

# Arrayed LiDAR Signal Analysis for Automotive Applications

Andreas Aßmann

Thesis submitted for the degree of Engineering Doctorate



Heriot-Watt University  
School of Engineering and Physical Sciences

December 2020

The copyright in this thesis is owned by the author. Any quotation from the thesis or use of any of the information contained in it must acknowledge this thesis as the source of the quotation or information.

# Abstract

Light detection and ranging (LiDAR) is one of the enabling technologies for advanced driver assistance and autonomy. Advances in solid-state photon detector arrays offer the potential of high-performance LiDAR systems but require novel signal processing approaches to fully exploit the dramatic increase in data volume an arrayed detector can provide.

This thesis presents two approaches applicable to arrayed solid-state LiDAR. First, a novel block independent sparse depth reconstruction framework is developed, which utilises a random and very sparse illumination scheme to reduce illumination density while improving sampling times, which further remain constant for any array size. Compressive sensing (CS) principles are used to reconstruct depth information from small measurement subsets. The smaller problem size of blocks reduces the reconstruction complexity, improves compressive depth reconstruction performance and enables fast concurrent processing. A feasibility study of a system proposal for this approach demonstrates that the required logic could be practically implemented within detector size constraints. Second, a novel deep learning architecture called LiDARNet is presented to localise surface returns from LiDAR waveforms with high throughput. This single data driven processing approach can unify a wide range of scenarios, making use of a training-by-simulation methodology. This augments real datasets with challenging simulated conditions such as multiple returns and high noise variance, while enabling rapid prototyping of fast data driven processing approaches for arrayed LiDAR systems.

Both approaches are fast and practical processing methodologies for arrayed LiDAR systems. These retrieve depth information with excellent depth resolution for wide operating ranges, and are demonstrated on real and simulated data. LiDARNet is a rapid approach to determine surface locations from LiDAR waveforms for efficient point cloud generation, while block sparse depth reconstruction is an efficient method to facilitate high-resolution depth maps at high frame rates with reduced power and memory requirements.

To Hristina  
To my friends and family

# Acknowledgements

I want to thank my supervisors Dr. Brian Stewart and Prof. Andrew Wallace for their continued support, mentorship and many fruitful, critical but constructive and always valuable discussions. Both have provided interesting and inspiring insights into industry and academia respectively as well as research and development in general. They have taken the time to proof-read this thesis and have provided detailed feedback, which has greatly helped to improve this thesis.

Thanks to STMicroelectronics and the Engineering and Physical Sciences Research Council (EPSRC) for funding and financial support, which made this project possible and allowed me to present my work at international conferences.

Thanks also to all staff at STMicroelectronics, in particular Arnaud and his team in Paris for their extensive knowledge and helpful advice on my work and the Centre for Doctoral Training in Applied Photonics for their support as well as their excellent organisation of the programme and all events hosted.

I want to thank all members of the vision lab and associated staff past and present, Sap, Marcel, Yun, João and in particular Paulo and Teun, for distractions, (many) coffee breaks, interesting discussions and chats about research and life in general.

Finally, I would like to thank my family, my friends Andreea, Blair, Teodora, Kim, Lukasz, Paul and especially Hristina, who kept me motivated to getting it done and have provided invaluable support throughout this period, both when we could all see each other in person and even when some of us could only interact virtually during a global pandemic. Thank you!



## Research Thesis Submission

|   |   |                |                       |
|---|---|----------------|-----------------------|
| Name:   | Andreas Aßmann                              |                |                       |
| School:   | School of Engineering and Physical Sciences |                |                       |
| Version: <i>(i.e. First, Resubmission, Final)</i> | Final                                       | Degree Sought: | Engineering Doctorate |

### Declaration


In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1. The thesis embodies the results of my own work and has been composed by myself
2. Where appropriate, I have made acknowledgement of the work of others
3. The thesis is the correct version for submission and is the same version as any electronic versions submitted\*.
4. My thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
5. I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.
6. I confirm that the thesis has been verified against plagiarism via an approved plagiarism detection application e.g. Turnitin.

\* Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.

|                         |   |       |            |
|-------------------------|---|-------|------------|
| Signature of Candidate: |  | Date: | 15/02/2021 |
|-------------------------|---|-------|------------|

### Submission

|  |  |
|--|--|
| Submitted By <i>(name in capitals)</i> : | ANDREAS AßMANN   |
| Signature of Individual Submitting:      |  |
| Date Submitted:                          | 15/02/2021   |

### For Completion in the Student Service Centre (SSC)

|  |           |     |  |    |  |          |     |  |    |  |
|--|-----------|-----|--|----|--|----------|-----|--|----|--|
| Limited Access   | Requested | Yes |  | No |  | Approved | Yes |  | No |  |
| <i>E-thesis Submitted (mandatory for final theses)</i> |           |     |  |    |  |          |     |  |    |  |
| Received in the SSC by <i>(name in capitals)</i> :     |           |     |  |    |  | Date:    |     |  |    |  |

# Contents

|   |             |
|---|-------------|
| <b>Abstract</b>                               | <b>i</b>    |
| <b>List of Publications</b>                   | <b>viii</b> |
| <b>List of Abbreviations</b>                  | <b>x</b>    |
| <b>List of Figures</b>                        | <b>xiii</b> |
| <b>List of Tables</b>                         | <b>xvi</b>  |
| <b>1 Introduction</b>                         | <b>1</b>    |
| 1.1 Motivation . . . . .                      | 2           |
| 1.2 Thesis outline . . . . .                  | 3           |
| <b>2 Light Detection and Ranging</b>          | <b>5</b>    |
| 2.1 Introduction . . . . .                    | 5           |
| 2.2 Depth Sensing . . . . .                   | 8           |
| 2.3 Photon Detection . . . . .                | 10          |
| 2.3.1 Single Photon Avalanche Diode . . . . . | 11          |
| 2.3.2 SPAD arrays . . . . .                   | 13          |
| 2.3.3 Laser Power Constraints . . . . .       | 15          |
| 2.4 LiDAR Architectures . . . . .             | 16          |
| 2.4.1 Mechanical Scanning Systems . . . . .   | 17          |
| 2.4.2 Solid-State LiDAR . . . . .             | 19          |
| 2.4.3 Commercial LiDAR Systems . . . . .      | 21          |
| 2.5 LiDAR Waveform . . . . .                  | 23          |
| 2.5.1 Waveform Signal Model . . . . .         | 24          |
| 2.5.2 LiDAR Signal Processing . . . . .       | 25          |
| 2.6 Conclusions . . . . .                     | 26          |
| <b>3 LiDAR Signal Simulation</b>              | <b>28</b>   |

|          |  |           |
|----------|--|-----------|
| 3.1      | Introduction . . . . .   | 28        |
| 3.2      | Datasets . . . . .   | 29        |
| 3.2.1    | LiDAR Datasets . . . . .   | 29        |
| 3.2.2    | Synthetic Depth Data . . . . .                                   | 30        |
| 3.3      | LiDAR Waveform Simulation . . . . .                              | 31        |
| 3.3.1    | Photon Count Rate Models . . . . .                               | 32        |
| 3.4      | LiDAR Simulation Toolchain . . . . .                             | 34        |
| 3.4.1    | Synthetic scenes to Full-Waveforms . . . . .                     | 36        |
| 3.4.2    | Waveforms with Multiple Returns . . . . .                        | 39        |
| 3.4.3    | Non-Linear Sampling . . . . .                                    | 42        |
| 3.4.4    | Hardware Prototyping . . . . .                                   | 44        |
| 3.5      | Conclusion . . . . .   | 45        |
| <b>4</b> | <b>Deep Learning for Surface Localisation in LiDAR Waveforms</b> | <b>47</b> |
| 4.1      | Introduction . . . . .   | 47        |
| 4.2      | Deep Learning . . . . .  | 49        |
| 4.2.1    | Convolutional Neural Network . . . . .                           | 51        |
| 4.2.2    | Convolutional Auto-Encoder . . . . .                             | 52        |
| 4.2.3    | Related work . . . . .   | 53        |
| 4.3      | LiDARNet . . . . .   | 53        |
| 4.3.1    | Introduction . . . . .   | 53        |
| 4.3.2    | Architecture . . . . .   | 55        |
| 4.4      | Training Methodology . . . . .                                   | 57        |
| 4.4.1    | Training Data Generation . . . . .                               | 57        |
| 4.5      | Validation Experiment . . . . .                                  | 60        |
| 4.5.1    | Two Retro-reflectors . . . . .                                   | 61        |
| 4.5.2    | Feature Tracing . . . . .  | 62        |
| 4.5.3    | Results . . . . .  | 65        |
| 4.6      | Automotive Experiment . . . . .                                  | 67        |
| 4.6.1    | Results . . . . .  | 69        |
| 4.7      | Summary and Conclusions . . . . .                                | 75        |
| <b>5</b> | <b>Concurrent Block Sparse Sensing LiDAR</b>                     | <b>78</b> |
| 5.1      | Introduction . . . . .   | 78        |
| 5.2      | Compressive Sensing . . . . .                                    | 80        |
| 5.2.1    | Linear transforms and projection patterns . . . . .              | 84        |
| 5.2.2    | Blocked Compressive Sensing . . . . .                            | 86        |
| 5.2.3    | Compressive Depth Imaging . . . . .                              | 89        |
| 5.3      | Small Scale Sparse Depth Sensing . . . . .                       | 96        |

|          |   |            |
|----------|---|------------|
| 5.3.1    | Observational Model . . . . .                       | 97         |
| 5.3.2    | Sparsity of Small Scale Signals . . . . .           | 106        |
| 5.4      | Time-of-Flight Solid-State Block Array . . . . .    | 114        |
| 5.4.1    | Signal Model . . . . .                              | 115        |
| 5.4.2    | Block Size Effects on Processing Speed . . . . .    | 117        |
| 5.4.3    | Total Variation Extension . . . . .                 | 117        |
| 5.5      | Applications and Results . . . . .                  | 120        |
| 5.5.1    | Practical Basis Transforms and Algorithms . . . . . | 121        |
| 5.5.2    | Arrayed Sparse LiDAR . . . . .                      | 122        |
| 5.5.3    | Scalable Imaging Arrays . . . . .                   | 137        |
| 5.6      | Summary and conclusions . . . . .                   | 138        |
| <b>6</b> | <b>Small Footprint Sparse Sensing Logic</b>         | <b>141</b> |
| 6.1      | Introduction . . . . .                              | 141        |
| 6.2      | Approximate Computing . . . . .                     | 142        |
| 6.2.1    | Reduced Precision . . . . .                         | 144        |
| 6.3      | Algorithm Hardware Optimisations . . . . .          | 145        |
| 6.3.1    | Lean ADMM . . . . .                                 | 146        |
| 6.4      | Precision Scaling Effects . . . . .                 | 148        |
| 6.4.1    | Compressive Depth Reconstruction . . . . .          | 153        |
| 6.4.2    | Sparse Depth Reconstruction . . . . .               | 156        |
| 6.5      | Case Study: Logic Footprint . . . . .               | 160        |
| 6.5.1    | SPAD Array Size . . . . .                           | 160        |
| 6.5.2    | Sparse Logic Component Size . . . . .               | 161        |
| 6.5.3    | Block Logic Core Size . . . . .                     | 162        |
| 6.6      | Summary and Conclusions . . . . .                   | 165        |
| <b>7</b> | <b>Conclusion and Future Work</b>                   | <b>167</b> |
| 7.1      | Summary of Main Findings . . . . .                  | 167        |
| 7.2      | Future Work . . . . .                               | 170        |
|          | <b>Bibliography</b>                                 | <b>172</b> |

# List of Publications

## Conference Proceedings

- **A. Aßmann**, B. Stewart, A. M. Wallace, "Deep Learning for LiDAR Waveforms with Multiple Returns," in European Signal Processing Conference (EUSIPCO), Amsterdam, Netherlands, 2020.
- **A. Aßmann**, Y. Wu, B. Stewart, A. M. Wallace, "Accelerated 3D Image Reconstruction for Resource Constrained Systems," in European Signal Processing Conference (EUSIPCO), Amsterdam, Netherlands, 2020.
- **A. Aßmann**, B. Stewart, J. F. C. Mota, A. M. Wallace, "Compressive Super-Pixel LiDAR for High Framerate 3D Depth Imaging," in IEEE Global Conference on Signal and Information Processing (GlobalSIP), Ottawa, Canada, 2019.

## Journals in Preparation

- **A. Aßmann**, J. F. C. Mota, B. Stewart, A. M. Wallace, "Parallel Block Compressive LiDAR Imaging," for IEEE Transactions on Computational Imaging.
- Y. Wu, **A. Aßmann**, B. Stewart, A. M. Wallace, "Efficient Approximate 3D Image Reconstruction" for IEEE Transactions on Computers.

## Patent Applications

- **A. Aßmann**, "Method and Devices for Enhanced Imaging", United States Patent Application US 16/894,221 filed on 05 June 2020.
- **A. Aßmann**, B. Stewart, "Scalable Depth Imager", United States Patent Application US 17/083,079 filed on 28 October 2020.
- **A. Aßmann**, "Under Display LiDAR", *in preparation*.

## Conference Presentations

- **A. Aßmann**, B. Stewart, A. M. Wallace, "Deep Learning for LiDAR Waveforms with Multiple Returns," in European Signal Processing Conference (EUSIPCO), Amsterdam, Netherlands, 2020/2021.
- **A. Aßmann**, Y. Wu, B. Stewart, A. M. Wallace, "Accelerated 3D Image Reconstruction for Resource Constrained Systems," in European Signal Processing Conference (EUSIPCO), Amsterdam, Netherlands, 2020/2021.
- **A. Aßmann**, B. Stewart, J. F. C. Mota, A. M. Wallace, "Compressive Super-Pixel LiDAR for High Framerate 3D Depth Imaging," in IEEE Global Conference on Signal and Information Processing (GlobalSIP), Ottawa, Canada, 2019.
- **A. Aßmann**, "Compressive Super-Pixel LiDAR for High-Framerate Depth Imaging," in Centre for Doctoral Training in Photonics Annual Conference, Edinburgh, Scotland, 2019.

## Posters

- **A. Aßmann**, B. Stewart, A. M. Wallace, "The Quest for Ground Truth in LiDAR: From Synthetic Data to Realistic Sampling," in Centre for Doctoral Training in Photonics Annual Conference, St Andrews, Scotland, 2018.

# List of Abbreviations

|              |   |
|--------------|---|
| <b>AC</b>    | approximate computing                           |
| <b>ADAS</b>  | advanced driver assistance systems              |
| <b>ADC</b>   | analogue-to-digital converter                   |
| <b>ADMM</b>  | alternating direction method of multipliers     |
| <b>ANN</b>   | artificial neural network                       |
| <b>APD</b>   | avalanche photodiode                            |
| <b>API</b>   | application programming interface               |
| <b>ASIC</b>  | application specific integrated circuit         |
| <b>BCS</b>   | block compressive sensing                       |
| <b>BPDN</b>  | basis pursuit de-noising                        |
| <b>CS</b>    | compressive sensing                             |
| <b>CBCS</b>  | checkerboard compressive sensing                |
| <b>CMOS</b>  | complementary metal-oxide semiconductor         |
| <b>CNN</b>   | convolutional neural network                    |
| <b>CUDA</b>  | compute unified device architecture             |
| <b>DL</b>    | deep learning                                   |
| <b>DCT</b>   | discrete cosine transform                       |
| <b>DMD</b>   | digital-micromirror-device                      |
| <b>DSP</b>   | digital signal processing                       |
| <b>dToF</b>  | direct Time-of-Flight                           |
| <b>DWT</b>   | discrete wavelet transform                      |
| <b>EB</b>    | encoding block                                  |
| <b>ECG</b>   | electrocardiogram                               |
| <b>FISTA</b> | fast iterative shrinkage thresholding algorithm |
| <b>FoV</b>   | field-of-view                                   |
| <b>FF</b>    | fill-factor                                     |
| <b>FFT</b>   | fast Fourier transform                          |
| <b>FPGA</b>  | field-programmable gate array                   |
| <b>FRI</b>   | finite-rate-of-innovation                       |

|                     |   |
|---------------------|---|
| <b>FWHM</b>         | full-width-at-half-maximum                              |
| <b>GPSR</b>         | gradient projection for sparse reconstruction           |
| <b>GPU</b>          | graphical processing unit                               |
| <b>HDL</b>          | hardware description language                           |
| <b>I/O</b>          | input and output  |
| <b>IRF</b>          | instrument response function                            |
| <b>laser</b>        | light amplification by stimulated emission of radiation |
| <b><i>lasso</i></b> | least absolute shrinkage and selection operator         |
| <b>LiDAR</b>        | light detection and ranging                             |
| <b>MP</b>           | matrix penciling  |
| <b>MCU</b>          | microcontroller unit                                    |
| <b>MRI</b>          | magnetic-resonance-imaging                              |
| <b>MSE</b>          | mean squared error                                      |
| <b>NIR</b>          | near-infrared   |
| <b>OMP</b>          | orthogonal matching pursuit                             |
| <b>OpenCL</b>       | open computing language                                 |
| <b>PDE</b>          | photon detection efficiency                             |
| <b>PL</b>           | projected Landweber                                     |
| <b>PRR</b>          | pulse-repetition-rate                                   |
| <b>PSNR</b>         | power signal-to-noise ratio                             |
| <b>RADAR</b>        | radio detection and ranging                             |
| <b>RAM</b>          | random access memory                                    |
| <b>ReLU</b>         | rectified linear unit                                   |
| <b>RIP</b>          | restricted isometric property                           |
| <b>RJMCMC</b>       | reverse jump Markov chain Monte Carlo                   |
| <b>ROC</b>          | receiver operating characteristics                      |
| <b>RSNR</b>         | reconstruction signal-to-noise ratio                    |
| <b>SAE</b>          | Society of Automotive Engineers                         |
| <b>SiPM</b>         | silicon photomultiplier                                 |
| <b>SLM</b>          | spatial light modulator                                 |
| <b>SoC</b>          | system-on-a-chip  |
| <b>SPAD</b>         | single photon avalanche detector                        |
| <b>SPL</b>          | smoothed projected Landweber                            |
| <b>SNR</b>          | signal-to-noise ratio                                   |
| <b>SSIM</b>         | structural similarity index measure                     |
| $\sigma$            | standard deviation                                      |
| <b>TCSPC</b>        | time-correlated single photon counting                  |
| <b>TDC</b>          | time-to-digital converter                               |



|              |  |
|--------------|--|
| <b>ToF</b>   | time-of-flight                         |
| <b>TV</b>    | total-variation                        |
| <b>VCSEL</b> | vertical-cavity-surface-emitting-laser |

# List of Figures

|      |   |    |
|------|---|----|
| 2.1  | A generic LiDAR system . . . . .                                    | 7  |
| 2.2  | Sun irradiance and eye transmittance illustration . . . . .         | 7  |
| 2.3  | The direct time-of-flight principle. . . . .                        | 9  |
| 2.4  | A standard photodiode diagram . . . . .                             | 11 |
| 2.5  | SPAD schematic and associated V-I characteristics . . . . .         | 12 |
| 2.6  | Illustration of beam steering LiDAR . . . . .                       | 17 |
| 2.7  | Illustration of mechanically scanning LiDAR . . . . .               | 18 |
| 2.8  | Illustration of solid-state LiDAR array . . . . .                   | 19 |
| 2.9  | Sampling times of various LiDAR architectures . . . . .             | 21 |
| 2.10 | LiDAR waveforms for various application scenarios . . . . .         | 23 |
| 3.1  | Base reflectivity map from class segmentation data. . . . .         | 37 |
| 3.2  | Conversion of RGB to brightness map. . . . .                        | 37 |
| 3.3  | Reflectivity map generation from class and brightness maps. . . . . | 38 |
| 3.4  | Depth and reflectivity maps for photon count rates. . . . .         | 38 |
| 3.5  | Multi-return generation from dense single return data . . . . .     | 40 |
| 3.6  | Multi-return waveform from high-resolution synthetic data . . . . . | 40 |
| 3.7  | Multi-return generation from dense single return data . . . . .     | 40 |
| 3.8  | A simulated multi-return LiDAR waveform . . . . .                   | 41 |
| 3.9  | LiDAR Simulation Toolchain . . . . .                                | 42 |
| 3.10 | Sampling mode illustrations for simulated LiDAR sensors . . . . .   | 43 |
| 3.11 | Sampling simulation flow. . . . .                                   | 44 |
| 3.12 | Hardware prototyping flow. . . . .                                  | 45 |
| 4.1  | Illustration of a typical artificial neuron. . . . .                | 49 |
| 4.2  | Illustration of a typical 1D convolutional building block. . . . .  | 51 |
| 4.3  | Generic convolutional auto-encoder architecture . . . . .           | 52 |
| 4.4  | Auto-encoder results applied to real and simulated data . . . . .   | 55 |
| 4.5  | LiDARNet Architecture . . . . .                                     | 56 |
| 4.6  | LiDARNet training with simulated inputs and ground truth. . . . .   | 59 |

|      |   |     |
|------|---|-----|
| 4.7  | Illustration of a two retro-reflectors in close proximity experiment . .  | 61  |
| 4.8  | Illustrative signal progression through LiDARNet in the <i>SuperRes</i><br>configuration . . . . .              | 63  |
| 4.9  | Resolving two close targets in LiDAR waveform . . . . .   | 66  |
| 4.10 | Automotive scenes for LiDARNet training data generation . . . . .   | 68  |
| 4.11 | ROC curves for LiDARNet . . . . .   | 70  |
| 4.12 | Multi-return issues with FRI-MP fitting too many surface locations<br>randomly. . . . .                         | 71  |
| 4.13 | Signal reconstructions from detected surface return positions for noisy<br>long-range LiDAR waveforms . . . . . | 73  |
| 4.14 | Processing time scaling for multiple concurrent waveforms . . . . .   | 74  |
| 5.1  | Compressive sensing visualisation . . . . .   | 82  |
| 5.2  | Illustration of a single pixel camera . . . . .   | 83  |
| 5.3  | Illustration of coded compressive depth system . . . . .  | 91  |
| 5.4  | Illustration of time-gated compressive single pixel depth system . . .  | 93  |
| 5.5  | Illustration of a single pixel depth system . . . . .   | 95  |
| 5.6  | Histogram building for compressive sensing . . . . .  | 98  |
| 5.7  | Measurements for compressive depth sensing . . . . .  | 99  |
| 5.8  | Compressive Super-Pixel LiDAR . . . . .   | 101 |
| 5.9  | Compressive Depth Sampling Ground Truth . . . . .   | 101 |
| 5.10 | Compressive Depth Sampling Illustration . . . . .   | 102 |
| 5.11 | Compressive small scale depth sensing visualisation . . . . .   | 104 |
| 5.12 | Compressive Depth Sampling Reconstruction . . . . .   | 104 |
| 5.13 | Illustrative scenes for block sparsity analysis . . . . .   | 106 |
| 5.14 | Sparsity comparison between depth sum and photon count . . . . .  | 107 |
| 5.15 | Compression potential for proxy signals . . . . .   | 109 |
| 5.16 | Sparsity effects on depth reconstruction via proxies . . . . .  | 112 |
| 5.17 | Block size effects on linear transform compression potential . . . . .  | 113 |
| 5.18 | Blocking structure for checkerboard compressive sensing . . . . .   | 114 |
| 5.19 | Super-Pixel LiDAR array system . . . . .  | 115 |
| 5.20 | Block processing time scaling . . . . .   | 117 |
| 5.21 | Scenes for evaluation of CBCS methodology . . . . .   | 120 |
| 5.22 | Depth compressive sensing comparison for compressive schemes . . .  | 130 |
| 5.23 | CBCS comparison for basis transforms . . . . .  | 132 |
| 5.24 | Comparison of CBCS TV extensions . . . . .  | 133 |
| 5.25 | Sample time comparison - CS and linear scan . . . . .   | 134 |
| 5.26 | Quality and frame time comparison for compressive depth reconstruc-<br>tion . . . . .                           | 134 |

---

|      |  |     |
|------|--|-----|
| 5.27 | Depth sparse random sensing comparison . . . . .                           | 135 |
| 5.28 | Noise comparison of sparse block depth sensing schemes . . . . .           | 136 |
| 5.29 | Integrated sparse sensing block . . . . .                                  | 137 |
| 5.30 | Scalable Sparse Depth Imaging Array . . . . .                              | 138 |
| 6.1  | Illustration of floating and fixed point number formats. . . . .           | 144 |
| 6.2  | Illustrative scenes for precision scaling effects . . . . .                | 150 |
| 6.3  | Precision effects on reconstruction quality for CBCS using $\ell$ ADMM .   | 155 |
| 6.4  | Illustration of quality degradation for precision scaling of $\ell$ ADMM . | 156 |
| 6.5  | Precision effects on reconstruction quality for CBCS using $d$ Sparse .    | 158 |
| 6.6  | Illustration of quality degradation for precision scaling of $d$ Sparse .  | 159 |
| 6.7  | Sparse Scale Block Components . . . . .                                    | 160 |
| 6.8  | Logic core size comparison chart . . . . .                                 | 163 |
| 6.9  | Size of logic core versus SPAD arrays . . . . .                            | 164 |
| 6.10 | Sparse scale block with component size scaling. . . . .                    | 165 |

# List of Tables

|     |  |     |
|-----|--|-----|
| 2.1 | Levels of driving automation . . . . .   | 6   |
| 2.2 | SPAD array sizes over the years . . . . .  | 14  |
| 2.3 | Typical selection of available scanning LiDAR systems . . . . .  | 22  |
| 2.4 | Close-to-market automotive solid-state LiDAR systems . . . . .   | 22  |
| 3.1 | Depth datasets for LiDAR simulation . . . . .  | 32  |
| 3.2 | Proposed LiDAR Specifications . . . . .  | 35  |
| 3.3 | Reflectance estimates . . . . .  | 36  |
| 4.1 | Auto-Encoder Parameter Configuration . . . . .   | 54  |
| 4.2 | LiDAR system parameters used to capture the super-resolution benchmark . . . . .   | 62  |
| 4.3 | LiDARNet <i>SuperRes</i> Parameter Configuration . . . . .   | 62  |
| 4.4 | Evaluation of a real super-resolution benchmark using LiDARNet . . . . .   | 66  |
| 4.5 | Simulated automotive solid-state LiDAR waveform dataset . . . . .  | 68  |
| 4.6 | LiDARNet Automotive Parameter Configuration . . . . .  | 69  |
| 4.7 | Type 2 Analysis of multi-return waveform peak detection . . . . .  | 70  |
| 4.8 | Evaluation of LiDARNet in <i>Automotive</i> configuration . . . . .  | 72  |
| 5.1 | Sparsity in blocked compressive depth sensing – Short range . . . . .  | 108 |
| 5.2 | Sparsity in blocked compressive depth sensing – Mid range . . . . .  | 108 |
| 5.3 | Sparsity in blocked compressive depth sensing – Long range . . . . .   | 108 |
| 5.4 | CBCS operating parameters for sparsity and compression illustrated on the <i>mid-range scene</i> . Operating point for examples in this chapter are highlighted. . . . . | 123 |
| 5.5 | CBCS block size processing and sampling time illustration . . . . .  | 124 |
| 5.6 | Performance evaluation across various scenes for sparse depth frameworks . . . . .   | 128 |
| 5.7 | Depth compressive sensing comparison using various approaches . . . . .  | 129 |
| 6.1 | FPGA logic primitives with NAND gate estimates . . . . .   | 149 |

|     |  |     |
|-----|--|-----|
| 6.2 | Properties affecting precision limits. . . . .                       | 151 |
| 6.3 | Estimates for minimum bit width for various scenes and algorithms. . | 153 |
| 6.4 | Resource utilisation on FPGA for $\ell$ ADMM . . . . .               | 154 |
| 6.5 | Resource utilisation on FPGA for $d$ Sparse . . . . .                | 157 |
| 6.6 | Physical dimensions of small SPAD arrays . . . . .                   | 161 |
| 6.7 | Physical dimensions of transistors in common nodes . . . . .         | 161 |
| 6.8 | DSP logic components for sparse imaging block . . . . .              | 162 |
| 6.9 | NAND gate count estimate for sparse imaging block logic . . . . .    | 163 |

# Chapter 1

## Introduction

The next generation of autonomous robots including self-driving cars require a suite of sensors that perform at least as well as the human sensory system. Depth perception is a key enabler for reliable navigation in complex environments. Humans achieve rudimentary depth perception by passive stereo vision facilitated by experience and continuous learning for a good relative judgement of how far away things are in static and dynamic scenarios. Other sensory inputs such as balance and acceleration inputs further aid depth and motion perception. While those are good approximations and have enabled humans to operate advanced machinery over the centuries, they rely on a wealth of sensory input information being processed by the extremely complex human brain.

