaneously estimates the reference and the target which are aligned along their edges. This is useful in simultaneous refinement of geometry and texture (Bódis-Szomorú et al., 2015).

8.2.3 Semantics and CNN

Many machine learning tasks require pixel-level accurate predictions, *e.g.* semantic segmentation, instance segmentation or masks, depth estimation and image synthesis (Chen et al., 2016c). FBS and DTS both can be embedded inside CNNs as a layer. These layers can either serve as refinement on top of the prediction to make it edge-aware, or they can process deep features in an edge-aware sense to create better feature representations. But, it is unclear whether CNNs need such edge-aware layers as refinement. In some cases CNNs produce high-quality results which are aligned along edges (He et al., 2017) and sometimes fail completely (Godard et al., 2017), which leaves a promising research direction to pursue.

Another interesting direction is to investigate what qualifies as an ‘edge’. Typically, color edges are used, but depth or semantic disparities can provide stronger cues. Simply utilizing semantic

labels or depth as additional dimensions along with color-pixel 5D space is a simple extension. But, more interesting is to address if the individual semantic labels constitute additional dimensions, or blurring across labels has any meaning? For example, it does not make sense to combine labels 'sky' and 'trees', but it might be useful to combine 'chair' and 'stool' labels. The confusion matrix of the algorithm which predicts the semantic labels might be useful to study how we can combine across labels. This allows us to model complex interactions, where the dimensions of the bilateral space can be hybrid, *e.g.* 'foreground-depth', instead of two individual dimensions: 'foreground' and 'depth'.

8.2.4 Data compression

Telepresence (Würmlin et al., 2004) and streamable free view-point videos (Collet et al., 2015) require good compression of color and depth or mesh data. Since edge-aware optimization is quite accurate and efficient for depth super-resolution and colorization tasks, the natural extension is to use it in data compression. One envisaged way to accomplish this is to sample a minimal number of points in the depthmap and create a sparse target, such that upon edge-aware processing we can recreate the original dense target with minimal artifacts.

REFERENCES

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282.

Adams, A., Baek, J., and Davis, M. A. (2010). Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29 (2), pages 753–762. Wiley Online Library.

Ait-Aider, O., Andreff, N., Lavest, J.-M., and Martinet, P. (2006). Exploiting rolling shutter distortions for simultaneous object pose and velocity computation using a single view. In *Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on*, pages 35–35. IEEE.

Ait-Aider, O., Bartoli, A., and Andreff, N. (2007). Kinematics from lines in a single rolling shutter image. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE.

Ait-Aider, O. and Berry, F. (2009). Structure and kinematics triangulation with a rolling shutter stereo rig. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1835–1840. IEEE.

Albl, C., Kukelova, Z., and Pajdla, T. (2015). R6p-rolling shutter absolute camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2292–2300.

Albl, C., Kukelova, Z., and Pajdla, T. (2016a). Rolling shutter absolute pose problem with known vertical direction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3355–3363.

Albl, C., Sugimoto, A., and Pajdla, T. (2016b). Degeneracies in rolling shutter sfm. In *European Conference on Computer Vision*, pages 36–51. Springer.

Allen, B. D. (2007). *Hardware design optimization for human motion tracking systems*. PhD thesis, University of North Carolina at Chapel Hill.

Anderson, R., Gallup, D., Barron, J. T., Kontkanen, J., Snavely, N., Hernández, C., Agarwal, S., and Seitz, S. M. (2016). Jump: virtual reality video. *ACM Transactions on Graphics (TOG)*, 35(6):198.

Andersson, L. A. and Nygards, J. (2008). C-sam: Multi-robot slam using square root information smoothing. In *2008 IEEE International Conference on Robotics and Automation*, pages 2798–2805. IEEE.

Azuma, R. T. (1997). A survey of augmented reality. *Presence: Teleoperators and virtual environments*, 6(4):355–385.

Bailey, R. E., Arthur, J. J., and Williams, S. P. (2004). Latency requirements for head-worn display s/evs applications. In *Enhanced and Synthetic Vision 2004*, volume 5424, pages 98–110. International Society for Optics and Photonics.

Bapat, A., Dunn, E., and Frahm, J.-M. (2016). Towards kilo-hertz 6-dof visual tracking using an egocentric cluster of rolling shutter cameras. *IEEE Transactions on Visualization and Computer Graphics*, 22(11):2358–2367.

Bapat, A. and Frahm, J.-M. (2019). The domain transform solver. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE.

Bapat, A., Price, T., and Frahm, J.-M. (2018). Rolling shutter and radial distortion are features for high frame rate multi-camera tracking. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4824–4833.

Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. (2009). Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24.

Barron, J. (2008). https://jonbarron.info/. `https://drive.google.com/file/d/0B4nuwEMaEsnmaDI3bm5VeDRxams/view?usp=sharing`. Online; accessed 8-March-2018.

Barron, J. T., Adams, A., Shih, Y., and Hernández, C. (2015). Fast bilateral-space stereo for synthetic defocus. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4466–4474.

Barron, J. T. and Poole, B. (2016). The fast bilateral solver. In *European Conference on Computer Vision*, pages 617–632. Springer.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer.

Belsley, D. A., Kuh, E., and Welsch, R. E. (2005). *Regression diagnostics: Identifying influential data and sources of collinearity*, volume 571. John Wiley & Sons.

Bishop, B., Welch, G., and Allen, B. (1995). Tracking: Beyond 15 minutes of thought: Siggraph 2001 course 11. Technical report, Technical report, University of North Carolina at Chapel Hill.

Bishop, G. and Fuchs, H. (1984). *Self-tracker: A smart optical sensor on silicon*. PhD thesis, University of North Carolina at Chapel Hill.

Blate, A., Whitton, M., Singh, M., Welch, G., State, A., Whitted, T., and Fuchs, H. (2019). Implementation and evaluation of a 50 khz, 28μs motion-to-pose latency head tracking instrument. *IEEE transactions on visualization and computer graphics*.

Bleyer, M., Rhemann, C., and Rother, C. (2011). Patchmatch stereo-stereo matching with slanted support windows. In *Bmvc*, volume 11, pages 1–11.

Bódis-Szomorú, A., Riemenschneider, H., and Van Gool, L. (2015). Superpixel meshes for fast edge-preserving surface reconstruction. In *Proceedings CVPR 2015*, pages 2011–2020.

Boud, A. C., Haniff, D. J., Baber, C., and Steiner, S. (1999). Virtual reality and augmented reality as a training tool for assembly tasks. In *1999 IEEE International Conference on Information Visualization (Cat. No. PR00210)*, pages 32–36. IEEE.

Bradley, D., Atcheson, B., Ihrke, I., and Heidrich, W. (2009). Synchronization and rolling shutter compensation for consumer video camera arrays. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 1–8. IEEE.

Brown, D. C. (1966). Decentering distortion of lenses. *Photometric Engineering*, 32(3):444–462.

Bussone, W. (2005). *Linear and angular head accelerations in daily life*. PhD thesis, Virginia Tech.

Cadena, C. and Košecká, J. (2014). Semantic segmentation with heterogeneous sensor coverages. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2639–2645. IEEE.

Caruso, D., Engel, J., and Cremers, D. (2015). Large-scale direct slam for omnidirectional cameras. In *Intl. Conference on Intelligent Robots and Systems*.

Chen, J., Adams, A., Wadhwa, N., and Hasinoff, S. W. (2016a). Bilateral guided upsampling. *ACM Transactions on Graphics (TOG)*, 35(6):203.

Chen, J., Paris, S., and Durand, F. (2007). Real-time edge-aware image processing with the bilateral grid. In *ACM Transactions on Graphics (TOG)*, volume 26 (3), page 103. ACM.

Chen, L.-C., Barron, J. T., Papandreou, G., Murphy, K., and Yuille, A. L. (2016b). Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4545–4554.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2016c). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*.