In most cases robotic systems have significantly less processing power and digital systems replicating human sensors will have the same limitations such as stereo cameras. Depth sensors in particular can be far more accurate if they leverage active sensing schemes. These measure a carrier wave's round trip time, called time-of-flight (ToF), to accurately measure the distance between object and sensor by means of the constant velocity of light waves. When using light waves to facilitate depth measurements this is called light detection and ranging (LiDAR). Modern LiDARs have been crucial components in most successful implementations of self driving cars, since the early 2000s. The most common form of LiDAR are point and line scanning variants, which mechanically scan a coherent light source such as light amplification by stimulated emission of radiation (laser) sources. They enable high accuracy measurements at short range in mobile phones as well as measurements across 200 metres or indeed for several metre long windows at several kilometres. However, high resolution LiDARs are often still very expensive due to the requirement for low tolerances in the calibration of the light source and detector for mechanically

moving systems and robust enclosures to reduce mechanical disruptions. For this reason significant work is being carried out to develop novel high resolution solid-state LiDAR arrays, which involve no moving parts, have a smaller footprint, can be integrated into robots seamlessly and can be mass produced at a lower cost. With more sensing elements, the amount of data that can be captured increases if the full potential of the array can be exploited, which as resolution increases pose significant hardware and software challenges. Further, fully autonomous systems operate in challenging conditions, with non trivial scenes and non-ideal reflectors. This requires complex processing schemes, which can impose prohibitive limit to operating times for dynamic applications.

The work described in this dissertation are as follows:

- Develop fast and efficient methods to process large volumes of data generated by a solid-state LiDAR system using single photon avalanche detector (SPAD) technology.
- Define a novel system architecture and develop novel methods to enable long range imaging at typical video frame rates or higher utilising solid-state photon detector arrays for automotive applications.

## 1.1 Motivation

This thesis addresses the issue of high data volume of solid-state arrayed LiDAR systems by means of two frameworks.

A machine learning approach is developed to localise returns from LiDAR waveforms in varying conditions with high throughput suitable for a large volume of detector responses in complex scenarios in a deterministic fashion. This approach makes use of deep learning techniques and is applicable to standard ways of scanning or burst illuminating an environment.

Additionally, for wide operating range depth systems e.g. 0-300 m, as required for self-driving cars, operating with significant ambient noise from sunlight and potentially other LiDARs, the required laser power to measure distance at the far end of the operating range can be dangerous to the human eye at close distances. To address this, a novel efficient LiDAR system is presented, which utilises the compressive sensing (CS) paradigm to reduce the data volume at the sampling stage, while also reducing the average laser output power by sparsifying the illumination stage resulting in low density illumination bursts, which incurs a cost in the processing stack. This work addresses this by parallelising the compressive reconstruction optimisa-



tion problem with a novel block compressive sensing system formulation for depth sensing with a wide operating range.

This is further supported by an investigation into efficient hardware implementations of the reconstruction problem exploiting approximate computing techniques to lower the hardware complexity and increase resource efficiency and further accelerate the processing stack for a pathway into application specific integrated circuit (ASIC) implementation for full sensor stacks. Either approach presents a meaningful way to process large scale solid-state LiDAR arrays in real-time in an integrated fashion.

The work for this thesis was carried out both at STMicroelectronics, a multinational company and pioneer in SPAD sensor systems for ToF imagers, and Heriot-Watt University.

## 1.2 Thesis outline

Chapter 2 introduces state-of-the-art LiDAR systems and their operational modes to measure depth. This chapter states the key challenges and shortcomings of current systems which form the basis of investigation for this thesis, being the challenges of using large ToF detector arrays from a signal processing perspective for real-time applications.

Chapter 3 presents datasets and signal simulation tools related to this work. This chapter also presents the signal model and custom simulation frameworks developed for this project to enable rapid prototyping of novel compressive sensing schemes as well as large volume data generation for deep learning approaches with underlying semantically relevant scenes for the automotive use case.

Chapter 4 provides an introduction to convolutional neural networks and their application to signal processing. This forms the basis of a novel network architecture designed for large scale multi-return peak localisation. The usage of deep learning to extract multi-path surface returns from ToF has not been previously demonstrated. This work presents a novel network architecture for surface localisation from LiDAR waveforms called LiDARNet with comparable performance to traditional model based signal processing while being faster with sub-millisecond processing times. It provides a high-throughput processing approach and thus is a natural fit to large scale array signal processing but further benefits from an adaptable validated training-by simulation approach. This allows future adaptations of various application scenarios into a single neural network, rather than multiple stages of traditional signal processing.

Chapter 5 presents the concept of CS and previous work on depth reconstruction in a compressive fashion. This work makes contributions to compressive depth sensing with a formal compressive depth reconstruction framework for small field-of-view imagers. It benefits from sparse random pattern exposures which reduces illumination density. The problem is distributed across a large solid-state ToF array to accelerate reconstruction times by using a novel active independent blocking method allowing concurrent processing schemes. This forms a novel compressive LiDAR architecture, which outperforms previous work, improving reconstruction quality for complex scenes, and reducing processing times with processing rates up to 1 kHz and maximum operating ranges well beyond 200 m.

This is further supported by an investigation into efficient hardware implementations of this block based CS LiDAR technique in Chapter 6, investigating the tolerance for reduced precision of numerical data types with fewer resource requirements and increased power efficiency. Resource utilisation for various precisions for a field-programmable gate array (FPGA) are translated into transistor counts to carry out a case study comparing a modern photon detector array area with estimates for logic area to perform the sparse sampling and depth reconstruction, finding that with moderate precision scaling, a system-on-a-chip (SoC) implementation of the proposed framework could be feasible within the size constraints of the detector.

Chapter 7 concludes the thesis by summarising the main findings of the two project paths and suggests future work building upon the work carried out for this dissertation.

# Chapter 2

## Light Detection and Ranging

### 2.1 Introduction

Over the last few decades cars have become safer due to improvements in manufacturing, testing and technology as a whole. The ability to sense impacts in a fraction of a second to deploy an airbag to reduce the impact for passengers and sensing when a wheel stops spinning to counteract skidding by modulating brakes are early examples of technology advancements to improve vehicle safety. More and more safety and convenience features have been added over the past decades, often summarised under the umbrella term advanced driver assistance systems (ADAS), which both support the human driver and improve safety of passengers and traffic participants alike.

This has enabled modern cars to perform some tasks of driving semi-autonomously. For example, a radar sensor can monitor the distance and speed of vehicles in front, while camera systems monitor the lane markings and traffic signs; this information is processed to control the acceleration of the vehicle and the steering on its own. Some cars can also perform emergency breaking to prevent accidents if sensors indicate a crash would be otherwise imminent. This kind of automation is deemed only as Level 2, with only few cars capable of such functionality currently on the market [1, 2, 3].

Autonomy in the context of driving is divided into 6 classes by the Society of Automotive Engineers (SAE) as shown in Table 2.1.

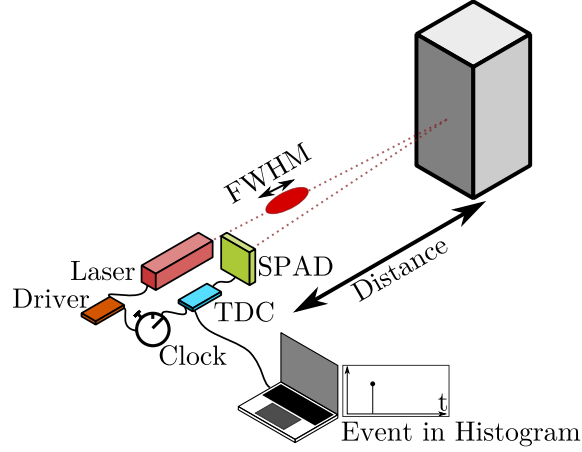
**Table 2.1:** Levels of driving automation as defined by the SAE [4].

|        | Level 0                             | Level 1                          | Level 2                          | Level 3                                | Level 4                   | Level 5                        |
|--------|-------------------------------------|----------------------------------|----------------------------------|--|---------------------------|--------------------------------|
| Driver |                                     | Active<br>Full<br>responsibility |                                  | Partial<br>responsibility              | Passive<br>responsibility | No<br>responsibility           |
| Car    | Warnings<br>and<br>minor assistance | steering<br>or<br>brake support  | steering<br>and<br>brake support | Autonomous<br>in<br>limited conditions |                           | Fully<br>autonomous<br>driving |

Only one car on the market, as of 2020, achieves Level 3 autonomy aspects utilising an extensive sensor suite including a solid-state light detection and ranging (LiDAR) line sensor [5, 3]. While some prototypes and a dedicated autonomous taxi service have launched since [6], these are still constrained to areas with good weather conditions and the total sensor ensemble is prohibitively expensive for the consumer market.

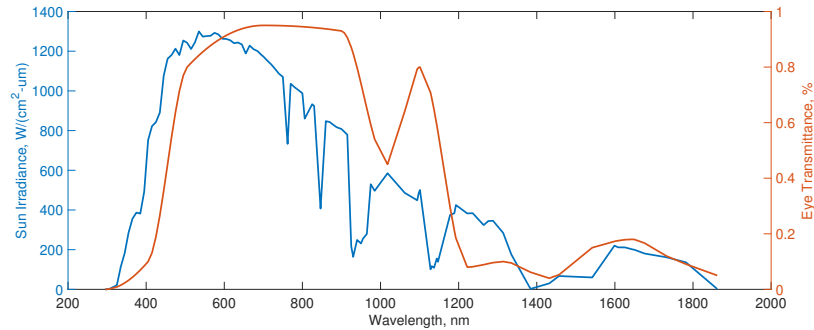
Driving is a very complex task and while a human driver is trained and then builds up extensive driving experience with access to a wealth of sensory inputs including depth estimation by means of stereo vision it is very difficult (and expensive) to perform the same tasks using sensors and control systems. The task of placing an autonomous actor into an environment and safely navigate the terrain is not a trivial task, in particular if (many) actors are involved in complex environments such as cities and while travelling at high speeds typical for transportation. Early demonstrations of autonomous driving have relied on extremely detailed mapping information [7] combined with very expensive sensors to perform Level 4 and Level 5 autonomy [8, 3]. This means specialised sensors are required, which perform vision tasks in a precise and timely fashion. While perception can be handled by normal passive camera systems, active measurements are the more reliable choice for speed and depth estimation [9].

Depth measurements in particular are important for navigation tasks as they allow accurate location of the autonomous vehicle (or main actor) with respect to a changing environment [9, 10]. This has made a high resolution high frame rate LiDAR sensor at a low cost a key enabler for level 5 autonomy [1, 2, 3]. A general LiDAR system is illustrated in Figure 2.1.



**Figure 2.1:** A generic LiDAR system. A coherent light source (e.g. a laser) is driven by a precise timing circuit to allow synchronised measurements with a photon detector (e.g. single photon avalanche detector (SPAD)) to measure the round trip time of emitted photons, which translates to range measurements.

From the early beginnings in space exploration [11], LiDAR systems have become more affordable but most systems are still prohibitively expensive for consumer applications. Further, the use of light under normal atmospheric conditions on earth introduces noise and safety challenges for cars as illustrated in Figure 2.2.



**Figure 2.2:** Sun irradiance adapted from [12] and eye transmittance values (how much of the light energy the eye absorbs) read from graph in [13] for illustrative purposes only. From a signal-to-noise ratio consideration there are pockets of little interference where operation is preferred, which is further constrained by eye-transmittance, which limits laser power and thus signal strength.

It is clear that the solar irradiance introduces significant noise challenges for self-driving cars with optical sensors at wavelengths, which can be manufactured at low cost and are invisible yet safe to the human eye. For this reason concurrent developments are under way to explore high resolution radio detection and ranging (RADAR) systems to perform similar tasks, but for now, LiDAR has advantages for mapping and navigation attributed to the potential for better spatial resolution [14, 15, 16, 17].

Modern advances in silicon photon-detector device using silicon complementary metal-oxide semiconductor (CMOS) technology [18] in the form of silicon SPAD arrays have created a valid path to develop large scale detector arrays for high resolution time-of-flight imaging at a relatively low cost. Combined with solid-state laser advancements in the form of vertical-cavity-surface-emitting-laser (VCSEL) arrays, which can also be manufactured at scale and are suitable for time-of-flight (ToF) applications, these two technologies enable novel solid-state LiDAR systems. They can be effectively integrated and manufactured at scale, which should drive down cost and reduce the optical and mechanical complexity of mechanical scanning LiDAR systems, which is one their key cost drivers [2, 19, 20, 3].

As the number of detectors increases to enable high resolution LiDAR, so does the required bandwidth of the processing system to cope with the processing for depth retrieval from LiDAR waveforms. This calls for efficient and fast processing strategies to deliver useful information to the full autonomous system. Furthermore, to illuminate the scene for such dense detector arrays within the wavelength limitations of silicon imposes limits on laser power and makes long distance ranging more difficult. This work aims to address these issues with investigations into a massively parallel processing approach (Chapter 4) and novel sparse sampling and processing schemes (Chapter 5) to enable efficient solid-state arrayed LiDAR systems.

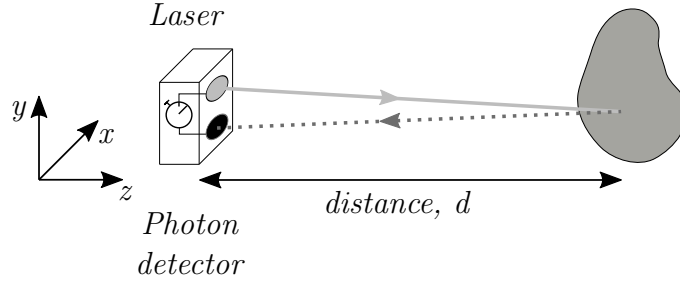
This chapter introduces the fundamental principles of LiDAR and solid-state photon detector arrays. A signal description of the full LiDAR waveform and its usefulness is presented. Finally, a discussion of relevant processing approaches and associated challenges for detector arrays over a full operating range LiDAR for long distance applications are stated.

## 2.2 Depth Sensing

There are two schools of ToF sensing, indirect and direct. Indirect ToF operates on the principles of phase or frequency difference and requires a continuous, but oscillating, illumination signal, which limits its usefulness in the near-infrared (NIR) due to high average power output and high laser cost [21, 2, 3]. Direct ToF operates on the principles of measuring the round trip time of photons directly. In particular single-photon counting systems can be sensitive to low photon counts at long ranges and are capable of dealing with high dynamic photon count ranges as well as the ability to capture full LiDAR waveforms making direct ToF LiDAR the preferred candidate for automotive LiDAR [22, 23].

### Direct Time-of-Flight

A photon travels at constant velocity in normal atmospheric conditions, most commonly referred to as the speed of light,  $c$ . By starting a timer as a laser pulse containing photons exits the emitter and recording the time those photons take to travel to and from a surface, one can readily determine the range,  $r$  as illustrated in Figure 2.3.



**Figure 2.3:** The direct time-of-flight principle illustrated with a solid-state emitter and detector pair. Light grey parts symbolise emission and dark grey detection, which applies to most diagrams in this chapter.

As a photon travels to and from a reflective surface, the range it covers is

$$r = c \frac{\delta t}{2}, \quad (2.1)$$

where  $c$  is the speed of light in air,  $r$  the distance to a reflective surface and  $\delta t$  the time it takes for the photon to travel to and from said surface. The depth resolution,  $s$  of a direct ToF system is defined by the smallest time step  $\delta t_s$  the sampling circuit can resolve [24], such that

$$s = \frac{\delta t_s c}{2}. \quad (2.2)$$

The components transforming the arrival to time measurements are called time-to-digital converters (TDC), they record the time at photon arrival and store a digital value often called a time stamp, which can be collected for further processing.

For a desired maximum range of a system,  $r_{max}$ , a histogram container can store this information in its quantised state with a total number of bins

$$p = \text{round}\left(\frac{r_{max}}{s}\right). \quad (2.3)$$

By measuring the ToF in different locations, a collection of depth values can be collected forming a discrete depth image,  $D \in \mathbb{R}^{n_x \times n_y}$ , where  $n_x$  and  $n_y$  are the

number of pixels in a sensor array as

$$D = \begin{bmatrix} d_{11} & \dots & d_{n_x 1} \\ \vdots & \ddots & \vdots \\ d_{1n_y} & \dots & d_{n_x n_y} \end{bmatrix}, \quad (2.4)$$

where  $d_{xy}$  is a single depth value at the array coordinate  $(x, y)$ .

Now let a histogram be a collection of photon counts,  $c_p$ , in each bin,  $p$ , such that

$$\mathbf{h} = [c_1, \dots, c_p] \in \mathbb{N}^p. \quad (2.5)$$

This is also called a LiDAR waveform, which will be further explored in Section 2.5. Prior to surface detection, the full sampling space is now a sample cube, such that

$$D_{ToF} = \begin{bmatrix} \mathbf{h}_1 & \dots & \dots \\ \vdots & \ddots & \vdots \\ \dots & \dots & \mathbf{h}_n \end{bmatrix}. \quad (2.6)$$

Assuming an operating range of 200 m and centimetre resolution a system can in theory produce up to 20,000 returns per pixel, resulting in a total data volume of 20 billion data points for a  $1000 \times 1000$  imaging resolution.

Since a waveform can contain up to  $K$  surface returns,  $\mathbf{d}_i \in \mathbb{R}^K$  and advanced algorithms can extract these multiple returns to form a point cloud from non-zero entries in the matrix,

$$D_{pc} = \begin{bmatrix} \mathbf{d}_1 & \dots & \dots \\ \vdots & \vdots & \vdots \\ \dots & \dots & \mathbf{d}_n \end{bmatrix}. \quad (2.7)$$

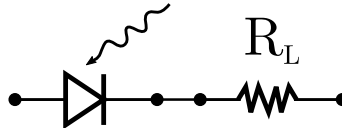
This illustrates the large data volume a LiDAR system can readily generate, even more so as the desired resolution increases for more demanding sensing tasks.

## 2.3 Photon Detection

The fundamental principal of photon detection is the process whereby the energy of a photon is absorbed into a photo-conductive material and the material converts the energy into an electrical current producing voltage, which can be measured as a voltage drop. Analogue devices such as vacuum tubes and in particular photon multiplier tubes (PMT) generally filled with low pressure gas to achieve amplification



of the photocurrent have been shown to have single photon precision [25]. This is important for precise timing of the arrival of a photon emitted from a synchronised system. However, analogue devices are fragile and large. To improve robustness of the system a solid-state device is desirable. Fortunately, modifications to a normal photodiode, as shown in Figure 2.4, can be made to achieve single photon detection.



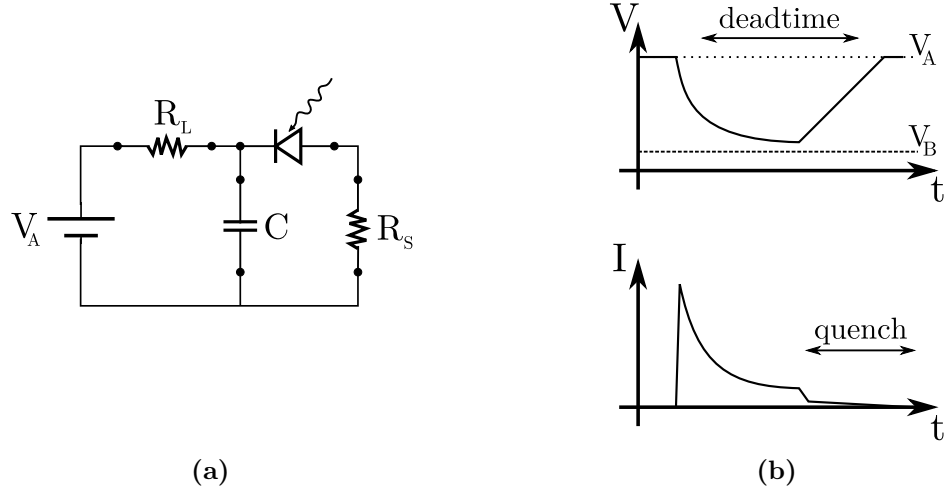
**Figure 2.4:** A standard photodiode diagram

A photodiode is a semi-conductor device made up of a positively and negatively doped semi-conductor material such as silicon. To generate a current when photons are absorbed, the diode is operated in reverse bias. By connecting the p-side to the negative terminal (anode) and the n-side to the positive terminal (cathode), the positive charge carriers (holes) are pulled towards the negative terminal away from the interface, and similarly the positive terminal attracts all electrons away from the interface. This creates a depleted region at the p-n junction, with a breakdown voltage,  $V_B$ , across. In a photodiode, photons absorbed cause a charge differential which promotes electron flow if the charge differential is greater than the breakdown voltage causing an electrical current. The generated current is proportional to the influx of photons and is amplified for analogue-to-digital conversion to record or use the generated signal [25, 26].

### 2.3.1 Single Photon Avalanche Diode

For a single photon sensitive photodiode, the principle of a voltage avalanche is exploited. To create a voltage avalanche a bias voltage,  $V_A > V_B$  is applied across the cathode and anode creating an excess voltage  $V_E = V_A - V_B$ . By increasing the excess voltage, the more likely an avalanche occurs due to photon absorption. When a photon is absorbed, a large voltage drop occurs causing a photocurrent and momentarily lowers  $V_A$ . However, this avalanche will continue unless  $V_A$  is restored to its bias value. By passing the current through a resistor the resultant voltage recharges the SPAD. This process is called passive quenching and allows to accurately detect a single photon by stopping the avalanche after a single photon event, as the device is not receptive to another photon until the bias voltage is restored [27, 28, 29] as illustrated in Figure 2.5.

Alternatively the SPAD can also be actively recharged (active quenching), however, this requires additional components and thus space, which is at a premium in SPAD



**Figure 2.5:** (a) Passive quench SPAD schematic and associated V-I characteristics for a photon event. The absorbed photon triggers an avalanche, causing  $V_A$  to drop resulting in a voltage across  $R_L$ , the resulting current is used to recharge  $V_A$  via  $R_S$ , quenching current flow to re-charge the SPAD for single photon detection [29].

arrays to maximise the photon sensitive area for increased efficiency [18, 30]. Without the quenching circuit and lower bias voltage the resultant device is simply an avalanche photodiode (APD) which acts as a linear amplifier [25]

While the normal APD has been successfully used in LiDAR devices, it not only amplifies the incoming light emitted by the laser but also the ambient photon influx. In contrast the SPAD is sensitive to single photons and thus is more susceptible to photons correlated with the associated emitter. This suppresses some ambient photon counts, which occur less due to quenching. This characteristic makes them ideal in long range LiDAR applications, where mean photon signal rates can be less than one photon [31, 32, 33, 34].

Due to their usefulness in many applications, significant effort has been afforded to industrialise the SPAD using CMOS manufacturing processes [35, 36, 24, 37, 38]. In recent years SPADs have been implemented in ever decreasing feature sizes (often referred to as technology nodes) and have been fully industrialised for mass manufacturing at a 40 nm technology node [39]. Such recent advancements have enabled increased SPAD array sizes and due to its mature manufacturing processes can make even larger yet affordable high resolution SPAD arrays a reality. Further, the smaller feature sizes enable the manufacturing of SPAD pixels which contain multiple SPADs. This further increases photon detection efficiency, reduces pile-up effects and thus increases signal-to-noise ratio (SNR). Photon pile-up occurs when multiple photons return within a laser pulse cycle, but a single photon detector is unable to detect any further photons after the first photon event due to insensitivity

during recharge. These type of SPAD pixels are often referred to as silicon photomultiplier (SiPM) [40, 41] or macro-pixels [42]. Such pixel designs allow the user to fine tune photon detection rates to prevent saturation and photon pile-up in shorter ranges when a system also needs to be suitable for long range applications [43].

### 2.3.2 SPAD arrays

Although arrays of SPADs have been demonstrated soon after the inception of the device itself [44, 45] such early prototypes were very expensive and had poor photon detection efficiency. With the increased interest in solid-state LiDAR driven by the demand of low-cost LiDARs for the automotive industry [1, 2], advancements in SPAD manufacturing have driven down cost, which combined with more advanced technology nodes (i.e. smaller device features) led to an increase in array size and SPAD density.

Early demonstrations of single photon detector arrays are very small at  $10 \times 1$  [44] and  $4 \times 4$  [45] in the late 1990s and early 2000s. Both were designed to be used in conjunction with visible light lasers (500-600 nm) making them unsuitable for autonomous driving applications. Further, their photo reactive area to sensor size ratio, the fill-factor (FF), and consequently their photon detection efficiency (PDE) were fairly low at 20-50% for visible light.

As solid-state manufacturing technologies improved, the first demonstrations of pixel arrays with multiple SPADs per pixel were demonstrated [40, 42, 46] which are either called digital silicon photomultiplier (dSiPM or SiPM) [40] or macro-pixel [42] in the 2010s. In particular [46] presented an increase of pixel array size to  $16 \times 2$  with  $64 \times 6$  individual detectors for 870 nm illumination. The use of multiple detectors per ToF pixel increases the FF to an impressive 70% by increasing the photo-active area with no or little increase in logic area. The array sizes are still too small for high resolution solid-state LiDAR, but such arrays can be utilised if scanning mechanisms are used to increase the effective imaging resolution.

With further improvements in manufacturing processes array sizes have increased further up to  $256 \times 256$  individual detectors, with effective pixel arrays of roughly  $64 \times 64$  using macro-pixels [47, 48, 49, 50]. These have various applications in biological imaging [51, 47, 52, 53] where in some cases detectors are not grouped or LiDAR type applications with more modest imaging resolutions [54, 55]. With advancements in optics for such small devices, fill-factor can also be increased using micro-lenses [56, 57] and thus focusing light onto the active detector area rather than being lost when interacting with surrounding electronics.

Range has increased to around 100 m, which makes them overall more usable for automotive applications. However, the range resolutions can vary dramatically between meters and several centimetres. Generally, the TDC either captures only small histogram sizes on-chip with usually coarse range steps or is performed off-chip with expensive data transfer and storage. The majority of the quoted frame rates for the binary frames are for every TDC time step, which is in of itself not very meaningful. The binary frames, i.e. single photon count events, have to be aggregated into waveforms for each pixel to enable distinction between signal and noise and then have to be processed further to derive useful information for mapping, navigation and/or classification.

More recent advances have addressed some of these issues and on-chip histogram accumulation at centimetre resolution for up to 80 m has been demonstrated [58]. However, for high frame rates the histogram size still has to be reduced, probably due to read-out limitations, limiting both spatial resolution and effective range and resolution [59, 60]. Other similar SPAD arrays with a size of about  $256 \times 256$  for various applications have been presented [61, 62] as well as dedicated LiDAR type sensors [63, 64, 65], which all use macro-pixels for better SNR, but often only quote the raw SPAD pixel count.

In the biological imaging realm larger arrays have been demonstrated at  $512 \times 512$  [66] but such arrays only require small dynamic range with high range resolution, which is often handled in external devices. A large scale  $1024 \times 500$  SPAD array has also been presented recently [67], which brings SPAD imagers ever closer to normal intensity imaging cameras in spatial resolution. Unfortunately, the reported dynamic range is only a few meters with centimetre resolution. An overview of the above advancements is illustrated in Table 2.2.

**Table 2.2:** Overview of SPAD array sizes and some figures of merit over the years.

| Parameter                  | 1997 [44]     | 2002 [45]    | 2010 [68]    | 2013 [46]     | 2016 [54]    | 2017 [48]      | 2019 [58]        | 2020 [67]         |
|----------------------------|---------------|--------------|--------------|---------------|--------------|----------------|------------------|-------------------|
| Imaging Size               | $10 \times 1$ | $4 \times 4$ | $4 \times 4$ | $16 \times 2$ | $9 \times 9$ | $64 \times 32$ | $64 \times 64$   | $1024 \times 500$ |
| Pixel Size                 | $1 \times 1$  | $1 \times 1$ | $2 \times 2$ | $4 \times 3$  | $1 \times 1$ | $1 \times 1$   | $4 \times 4$     | $1 \times 1$      |
| Array size                 | $10 \times 1$ | $4 \times 4$ | $8 \times 8$ | $64 \times 6$ | $9 \times 9$ | $64 \times 32$ | $256 \times 256$ | $1024 \times 500$ |
| Pixel pitch, $\mu\text{m}$ | 10 – 50       | 30           | n/a          | 25            | 24           | n/a            | 38.4             | 9.4               |
| Fill-factor, %             | n/a           | 7            | n/a          | 70            | 43           | n/a            | 51               | 13.4              |
| PDE, %                     | 50            | 20           | 11 to 13     | n/a           | 2 – 30       | 5              | 23               | 27                |
| Wavelength, nm             | 500 – 620     | 532          | n/a          | 870           | 870          | 808            | 671              | 785               |
| Sample Rate, Hz            | n/a           | 8 MHz        | n/a          | n/a           | 11kHz        | n/a            | 200 MHz          | 24 kHz            |
| Dynamic Range, m           | n/a           | n/a          | n/a          | n/a           | 80           | 45             | 84               | 1.14              |
| Range Resolution, cm       | n/a           | n/a          | n/a          | n/a           | 1.26         | n/a            | 2                | 3.6               |
| Type                       | SPAD          | SPAD         | dSiPM        | SPAD          | SPAD         | SPAD           | SPAD             | SPAD              |
| ToF                        | dToF          | dToF         | dToF         | dToF          | dToF         | iToF           | dToF             | dToF              |

A similar array size progression can be also observed for non-Silicon SPADs devices, albeit their cost and array size is several years behind the Silicon SPAD developments [69, 70, 71, 72, 73, 74]. Nonetheless, the progression of  $> 1000$  nm LiDAR arrays is important in the future to further improve SNR and potentially range for LiDAR

systems and findings of this work should translate readily to any type of ToF imaging array.

It is obvious from the array size trajectory over the past few years that large scale arrays are possible, but as array size increases, maintaining on-chip histogram creation becomes challenging for large dynamic ranges of 0-300 m and beyond. Therefore, if read-out limitations can be overcome for high resolution arrays, large scale processing capable of high bandwidth and rapid processing is required at a massively parallel scale for typical video frame rates or above ( $> 30$  Hz). This is ideally done on the full histograms or LiDAR waveform for each pixel of the array, which will be discussed in more detail Section 2.5, as it provides the most information the system can provide [17]. To address this particular problem, this work investigates a high throughput approach using deep learning techniques in Chapter 4.

Alternatively, if the sampling methodology was to be adjusted and histograms were accumulated for sub arrays or circumvented altogether in conjunction with flexible illumination schemes, the bandwidth required to read-out ever increasing array sizes could become more feasible. This motivated the investigation of applying compressive sensing (CS) [75, 76, 77] principles to automotive LiDAR applications with a large dynamic range for solid-state SPAD arrays as presented in Chapter 5 to reduce sample and processing time as well as illumination power.

### 2.3.3 Laser Power Constraints

For uniform illumination of a direct ToF system and the ideal case of Lambertian objects with reflectivity  $\rho$ , the received power,  $P_{in}$ , at a single detector is related to the emitted power,  $P_{out}$ , from the radar equation [78] as

$$P_{in} = \frac{P_{out} D^2}{4\pi r^4 d_b^2} \rho, \quad (2.8)$$

where  $D$  is the aperture of the collection lens,  $r$  the operating range and  $d_b$  the beam diameter of the outgoing laser pulse. For a constant field-of-view (FoV) an increase in resolution, i.e. pixel count in the array, generally results in a decrease in aperture and beam diameter as the emitter array scales similarly. This means that less light can be collected. This is worsened as laser power further decreases inversely to the power of 4 with respect to operating range. Although a smaller beam diameter also inversely squares and increases the power density marginally, this illustrates the significant challenges associated with long range high resolution LiDAR and in particular for solid-state implementations relying on per pixel or uniform scene illumination (often referred to as burst or flash illumination). Not only is power

lost due to smaller apertures but more importantly to effectively capture returns from several hundred meters away, a high laser power would be required, which can interact with surfaces and other optical systems such as cameras and most critically the human eye and can be dangerous at shorter ranges.

This has led to many of the upcoming automotive solid-state LiDAR systems to adopt a solid-state scanning approach, which decreases the emitter power density to a single line at any given time. This imposes limits on the sampling speed and is an approach which is contrary to realising the full potential of a solid-state array sensor architecture.

As current systems are only of modest array size this might be an acceptable compromise, but if future LiDARs are to match high definition camera resolutions and beyond at several mega-pixels ( $> 10^6$ ) the scanning time will become a significant limitation on performance. The required laser power per pixel will need to be as high as permissible within eye-safety constraints. While laser budget can be increased by means of chosen emission wavelength, detector cost outside of Silicon manufacture is still prohibitive. This work will address these constraints by exploring novel sparse illumination schemes and associated depth reconstruction algorithms, which can also boost resolution for fewer detector elements enabling high resolution not only via large scale arrays but also for smaller arrays with exotic and costly materials. This work is outlined in Chapter 5.

To further illustrate the sampling limitations of current and future systems, the most common system architectures of current LiDAR systems are presented and discussed in terms of frame time from a purely sampling budget point of view, noting that processing budget is also significant and in most cases more costly than the acquisition time budget.

## 2.4 LiDAR Architectures

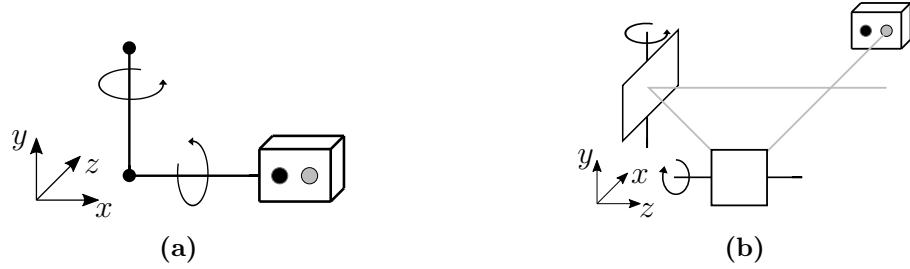
In this section illustrations of the most common scanning and sampling schemes are presented for LiDAR systems. The analysis will focus on frame time budget as defined by the sampling speed in terms of array resolution, which can limit frame rates dramatically to a point where any further processing will adversely affect frame rate making video-like frame rates of 30 Hz and above very challenging for many of the scanning architectures.

### 2.4.1 Mechanical Scanning Systems

The first generation of imaging LiDAR systems can be generally classified as mechanically scanning. Most early research arrangements relied on mechanical systems to scan the scene in the  $x$ - $y$  plane by steering the beam and/or the detector, with the scanning step size defining both vertical and horizontal resolution. Another popular method is a line of emitters and detectors (i.e a one-dimensional array) being rotated continuously at discrete time steps.

#### Point scan

A point scanning system generally relies on either a gimble type scanning system, where the emitter and detector are both moved to scan the scene as shown in Figure 2.6(a). These types of arrangements are still found in state-of-the-art space LiDARs [79]. They provide robust performance, as they can compensate for shock, but due to their moving parts and tight tolerances are very expensive. Other popular prototypes use beam steering by means of moving mirrors (e.g. mirror galvanometers) shown in Figure 2.6(b) or by moving the entire system on a  $x$ - $y$  stage [80].



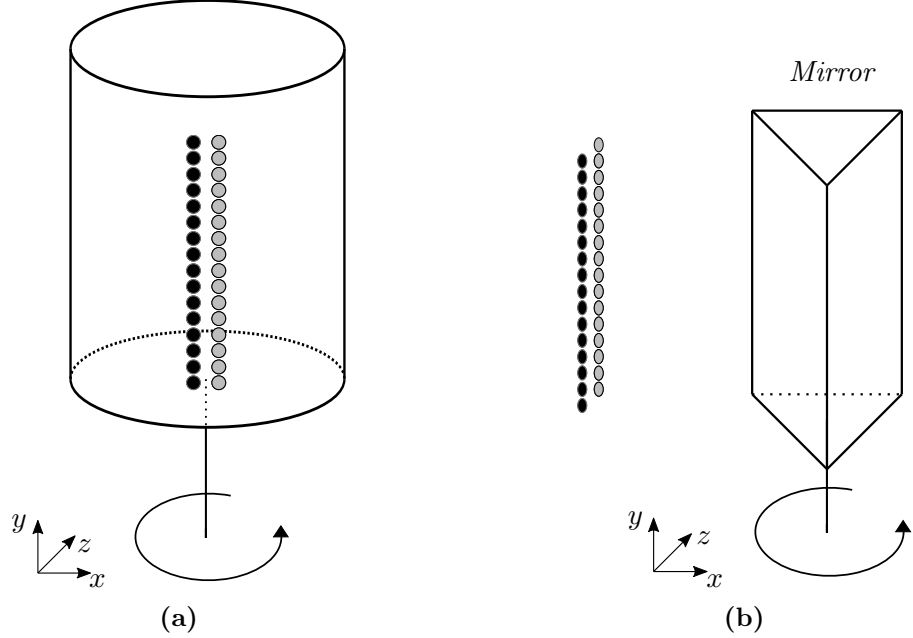
**Figure 2.6:** Illustration of point scanning LiDAR with arrows for moving axis, (a) gimble type and (b) galvo mirror type.

In a point scan system, a single coherent photon emitter,  $t_x$ , is paired with a single pixel photon detector,  $r_x$ . To enable a spatial resolution beyond a single point, the laser beam is optically steered across the scene in the  $x, y$  plane of size  $n_x \times n_y$ . This can be facilitated by physically rotating the laser-detector pair with a two axis rotating arm or by steering the laser beam using two mirrors which are mechanically actuated. This process is fairly slow and is unsuitable for real-time high resolution LiDAR imaging. The frame sample time  $F_t$  is defined by the number of points in the  $x, y$  scan, the number of laser pulses required to overcome ambient and device noise for a robust SNR governed by the pulse repetition rate,  $PRR$  and the maximum system range i.e. the time-of-flight for the maximum range,  $\delta t_{max}$ , such that

$$(F_t)_{\text{point}} = \frac{n_x n_y \delta t_{max}}{PRR}. \quad (2.9)$$

### Line scan

The most common type of commercially available LiDAR system for the initial autonomous prototype vehicles [8, 7, 81] is the mechanical line scan as illustrated in Figure 2.7.



**Figure 2.7:** Illustration of mechanical scanning lidar with arrows for moving axis, (a) rotation type, where lasers and detectors are physically rotated and (b) mirror spin type, where a stationary laser and emitter line is swept directionally with the mirror rotation.

In a mechanical line scan, a system is comprised of multiple laser-detector pairs often stacked vertically, which are then rotated mechanically around the  $y$ -axis. It can also be optically steered using a rotating mirror, with a stationary linear array. The vertical resolution,  $r_y$  is therefore defined by the number of pixels,  $n_y$  or channels and the optical system capturing a slice of the scene with a system specific field-of-view (FoV). Each pixel often has a vertical optical resolution in degree,

$$r_y = \frac{\text{FoV}_y}{n_y}. \quad (2.10)$$

The horizontal resolution is defined by line sample time and the rotational or scan speed of the system, which we shall define via the frame rate of the system,  $f$ , being the number of full scans per second. This in turn dictates the number of horizontal sample points,

$$n_x = \frac{\delta t_{max}}{f \cdot PRR} \quad (2.11)$$



The frame time is therefore

$$(F_t)_{\text{line}} = \frac{n_x \delta t_{\text{max}}}{PRR}. \quad (2.12)$$

This decreases the frame time proportionally to the stationary axis compared to a point scan system. In terms of the sampling signal model, the emission is aligned with the number of photon detector pixels,  $n$ , and is therefore for a single line

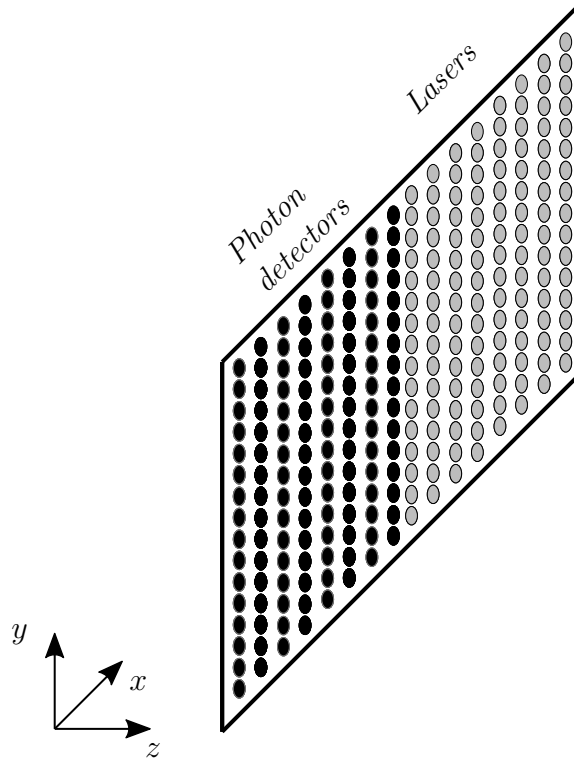
$$T_x = [1, \dots, 1] \in \{1\}^n. \quad (2.13)$$

The corresponding photon detector line captures histograms at each pixel location, thus

$$R_x = [h_1, \dots, h_n] \in \mathbb{N}^{n \times k}. \quad (2.14)$$

### 2.4.2 Solid-State LiDAR

Full solid-state systems are taking advantage of advances made in detector arrays as outlined in Section 2.3.2 and in solid-state laser arrays, in particular VCSEL arrays [19, 20, 2]. This enables a fully solid-state LiDAR architecture as illustrated in Figure 2.8.



**Figure 2.8:** Solid-state LiDAR with full photon and laser array.

For a fully solid-state array LiDAR, the transmitter usually illuminates all respective pixels. In the case that the entire scene can be illuminated at once, the transmitter and receiver are therefore

$$T_x = \begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix} \in \{1\}^{n_x \times n_y} \quad R_x = \begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix} \in \{1\}^{n_x \times n_y} \quad (2.15)$$

This is often called flash or burst illumination LiDAR. It is obvious that for this kind of illumination and sampling scheme, the amount of data for each frame when capturing full waveforms will require a high bandwidth sampling interface and a large amount of memory to store the sampling data cube and perform any additional processing steps. The dimensions of this data cube are

$$D_{\text{array}} \in \mathbb{N}^{n_x \times n_y \times p}, \quad (2.16)$$

with  $p$  number of histogram bins as before. This is the fastest sampling scheme available with a frame sampling time of

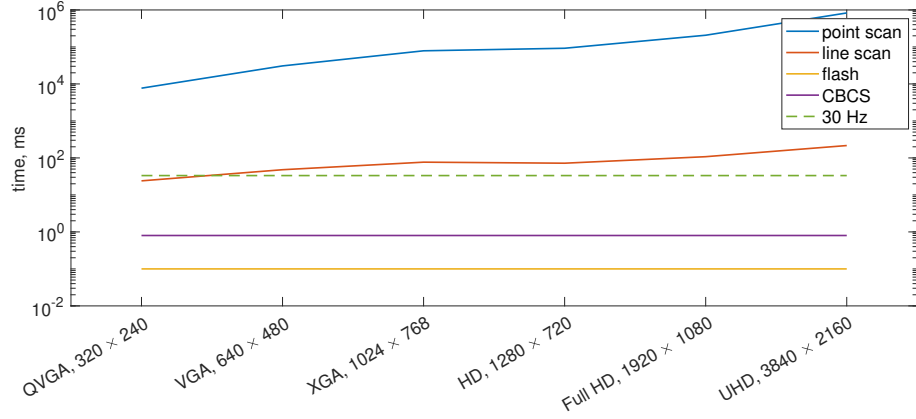
$$(F_t)_{\text{array}} = \frac{\delta t_{\text{max}}}{PRR}. \quad (2.17)$$

It is not only challenging from a data aspect, but it is also challenging from a safety perspective. Eye-safety regulations state a limit of permissible laser power per unit area. For short range applications, which require only moderate photon output, this is not an issue. For long range applications, where optical transmission loss ( $1/d^4$  [78]) is not negligible, this becomes a crucial design parameter and often means that for any usable FoV the laser power density becomes prohibitive for eye-safe operation. This has led to solid-scanning systems, which rather than spinning a line sensor, use a solid-state laser array to illuminate the scene in lines e.g. row wise,

$$(T_x)_1 = \begin{bmatrix} 1 & \dots & 1 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \\ 0 & \dots & 0 \end{bmatrix} \quad (T_x)_2 = \begin{bmatrix} 0 & \dots & 0 \\ 1 & \dots & 1 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \quad (T_x)_{n_y} = \begin{bmatrix} 0 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \\ 1 & \dots & 1 \end{bmatrix}. \quad (2.18)$$

$R_x$  will line scan in the same way aligned with  $T_x$ . The frame time is therefore the same as for mechanical line scanning systems albeit with the advantage of no moving parts and the scan direction is software defined. This however, does not take full advantage of the solid-state nature and the benefits of larger arrays as resolution

increases. The time it takes to sample each of the presented sampling behaviours is further illustrated in the graph shown in Figure 2.9, where any active point, line or full array is illuminated for  $100\text{ }\mu\text{s}$  each (50 laser pulses for 300 m range) for a variety of array resolutions. Lines are scanned across the array’s smallest dimension for the shortest possible sample time.



**Figure 2.9:** Sampling times of various sampling approaches for LiDAR system architectures, including the novel approach presented in this work, CBCS, with initial findings in [82]. A frame-time limit for 30 Hz is also shown.

This graphs illustrates, the benefit of flash and compressive illumination schemes, as the total illumination time remains constant as array sizes increase.

While current systems with modest resolutions can achieve sampling rates just below and up to 30 Hz, as resolution increases, normal sampling strategies limit the overall frame rate significantly, this applies to solid-state scanning and mechanical line scanning alike. This limits processing time significantly at low resolutions and prohibits fast frame rates above 30 Hz for higher resolutions. Flash illumination would be the best illumination and sampling scheme but is difficult to achieve for long range applications in the NIR spectrum. The approach presented in this work aims to tackle this problem and is further described in Chapter 5 by using a sparse multi-pattern sampling approach. This enables constant sampling rates and illumination densities similar to line scanning approaches, while being closer in sample time to full flash illumination.