Collet, A., Chuang, M., Sweeney, P., Gillett, D., Evseev, D., Calabrese, D., Hoppe, H., Kirk, A., and Sullivan, S. (2015). High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (ToG)*, 34(4):69.

Conrady, A. E. (1919). Decentred lens-systems. *Monthly notices of the royal astronomical society*, 79(5):384–390.

Cox, D. A., Little, J., and O'shea, D. (2006). *Using algebraic geometry*, volume 185. Springer Science & Business Media.

Dahmouche, R., Ait-Aider, O., Andreff, N., and Mezouar, Y. (2008). High-speed pose and velocity measurement from vision. In *Robotics and Automation. ICRA. IEEE Intl. Conference on*, pages 107–112.

Dai, Y., Li, H., and Kneip, L. (2016). Rolling shutter camera relative pose: Generalized epipolar geometry. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dou, M., Guan, L., Frahm, J.-M., and Fuchs, H. (2012). Exploring high-level plane primitives for indoor 3d reconstruction with a hand-held rgb-d camera. In *Asian Conference on Computer Vision*, pages 94–108. Springer.

Dou, M., Guan, L., Frahm, J.-M., and Fuchs, H. (2013). Exploring high-level plane primitives for indoor 3d reconstruction with a hand-held RGB-D camera. In *ACCV 2012 Workshops*, pages 94–108. Springer.

Durand, F. and Dorsey, J. (2002). Fast bilateral filtering for the display of high-dynamic-range images. In *ACM transactions on graphics (TOG)*, volume 21 (3), pages 257–266. ACM.

Elad, M. (2002). On the origin of the bilateral filter and ways to improve it. *IEEE Transactions on image processing*, 11(10):1141–1151.

Engel, J. and Schöps (2014). LSD-SLAM, General Notes on Good Results. `https://github.com/tum-vision/lsd_slam#316-general-notes-for-good-results`. [Online; accessed 12-November-2017].

Engel, J., Schöps, T., and Cremers, D. (2014a). LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*.

Engel, J., Schöps, T., and Cremers, D. (2014b). Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer.

Engel, J., Stueckler, J., and Cremers, D. (2015). Large-scale direct slam with stereo cameras. In *Intl. Conference on Intelligent Robots and Systems*.

Farbman, Z., Fattal, R., Lischinski, D., and Szeliski, R. (2008). Edge-preserving decompositions for multi-scale tone and detail manipulation. In *ACM Transactions on Graphics (TOG)*, volume 27 (3), page 67. ACM.

Fattal, R. (2009). Edge-avoiding wavelets and their applications. *ACM Transactions on Graphics (TOG)*, 28(3):22.

Fattal, R., Agrawala, M., and Rusinkiewicz, S. (2007). Multiscale shape and detail enhancement from multi-light image collections. In *ACM Transactions on Graphics (TOG)*, volume 26 (3), page 51. ACM.

Ferstl, D., Reinbacher, C., Ranftl, R., Rüther, M., and Bischof, H. (2013). Image guided depth upsampling using anisotropic total generalized variation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 993–1000. IEEE.

Forssén, P.-E. and Ringaby, E. (2010). Rectifying rolling shutter video from hand-held devices. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 507–514. IEEE.

Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE.

Gallup, D., Frahm, J.-M., Mordohai, P., and Pollefeys, M. (2008). Variable baseline/resolution stereo. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

Gallup, D., Frahm, J.-M., Mordohai, P., Yang, Q., and Pollefeys, M. (2007). Real-time plane-sweeping stereo with multiple sweeping directions. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

Gastal, E. S. and Oliveira, M. M. (2011). Domain transform for edge-aware image and video processing. In *ACM Transactions on Graphics (ToG)*, volume 30, page 69. ACM.

Gastal, E. S. L. and Oliveira, M. M. (2012). Adaptive manifolds for real-time high-dimensional filtering. *ACM TOG*, 31(4):33:1–33:13. Proceedings of SIGGRAPH 2012.

Ge, X. (2012). The design of a global shutter CMOS image sensor in 110nm technology. Master's thesis, Delft University of Technology, the Netherlands.