### 2.4.3 Commercial LiDAR Systems

To give an idea of the current LiDAR market with respect to typical specifications and cost, an illustrative selection is provided in Table 2.3. These are mostly automotive systems, with the exception of the Riegl. This LiDAR system is included as it is one of the only readily available systems providing full-waveform output, while

all other systems pre-process the raw waveform internally with no end-user access. A more comprehensive review of current LiDAR systems can be found in [83].

**Table 2.3:** Typical selection of available scanning LiDAR systems

|                        | Velodyne<br>AlphaPrime [84] | Velodyne<br>VLP-16 [85] | Ibeo<br>LUX 4L [86] | Ouster<br>OS2-32 [87] | Riegl<br>VUX-1LR [88] |
|------------------------|-----------------------------|-------------------------|---------------------|-----------------------|-----------------------|
| Pixels<br>Scan         | 1 × 128<br>rotate           | 1 × 16<br>rotate        | 1 × 4<br>mirror     | 1 × 32<br>rotate      | 1 × 1<br>mirror       |
| FoV, °                 | 360 × 41.33                 | 360 × 30                | 110 × 3.2           | 360 × 22.5            | 330 × 0.02            |
| Vertical Resolution, ° | 0.2                         | 2                       | 0.8                 | 0.7/0.35/0.18         | 0.004                 |
| Range (max@10%), m     | 220                         | 100                     | 50                  | 120                   | 110-820               |
| Range Resolution, cm   | 3                           | 3                       | 4                   | 0.3                   | 1.5                   |
| Effective Resolution   | 1800 × 128                  | 720 × 16                | 138 × 4             | 512/1024/2048 × 32    | n/a                   |
| Wavelength, nm         | 903                         | 903                     | 905                 | 865                   | NIR                   |
| Frame rate, Hz         | 5 – 20                      | 5 – 20                  | 25                  | 10, 20                | 10 – 200              |
| Full-Waveform          | No                          | No                      | No                  | No                    | Yes                   |
| Peak output            | First, Max, Last            | Strongest, Last         | First, Max, Last    | Max                   | n/a                   |
| Price, \$              | 100,000                     | 4,000                   | 10,000              | 16,000                | 50,000                |

The selection provides a broad overview of the compromises present in available LiDAR sensors. For high resolution systems the cost is extremely high and even the cheaper systems are still cost prohibitive for mass deployment in consumer applications. Frame rates are also lower than normal camera systems in most cases and range is also fairly limited and often associated with the lower frame rate specified. Such low frame rates will limit the use cases for these systems, making motorway applications more challenging, which generally require the fastest response time possible. Most systems also only report the strongest return, which may not be the first true return in some cases. However, the more advanced systems do also provide first and last return.

In terms of solid-state LiDAR, despite a large number of start-ups in the space, reliable specifications are not readily available, with only few companies providing a comprehensive datasheet of which two are summarised in Table 2.4.

**Table 2.4:** Close-to-market automotive solid-state LiDAR systems

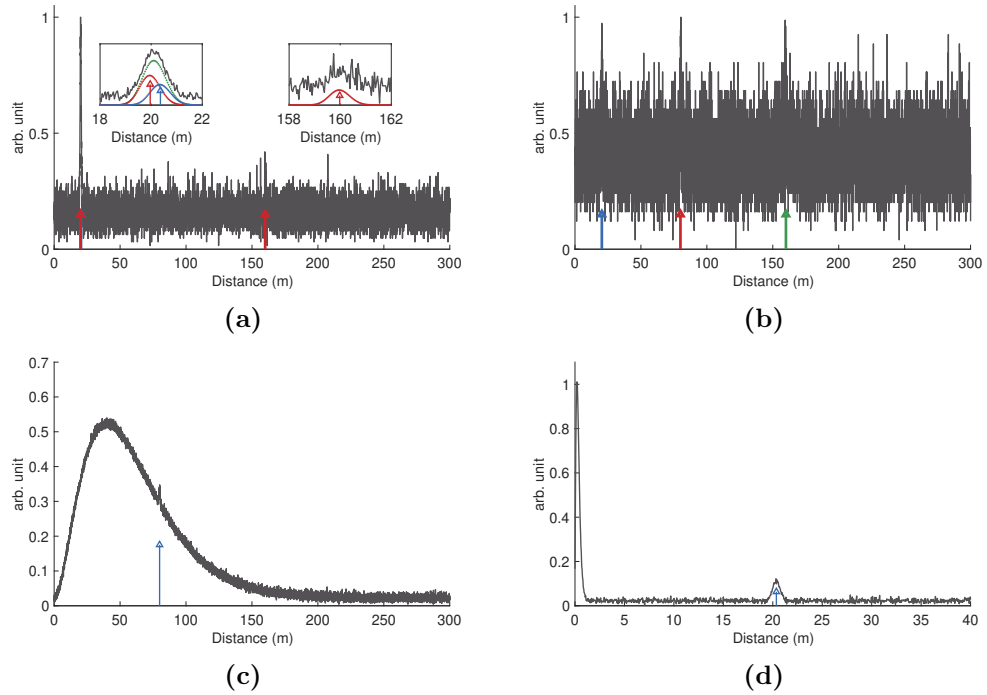
|                      | Ibeo<br>Next [89]                    | Ouster<br>ES2 [90]                    |
|----------------------|--------------------------------------|---------------------------------------|
| Pixels<br>Type       | 80 × 128<br>Solid-State (VCSEL+SPAD) | 260 × 130<br>Solid-State (VCSEL+SPAD) |
| FoV, °               | 11.2 – 60 × 41.33                    | 26 × 13                               |
| Range (max@10%), m   | ~ 250                                | 200                                   |
| Range Resolution, cm | 3                                    | 5                                     |
| Wavelength, nm       | NIR                                  | 905                                   |
| Frame rate, Hz       | 25                                   | 10 – 30                               |
| Full-Waveform        | No                                   | No                                    |
| Peak output          | n/a                                  | Strongest, Max, Last                  |
| Price aim, \$        | < 1,000                              | < 1,000                               |
| Release target       | 2020                                 | 2022                                  |

With pricing information being only speculation at this point, the only conclusion to draw is that there is an aim to reduce the price well below the cheapest scanning system, while providing significantly higher resolution for the same FoV. Both systems use VCSEL and SPAD technology and quote a fairly high maximum range in both cases. However, the frame rate still remains in the 10-30 Hz range,

illustrating how challenging high frame rate long distance imaging is due to high data volume generated and the laser power limitations potentially driving up sample time.

## 2.5 LiDAR Waveform

Although the storage and processing of a full LiDAR waveform is generally expensive, it can provide additional information beyond a first return [91, 92, 93, 31] and is particularly useful in challenging situations with high ambient noise or due to adverse weather conditions [94, 17] and can also provide information about scene content as well as mapping information [95]. Typical waveforms for various scenarios are shown in Figure 2.10, which have been simulated for illustrative purposes only.



**Figure 2.10:** Considered for this work: (a) Multi-path returns contained in waveforms with moderate noise. (b) Very noisy waveforms with multiple returns. And for illustration but not addressed in this work: (c) a waveform with a target in fog and (d) sensor artefacts such as cross-talk, where a target appears due to internal sensor reflections. Custom waveforms were generated using the simulations tools from this work for illustration purposes.

It is therefore worthwhile investigating depth recovery from said waveform compatible with large scale SPAD ToF imaging arrays.

### 2.5.1 Waveform Signal Model

An ideal LiDAR measurement observing  $K$  surface returns can be described as a continuous signal being an cumulative sum of impulse returns, such that

$$h(t) = \sum_{k=0}^{K-1} \Gamma_k \delta(t - t_k), \quad (2.19)$$

where  $\delta$  is the Dirac distribution,  $\{\Gamma\}_{k=0}^{K-1}$  denotes the reflectivity of the  $k^{\text{th}}$  return and  $\{t_k\}_{k=0}^{K-1}$  the respective time delay [91, 96].

A practical LiDAR system utilises a photon sensitive device, of which some are considered in Section 2.3. In this work SPAD devices for single photon detection are primarily considered if not otherwise mentioned. The choice of detector alongside optics and overall system characteristics alters the shape and amplitude of the overall return signal for each of the  $k$  signal returns. For example for passive quenching SPADs, due to their dead time [91], their sharp impulse rising edge is followed by a slowly decaying tail, where the SPAD recharges. There are a plethora of further influences on the shape of the recorded return, the reflectivity, the atmospheric conditions and system loss to name a few [78]. It is therefore important to take as much of these unique features into account when modelling the signal. This is often done using a representative instrument response function (IRF) for a single return,  $\rho \in \mathbb{R}_+^p$ . A practical IRF can then be described as

$$\mathbf{r} = \delta \star \rho, \quad (2.20)$$

where  $\star$  is the discrete convolution operator. This leads to a photon rate distribution

$$c_i(t) = \sum_{k=0}^{K-1} (\Gamma_k \mathbf{r}(t - t_k)) + \beta, \quad (2.21)$$

for every pixel,  $i$ , where  $\beta$  is the background photon count rate. This distribution is then used to draw counts from the Poisson distribution  $\mathcal{P}(\cdot)$ . A full LiDAR waveform is then

$$f_i \sim \mathcal{P}(c_i), \quad (2.22)$$

where  $i \in \mathbb{N}^N$  indicates the  $i^{\text{th}}$  pixel out of a vectorised grid of photon detector pixels [91, 31, 33, 97, 17].

### 2.5.2 LiDAR Signal Processing

The shape of the waveform, in particular if capturing a wide field of view, can be seen as a scene signature. It can contain information about atmospheric conditions [94] and therefore some work has focused on classifying abstract information contained in the waveform [95] to identify typical urban objects. This approach was also applied to canopy and other remote sensing tasks [98, 99].

For depth extraction from the waveform on a per pixel basis, early work focused on finding the underlying parameter set and thus the ideal information including the time delays of returns and therefore depth. An estimate was iteratively fitted to the observation using maximum likelihood estimation (MLE) [80] and was later improved by using more complex methods exploiting Bayesian Statistics and Markov chains [31, 100]. While it is a very precise approach, which generalises well to varying dynamic ranges and works across a range of noise and photon count levels it is rather slow to compute. The approach was refined and also parallelised in work following the initial demonstrations [101, 102, 103]. Although achieving dramatic reductions in processing time for this approach, the methodology focuses on single waveform processing and would be challenging to scale up in a massively parallel fashion for large detector arrays.

The work was further extended for very sparse photon count data, but the issue of speed still remained with reported processing times of several hours for reasonable image frame sizes [32, 104, 105]. More recent work proposes down-sampling of the data to derive priors to improve and accelerate the full frame problem with good results [106]. However, it is curious to observe that most reconstructions occur in a fairly small window of few metres, albeit at large stand-off distance. Such small observations windows further constrain the reconstruction to a short range reconstruction, albeit at low photon count rates. Nonetheless, the processing time per pixel has been improved to a few seconds from several minutes in prior work.

Rather than utilising Bayesian methods, another approach was to utilise finite-rate-of-innovation (FRI) principles [107, 108] and fast computational methods [109] to unmix the signal in the Fourier domain [110]. While this approach reports very fast processing time, and if applied on a per pixel basis would be suitable for real-time applications, the framework requires significant prior information in particular the number of peaks and has not been demonstrated under high ambient noise. However, it is one of the faster approaches with good parameter estimation in suitable low noise applications.

Other approaches consider spatial correlation and sparsity regularisation to con-

strain the problem formulation [111, 112, 113, 114, 115]. They exploit the signal statistics to enforce sparsity and promote some spatial correlation by considering neighbouring pixels. Recent work also achieves impressive run-times using graphical processing unit (GPU) acceleration [116]. However, a lot of the analysis is once again carried out on fairly small range windows of less than 10 m dynamic range. This constitutes a problem relaxation and might promote some of the spatial constraints considered in the first place, because the likelihood for dramatic range differentials is much less likely when only a small observance window of  $< 10$  m is considered. This might allow to make smoothness and surface assumptions due to the small observation window. The claim to long range imaging is somewhat misleading, as only a small window at a large stand-off distance (even at 100s of metres) is considered, the long range here only acts as a photon count attenuator and introduces optical challenges, but when only considering a small window the reconstruction problem is essentially still a short range problem. This confirms further the significant challenges associated with full range depth reconstruction from long range LiDAR waveforms.

## 2.6 Conclusions

This chapter has reviewed the fundamental light detection and ranging (LiDAR) principles and some of the major challenges associated with optical depth measurements using time-of-flight (ToF) principles for self-driving cars. Some current and future commercial LiDAR systems are reviewed illustrating the current prohibitive cost of the technology for mass deployment. An introduction to ToF was provided with a formulation for typical LiDAR depth outputs. Further, a LiDAR signal model for the full-waveform is provided, which can provide more information, particularly in challenging conditions important for full autonomy.

To enable low-cost fully autonomous cars novel sensors are required, which provide high resolution and high frame rates in a robust package, which in turn calls for solid-state systems which can be mass manufactured. Major developments in silicon single photon avalanche detector (SPAD) technology have enabled ever larger SPAD arrays. This development was reviewed with a brief introduction into the fundamental workings of the avalanche photodiode. SPAD arrays trend towards denser and more efficient arrays, making them ideal candidates for future automotive LiDAR systems.

Current LiDAR signal processing techniques are discussed in the context of larger ToF arrays noting in particular the often long processing times and some of the



rather limiting assumptions to relax the problems for more efficient processing approaches. This is often coupled with short dynamic range data or windowed approaches, which if extended to full range applications would incur a significant time and processing penalty.

This motivates the need for novel processing techniques to process the drastic increase in measurements as large ToF imaging array become a reality. To address this particular issue, an investigation into a parallel processing approach using deep learning is carried out in Chapter 4.

Typical LiDAR architectures are illustrated alongside a sample time analysis. This section concludes that with ever increasing resolution, the sample time with any kind of scanning approach is a limiting factor. Although flash illumination would be the best sampling case in terms of speed, it is not possible for Silicon compatible wavelengths due to eye-safety concerns. This motivates an investigation into a novel sparse sampling approach to LiDAR processing which both improves the sampling speed with low illumination density, which further reduces sample bandwidth by exploiting compressive sensing (CS) principles presented in Chapter 5 with efficient hardware considerations of this approach being explored in Chapter 6.

# Chapter 3

## LiDAR Signal Simulation

### 3.1 Introduction

Developing frameworks and algorithms for future sensor systems requires realistic data to verify and evaluate the approaches as well as any underlying assumptions. While several companies are currently developing solid-state light detection and ranging (LiDAR) systems, it is very difficult to gain access to development systems which could provide raw unprocessed data. Although several datasets exist for point cloud data, the final output of a LiDAR system, such as [117, 118, 83], datasets of raw photon count data for complex scenes at high resolution do not seem to be publicly available at the time of writing. To enable a thorough investigation of algorithms which can process raw photon count data at scale it is imperative to work with suitable raw data. In the absence of a real system, the best option is to simulate such data.

Photon count simulation and the modelling of full LiDAR waveforms is an active field of research (e.g. [92, 93, 119, 120, 121]) and thus the shape and characteristics of LiDAR returns are fairly well understood and suitable methodologies to generate photon count rates will be introduced in this chapter. Another significant aspect of this work is to provide methods for depth reconstruction at scale for an autonomous vehicle. This means that sequences of data are desirable but more importantly the scene composition should be as close as possible to the envisioned application. This means that the type of scenes to be considered should be as close as possible to real life driving environments ranging from highways to dense urban environments with many actors and objects in the scene. An approach is presented which allows leverage of synthetic and real depth datasets generated for the purpose of training and evaluating autonomous decision making and other visual processing tasks. Hence,

algorithms and code were developed in this work to generate photon count data at variable resolutions with feature rich underlying semantic data and ground truth. The ground truth enables straightforward evaluation of any framework developed. This also means that once the simulated raw data is generated, sampled and then processed, the final output can be compared to a ground truth with actual scene content making it possible to interpret results both quantitatively and qualitatively, by visual inspection.

While a real solid-state arrayed LiDAR system would be more desirable than simulating photon count data, this approach enables investigations into novel sensing and sampling schemes, which may not be possible with existing systems. And although this work had access to a few LiDAR sampling cubes from a point scan LiDAR system, they only contain information over a small range window for very specific non-automotive applications. Further, due to the abundance of rich depth image datasets, the volume of underlying data available allows the investigation of modern machine learning approaches making use of big data. This work takes full advantage of these enabling capabilities of simulating raw LiDAR system signals rather than working with pre-defined sampling and system limitations.

## 3.2 Datasets

The form of depth data available can be broadly put into three categories: point cloud datasets, depth datasets and depth generation from 3D software. While point cloud datasets provide a great resource for decision making algorithm development further down the processing flow of autonomous compute systems, they are a highly processed form of depth data and a lot of information from the raw LiDAR waveform is discarded. This work aims to output such data but perform processing on raw LiDAR sensor sampling data. To provide suitable ground truth two types of depth datasets were identified, one being synthetic i.e. computer generated depth scenes [122, 123] and the other being augmented real datasets containing depth, image and segmentation data [124]. Using a dense depth ground truth and augmenting this data with auxiliary information to supplement reflectivity information gives full flexibility in the simulation stack allowing for full control of sampling methodology, field-of-view (FoV) and capture resolution in all dimensions.

### 3.2.1 LiDAR Datasets

Although real full point-cloud datasets exist, they are locked to a specific sensor, which means that the raw data is not available and has been processed; often with

unknown assumptions from the manufacturer. This makes them a good resource for decision making and more advanced scene analysis, where a specific LiDAR sensor is expected to be part of the input. The very popular KITTI vision benchmark [117] was one of the first datasets which found widespread popularity for scene analysis and autonomous decision making as it provides various synced sensing modalities, which were state-of-the-art at the time it was published. A more recent and comprehensive LiDAR point cloud dataset is the Waymo point-cloud dataset [118], it provides a large amount of point-cloud frames making it a great choice for further point-cloud processing. There are many more such datasets as outlined in [83], but they are not suitable for this work, since the signal processing at sensor level is already performed. This means that the ground truth is often not available unless, in the best case, it is an estimate from a secondary sensor source and, being based on real systems, all their current limitations are imposed on the final output data provided in these datasets. One main limitation of most current commercial LiDAR systems aimed at the automotive space is the limit of point outputs per pixel, which is often only the first return.

Further, mechanical scanning systems, mostly used in these datasets, often have a fairly limited vertical resolution defined by the number of channels and with increasing channel numbers the cost of such mechanical LiDAR systems significantly increases due to expensive calibration and tight tolerances. This makes it difficult to up-scale such data to more high-resolution solid-state system specifications, which can enable a higher pixel density and does not suffer from difficult calibration processes due to having no moving parts. It is therefore desirable to have dense underlying depth data with known ground truth for flexible raw LiDAR waveform generation, with full control over assumptions and subsequent signal processing.

### 3.2.2 Synthetic Depth Data

As the objective of this work is to develop signal processing algorithms for solid-state LiDAR arrays, the usage of output data from scanning systems seems counter-intuitive. They provide a very different field-of-view and the sampling behaviour is fixed and defined by its underlying mechanical system. Further, it is desirable to generate raw photon count data with flexibility for solid-state single photon avalanche detector (SPAD) array specifications rather than other detector modalities in pre-determined configurations. To generate full waveform data for solid-state SPAD arrays based on feature rich scenes, some form of underlying ideal depth and reflectivity data is required. A few different approaches exist to generate such data for the desired application of autonomous driving sensor systems.

A full 3D scene simulator based on a 3D game engine can be employed to generate custom scenes and animate a camera pathway through a pre-defined scene. This approach has significant drawbacks due to the extremely time consuming aspect of content creation. Luckily, this is an active area of research and some open-source autonomous driving simulators exist, such as CARLA [125]. These can be modified, to implement sub-routines and create sensor models placed on vehicles in the simulated world to extract relevant depth, speed, acceleration, video information and other measurements to generate ground truth data. One particular advantage to this approach is the ability to capture information at any frame rate. However, such involved methods are only necessary if huge amounts of data at a specific frame rate are required. Despite being a very flexible approach, it is still very time consuming to generate custom pathways, a well defined sensor system in the selected 3D game engine and significant computational resources are required to perform ray-tracing on-top of the 3D scene generation.

Another more computationally efficient approach is to use existing feature rich datasets, which not only provide depth information but also intensity and scene segmentation. Two such synthetic datasets are available delivering scenarios comparable to the desired application: Virtual KITTI [122] and SYNTHIA [123]. Virtual KITTI is a recreation of the original KITTI vision benchmark [117] at a much higher resolution in particular for the depth data, which was lacking due to the employment of a first-generation Velodyne LiDAR system at the time. Additionally an indoor depth dataset was chosen as a depth information source to evaluate indoor short range performance [124]. One downside to this approach is the lack of multiple returns in the presence of opaque materials or wide laser beams. To mitigate this, an approach is being proposed which enables to generate multiple pseudo-returns from the high resolution of the base data by down-sampling to the final spatial resolution of most LiDAR sensors but retaining all depth information. This is presented in section 3.4.2 in more detail. Some key specifications of the three datasets are presented in Table 3.1.

### 3.3 LiDAR Waveform Simulation

The simulation of LiDAR waveforms (e.g. see Figure 2.10 for an illustration) is useful, because many LiDAR applications are cost prohibitive to acquire real waveforms at scale. LiDAR was first applied in space applications [11] and has found more widespread application in aerospace applications. It is therefore not surprising that a lot of simulation work has focused on space and aerial LiDAR applications [92, 93, 119, 120, 121]. In combination with the well studied signal model of LiDAR

**Table 3.1:** Synthetic and augmented real depth datasets for LiDAR waveform data simulation.

| Parameter              | Symbol           | Virtual KITTI [122] | SYNTHIA [123]     | NYU Depth [124]     |
|------------------------|------------------|---------------------|-------------------|---------------------|
| Observation            |                  | synthetic           | synthetic         | real                |
| Sensor                 |                  | -                   | -                 | Microsoft Kinect V1 |
| Total number of frames |                  | 17000               | 200000            | 1449                |
| Scene types            |                  | automotive          | automotive        | indoor              |
| Resolution             | $N_x \times N_y$ | $1242 \times 375$   | $1280 \times 760$ | $640 \times 480$    |
| Depth Ground Truth     | $D$              | ideal               | ideal             | estimate            |
| RGB                    | $I$              | yes                 | yes (stereo)      | yes                 |
| Multiple viewpoints    |                  | yes                 | yes               | no                  |
| Weather                |                  | yes                 | yes               | no                  |
| Daytime cycle          |                  | yes                 | yes               | no                  |
| Class Segmentation     | $S$              | yes                 | yes               | yes                 |
| Focal length in $x$    | $f_x$            | 725.00              | 532.74            | 582.62              |
| Focal length in $y$    | $f_y$            | 725.00              | 532.74            | 582.69              |
| Optical centre in $x$  | $c_x$            | 620.50              | 580.00            | 313.04              |
| Optical centre in $y$  | $c_y$            | 187.00              | 320.00            | 238.44              |
| Value Scaling          | $w$              | 100                 | 100               | 1                   |

signals, simulating waveforms is a reasonable approach to prototype and model future systems in the absence of prototypes or to produce data at scale without costly acquisition trials. The cost of trials is not only prohibitive for aerospace applications but is equivalently prohibitive at scale for automotive applications. Many scenes are ideally observed in varying conditions making the acquisition of comprehensive real datasets a time consuming task. It is not surprising that even if such datasets exist from major companies in the field, that these are not publicly available as they are an extremely valuable asset.

The aerospace focused simulation frameworks can not only include atmospheric effects but also surface and material properties. The simulation approaches are often extremely complex and time consuming still, because most are designed for the current state-of-the art of scanning laser systems with fewer detectors. This work aims to provide a suitable simulation chain for solid-state LiDAR prototyping, which provides the facility to model the sampling behaviour alongside robust yet fast waveform generation. This establishes a convenient framework to generate large waveform datasets of entire scenes and scene sequences at variable resolutions.

### 3.3.1 Photon Count Rate Models

The LiDAR waveform signal model defined in equation (2.22) requires a mean photon count distribution  $c$  for each pixel made up of a mean count for returns and ambient conditions. The photon counts are then drawn from a Poisson process according to the mean compound photon distribution i.e. a mean photon count value

for each waveform bin containing signal and noise contributions. This work considers two photon count models for surface returns. The first is focused on aerial LiDAR applications with the main parameter being output power and a compact atmospheric model. The other is from the solid-state photonics community, which focuses on photon count rates for SPAD devices with more simplistic atmospheric models.

### Airborne Laser Scan Waveforms

A reference point for laser power propagation in airborne LiDAR systems is the work by [92, 93, 126]. This puts forward one path to obtain energy and power estimates for a LiDAR system, which is briefly summarised below.

Let  $e_\lambda = \frac{hc}{\lambda}$  be the energy of a photon at wavelength,  $\lambda$ , with  $c$  the speed of light and  $h$  the Planck constant, the photon return rate depends on the spot size,  $d$  incident on the active area of the photon detector device, the transmitted power,  $P_{\text{out}}$ , the reflectivity of the surface,  $\rho$ , (this includes geometric loss here too) and atmospheric loss over the round trip distance,  $r$ , the received power is then in its simplest form

$$P_{\text{in}} = \frac{P_{\text{out}} d^2}{16r^2} \eta_{\text{sys}} \eta_{\text{atm}} \rho, \quad (3.1)$$

where  $\eta_{\text{sys}}$  and  $\eta_{\text{atm}}$  are transmission factors for system and atmosphere respectively [126].

The mean photon count per laser pulse is then

$$c_{\text{pulse}} = \frac{P_{\text{in}}}{e_\lambda} \text{FWHM}, \quad (3.2)$$

with the full-width-at-half-maximum (FWHM) of the laser pulse i.e. the time envelope around the pulse centre. For an exposure time, defined by the desired frame rate,  $f$ , in Hz, the maximum incoming photons per illumination cycle, with a single pulse per time-of-flight (ToF),  $\delta t$ , is then

$$c_{\text{max}} = \frac{1/f}{\delta t} c_{\text{pulse}}. \quad (3.3)$$

### SPAD Time-of-Flight Arrays

This work considers some specific aspects unique to solid-state LiDAR arrays using SPAD detectors. Most SPAD pixels are made up of multiple individual SPADs, this configuration is often called silicon photomultiplier (SiPM) or macro-pixels (see Chapter 2.3.1). This design allows reduction of the effects of SPAD dead-time and

increases signal rate as it allows to capture more photons from a single timed laser pulse. Simulation models exist, which can quantify the optimal photon count rate,  $n_P$ , based on number of SPADs for SiPM pixel designs [41]. For the considered pixel design in [43], this work assumes 16 SPADs in each pixel making up the solid-state LiDAR array.

### 3.4 LiDAR Simulation Toolchain

The LiDAR simulation tools developed for this work will follow the solid-state approach with a SPAD array similar to [43] in mind to match the scope of this work better. Details of the system parameters and the signal irradiance values are provided in Table 3.2. In particular the irradiance values were provided by the authors of [43] in collaboration with optical engineers at STMicroelectronics and are derived from an optical model in the optics software ZeMax. Similarly, the solar irradiance is obtained from the same optical model and the value is normalised to 1 lux. Alternatively, photon count rates can be obtained by equations (3.2) for a more generic approach. To generate synthetic full-waveform data comparable to real data, the ideal depth data has to be processed to reflect the typical signal propagation in LiDAR. In general, this process can be described as a series of convolutions. A laser light source generates pulsed waveforms at a specified pulse-repetition-rate (PRR) of particular pulse-width, defined by its FWHM. This pulse-width dictates the uncertainty in any individual depth measurement. Another important specification limiting system performance is the light source itself, in particular the beam divergence and lasing power have significant implications on range and resolution. As such solid-state sensors are in concurrent development to the processing framework, some assumptions in this work are based upon an early published development sensor chip.

This allows to calculate a per bin mean photon count as shown in Equation 3.5. Let  $A_s = \frac{\pi d^2}{4}$  be the laser spot area, then the photon count expected at range  $r$  is

$$f_P(r) = \frac{A_s I_{ST}}{r^2 e_\lambda}, \quad (3.4)$$

with  $e_\lambda = \frac{hc}{\lambda}$  being the energy of a photon in 905 nm. The total photon count per bin for a pulse is then

$$c_P(r) = P \times \Gamma \times f_P(r) \times \text{FWHM}, \quad (3.5)$$

with  $\Gamma$  being the reflectance of a target. For a realistic scene to waveform transla-



**Table 3.2:** Solid-state LiDAR specifications based on development chip in [43] as a simulation test platform.

| Parameter                   | Symbol         | Value                    | Unit                          |
|-----------------------------|----------------|--------------------------|-------------------------------|
| Laser power                 | $P$            | 4.5                      | W                             |
| Wavelength                  | $\lambda$      | 905                      | nm                            |
| Signal Irradiance           | $I_{ST}$       | $857.266 \times 10^{-3}$ | $\frac{\text{W}}{\text{m}^2}$ |
| Photon detection efficiency | PDE            | 0.7                      |                               |
| Pulse width                 | FWHM           | 133.3                    | ps                            |
| Spot diameter               | $d$            | 50                       | $\mu\text{m}$                 |
| Beam divergence             | $\sigma$       | 0.5                      | $^\circ$                      |
| Maximum ambience            | $\Omega_{max}$ | 100                      | klux                          |
| Solar Irradiance            | $I_{solar}$    | $38 \times 10^{-6}$      | $\frac{\text{W}}{\text{m}^2}$ |
| Pixels in X                 | $N_x$          | 128                      |                               |
| Pixels in Y                 | $N_y$          | 128                      |                               |
| SPADs per pixel             | $n_{SPAD}$     | $16(4 \times 4)$         |                               |
| Pulse repetition rate       | PRR            | 500                      | kHz                           |
| Range                       | $r_{max}$      | 300                      | m                             |
| Sampling rate               | $S$            | 3.75                     | GHz                           |
| Bin width                   | $t_{bin}$      | 133.3                    | ps                            |

tion an estimate for reflectance is therefore required. This work will make use of segmentation maps of synthetic datasets to introduce an approximation of material properties and thus reflectance. The photon counts per bin are used to populate a histogram based on an underlying depth scene. The SPAD sampling behaviour is modelled with a focus on fast waveform generation, but the option exists to allow for a full SPAD sampling behaviour as used in [43]. The total count per bin is limited to  $n_P = \text{PDE} \times n_{SPAD}$  per exposure cycle  $\eta_P$  [41].

### Solar background photon rate

For a particular wavelength,  $\lambda$ , the sun's energy flux will have significant impact on the total photon rate often referred to as background count rate. The ambient count rate is derived from the solar irradiance at 905 nm,  $I_{solar}$  normalised to 1 lux. The background rate, i.e. total solar photons arriving per second, is then

$$f_{amb} = \frac{I_{solar}\Omega}{e_\lambda}, \quad (3.6)$$

where  $\Omega$  is solar influx in lux. This results in a mean background count  $\beta$  per histogram bin or the mean DC noise level, i.e. solar photons arriving in a bin time interval  $t_{bin}$  as

$$\beta = f_{amb} \times t_{bin}. \quad (3.7)$$

This allows to compute the combined bin count distribution  $c_i = c_P + \beta$  to generate a full waveform as per equation (2.22).

### 3.4.1 Synthetic scenes to Full-Waveforms

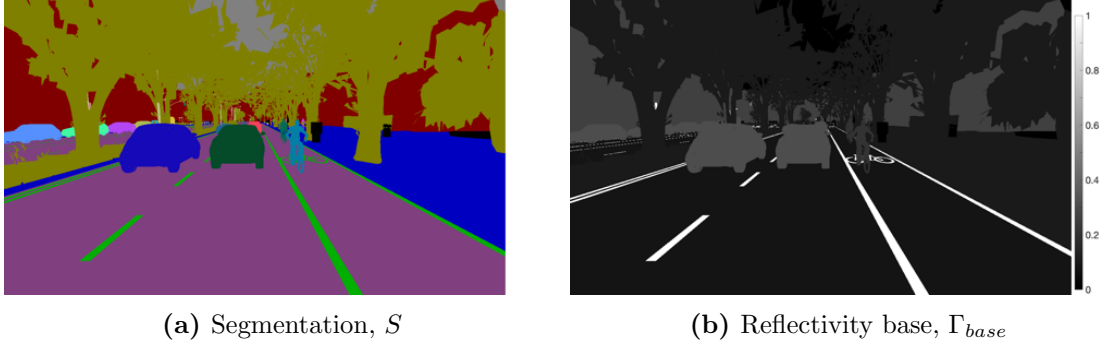
To facilitate a realistic signal-to-noise ratio (SNR) it is important to consider the reflectance,  $\Gamma$ , of objects in the scene. While material properties and full knowledge of the geometry could be embedded in full 3D engine based models (e.g. as an extension to [125]) and reflections can be realistically simulated using ray-tracing. This information is not available for the offline datasets outlined above. To mitigate this, some rudimentary Lambertian reflectance information for a scene is derived from the intensity,  $I$ , and segmentation map,  $S$ , to generate reflectance estimates for the known object classes within the datasets with base reflectivity values as indicated in Table 3.3. This makes various assumptions in terms of most common materials used for particular classes based upon common spectral libraries [127, 128]. In particular for the segmentation data for [124] some classes are estimated with variations around a particular material type. It is assumed that the sky yields no reflections.

**Table 3.3:** Estimates for reflectance for selection of defined classes within datasets.

| Class              | Reflectance $\Gamma_{base}$ |
|--------------------|-----------------------------|
| Sky                | 0.00                        |
| Terrain            | 0.17                        |
| Tree               | 0.15                        |
| Vegetation         | 0.25                        |
| Building           | 0.25                        |
| Road               | 0.08                        |
| Lanemarkings       | 1.00                        |
| Sidewalk           | 0.10                        |
| Guardrail          | 0.24                        |
| Fence              | 0.10                        |
| Traffic sign       | 1.00                        |
| Traffic light      | 0.50                        |
| Pole               | 0.25                        |
| Cars               | 0.30                        |
| Pedestrian         | 0.20                        |
| Cyclist            | 0.20                        |
| Textiles           | 0.1                         |
| Wooden surfaces    | 0.6                         |
| Lacquered surfaces | 0.9                         |
| Metal objects      | 0.8                         |
| Glass surfaces     | 0.1                         |
| Misc.              | 0.20                        |

To generate the base reflectivity map  $\Gamma_{base} \in \mathbb{R}^{n=N_x N_y}$  each class identifier is mapped to the reflectivity values above as shown in Figure 3.1.

To introduce some variance across objects in the final reflectivity map  $R \in \mathbb{R}^{n=N_x N_y}$  (e.g. tyres are not separate from a car in the source segmentation data but are

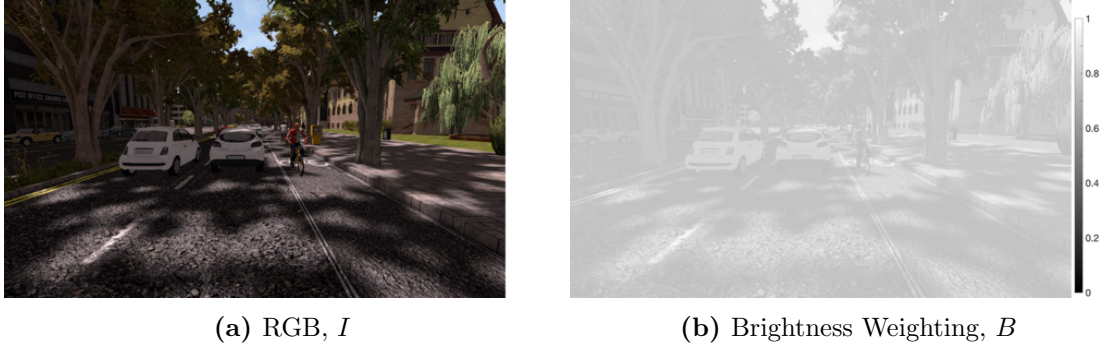


**Figure 3.1:** (a) any class defined in the segmentation map, here illustrated with unique colours with classes having multiple values, is mapped onto a reflectivity base in (b)

of lower reflectivity), the RGB image,  $I \in \mathbb{R}^{3N_x N_y}$  is converted into a brightness weighting map as  $B \in \mathbb{R}^{n=N_x N_y}$ , such that

$$B = \gamma \frac{r + g + b}{r_{max} + g_{max} + b_{max}} + (1 - \gamma), \quad (3.8)$$

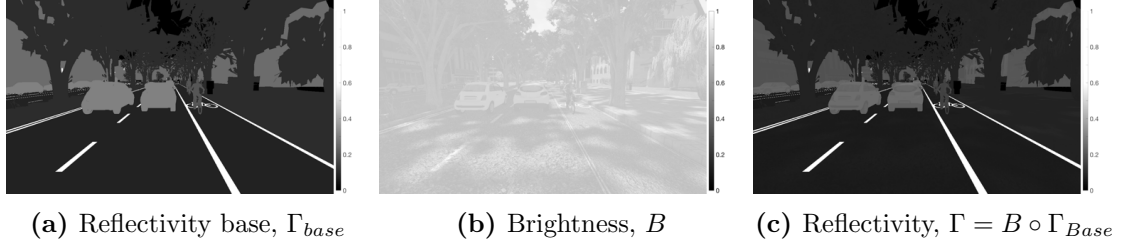
where  $r, g, b$  are the respective colour channel of  $I$  as shown in Figure 3.2 and  $\gamma$  is a weighting constant which allows to adjust the influence of brightness, in this case  $\gamma = 0.25$ .



**Figure 3.2:** The RGB image in (a) is converted in a brightness weighting map shown in (b) using the intensity values derived from the colour channels and a weighting constant.

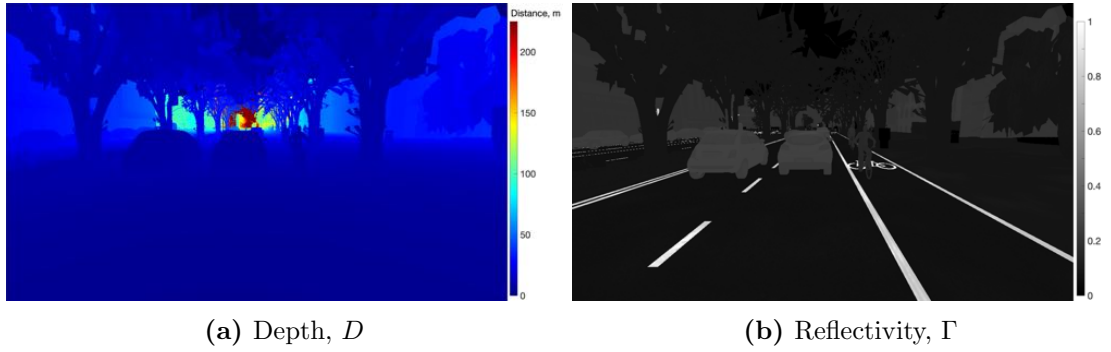
This assumes that bright objects will absorb less laser power than dark objects. This value is then used to scale the reflectivity values for each known class from the respective segmentation map to provide a more varied estimate for the expected photon count at the receiver. The steps to arrive at the reflectivity map for this example are shown in Figure 3.3 and is defined as

$$\Gamma = B \circ \Gamma_{base}. \quad (3.9)$$



**Figure 3.3:** The reflectivity base in (a) is multiplied with the brightness weighting in (b) to result in a reflectivity map in (c) with some object colour influence.

The reflectance information is pixel matched to the ground truth depth data and thus can be sampled in the same way as the depth scene. This makes it possible to simulate photon count rates with flexible sampling schemes with both known depth ground truth and rudimentary estimates of reflectivity for equation (3.5), where at pixel  $i$  the value for distance,  $r_i = D(i)$  for a vectorised depth map  $D \in \mathbb{R}^{n=N_x N_y}$  and  $\Gamma_i = \Gamma(i)$  for the vectorised reflectance map are known. An example of such maps for a scene is shown in Figure 3.4.



**Figure 3.4:** Finally, the depth map in (a) and the reflectivity map in (b) can be used to generate values for photon count rates.

It is important to realise that the depth images in these datasets are stored in the image pixel space rather than in 3D coordinates or spherical coordinates. To properly simulate the sensor sampling behaviour, the data has to be transformed into spherical coordinates to capture the depth as seen along a laser beam at an angle. To obtain the radial information for the observed depth from a LiDAR sensor, the dense orthographic depth data from the datasets is transformed into the XYZ domain using the pinhole camera model equations with parameters for each dataset.

presented in Table 3.1,

$$\begin{aligned} Z(x, y) &= \frac{d(x, y)}{w}, \\ X(x, y) &= \frac{(x - c_x)Z}{f_x}, \\ Y(x, y) &= \frac{(y - c_y)Z}{f_y}, \end{aligned}$$

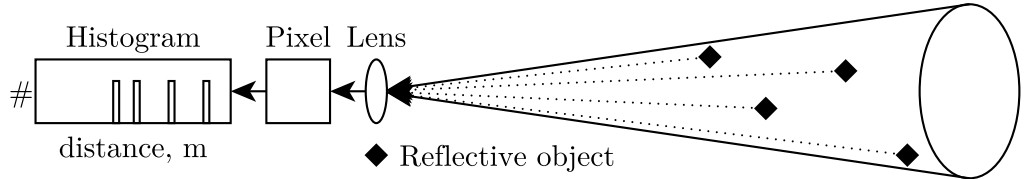
with the intrinsic camera parameters, focal length in  $x$  and  $y$ ,  $f_x, f_y$  respectively normalised to the depth scale factor,  $w$  and the optical centres,  $c_x$  and  $c_y$  as pixel positions. With 3D image coordinate data it is straightforward to represent the XYZ data in spherical coordinates with the desired radius information on a per pixel basis at their respective  $x$  and  $y$  positions in their original pixel representation,  $r(x, y)$  and the associated azimuth,  $\theta(x, y)$  and elevation  $\phi(x, y)$  angles. For convenience this thesis will use depth and radius interchangeably, but the default is radial depth information if not otherwise specified. This makes it possible to simulate the ToF as observed by a LiDAR sensor and its optical components. With the forward and backward transforms in place, this toolchain can readily generate point clouds from ground truth and recovered depth images or represent depth data in pixel representation.

### 3.4.2 Waveforms with Multiple Returns

Having defined the spherical calibration of the data capture, the tools presented in this chapter can also simulate multiple returns in a single waveform. In real LiDAR returns multi-path reflections can occur if an object is occluded by a transmissive material or if the beam width of the laser is wide enough to illuminate several surfaces in its line of sight. The main depth data source is a single dense layer of information and thus occlusion data is not readily available. To simulate transmissive objects in a laser path, a full 3D scene with material properties has to be simulated and ray-traced to capture this complex scenario. This is a very time consuming process and does not lend itself to frame sequences and prototyping of fast sample and processing stacks. It could be implemented for full 3D scene generators mentioned earlier to generate offline datasets, but this work will focus on multi-return generation from multiple surfaces being illuminated in the same optical path. This is a very fast and efficient and therefore does not add much time to the simulation frame time, but is also a more common scenario. In this case multiple scattering events are captured without any loss in transmissive layers, which makes it more probable to detect. For occluded object returns each scatter event with the transmissive media or layer will

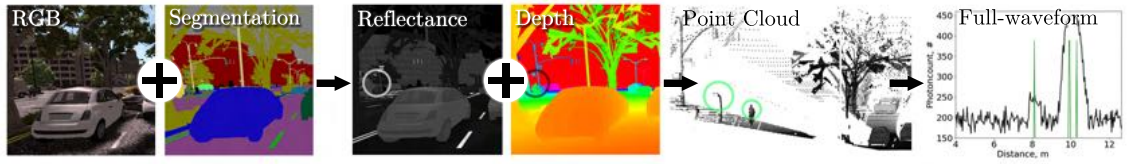
attenuate the signal both on forward and return path. This reduces photon count rates even further, reducing the likelihood of significant photon returns.

To simulate multi-surface returns, it is assumed that the beam has enough divergence over the entire range to generate multiple scatter events from different areas of a surface or multiple surfaces which fall within the optical path of the emission and detector pair. This is only possible if the final LiDAR array resolution is less than the initial depth images. This is illustrated in Figure 3.5



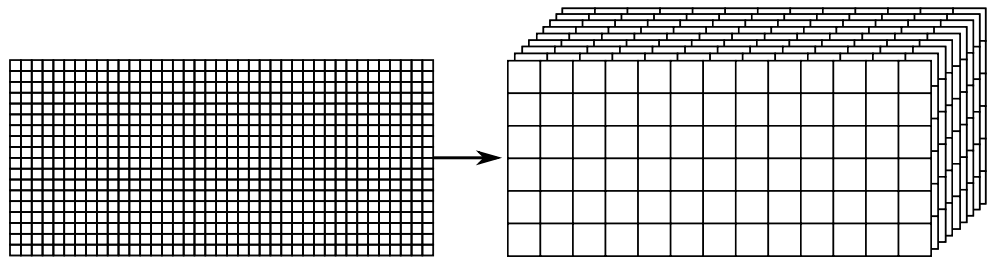
**Figure 3.5:** Multi-return generation from dense single return data

Combining this approach with the reflectance and depth information available, multiple returns can be sampled along the line-of-sight of pixel blocks as illustrated for the previously considered scene in Figure 3.6.



**Figure 3.6:** Using RGB and segmentation information, a reflectance map is generated which is combined with depth information and a multi-return sampling operator to generate multiple objects from reflections along a line-of-sight. This allows to simulate a full LiDAR waveform with multiple surface returns. Point cloud for illustration purposes of line-of-sight only.

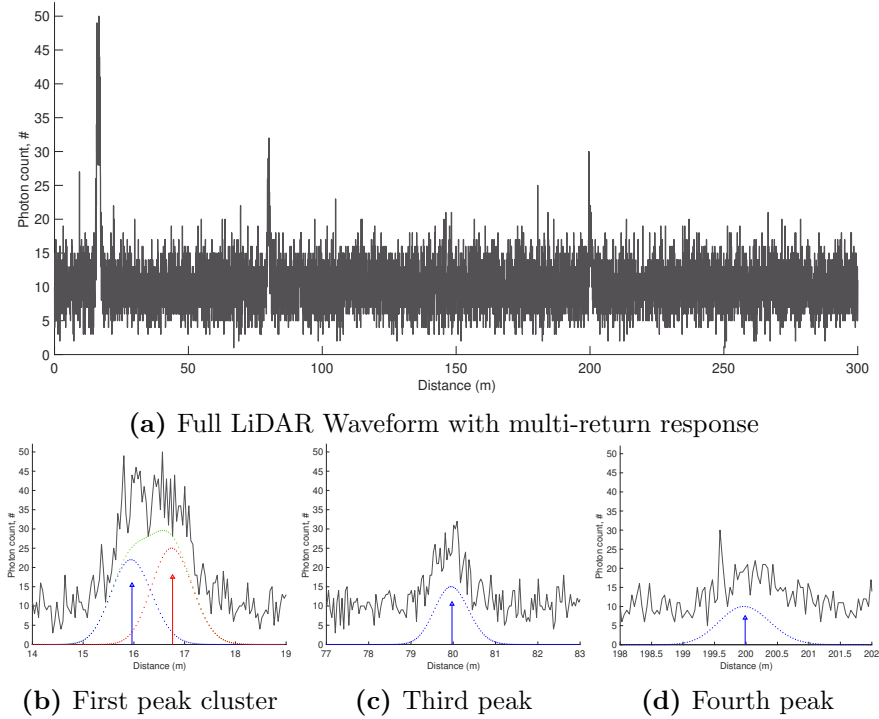
This can also be visualised as a transformation from a single high resolution depth map to a lower resolution depth cube as shown in Figure 3.7.



**Figure 3.7:** Multi-return generation from dense single return data

Mathematically, each depth value  $d_{x,y}$  in the dense depth map,  $D \in \mathbb{R}^{N_x \times N_y}$ , is grouped into macropixel vectors forming a lower resolution depth cube containing multi path information,  $D_{mp} \in \mathbb{R}^{n_D \times N_x/\varsigma \times N_y/\varsigma}$ , where  $n_D = \varsigma^2$  is the maximum number of depth returns in a macropixel and  $\varsigma$  is a spatial down-sampling factor. For example, for the first macropixel the depth vector is  $(D_{mp})_1 = D(1:\varsigma, 1:\varsigma)$ . It is important to note, the total number of depth values is not reduced and the same field-of-view is observed at the slight cost of spatial resolution. The beam shape is assumed to be square shaped if not otherwise mentioned to capture all base data conveniently. However, it is straightforward to implement other shapes for the re-sample operator. The exact same re-sampling is applied to the associated reflectance map,  $\Gamma$ , resulting in  $\Gamma_{mp} \in \mathbb{R}^{n_D \times N_x/\varsigma \times N_y/\varsigma}$ .

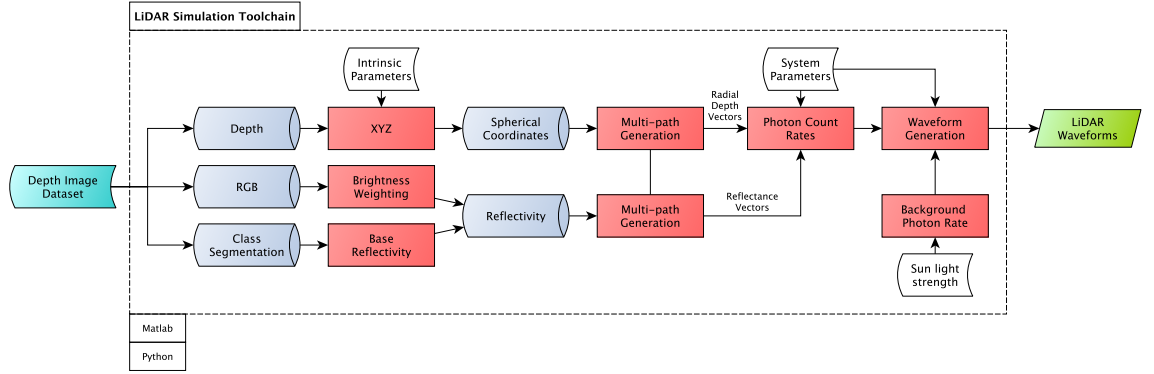
Once re-sampled, each macropixel can yield a multi-return LiDAR waveform for each macro-pixel vector pair. However, it is perfectly reasonable to obtain a seemingly single return if a planar surface is illuminated and all returns are at the same depth. If gradients or multiple surfaces are present, a multi-return waveform is generated as illustrated in Figure 3.8 for three surface clusters at different ranges with varying reflectance values.