Gelper, S., Fried, R., and Croux, C. (2010). Robust forecasting with exponential and holt–winters smoothing. *Journal of forecasting*, 29(3):285–300.

Geyer, C., Meingast, M., and Sastry, S. (2005). Geometric models of rolling-shutter cameras. *arXiv preprint cs/0503076*.

Gharbi, M., Chen, J., Barron, J. T., Hasinoff, S. W., and Durand, F. (2017). Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):118.

Godard, C., Mac Aodha, O., and Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279.

Grundmann, M., Kwatra, V., Castro, D., and Essa, I. (2012a). Calibration-free rolling shutter removal. In *2012 IEEE international conference on computational photography (ICCP)*, pages 1–8. IEEE.

Grundmann, M., Kwatra, V., Castro, D., and Essa, I. (2012b). Effective calibration free rolling shutter removal. In *Computational Photography, 2012 International Conference on*. IEEE.

Handa, A., Newcombe, R. A., Angeli, A., and Davison, A. J. (2012). Real-time camera tracking: When is high frame-rate best? In *European Conference on Computer Vision*, pages 222–235. Springer.

Hanning, G., Forslöw, N., Forssén, P.-E., Ringaby, E., Törnqvist, D., and Callmer, J. (2011). Stabilizing cell phone video using inertial measurement sensors. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1–8. IEEE.

Haralick, R. M., Lee, C.-n., Ottenburg, K., and Nölle, M. (1991). Analysis and solutions of the three point perspective pose estimation problem. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 592–598. IEEE.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

Hedborg, J., Forssén, P.-E., Felsberg, M., and Ringaby, E. (2012). Rolling shutter bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1434–1441. IEEE.

Hedborg, J., Ringaby, E., Forssén, P.-E., and Felsberg, M. (2011). Structure and motion estimation from rolling shutter video. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 17–23. IEEE.

Heflin, B., Scheirer, W., and Boult, T. E. (2010). Correcting rolling-shutter distortion of cmos sensors using facial feature detection. In *2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–6. IEEE.

Hirschmuller, H. and Scharstein, D. (2007). Evaluation of cost functions for stereo matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

Horn, B. K. (2000). Tsai's camera calibration method revisited. *Online: http://people. csail. mit. edu/bkph/articles/Tsai_Revisited. pdf*.

Hu, X. and Mordohai, P. (2012). A quantitative evaluation of confidence measures for stereo vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2121–2133.

Ito, E. and Okatani, T. (2017). Self-calibration-based approach to critical motion sequences of rolling-shutter structure from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 801–809.

Jerald, J. and Whitton, M. (2009). Relating scene-motion thresholds to latency thresholds for head-mounted displays. In *2009 IEEE Virtual Reality Conference*, pages 211–218. IEEE.

Jerald, J. J. (2010). *Scene-motion-and latency-perception thresholds for head-mounted displays*. PhD thesis, The University of North Carolina at Chapel Hill.

Kalekar, P. S. (2004). Time series forecasting using holt-winters exponential smoothing. *Kanwal Rekhi School of Information Technology*, 4329008:1–13.

Kanter, D. (2015). Graphics processing requirements for enabling immersive vr. *AMD White Paper*.

Karpenko, A., Jacobs, D., Baek, J., and Levoy, M. (2011). Digital video stabilization and rolling shutter correction using gyroscopes. *CSTR*, 1:2.

Kerl, C., Stuckler, J., and Cremers, D. (2015). Dense continuous-time tracking and mapping with rolling shutter rgb-d cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2264–2272.

Khor, W. S., Baker, B., Amin, K., Chan, A., Patel, K., and Wong, J. (2016). Augmented and virtual reality in surgery—the digital surgical environment: applications, limitations and legal pitfalls. *Annals of translational medicine*, 4(23).

Khoshelham, K. and Elberink, S. O. (2012). Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454.

Kim, B., Kaess, M., Fletcher, L., Leonard, J., Bachrach, A., Roy, N., and Teller, S. (2010). Multiple relative pose graphs for robust cooperative mapping. In *2010 IEEE International Conference on Robotics and Automation*, pages 3185–3192. IEEE.