**Figure 3.8:** An illustration of a simulated multi-return LiDAR waveform for three hand placed clusters using the simulation tools presented. The first cluster contains two merged peaks while the others contain single returns.

This method also accommodates the simulation of the LiDAR resolution problem, where multiple surface returns merge into a single peak. This LiDAR resolution problem is under active investigation [31, 96] and will be addressed in Chapter 4.

To summarise, the simulation toolchain presented thus far is capable of generating LiDAR histograms with user definable system parameters from synthetic high resolution depth data sequences with auxiliary data available to replicate material effects in a rudimentary fashion. The flow of the waveform simulation is shown in Figure 3.9. The toolchain allows large scale depth datasets to be converted into



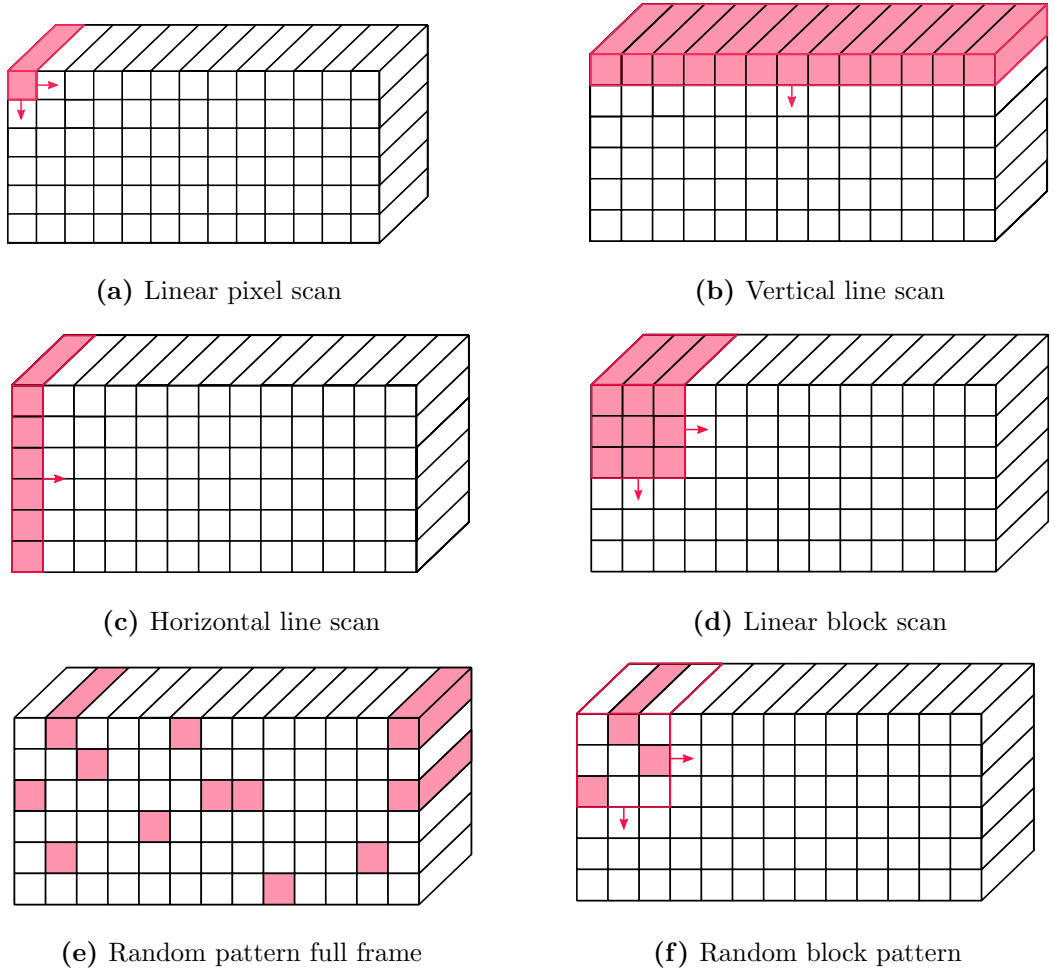
**Figure 3.9:** The full LiDAR simulation toolchain for conversion of dense depth data to multi-return histograms.

LiDAR waveform datasets, which enables the exploration of machine learning approaches to perform LiDAR waveform processing presented in Chapter 4. For this reason the toolchain was not only implemented in Matlab but also in Python, as machine and deep learning frameworks have well established development tools in Python rather than Matlab.

### 3.4.3 Non-Linear Sampling

With the capability of simulating full LiDAR waveforms, the sampling behaviour of the sensor is modelled. The simplest case is the full pixel scan, where each macropixel (or pixel) is sampled individually as shown in Figure 3.10(a). As previously discussed, this is a fairly slow process and does not take full advantage of a solid-state LiDAR array. Similarly, the depth cube can be sampled in blocks or lines as illustrated in Figure 3.10(a), which is the most common mode of operation of most current LiDAR systems.



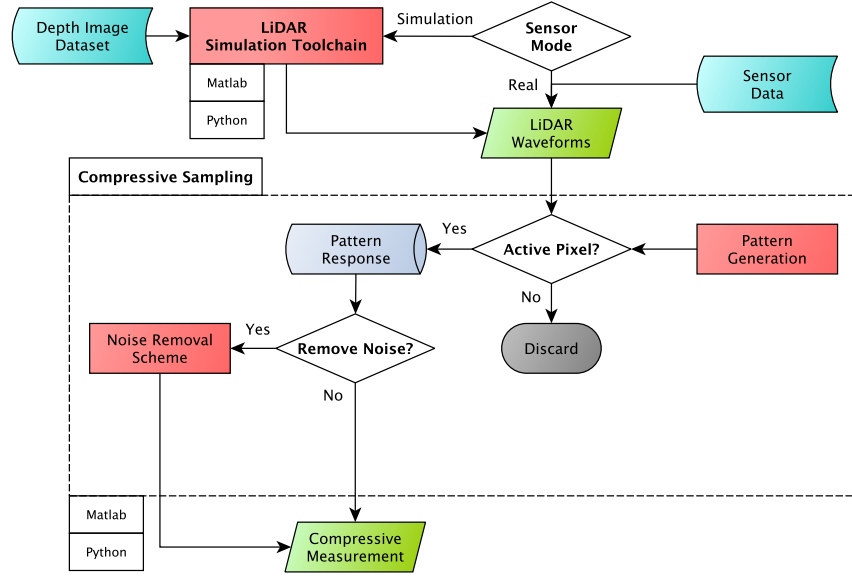


**Figure 3.10:** Sampling mode illustrations for simulated LiDAR sensors

Since the simulation is not constrained by existing system paradigms, the depth cube can also be sampled in non-linear patterns and paths. This enables the exploration of compressive sensing methods applied to LiDAR processing, which relies on often random or specific pattern sampling methods [77]. This will be further explored in Chapter 5. A typical random sampling instance is shown in Figure 3.10(c) and a blocked random sampling [129] instance is illustrated in Figure 3.10(d). The scene can be sampled randomly or with a specific basis function in mind and the block operator can be defined with and without overlap. In this work, the blocking default scheme is non-overlapping with pseudo-random pattern generation as outlined in Chapter 5.

This results in a simulation flow for compressive type sampling of LiDAR data. More details in general and in particular on the pattern generation and noise removal schemes are presented in Chapter 5.3.1. This is not only applicable to simulated data but can also be interfaced with full-frame real LiDAR waveform data as shown

in the flow diagram in Figure 3.11.

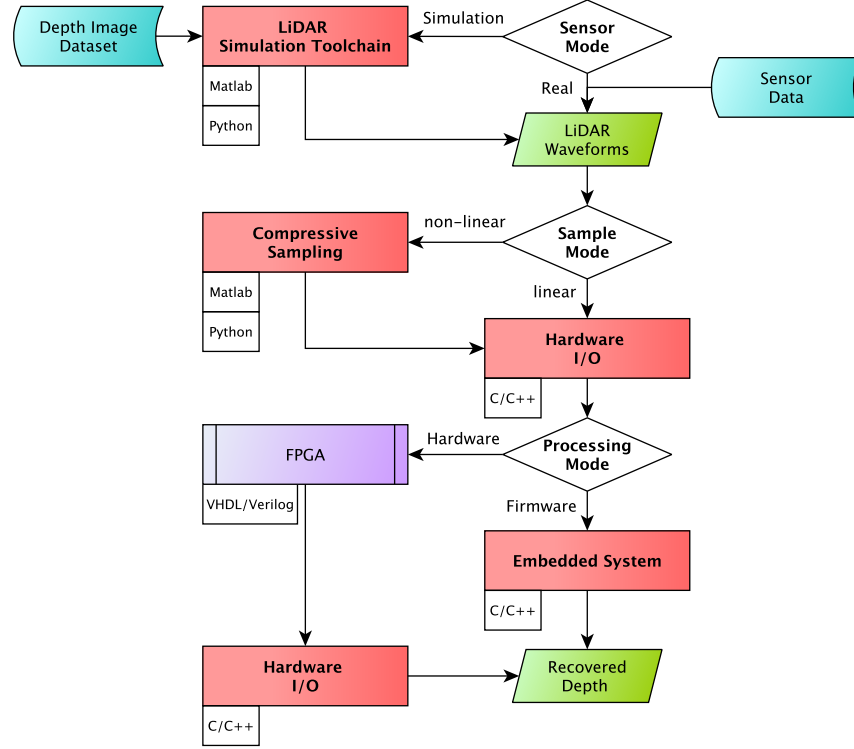


**Figure 3.11:** Sampling simulation flow.

### 3.4.4 Hardware Prototyping

With data generation and sampling behaviour in place, the last piece covered in the simulation suite developed for this work is the capability of interfacing all output data generated from this toolchain to interface with hardware compatible test-beds or indeed hardware directly. To cover a wide scope of hardware implementations, a C/C++ input and output (I/O) interface was created which is paired with export and import functions in Matlab to prepare data for hardware processing and re-import hardware output data for analysis. This enables testing of algorithms in the C environment often used for firmware and software running on microcontroller unit (MCU) or system-on-a-chip (SoC) devices, representative of embedded system implementations. The C interface has the capability to directly interface with a Virtex 7 based field-programmable gate array (FPGA) (AlphaData ADM-XRC-7V1) using a PCI-Express connection.

The I/O module on the FPGA was implemented as a sequential shift register. The shift register is then accessed by a logic state machine. This type of operational mode was tested by implementing a simple cross-correlator in Verilog using Vivado Synthesis tools. The output was verified against a Matlab and C implementation of the same algorithm. This hardware interface enables the exploration of more hardware specific optimisation such as the work presented in Chapter 6. The hardware capabilities of the simulation tools are shown in Figure 3.12 with all implementation variations of each tool block if multiple exist.



**Figure 3.12:** Hardware prototyping flow.

## 3.5 Conclusion

This chapter has presented the methodology and capabilities of the simulation tools developed for this work. The LiDAR tool-chain developed is capable of converting feature rich synthetic depth datasets into multi-return LiDAR waveform data. To introduce scope for reflectivity, the segmentation and brightness layers of the underlying dataset are leveraged to generate a class based approximation for reflectivity. This introduces variation in the generation of the waveforms replicating real world surface returns and by doing so programmatically enables the generation of large LiDAR datasets at scale. This is a very useful feature for modern machine learning type processing methodologies, which require a lot of data to derive weights for large scale inference models.

Further, the tool-chain includes full sensor models with capabilities of modelling a basic SPAD behaviour as well as linear and non-linear sampling behaviours. This allows compressive sensing simulations without committing to specialised sampling hardware. The sampling simulations can also be applied to existing real full-frame LiDAR waveform data. This makes it an excellent tool to investigate compressive sensing algorithms for time-of-flight applications as explored in Chapter 5.

Finally, a hardware interface was developed which enables further prototyping of

algorithms with a focus on embedded systems. This effectively turns the LiDAR simulation tools into a synthetic sensor which can be interfaced with hardware. The operation of these interfaces was verified by implementing a cross-correlator in Verilog and C and was also run on an actual FPGA yielding identical results to the Matlab and C implementations. Hardware specific optimisation for the compressive processing method is further explored in Chapter 6.

The simulation tools presented offer an effective solution to ToF sensor prototyping for novel paradigms and edge cases which are not yet commercially explored in a solid-state fashion. Further, the integration of readily available depth datasets as the ground truth enables large scale synthetic LiDAR datasets to be generated at runtime and offline. This enables the investigation of novel deep learning approaches, presented in Chapter 4, for LiDAR waveform processing with a ground truth for in-depth analysis.

# Chapter 4

## Deep Learning for Surface Localisation in LiDAR Waveforms

### 4.1 Introduction

Moving away from a mechanical scanning light detection and ranging (LiDAR) system to a solid-state array LiDAR increases the amount of data being sampled and thus to be processed concurrently when capturing full waveforms. Modern prototype scanning systems employ up to 128 emission and detection channels to capture data at each scanning step to form a high-resolution depth scan. In contrast to this, a solid-state system capturing at or above the same final resolution can easily produce 100 times the volume of data, because it can capture waveforms for each pixel in the entire field-of-view concurrently. As technology progresses to increase resolution beyond scanning systems, the need to process tens or hundreds of thousands LiDAR waveforms is inevitable. If camera resolution is to be matched at megapixel resolution, millions of raw LiDAR signals would need to be processed in real-time. This increases the demands on the processing bandwidth dramatically from only 128 concurrent waveforms by several orders of magnitude.

To match future high bandwidth solid-state systems, novel processing methods are required to cope with the dramatic increase in raw information. Although efforts have been made to accelerate raw LiDAR processing using statistical approaches, this often discards a lot of the information contained in the raw waveform or makes various assumptions to simplify the problem. This often reduces the processing task to a modest multi-dimensional image recovery problem. However, this often implies relatively ideal conditions and requires different algorithms for specific conditions. This makes a unified processing solution which can scale to high resolutions even

more challenging. The LiDAR problem in the automotive context has to operate in various conditions, from night time to bright sunlight and adverse weather conditions. This introduces further challenges to the processing stack and is often a complex task in itself for single waveforms, let alone thousands of concurrent waveforms.

In recent years, machine learning has improved dramatically driven by the need to process successfully the ever growing amount of data available in many machine vision tasks. In particular deep neural networks have become viable with the advancements of modern graphical processing unit (GPU) hardware with highly parallel and efficient compute stacks. This enables the user to tackle a problem using a data driven approach as well as the classical modelling methodologies. A large volume of relevant data is the key to successful machine learning methods. This means that data has to be diverse yet consistent. In the absence of real high-resolution solid-state LiDAR, this work exploits the simulation modalities presented in Chapter 3 to generate diverse sets of LiDAR waveforms derived from synthetic scenes with varying scene content in terms of reflectivity and with varying ambient conditions ranging from night to strong mid-day sunlight.

The aim of this thesis work is to develop and demonstrate a deep neural network capable of determining surface locations for complex multi-return LiDAR waveforms rapidly at scale as a direct high throughput processing solution for solid-state single photon avalanche detector (SPAD) time-of-flight (ToF) arrays.

This work makes the following contributions to deep learning applied to LiDAR waveforms:

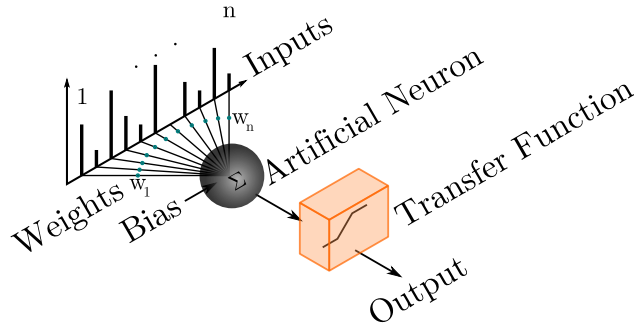
- A novel deep neural network architecture capable of direct and fast surface localisation from full LiDAR waveforms with high throughput: LiDARNet.
- A training-by-simulation approach allowing to train the network in absence of real data for a specific system or augment existing datasets.
- Demonstrating the capability of LiDARNet to perform fast and robust surface localisation even for surfaces beyond the temporal pulse resolution of the system defined by the pulse width and across typical sunlight exposures for long range automotive LiDAR imaging.

The chapter introduces the key elements of deep neural networks using convolutional building blocks and a popular architecture in the form of a convolutional auto-encoder which mimics model-based sparse computational approaches in a data driven fashion in Section 4.2. A novel network architecture tailored to LiDAR wave-

forms is then presented, in Section 4.3 which is designed to be trained primarily with simulation data but is capable of transferring its capabilities to unseen real data. LiDARNet is validated against the signal model and evaluated using an existing LiDAR benchmark dataset in Section 4.5. Its performance in the automotive context of long range LiDAR waveforms is assessed with varying background noise in Section 4.6 and the chapter concludes in Section 4.7.

## 4.2 Deep Learning

Machine learning has seen a massive increase in popularity due to its impressive increase in capability in recent years. In particular deep neural networks have become a very popular and effective approach in tackling complex problems using a data driven approach. One of the fundamental building blocks is the artificial neuron shown in Figure 4.1.



**Figure 4.1:** Illustration of a typical artificial neuron.

Every input  $x_i \in \mathbb{R}^n$  into an artificial neuron is associated with a weight  $w_i$  to aggregate a response with a bias,  $B$ , which ultimately passes through a transfer function,  $\phi$  to form an output  $y$ , or mathematically

$$y = \phi\left(\sum_{i=1}^n (w_i x_i) + B\right). \quad (4.1)$$

By forming complex networks of multiple such neurons in layers a basic neural network is constructed. After a signal or image is fed into a network, the principal data is passed through many neurons. The initial measurements get transformed in this process and combined into another domain. For the network to extract useful information a training process is required to find and set suitable values for each weight of a neuron in an iterative fashion. This can take place in a supervised fashion, where labelled output data is provided to auto-tune the network weights to match the final output for an input. Alternatively, unsupervised training tunes

the network by latching onto dominant features in the given input dataset without outside information. Deep neural networks are inspired by the visual cortex, which is made up of complex and simple cells. This is regarded as the main inspiration for the modern deep neural network [130, 131, 132].

While a single neuron performs a predictable output, combinations of complex multi-layered clusters can perform more advanced operations with the layers between input and output often called hidden layers [133]. However, while many artificial neurons will perform more complex tasks, the number of weights required to be stored also increases dramatically. To advance their functionality, deep neural networks require an ever increasing amount of computational resources to become more effective at increasingly complex tasks and become increasingly difficult to train with simplistic training approaches [134].

The performance and increasing capabilities of artificial neural networks closely follow the dramatic growth in compute power, advances in parallel compute architectures in particular for matrix arithmetics and memory density enabled by Moore's law well into the 2010s [135, 136]. In particular GPU architectures and their consistent performance improvements with ever increasing competence in general purpose computing, driven by demand for advanced physics in modern 3D gaming engines relying heavily on matrix arithmetic, enabled modern deep neural networks, such as the now famous AlexNet [137], to outperform many traditional model based approaches.

Graphic processors are a natural fit for deep neural networks, as they are designed to accelerate large scale vector and matrix operations at scale and speed to draw 3D image frames with massive parallel architectures. Nvidia enabled access to their compute capability in form of their compute unified device architecture (CUDA) in the early 2000s. It quickly became the standard application programming interface (API) embraced by the machine learning community for their deep learning tools to enable larger networks and accelerate training at scale. While CUDA is a proprietary architecture from Nvidia there is also an open source equivalent supported by AMD, the open computing language (OpenCL) which isn't as widely supported by most standard deep learning backends such as Tensorflow [138].

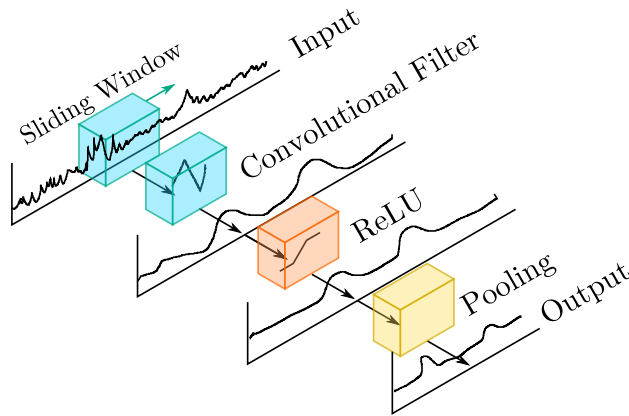
While many of the earlier networks were purely operated in a feed-forward fashion, the concept of back-propagation [139] is another important concept which has contributed to an increase in deep neural network performance allowing for deeper and more complex networks. By using back-propagation, network weights are adjusted both on the forward pass and also on a backward pass, where the errors are



evaluated in both directions for each iteration or epoch. The backward pass is facilitated by evaluating derivatives using stochastic gradient descent to feed back the error gradient and improve training and application performance [140, 141, 142]. Combined with the increase of computational resources in GPU systems, neural networks could perform more error back-propagation into more hidden layers in less time, contributing to better deep neural network architectures and subsequent impressive application performances.

### 4.2.1 Convolutional Neural Network

Although deep (i.e. multi-layered) neural network systems can perform complex tasks, they can scale exponentially with additional hidden layers, making them very memory demanding even with modern compute capabilities. Rather than assigning a single weight to every input, a small collection of weights is applied to a sliding window or kernel across the input vector. This is essentially a signal convolution and was consequently termed a convolutional layer, which is more memory efficient due to weight sharing for various input collections [143] and was effectively demonstrated in the 1990s combined with multi-layered neural systems performing handwritten digit classification [140, 144]. The modern convolutional building block is shown for a 1D signal in Figure 4.2.



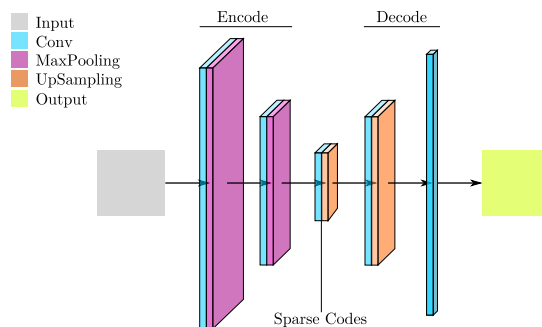
**Figure 4.2:** Illustration of a typical 1D convolutional building block.

As with the artificial neuron, the activations of the convolutional layer or filter are followed by an activation or transfer function; in this case a rectified linear unit (ReLU) is shown. While there are other activation functions, the ReLU has been shown to outperform other activation functions in most applications [145, 146, 147]. This makes it a good default starting point, but for certain applications other functions might be more suitable. Another key component of the modern

convolutional layer is the pooling operator, which has been successfully applied to neural networks [148, 149, 150]. It reduces the dimensionality of the signal and promotes invariant feature learning in deeper layers and are a key component of recent neural networks performing state-of-the-art complex vision and signal tasks e.g. [151, 137, 152, 153, 154, 155] and many more.

### 4.2.2 Convolutional Auto-Encoder

A popular and powerful deep neural network architecture in recent years is the convolutional auto-encoder. Made up of two stages, *encode* forming a code representation and *decode* to reconstruct the principal information. It provides an architecture which often exploits sparsity, has been shown to aid pre-training but also works for classification and de-noising applications on its own [156, 142, 157, 158]. Generally a single convolutional stage is made up of many convolutional filters within a single layer, with many hidden layers making the network *deep*. An illustration of a modern auto-encoder architecture with such multi-layered stages is shown in Figure 4.3 adapted from [156].



**Figure 4.3:** Generic convolutional auto-encoder architecture [156]. The multi-layered fashion of the deep network is illustrated by various sizes of the individual stages, illustrating that a single input is passing through many layers.

The auto-encoder or also auto-associator reduces the dimensionality of the data and thus can identify features in lower dimensions similar to compression in traditional signal processing. Being only made up of convolutional building blocks both training and application are also extremely efficient thanks to modern GPU cards and deep learning accelerators. Given its overall simplicity and potential for speed, it is an excellent starting point for fast and efficient processing for many signal processing tasks.

### 4.2.3 Related work

Application of deep learning to image and signal processing is a very active area of research. In terms of LiDAR applications, they often work with point clouds to perform object classification [159, 160], scene segmentation [161] or sensor fusion [162, 163, 164, 165]. This work aims to process LiDAR waveform directly to generate such point clouds efficiently from complex multi-return raw full-waveform LiDAR data. There are some similarities with work being carried out in electrocardiogram (ECG) analysis such as [166, 167], but in that case the final output is simpler with few output classes, whereas thousands of potential surface locations are present in LiDAR waveforms.