Kim, J.-H., Cadena, C., and Reid, I. (2016). Direct semi-dense slam for rolling shutter cameras. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1308–1315. IEEE.

Kim, J.-H., Latif, Y., and Reid, I. (2017). Rrd-slam: Radial-distorted rolling-shutter direct slam. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 5148–5154. IEEE.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE Computer Society.

Klein, G. S. and Drummond, T. W. (2004). Tightly integrated sensor fusion for robust visual tracking. *Image and Vision Computing*, 22(10):769–776.

Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117.

Kumar, R. K., Ilie, A., Frahm, J.-M., and Pollefeys, M. (2008). Simple calibration of non-overlapping cameras with a mirror. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE.

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE.

LaValle, S. M., Yershova, A., Katsev, M., and Antonov, M. (2014). Head tracking for the oculus rift. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 187–194. IEEE.

Lee, C.-R., Yoon, J., and Yoon, K.-J. (2018). Calibration and noise identification of a rolling shutter camera and a low-cost inertial measurement unit. *Sensors*, 18(7):2345.

Lee, C.-R., Yoon, J. H., Park, M.-G., and Yoon, K.-J. (2019). Gyroscope-aided relative pose estimation for rolling shutter cameras. *arXiv preprint arXiv:1904.06770*.

Lee, C.-R. and Yoon, K.-J. (2017). Inertial-aided rolling shutter relative pose estimation. *arXiv preprint arXiv:1712.00184*.

Leutenegger, S., Chli, M., and Siegwart, R. (2011). Brisk: Binary robust invariant scalable keypoints. In *2011 IEEE international conference on computer vision (ICCV)*, pages 2548–2555. Ieee.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334.

Levin, A., Lischinski, D., and Weiss, Y. (2004). Colorization using optimization. In *ACM Transactions on Graphics (ToG)*, volume 23 (3), pages 689–694. ACM.

Li, B., Heng, L., Koser, K., and Pollefeys, M. (2013a). A multiple-camera system calibration toolbox using a feature descriptor-based calibration pattern. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1301–1307. IEEE.

Li, M., Kim, B. H., and Mourikis, A. I. (2013b). Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera. In *2013 IEEE International Conference on Robotics and Automation*, pages 4712–4719. IEEE.

Liang, C.-K., Chang, L.-W., and Chen, H. H. (2008). Analysis and compensation of rolling shutter effect. *IEEE Transactions on Image Processing*, 17(8):1323–1330.

Lincoln, P., Blate, A., Singh, M., Whitted, T., State, A., Lastra, A., and Fuchs, H. (2016). From motion to photons in 80 microseconds: Towards minimal latency for virtual and augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 22(4):1367–1376.

List, U. H. (1983). Nonlinear prediction of head movements for helmet-mounted displays. Technical report, AIR FORCE HUMAN RESOURCES LAB BROOKS AFB TX.

Liu, F., Gleicher, M., Wang, J., Jin, H., and Agarwala, A. (2011). Subspace video stabilization. *ACM Transactions on Graphics (TOG)*, 30(1):4.

Liu, S., De Mello, S., Gu, J., Zhong, G., Yang, M.-H., and Kautz, J. (2017a). Learning affinity via spatial propagation networks. In *Advances in Neural Information Processing Systems*, pages 1520–1530.

Liu, W., Chen, X., Shen, C., Liu, Z., and Yang, J. (2017b). Semi-global weighted least squares in image filtering. In *The IEEE International Conference on Computer Vision (ICCV)*.

Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133.

Lowe, D. G. et al. (1999). Object recognition from local scale-invariant features. In *iccv*, volume 99–2, pages 1150–1157.

Lu, J., Yang, H., Min, D., and Do, M. N. (2013). Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1854–1861. IEEE.

Magerand, L., Bartoli, A., Ait-Aider, O., and Pizarro, D. (2012). Global optimization of object pose and motion from a single rolling shutter image with automatic 2d-3d matching. In *European Conference on Computer Vision*, pages 456–469. Springer.