There is some recent work applying deep learning directly to the LiDAR waveform to either de-noise the waveform and remove minor artefacts [168] or up-scale the waveform [169]. However, both of these approaches only achieve marginal, if any, improvements to the input waveform with no abstraction of the data into useful information. Other approaches attempt more useful application of neural networks to LiDAR data, such as direct object reconstruction in very constrained application settings [170, 171] or surface segmentation [172].

The work presented in this chapter aims to determine surface locations for multi-return waveforms and thus enable fast and efficient point cloud generation at scale for solid-state LiDAR arrays.

## 4.3 LiDARNet

### 4.3.1 Introduction

The following sections present work to process raw LiDAR waveforms using deep neural networks to identify true surface returns rapidly and with high throughput compatible with high-resolution solid-state systems and scope for increasing resolution and additional functionality in the future. The large volume of data being sampled by such devices calls for parallel, efficient and high throughput approaches comparable to big data applications at which deep learning approaches excel. The network is ideally capable of processing waveforms under various conditions including a high dynamic range of background noise as well as adverse weather conditions or other influences on the waveform all within a single pass through the network. The architecture is designed to adapt to a specific sensor output based primarily on the waveform size. Two architectures are presented; one for a short range LiDAR system where a small amount of real data is available to validate the training

approach using simulated data and another for the proposed automotive LiDAR system.

As an initial starting point a convolutional auto-encoder architecture shown in Figure 4.3 was used to investigate if deep learning is suitable for raw LiDAR waveform processing.

The auto-encoder was set up with parameters shown in Table 4.1 to be compatible with the LiDAR system from [173] and their real super-resolution dataset which is described in more detail in Section 4.5 and Table 4.2. Super-resolution describes the ability to resolve close surfaces with separations less than the full-width-at-half-maximum (FWHM) of the LiDAR system. Like for all experiments in this chapter, the network is trained with a binary surface location vector of the same length as the corresponding LiDAR waveform.

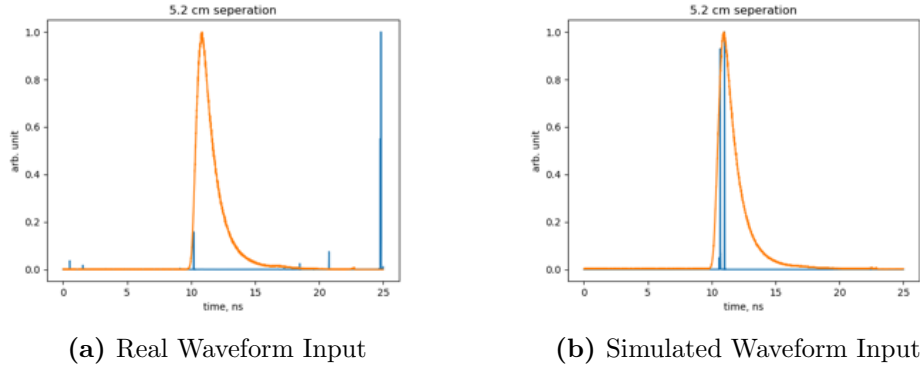
**Table 4.1:** Network parameters (F for number of filters and W for convolutional window length) for auto-encoder to process 1D LiDAR waveforms, pooling and up-sampling by a factor of 2 compiled with AdaDelta and a binary cross-entropy loss function. Trained with 10000 simulated waveforms for 200 epochs.

|        | Auto-Encoder |         | Activation |
|--------|--------------|---------|------------|
|        | F, W         | #params |            |
| Conv1D | 32, 40       | 1056    | ReLU       |
| Conv1D | 16, 40       | 20496   | ReLU       |
| Conv1D | 8, 40        | 5128    | ReLU       |
| Conv1D | 32, 40       | 10272   | ReLU       |
| Conv1D | 1, 40        | 1281    | Sigmoid    |
| total  |              | 38233   |            |

Using this auto-encoder yielded preliminary results with seemingly good success when simulated data was used for training and testing as illustrated in Figure 4.4 but did not transfer to unseen real data, motivating a different network architecture.

Further, from this preliminary experiment, the auto-encoder architecture only performed well if the background noise level was within a small margin of a constant mean. Any deviation from this value for testing would result in false activations. Therefore, different network weights would be required for different noise levels. This is similar to model based approaches, which tend to be more compact for singular scenarios. This suggests that the considered convolutional auto-encoder did not learn to identify principal components but rather learned locations from the training data, since it had limited success on like-for-like simulated data.

While both the real and simulated input waveforms look visually identical, the network seems to be biased towards training data and fails to generalise the features. This not only led to inconsistent performance when applying the auto-encoder to



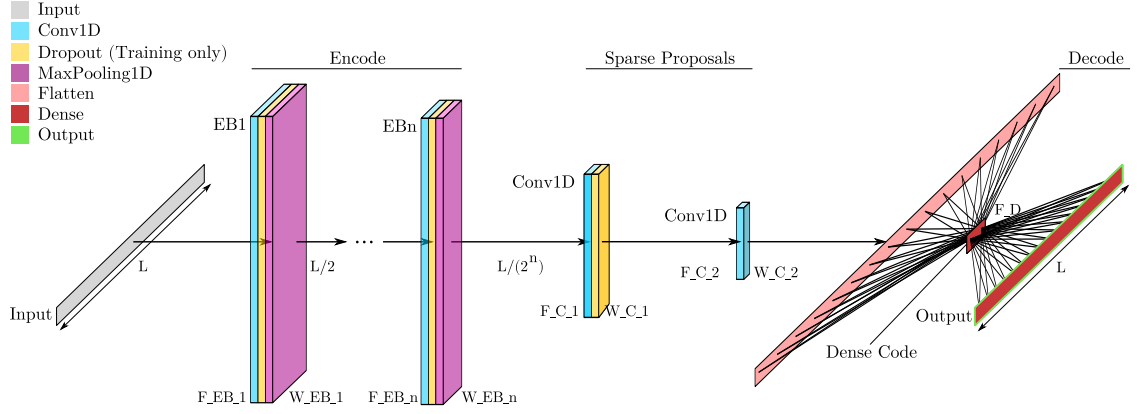
**Figure 4.4:** Auto-encoder results applied to real and simulated data for super-resolution benchmark for 5.2 cm separation. (Orange) input waveform and (blue) auto-encoder proposal. (a) The auto-encoder did not transfer simulated training results to unseen real-data resulting in many false proposals. (b) For simulated data the auto-encoder identifies the correct surfaces locations for the given separation.

real data but also limited the application to very specific operating conditions. This suggests that there is an issue in the considered architecture, which may not have become immediately obvious if a large scale real dataset would have been used. Achieving a robust singular network which can discriminate various scenarios is unlikely for this type of architecture.

### 4.3.2 Architecture

The auto-encoder demonstrated, that deep learning is able to encode some LiDAR waveform features in theory using the encoding stage, but failed to generalise the latent and often sparse information space back to return proposals within the appropriate information space.

To address this, a new architecture was developed, which refined the encoding stage to include mechanisms to minimise data bias in training and improve feature encoding. This was achieved by enforcing dropouts during the encoding stage [174] and combined with a maximum pooling operator this forms an encoding block (EB) as shown in Figure 4.5. Each EB reduces the dimension of the signal by a factor of 2. Depending on the input signal’s dimensions, a number of encoding blocks can be placed in series to fine-tune the appropriate latent space dimension. Once the latent space encodes significant information, a sparse proposal stage is introduced as shown above. This uses two convolution blocks in series, where the first has a stacked dropout layer and the second has a smaller signal window than the first. This stage generally refines the features after encoding and discards unwanted artefacts from prior convolutions.



**Figure 4.5:** LiDARNet Architecture. A signal of length  $L$  is fed into LiDARNet. The height of the blocks illustrates the number of layers ( $F$ ), while the width of a block indicates relative size of the window size of the convolutional filters ( $W$ ).

Additionally, rather than relying on a convolutional decoding stage which reduces the latent dimension via convolutions with decreasing number of layers and up-sampling, a fully connected dense decoding stage is introduced more akin to traditional classification stages. This was chosen to allow for the sparse proposal information space to be fully rolled out by a flatten operation. The benefit is that no information is discarded after encoding the information in a multi-dimensional space. The information is then condensed into a reconstruction layer, where the latent space forms a reconstruction code, from which a final output with surface proposals is generated.

The final stage is therefore set up as a classification problem where each waveform bin is a possible class or potential location of a true surface return. The advantage of this approach is that all identified features are considered in a complex fully connected neural network. This comes at the cost of complexity and increases the number of parameters of the network. But even for the more complex case of the automotive scenario with 6.5 million parameters, the network size is still modest for modern deep learning architectures. This architecture is designed to generalise the data and allows the user to perform training on simulation data with results also transferring over to unseen real data. It further works across the full background operating range, achieving the goals set out for this work. The network has to be configured based primarily on input waveform size. Information such as number of peaks is not directly provided but implicitly bound by the labelled ground truth.

Generally, the more layers are present, the larger the parameter space as more connections between layers have to be created with associated weights. The window ( $W$ ) sizes have been chosen to capture a typical surface cluster (a collection of re-

turns in close proximity to each other) within the operating conditions. While the number of filters ( $F$ ) has been chosen to promote learning, i.e. to facilitate enough parameters to encode the relevant information, but not too many to avoid redundancy in the network. This was done empirically, where filter sizes were reduced until the learning rate per epoch worsens. The training approach and fine-tuning of these parameters are outlined in the next section.

## 4.4 Training Methodology

For data-driven approaches such as deep learning and machine learning in general, the quality of the input information is crucial as it drives feature detection and forms sensible paths through an artificial neural network (ANN). To facilitate robust learning and ultimately successful operation of an ANN, the training data has to be consistent yet diverse and representative of the entire scope of the desired application. This introduces challenges to the training methodology and the training data. As discussed in Chapter 3 large scale diverse datasets for full waveform LiDAR data are not readily available at the time of writing. This work proposes a training approach via simulation to generate diverse and consistent training data with fully labelled ground truth data.

However, simulations can be too consistent and might not have the same type of variation real systems produce, despite best efforts to account for such signal properties. It is therefore critical, as with all simulation approaches, to validate the simulation approach by testing against real data. A real experiment and system setup from [173, 31], where two retro-reflectors are moved in close proximity beyond the resolution of the FWHM is implemented in the simulation framework presented in Chapter 3. A LiDARNet variant has been created for this particular type of system output, herein referred to as *SuperRes* with parameters presented earlier in Table 4.6. The general training approach is summarised below and the differences between the constrained experiment for validation and the more general *Automotive* use case are also discussed.

### 4.4.1 Training Data Generation

In this chapter a discrete quantised observation model of equation (2.22) is adopted. The respective time delay  $t_k$  of each true surface return can be represented in the form of an ideal binary signal representing the impulse train described in equation (2.19).

For  $K$  surfaces in the field-of-view (FoV) of the emitter and associated detector all

surface locations of a LiDAR waveform can be described as a collection of distance and reflectivity pairs. For simplicity, all attenuation is to be encapsulated in a single parameter,  $g_k$ . In this chapter  $g_k$  is calculated using the envisioned LiDAR system in Table 3.2 with the procedure described in Chapter 3 for the *Automotive* use case. A surface group, i.e. all returns from a single detector element, is then

$$\mathbf{g} = [(d_0, g_0), \dots, (d_{K-1}, g_{K-1})]. \quad (4.2)$$

Each value pair of distance,  $d_k$ , and attenuation including reflectivity,  $g_k$ , is used to generate the surface mean return photon count with a global background mean,  $\beta$  as outlined in Chapter 3 to assemble a count distribution to generate a waveform,  $\mathbf{f}$ , using a Poisson counting process as defined in equation (2.22).

A discrete LiDAR waveform with  $p$  bins has an associated discrete distance vector

$$\mathbf{d} = [D_0 \dots D_p], \quad (4.3)$$

which encodes the respective quantised distances associated with each bin of a histogram. The principal location information of a histogram  $\mathbf{h} \in \{0, 1\}^p$  of  $K$  returns can be visualised as

$$\mathbf{h} = [0 \dots 1 \dots 1 \dots 0], \quad (4.4)$$

where up to  $K$  non-zero entries (equal to 1) represent the true location of each return in its respective bin location related to its discrete distance,  $d_k$  with the same distance definitions as in equations (2.2)-(2.3). This ideal histogram formulation constitutes the *ground truth* input to the network during training.

To recover depth measurements in this ideal case, the depth is a sparse vector containing the peak locations directly as

$$\mathbf{m} = \mathbf{h} \circ \mathbf{d} = [0 \dots d_0 \dots d_{K-1} \dots 0], \quad (4.5)$$

where  $d_k$  is the distance of the  $k^{th}$  return and  $\circ$  is the Hadamard (component-wise) product,  $\mathbf{m}$  is therefore a vector aligned to the bins of  $\mathbf{h}$  where entries are the non-zero distances to each of the surfaces. This allows to construct a list of surface returns according to

$$T(k) \leftarrow \mathbf{m}(j) \text{ if } \{\mathbf{h}(j) > 0\}. \quad (4.6)$$

To motivate training and enforce features, deep learning often benefits from nor-



malisation of the input data as a consistent scale aids gradient descent during the application of back-propagation [175]. Two normalisation schemes being

$$\text{norm}_{\min-\max}(\mathbf{f}) = \frac{\mathbf{f} - \min(\mathbf{f})}{\max(\mathbf{f}) - \min(\mathbf{f})} \text{ and} \quad (4.7)$$

$$\text{norm}_{\max}(\mathbf{f}) = \frac{\mathbf{f}}{\max(\mathbf{f})} \quad (4.8)$$

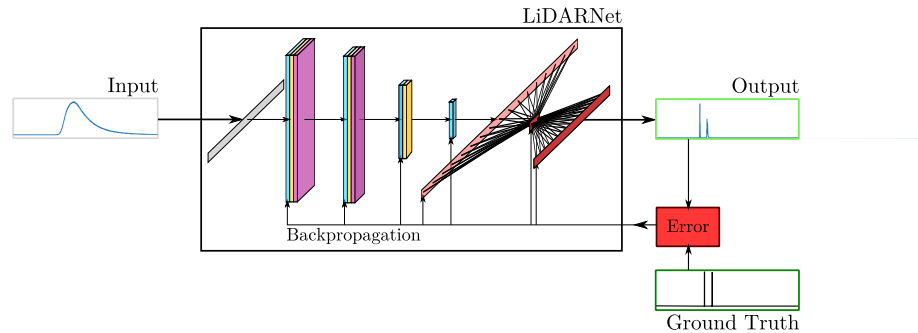
are considered, where  $\mathbf{f} \in \mathbb{N}^p$  is the input signal vector, here a LiDAR waveform.

The output of LiDARNet is a confidence vector of surface locations,  $\mathbf{x} \in \mathbb{R}^p$ . To derive the estimate for surface locations,  $T^*$ , a confidence threshold,  $\tau$ , is introduced such that

$$T^*(k) \leftarrow \mathbf{m}(j) \text{ if } \{\mathbf{x}(j) > \tau\}. \quad (4.9)$$

Results of using  $\text{norm}_{\max}$  were presented initially in [97] but this work later found that  $\text{norm}_{\min-\max}$  further improved performance of LiDARNet where the background rate is not constant in the dataset such is the case in the *Automotive* experiments. For the super-resolution experiment outlined in the next section (*SuperRes* configuration), no noticeable difference was observed. However, neither mode is able to train robust convolutional filter weights to encode useful information without any normalisation.

A dataset consists of  $N_{\text{train}}$  waveforms for training and  $N_{\text{test}}$  waveforms and associated binary ground truth histograms  $\mathbf{h}$ . The data split in this work is  $\frac{N_{\text{test}}}{N_{\text{train}}} = \frac{1}{25}$ . To introduce additional variance in a dataset, some parameters are randomised in the data generation. An illustration of the data inputs for the training stage is shown in Figure 4.6.



**Figure 4.6:** LiDARNet training with simulated inputs and ground truth.

All networks were created using Keras with a Tensorflow 2.x backend running on

Python 3.6 running on a workstation<sup>1</sup>. The networks were compiled with an Adamax optimiser and binary cross-entropy was chosen to evaluate training loss or the error as is often case for classification type problems in deep learning.

## 4.5 Validation Experiment

All training input data for LiDARNet in this work is made up of simulated waveforms. Simulations will always be approximations of real-world signals and as such will have unique features from assumptions just like real data will have unique features due to system characteristics. However, these variations are not the same and thus a data driven approach based on simulated data can introduce a data bias which may hinder the generalisation of features during training making it challenging to apply the trained network to different new data including real data. The LiDARNet architecture has been specifically designed to be able to apply a network trained with simulated data to real data applications.

The labelled data in the training of the network is the ideal impulse train without any loss described in (2.19) with  $\Gamma_k = 1$  discretised as  $\mathbf{h}$  as defined in equation (4.4). Previous work has shown that this underlying ideal signal model holds in combination with drawing counts from a random Poisson process for a simulated waveform [31, 102, 110, 112, 105] (see equation (2.22)). This means that it is entirely possible that LiDARNet will encode features comparable to traditional statistical models.

The goals of this validation experiment are to demonstrate the capability of generalising features, the application of a network trained by simulation to real data and to explore if the network learns features comparable to the underlying signal model assumptions.

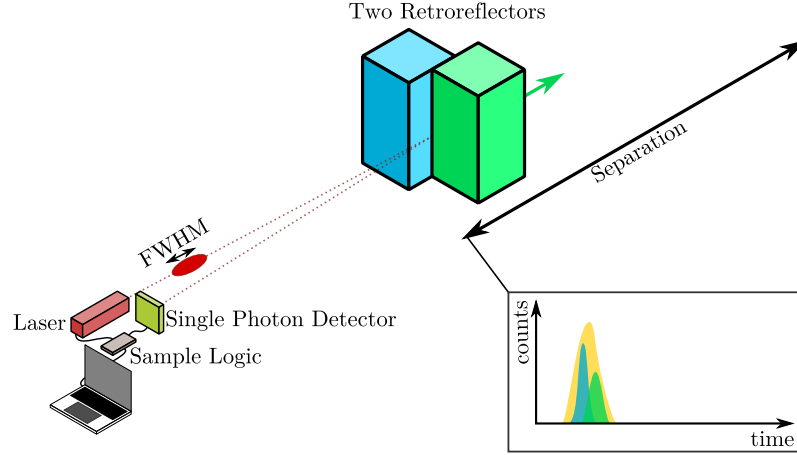
The network’s internal workings are investigated to identify such similarities qualitatively. The output of the neural network provides immediate surface location proposals as per equation (4.9) similar to parameter estimations methods for waveforms. To assess the network’s performance, LiDARNet is compared quantitatively with two statistical approaches, which provide location estimates and have demonstrated competent results on the real dataset [31, 110].

---

<sup>1</sup>Windows 10 workstation: x86-64 with dual Intel Xeon E5-2630v3 (2.4 GHz), 48 GB RAM and NVidia RTX 2080 Ti (12 GB)

### 4.5.1 Two Retro-reflectors

For validation, the system and experimental design from [173, 31] was replicated as a pseudo-random data simulation. Two retro-reflectors  $S_1, S_2$  are placed next to each other and a ToF system is pointed illuminating both of their facing edges as shown in Figure 4.7.



**Figure 4.7:** Illustration of a two retro-reflectors in close proximity experiment first described in [173] and used in [31] and [110] for performing super-resolution using a superconducting single photon detector (SSPD) based on a nanowire and a wide field-of-view. In yellow the merged peak shape is shown with individual peak contributions of each reflector shown in blue and green respectively.

The real benchmark dataset as used in [31, 110] contains 18 measurements selected from data first presented in [173], where one reflector is moved away from another stationary reflector in small cm increments. The dataset considered has a starting separation of 1.7 cm and a final separation of 71.2 cm at a stand-off distance of 330 m.

For the training data generation of this experiment the two surfaces, i.e.  $K = 2$ , the parameters to be randomised for data generation are the amplitude  $g$  and the distance between the two reflectors. To truthfully replicate measurement variance, the surface  $S_1$  only varies slightly around a fix position, while  $S_2$  is moved randomly in either direction for separations of  $\pm 100$  cm. The instrumental  $r$  is the real system instrumental function. The relevant LiDAR system parameters are summarised below in Table 4.2.

A total of  $N_{\text{train}} = 25000$  and  $N_{\text{test}} = 1000$  were generated. For the network training the data is fed in chunks of size 1000. This is primarily due to memory limitations of the GPU used. During a training run, data and label data i.e. ground truth is randomly fed from a chunk to the network for a number of epochs,  $n_{\text{epochs}}$ , cycling

**Table 4.2:** LiDAR system parameters used to capture the super-resolution benchmark dataset [173]

| Parameter   | Symbol    | metric  | time     | type |
|-------------|-----------|---------|----------|------|
| Wavelength  | $\lambda$ | 1550 nm |          | SSPD |
| Sensor      |           |         |          |      |
| Pulse width | FWHM      | 1.5 cm  | 50 ps    |      |
| Aperture    |           | 200 mm  |          |      |
| Channels    | bins, $p$ | 4096    |          |      |
| Bin width   |           | 9.2 mm  | 3.073 ps |      |

through the entire dataset. This is repeated for  $n_{\text{runs}}$ . This data and training delivery is the same for any LiDARNet configuration (*SuperRes* and *Automotive*) presented.

The network was configured for the histogram size as presented first in [97] and is shown in Table 4.3.

**Table 4.3:** Network parameters for *SuperRes* LiDARNet configuration [97] for the super-resolution benchmark from [173].

|        | SuperRes |         | Activation |
|--------|----------|---------|------------|
|        | F, W     | #params |            |
| EB1    | 64, 64   | 4160    | ReLU       |
| EB2    | 64, 32   | 131136  | ReLU       |
| Conv1D | 32, 32   | 65568   | ReLU       |
| Conv1D | 16, 32   | 16400   | ReLU       |
|        | C        | #params |            |
|        |          |         |            |
| Dense  | 128      | 2097280 | ReLU       |
| Dense  | 4096     | 528384  | SoftMax    |
| total  |          | 2842928 |            |

*SuperRes* LiDARNet is trained in this work with  $n_{\text{runs}} = 5$  and  $n_{\text{epochs}} = 10$  epochs per run for the entire dataset resulting in a total training time of about 35 minutes.

### 4.5.2 Feature Tracing

To illustrate the network’s inner workings, a signal from the real super-resolution dataset is passed through the simulation trained network. In particular the convolutional layers are under investigation as they perform the feature encoding which if working correctly should identify principal components in a latent space. A comprehensive selection of intermediate signals and their associated convolutional filter weights are shown in Figure 4.8 for a real waveform with two returns with a separation distance of 3.2 cm.



The following analysis of above figure is structured into the three main stages identified for LiDARNet in Figure 4.5; *Encode*, *Proposals* and *Decode*.

*Encode* – Two encoding blocks are present in this configuration, which reduces the principal signal dimension by a factor of 4, here from 4096 to 1028. The initial *Feature Detection* identifies recognisable features of the LiDAR waveform with a selection of weights with the largest activation values (Conv1D\_1). A masking of the region of interest with compensation for the SPAD tail (top-left), the down-sampled waveform (top-right), a region of interest where peaks might be (bottom-right) and finally an inverse of SPAD characteristics and a system artefact at the end of the histogram (bottom-left). It is quite curious that the network consistently produces at least one significant output identifying system characteristics without signal content. Next the detected features are passed through another set of convolution filters (Conv1D\_2). The resultant refined features (*Refined Feature Encoding*) are consistently focusing on the region of interest, and a clear separation of peaks proposals starts to emerge. However, the intermediate signal at this point still contains many system artefacts and indeed new artefacts from the network itself.

*Proposals* – After this initial encoding stage, the refined features are further passed through two convolutional layers (Conv1D\_3 with dropout and Conv1D\_4 without dropout). At this point no pooling occurs and the dimensionality of the signal passing through remains constant. Focusing on the first convolutional layer with dropouts and its filter weights, the filters target much more specific areas of the signal, in particular an amplification of the signal area and attenuation of the SPAD tail can be observed in many of the weights. The existence of multiple variants suggests a wide scope, which is separation agnostic for this configuration. *Sparse Proposals* are formed which resemble the expected output more closely, but at this stage the signal’s dimension is lower and the proposals can’t be directly mapped to the original signal dimension. It is further obvious that the process can still introduce false activations in particular at the beginning and end of the processed signal. To remove the new artefacts and improve the sparse representation, another compact convolutional layer refines the latent information space, which successfully encodes the information to consistent activations resembling two peak position proposals as shown in *Refined Sparse Proposals*. The weights associated with this final convolutional layer seem to encode further noise removal with some variation around a mean line combined with notch. Given that false activations at the beginning and end are observed in the previous layer output, these notches seem to remove false activations.

*Decode* – All outputs from the last convolutional layer in the encoding stage are rolled out using a Flatten operator. This expands the entire information space into a single vector, which has the advantage of retaining all available information before using a fully connected neural network to form a reconstruction code (*Recon Code*) from all encoded information, which is then expanded out into the initial information space. This results in two spikes at the corresponding positions of the retro-reflectors in this constrained experiment.