Mazumdar, A., Alaghi, A., Barron, J. T., Gallup, D., Ceze, L., Oskin, M., and Seitz, S. M. (2017). A hardware-friendly bilateral solver for real-time virtual reality video. In *Proceedings of High Performance Graphics*, page 13. ACM.

Meilland, M., Drummond, T., and Comport, A. (2013). A unified rolling shutter and motion blur model for 3d visual registration. In *Proceedings of the IEEE Intl. Conference on Computer Vision*, pages 2016–2023.

Milgram, P., Takemura, H., Utsumi, A., and Kishino, F. (1995). Augmented reality: A class of displays on the reality-virtuality continuum. In *Telemanipulator and telepresence technologies*, volume 2351, pages 282–293. International Society for Optics and Photonics.

Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572. IEEE.

Nafis, C., Jensen, V., Beauregard, L., and Anderson, P. (2006). Method for estimating dynamic em tracking accuracy of surgical navigation tools. In *Medical Imaging 2006: Visualization, Image-Guided Procedures, and Display*, volume 6141, page 61410K. International Society for Optics and Photonics.

Naimark, L. and Foxlin, E. (2005). Encoded led system for optical trackers. In *Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 150–153. IEEE Computer Society.

NDI (2018). Optotrak Certus Technical Specifications. `https://www.ndigital.com/msci/products/optotrak-certus/#optotrak-certus-technical-specifications`. [Online; accessed 28-March-2018].

Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011). Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE.

Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Ong, S. K. and Nee, A. Y. C. (2013). *Virtual and augmented reality applications in manufacturing*. Springer Science & Business Media.

Oth, L., Furgale, P., Kneip, L., and Siegwart, R. (2013). Rolling shutter camera calibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1360–1367.

Paris, S. and Durand, F. (2006). A fast approximation of the bilateral filter using a signal processing approach. In *European conference on computer vision*, pages 568–580. Springer.

Paris, S., Kornprobst, P., Tumblin, J., Durand, F., et al. (2009). Bilateral filtering: Theory and applications. *Foundations and Trends® in Computer Graphics and Vision*, 4(1):1–73.

Park, F. C. and Martin, B. J. (1994). Robot sensor calibration: solving ax= xb on the euclidean group. *IEEE Transactions on Robotics and Automation (Institute of Electrical and Electronics Engineers)*, 10(5).

Patron-Perez, A., Lovegrove, S., and Sibley, G. (2015). A spline-based trajectory representation for sensor fusion and rolling shutter cameras. *International Journal of Computer Vision*, 113(3):208–219.

Pausch, R., Crea, T., and Conway, M. (1992). A literature survey for virtual environments: Military flight simulator visual systems and simulator sickness. *Presence: Teleoperators & Virtual Environments*, 1(3):344–363.

Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence*, 12(7):629–639.

Persa, S.-F. (2006). *Sensor fusion in head pose tracking for augmented reality*. TU Delft, Delft University of Technology.

Pham, T. Q. and Van Vliet, L. J. (2005). Separable bilateral filtering for fast video preprocessing. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 4–pp. IEEE.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.

Porikli, F. (2008). Constant time o (1) bilateral filtering. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

Purkait, P. and Zach, C. (2018). Minimal solvers for monocular rolling shutter compensation under ackermann motion. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 903–911. IEEE.

Purkait, P., Zach, C., and Leonardis, A. (2017). Rolling shutter correction in manhattan world. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 882–890.

Pustka, D., Willneff, J., Wenisch, O., Lükewille, P., Achatz, K., Keitler, P., and Klinker, G. (2010). Determining the point of minimum error for 6dof pose uncertainty representation. In *2010 IEEE International Symposium on Mixed and Augmented Reality*, pages 37–45. IEEE.

Rengarajan, V., Balaji, Y., and Rajagopalan, A. (2017). Unrolling the shutter: Cnn to correct motion distortions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2291–2299.

Rengarajan, V., Rajagopalan, A. N., and Aravind, R. (2016). From bows to arrows: Rolling shutter rectification of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2773–2781.

Revaud, J., Weinzaepfel, P., Harchaoui, Z., and Schmid, C. (2015). Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1164–1172.

Ringaby, E. and Forssén, P.-E. (2012). Efficient video rectification and stabilisation for cell-phones. *Intl. Journal of Computer Vision*, 96(3):335–352.

Rolland, J. P., Davis, L., and Baillot, Y. (2001). A survey of tracking technology for virtual environments. *Fundamentals of wearable computers and augmented reality*, 1:67–112.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer.

Rosten, E., Porter, R., and Drummond, T. (2010). Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):105–119.

Rowson, S., Brolinson, G., Goforth, M., Dietter, D., and Duma, S. (2009). Linear and angular head acceleration measurements in collegiate football. *Journal of biomechanical engineering*, 131(6):061016.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. R. (2011). Orb: An efficient alternative to sift or surf. In *ICCV*, volume 11–1, page 2. IEEE.

Sanchez-Vives, M. V. and Slater, M. (2004). From presence towards consciousness. In *8th Annual Conference for the Scientific Study of Consciousness*.

Saurer, O., Koser, K., Bouguet, J.-Y., and Pollefeys, M. (2013). Rolling shutter stereo. In *Proceedings of the IEEE Intl. Conference on Computer Vision*, pages 465–472.

Saurer, O., Pollefeys, M., and Lee, G. H. (2015). A minimal solution to the rolling shutter pose estimation problem. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1328–1334. IEEE.

Saurer, O., Pollefeys, M., and Lee, G. H. (2016). Sparse to dense 3d reconstruction from rolling shutter images. In *IEEE Computer Vision and Pattern Recognition*. IEEE.

Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nešić, N., Wang, X., and Westling, P. (2014). High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition*, pages 31–42. Springer.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42.

Schönberger, J. L. and Frahm, J.-M. (2016). Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Schöps, T., Engel, J., and Cremers, D. (2014). Semi-dense visual odometry for AR on a smartphone. In *International Symposium on Mixed and Augmented Reality*.

Shah, M., Eastman, R. D., and Hong, T. (2012). An overview of robot-sensor calibration methods for evaluation of perception systems. In *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*, pages 15–20. ACM.

Shi, J. et al. (1993). Good features to track. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600. IEEE.

Šmıd, M. and Matas, J. (2017). Rolling shutter camera synchronization with sub-millisecond accuracy. In *Proc. 12th Int. Conf. Comput. Vis. Theory Appl*, page 8.

Su, S. and Heidrich, W. (2015). Rolling shutter motion deblurring. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1529–1537. IEEE.

Sun, D., Roth, S., and Black, M. J. (2010). Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439. IEEE.

Tateno, K., Navab, N., and Tombari, F. (2018). Distortion-aware convolutional filters for dense prediction in panoramic images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 707–722.

Tomasi, C. and Detection, T. K. (1991). Tracking of point features. Technical report, Tech. Rep. CMU-CS-91-132, Carnegie Mellon University.

Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE.

Tsai, R. (1987). A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344.

Tsai, R. Y. and Lenz, R. K. (1989). A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. *IEEE Transactions on robotics and automation*, 5(3):345–358.

Valentin, J., Kowdle, A., Barron, J. T., Wadhwa, N., Dzitsiuk, M., Schoenberg, M., Verma, V., Csaszar, A., Turner, E., Dryanovski, I., Afonso, J., Pascoal, J., Tsotsos, K., Leung, M., Schmidt, M., Guleryuz, O., Khamis, S., Tankovitch, V., Fanello, S., Izadi, S., and Rhemann, C. (2018). Depth from motion for smartphone ar. *SIGGRAPH Asia*.

Ventura, J., Arth, C., Reitmayr, G., and Schmalstieg, D. (2014). Global localization from monocular slam on a mobile phone. *Visualization and Computer Graphics, IEEE Transactions on*, 20(4):531–539.

Von Itzstein, G. S., Billinghurst, M., Smith, R. T., and Thomas, B. H. (2017). Augmented reality entertainment: Taking gaming out of the box. *Encyclopedia of Computer Graphics and Games*, pages 1–9.

Vorozcovs, A., Stürzlinger, W., Hogue, A., and Allison, R. S. (2006). The hedgehog: a novel optical tracking method for spatially immersive displays. *Presence: Teleoperators & Virtual Environments*, 15(1):108–121.

Waechter, M., Moehrle, N., and Goesele, M. (2014). Let there be color! large-scale texturing of 3d reconstructions. In *European Conference on Computer Vision*, pages 836–850. Springer.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

Weiss, B. (2006). Fast median and bilateral filtering. In *Acm Transactions on Graphics (TOG)*, volume 25 (3), pages 519–526. ACM.

Welch, G., Bishop, G., Vicci, L., Brumback, S., Keller, K., and Colucci, D. (2001). High-performance wide-area optical tracking: The hiball tracking system. *presence: teleoperators and virtual environments*, 10(1):1–21.

Welch, G., Bishop, G., Vicci, L., Brumback, S., Keller, K., et al. (1999). The hiball tracker: High-performance wide-area tracking for virtual and augmented environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 1–ff. ACM.

Welch, G. and Foxlin, E. (2002). Motion tracking: no silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications*, 22(6):24–38.

Welch, G. F. and Bishop, G. (1996). *SCAAT: Incremental tracking with incomplete information*. PhD thesis, University of North Carolina at Chapel Hill.

Wiersma, R. D., Tomarken, S., Grelewicz, Z., Belcher, A. H., and Kang, H. (2013). Spatial and temporal performance of 3d optical surface imaging for real-time head position tracking. *Medical physics*, 40(11).

Wilburn, B. (2004). *High performance imaging using arrays of inexpensive cameras*. PhD thesis, Citeseer.

Wu, H., Zheng, S., Zhang, J., and Huang, K. (2018). Fast end-to-end trainable guided filter. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1838–1847.

Würmlin, S., Lamboray, E., and Gross, M. (2004). 3d video fragments: Dynamic point samples for real-time free-viewpoint video. *Computers & Graphics*, 28(1):3–14.

Xu, L., Ren, J., Yan, Q., Liao, R., and Jia, J. (2015). Deep edge-aware filters. In *International Conference on Machine Learning*, pages 1669–1678.

Yair, W. (2004). Colorization Using Optimization webpage. `https://www.cs.huji.ac.il/~yweiss/Colorization/index.html`. Online; accessed 12-March-2018.

Yang, Q., Ahuja, N., and Tan, K.-H. (2015). Constant time median and bilateral filtering. *International Journal of Computer Vision*, 112(3):307–318.

Yang, Q., Wang, S., and Ahuja, N. (2010). Svm for edge-preserving filtering. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1775–1782. IEEE.

Yang, R. and Pollefeys, M. (2003). Multi-resolution real-time stereo on commodity graphics hardware. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I. IEEE.

Yatziv, L. and Sapiro, G. (2006). Fast image and video colorization using chrominance blending. *IEEE transactions on image processing*, 15(5):1120–1129.

Zbontar, J. and LeCun, Y. (2016). Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2.

Zhang, M. and Gunturk, B. K. (2008). Multiresolution bilateral filtering for image denoising. *IEEE Transactions on image processing*, 17(12):2324–2333.

Zhang, X., Fronz, S., and Navab, N. (2002). Visual marker detection and decoding in ar systems: A comparative study. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 97. IEEE Computer Society.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334.

Zheng, F., Whitted, T., Lastra, A., Lincoln, P., State, A., Maimone, A., and Fuchs, H. (2014). Minimizing latency for augmented reality displays: Frames considered harmful. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 195–200.

Zhou, F., Duh, H. B.-L., and Billinghurst, M. (2008). Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 193–202. IEEE Computer Society.

Zhuang, B., Cheong, L.-F., and Hee Lee, G. (2017). Rolling-shutter-aware differential sfm and image rectification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 948–956.