As can be seen from the analysis, some the early signal features in the encoding stage resemble the underlying signal model but the network also produces very different features. Just from visual inspection these might not seem useful, but get transformed throughout the network into valid outputs. This suggests that the network identifies valid features from a signal model perspective but also learns features directly from the data, a key benefit of machine learning approaches as it can provide a wider scope with more variety of generic features considered significant.

Next the full real dataset is evaluated and compared against the state-of-the-art statistical approaches to fully validate the simulation and training approach.

### 4.5.3 Results

For a more quantitative analysis of LiDARNet’s performance, the real super-resolution benchmark is evaluated and compared against two statistical approaches which provide direct surface locations to validate that the network learns general features which translate to real data.

The analysis focuses on the case where the two returns are merged into a unique peak shape, since the fully separated case is less challenging and all approaches perform very well in this case. Both statistical model approaches are given the number of peaks for this experiment (i.e.  $K = 2$ ), while LiDARNet has an implicit bound of  $K \leq 2$  as it has been trained with only up to two surface returns present in any presented waveform. The results for all three approaches are shown in Table 4.4.

As can be seen, LiDARNet performs comparably to the statistical approaches with the clear ability to resolve two targets which are separated at less than the system’s FWHM. LiDARNet has more variance than the statistical approaches, but nevertheless performs well while being fast and scalable to multiple inputs and outputs. The results are also presented visually in Figure 4.9.

All approaches perform at sub-centimetre resolution and consistently separate all peaks both in the merged and in the separated cases. This demonstrates that LiDARNet, despite being trained on simulated data only, can also process unseen

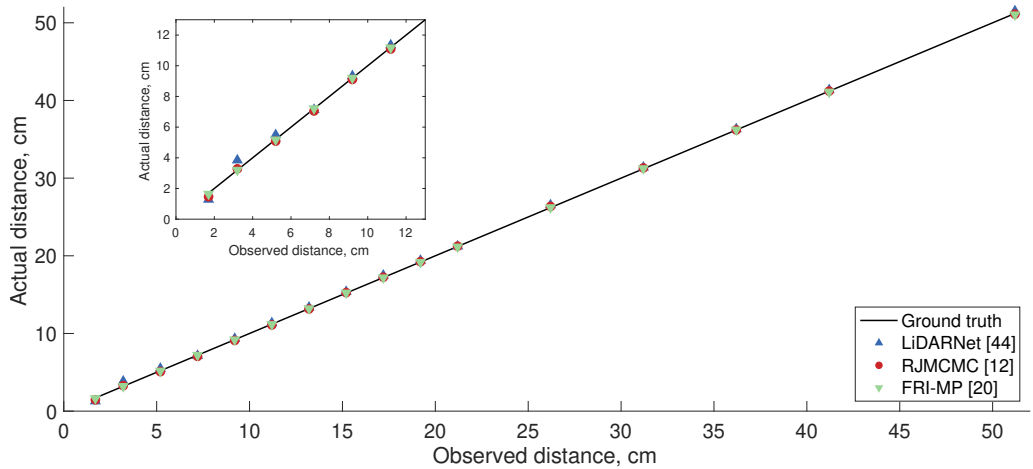
**Table 4.4:** Evaluation of a real super-resolution benchmark using LiDARNet (trained for 5 runs and 10 epochs each on simulated training data).

| ground truth<br>(cm) | RJMCMC [31]      |               | FRI-MP [110]     |               | LiDARNet [97]    |               |
|----------------------|------------------|---------------|------------------|---------------|------------------|---------------|
|                      | estimate<br>(cm) | error<br>(cm) | estimate<br>(cm) | error<br>(cm) | estimate<br>(cm) | error<br>(cm) |
| 1.7                  | 1.462            | 0.238         | 1.640            | 0.060         | 1.281            | 0.419         |
| 3.2                  | 3.281            | 0.081         | 3.205            | 0.005         | 3.843            | 0.643         |
| 5.2                  | 5.086            | 0.114         | 5.180            | 0.020         | 5.489            | 0.289         |
| 7.2                  | 7.053            | 0.147         | 7.213            | 0.013         | 7.136            | 0.064         |
| 9.2                  | 9.108            | 0.092         | 9.201            | 0.001         | 9.332            | 0.132         |
| 11.2                 | 11.092           | 0.108         | 11.176           | 0.024         | 11.345           | 0.145         |
| 13.2                 | 13.155           | 0.045         | 13.209           | 0.009         | 13.357           | 0.157         |
| 15.2                 | 15.255           | 0.055         | 15.234           | 0.034         | 15.370           | 0.170         |
| 17.2                 | 17.242           | 0.042         | 17.192           | 0.008         | 17.474           | 0.274         |
| 19.2                 | 19.239           | 0.039         | 19.180           | 0.020         | 19.396           | 0.196         |
| 21.2                 | 21.237           | 0.037         | 21.161           | 0.039         | 21.226           | 0.026         |
| 26.2                 | 26.355           | 0.155         | 26.226           | 0.026         | 26.532           | 0.332         |

real LiDAR data from only requiring a few model parameters from the measurement system, allowing to adapt the network to multiple scenarios yielding a flexible and fast high throughput processing method.

This validates the training-by-simulation approach and although one would expect even better performance if a real large scale dataset was available, the benefit of simulated training is a solid starting point in absence of real data at scale. The simulation approach can further augment real world datasets if required. System specifications can also change throughout a development cycle and environmental noise such as temperature fluctuations can make real datasets as biased as a poorly designed simulation approach.

However, diversifying a simulated dataset is straightforward by adding different model components for various scenarios, making it an ideal prototyping approach, which can be deployed to real sensors as soon as available and can then be refined using real data from the system. This validation demonstrates that a neural network

**Figure 4.9:** Resolution experiment from [31] and [110] with this work's results from [97]



can be trained with simulated data which is based only a few key parameters of the LiDAR system, which should be available early in the development cycle of the system.

To evaluate this approach further, the network architecture is adapted for the *Automotive* use case and is evaluated using prototype LiDAR specifications outlined in Table 3.2.

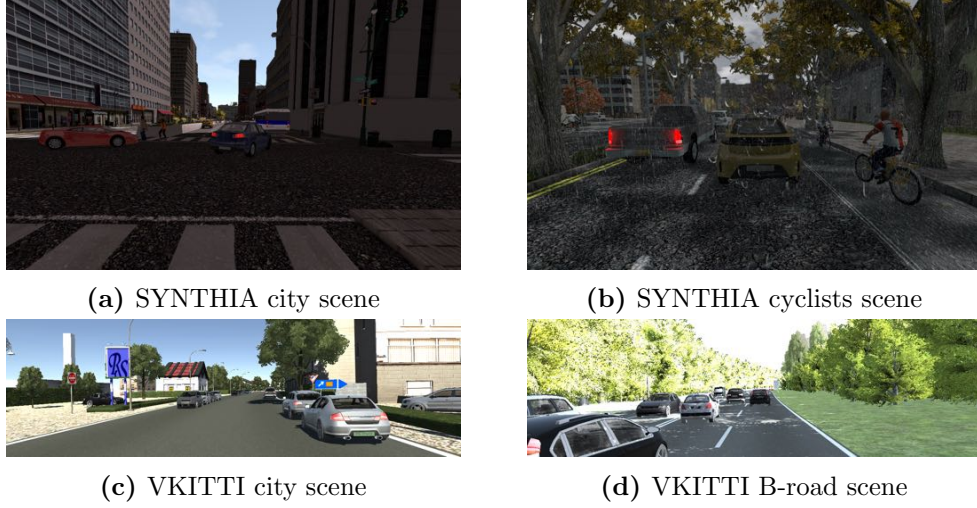
## 4.6 Automotive Experiment

With the training-by-simulation approach validated, the network configuration was adapted to tackle the more complex task of processing long range LiDAR waveforms in varying background lighting conditions. The spread of the background rate represents sensor operation at night and during bright mid-day sunlight. While night time LiDAR without any adverse weather effects is an almost ideal application scenario for LiDAR, the other end of the spectrum is extremely challenging from a signal-to-noise ratio (SNR) point of view. It is also important to note, that the dynamic range of the mean count rate varies dramatically as ambient light exposure increases. This is one of the reasons the data driven approach requires some form of normalisation to promote learning, to ensure a consistent value range propagating through the network.

To test the capabilities of LiDARNet applied to waveforms which not only contain multiple surface returns but also have varying levels of sun light photon counts a dataset is generated for this particular use case using automotive synthetic scenes. Two scenes from VirtualKITTI [122] and another two scenes from SYNTHIA [123] for a total of 4 scenes shown in Figure 4.10 are used for waveform generation.

The multi-return mode is set to capture  $K \leq 9$  surfaces per detector. The background rate is added to the depth information and is randomly varied for each waveform between 0.04 and 38.28 to simulate sun light ranging from 0.1 to 100 klux, which replicates a range from night time to day time with bright sunshine. Pixels are selected at random across all scenes, but locations are stored to avoid duplicate entries and ensure an independent test dataset. Every waveform is aggregated from 10 simulated laser pulses. The training parameters are summarised in Table 4.5

To work with the increased histogram vector length over the *SuperRes* configuration and to allow for more variation introduced by the large ambient count range, LiDARNet is configured in an *Automotive* configuration as shown in Table 4.6, which



**Figure 4.10:** Automotive scenes for LiDARNet training data generation

**Table 4.5:** Simulated automotive solid-state SPAD LiDAR waveform dataset derived from synthetic scenes for LiDARNet training.

| Parameter            | Symbol             | metric                    | time   | type         |
|----------------------|--------------------|---------------------------|--------|--------------|
| Wavelength           | $\lambda$          | 905 nm                    |        | SPAD         |
| Sensor               |                    |                           |        |              |
| Pulse width          | FWHM               | 4 cm                      | 133 ps |              |
| Channels             | bins, $p$          | 7500                      |        |              |
| Bin width            |                    | 4 cm                      | 133 ps | 0.1-100 klux |
| Background mean rate | $\beta$            | 0.04-38.28 counts per bin |        |              |
| Number of surfaces   | $K$                | 0-9                       |        |              |
| Training waveforms   | $N_{\text{train}}$ | 100000                    |        |              |
| Test waveforms       | $N_{\text{test}}$  | 4000                      |        |              |

were also first presented in [97].

As expected, due to the increased complexity of the *Automotive* configuration and input waveform structure, this configuration requires more epochs and due to the larger network and larger input data chunk memory size training takes significantly longer. It should be noted, that the *Automotive* case has to encode a lot more information due to the increased surface count and varying background rate.

It should be noted, that the surface number bound is data defined and can be readily adapted based on expected maximum returns or indeed observed parameters from an existing system for the desired application. As mentioned before, a data driven approach allows full flexibility to include as many scenarios as desired and should in theory allow the user to expand the network’s capability to handle many varying operating conditions and by re-training the network with the additional data all contained within a single network. In contrast to requiring multiple models and a model selection process which has to identify the current conditions reliably prior waveform processing introducing latency in a real-time system.

**Table 4.6:** Network parameters for *Automotive* long range variable noise LiDARNet configuration [97]

|        | Automotive |         | Activation |
|--------|------------|---------|------------|
|        | F, W       | #params |            |
| EB1    | 96, 48     | 4704    | ReLU       |
| EB2    | 96, 48     | 442464  | ReLU       |
| EB3    | 64, 24     | 147520  | ReLU       |
| Conv1D | 32, 24     | 49184   | ReLU       |
| Conv1D | 16, 24     | 12304   | ReLU       |
|        | C          | #params |            |
|        |            |         |            |
| Dense  | 256        | 3842304 | ReLU       |
| Dense  | 7500       | 1927500 | SoftMax    |
| total  |            | 6425980 |            |

LiDARNet in the *Automotive* configuration (Table 4.6) was trained for  $n_{\text{runs}} = 6$  and  $n_{\text{epochs}} = 25$  each resulting in a total training time of 14 hours with the dataset outlined in Table 4.5.

#### 4.6.1 Results

Before discussing the performance of LiDARNet on long range automotive waveforms, the effects of input normalisation are assessed based on the final performance of LiDARNet by analysing the receiver operating characteristics (ROC) of the trained networks with different input normalisations.

A proposal is considered a true-positive (TP) if it is within  $\pm 12$  cm of a ground truth positive (P) and if the proposal made by the network is greater than a threshold,  $\tau$ . The other quantities to be evaluated are true-negatives (TN) i.e. if the ground truth is also negative (N), a false-positive (FP), which means a surface location was proposed in absence of a real surface in the scene location and false-negatives (FN), where a true surface return has not been identified. This also allows to calculate a true-positive rate (TPR), false-positive rate (FPR) and an overall accuracy metric (ACC) as

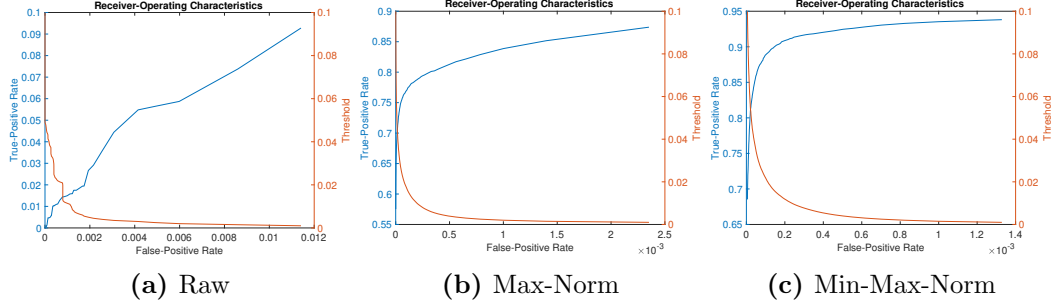
$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad (4.10)$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \text{ and} \quad (4.11)$$

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}. \quad (4.12)$$

The resultant ROC curves are shown in Figure 4.11 for the raw waveform and the normalisations considered as per equations (4.7)-(4.8) for  $\tau = \{0.001, 0.1\}$  across all 4000 test waveforms from 4 independent scenes.

The raw input fails completely with true-positive rates (TPR) of below 10%. If the



**Figure 4.11:** ROC for LiDARNet for different normalisations of input data, noting the dramatic decrease of false-positive rate from (a) no normalisation to (c) min-max normalisation.

input data is max-normalised as presented in [97] and in particular with min-max-normalisation, LiDARNet performs very well with TPRs above 80% for maximum and well above 90% for min-max. Due to the large number of zero entries in the dataset, the overall accuracy figure at 99.98% for both approaches is not as meaningful as the TPR. Comparing the effectiveness of peak detections and thus the quality of surface localisations of LiDARNet versus reverse jump Markov chain Monte Carlo (RJMCMC) [31] and finite-rate-of-innovation (FRI)-matrix penciling (MP) [110] in Table 4.7 will emphasise this.

**Table 4.7:** Type 2 Analysis of multi-return waveform peak detection for resolution confidence interval of  $\pm 12$  cm, a total of 12976 true returns are contained in 4000 waveforms. LiDARNet threshold is  $\tau = 0.05$

|                        | TP<br>$\triangle$ | FP<br>$\nabla$ | FN<br>$\nabla$ | TPR<br>$\triangle$ | FPR<br>$\nabla$ | ACC<br>$\triangle$ |
|------------------------|-------------------|----------------|----------------|--------------------|-----------------|--------------------|
| RJMCMC [31]            | 9769              | <b>133</b>     | 3207           | 0.7529             | <b>0.000004</b> | 0.9999             |
| FRI-MP [110]           | 7628              | 9599           | 5348           | 0.5879             | 0.000320        | 0.9995             |
| <b>LiDARNet-MinMax</b> | <b>10650</b>      | 757            | <b>2326</b>    | <b>0.8207</b>      | 0.000025        | 0.9999             |
| <i>LiDARNet-Max</i>    | 8997              | 330            | 3979           | 0.6934             | 0.000011        | 0.9999             |

The RJMCMC [31] approach outperforms all other approaches in terms of false-positives, but also misses significantly more peaks than LiDARNet-MinMax. FRI-MP [110] performs the worst out of all considered methods, primarily due to the very high false-positive rate despite being given the correct number of peaks *a priori*, something which would be difficult to achieve in real world applications.

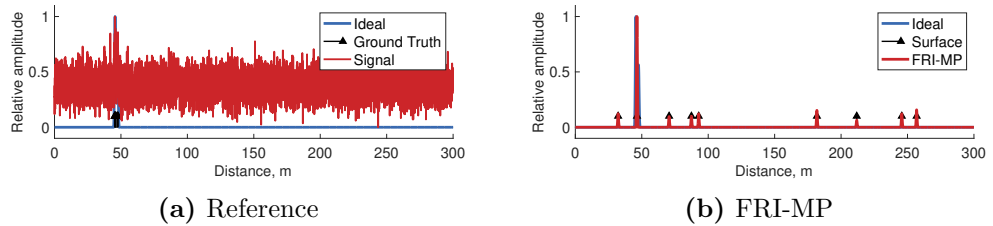
LiDARNet-MinMax has the highest success in identifying true-positives at about 82%, while still rejecting most false positives successfully. For LiDARNet this rejection rate can be adjusted with the threshold, but at the cost of rejecting true positives with low confidence values. LiDARNet-Max demonstrates this well as the threshold is relatively high for this trained network, which reduces both true and false positive rates.

It should be stressed that both LiDARNet and RJMCMC [31] have no prior knowledge of the exact number of positives contained in each waveform in this experiment. In the case of RJMCMC [31] the expected bounds have to be set (here  $K = \{0, 9\}$ ) and for LiDARNet the number of peaks is implicitly embedded in the network due to training with waveforms only containing up to  $K \leq 9$  positives.

It is curious that FRI-MP [110] has such major issues with this type of application, despite being given the correct number of returns. One key difference between this experiment and the super-resolution experiment is the type of background rate considered, which is low and constant for the super-resolution case but varies significantly in the case of the comprehensive simulated automotive dataset.

Despite demonstrating a very competent performance in separating two peaks in close proximity in low noise settings, the method tends to fit a single surface similar to RJMCMC in the automotive case, even though many of those peaks are made up of multiple individual returns from angled surfaces. Since RJMCMC makes a decision on how many peaks are estimated on a per waveform basis, it generally underestimates the number of true peaks, but with the given resolution confidence this is acceptable.

FRI-MP requires a precise number of returns in a waveform to fit the correct number of responses. When it identifies surface return cluster as a single return, it then seems to randomly fit surfaces throughout the solution space. This is a major downside of this approach, despite its otherwise excellent performance and overall simplicity. A failure case is illustrated with an example in Figure 4.12.



**Figure 4.12:** Multi-return issues with FRI-MP fitting too many surface locations randomly.

It can be seen above that while the approach does fit the main cluster correctly, it also randomly allocates the number of expected returns throughout the range, rather than fitting multiple returns under the cluster. This is the same waveform shown in Figure 4.13b, 4.13f.

In future work the false-positive rate should be reduced further for LiDARNet (and all other approaches), as they can be as dangerous in an autonomous driving situa-

tion as a false-negative and thus are not acceptable at these levels for safety critical tasks.

In terms of signal quality improvement, the ideal noiseless test waveforms are min-max normalised ( $x$ ) and all return estimates are convolved with the instrumental function and then min-max normalised to have a consistent comparison between all approaches. The reconstruction signal ( $y$ ) is evaluated using mean squared error (MSE) and power signal-to-noise ratio (PSNR) defined below for a reference  $x$  and a signal  $y$  as

$$\begin{aligned} \text{MSE}(x, y) &= \frac{1}{n} \sum_{i=0}^n (x - y)^2, \\ \text{PSNR}(x, y) &= 10 \log_{10} \frac{\max(x)}{\text{MSE}(x, y)}. \end{aligned} \quad (4.13)$$

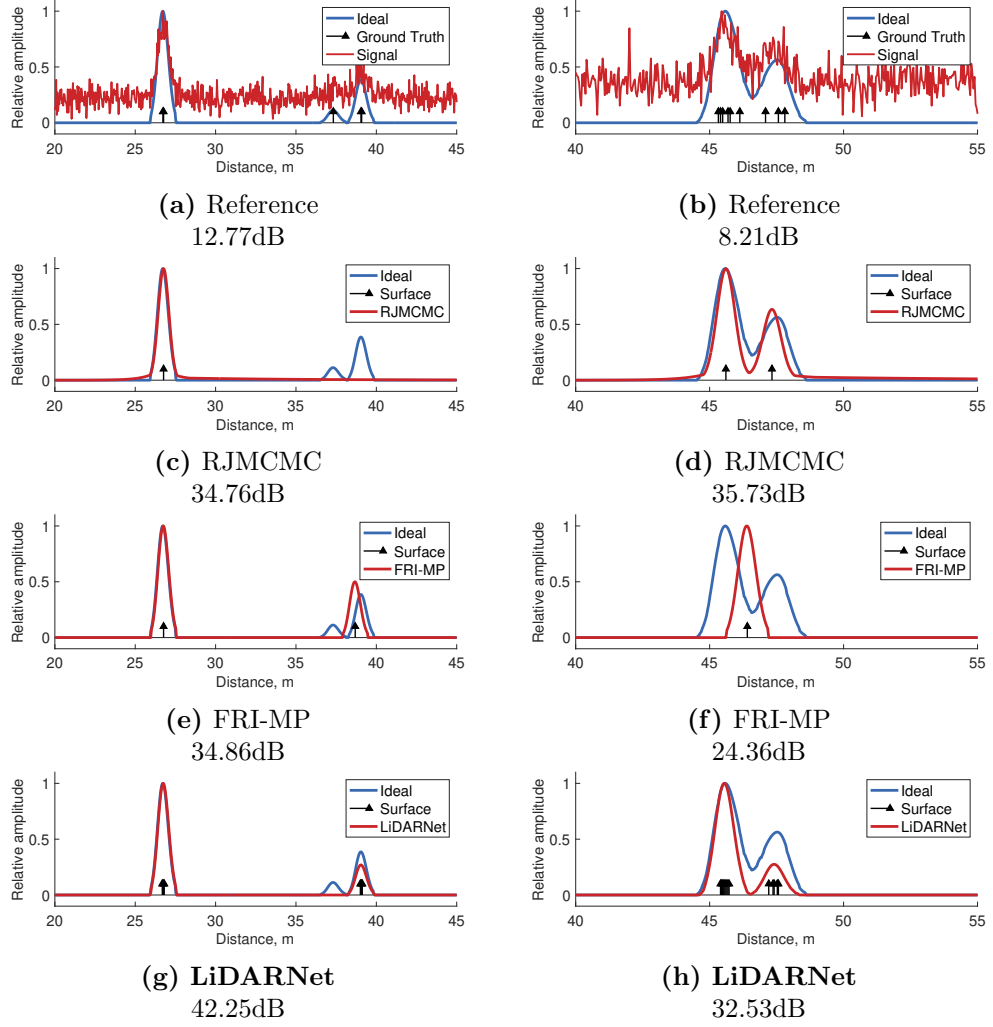
Both statistical approaches can recover estimates for actual photon return rates. LiDARNet discards this information currently, but still recovers relative amplitude between surface returns. The SNR evaluation is shown in Table 4.8.

**Table 4.8:** Evaluation of LiDARNet in *Automotive* configuration for 4000 simulated waveforms ( $K \leq 9$ ) compared to ideal noiseless waveform and processing times for a single waveform.

| Metric | Unit | Noisy<br>Waveform | RJMCMC [31]    | FRI-MP [110] | <b>LiDARNet<br/>MinMax</b> | <i>LiDARNet<br/>Max</i> |
|--------|------|-------------------|----------------|--------------|----------------------------|-------------------------|
| PSNR   | dB   | 13.12             | 47.59          | 47.04        | <b>51.45</b>               | 50.98                   |
| MSE    |      | 0.19344           | <b>0.00013</b> | 0.00045      | 0.00027                    | 0.00042                 |
| Time   | s    | -                 | 14.7362        | 0.0092       | 0.0044                     | 0.0044                  |

Over the entire dataset of 4000 simulated waveforms with  $K \leq 9$ , LiDARNet-MinMax recovers the signal with the highest PSNR, followed by LiDARNet-Max and then RJMCMC. FRI-MP overall improves signal quality dramatically as well despite the issues with false positives, since the randomly placed returns are generally of lower amplitude than the estimates within true return clusters. RJMCMC has the lowest MSE closely followed by LiDARNet-MinMax.

To further investigate the reconstruction behaviour of the respective approaches, a few sample waveforms are considered in Figure 4.13 for LiDARNet-MinMax in the *Automotive* configuration and  $\tau = 0.025$ . All waveforms are min-max normalised to be comparable throughout. Noiseless waveforms are also shown, which are reconstructed from surface locations by convolution with the instrumental function for all methods considered.



**Figure 4.13:** Signal reconstructions from detected surface return positions for noisy long-range LiDAR waveforms. (a) a mid-range scenario with moderate noise and 3 peaks – (b) a mid-range scenerio with high levels of noise and multiple peaks in close proximity. All approaches are shown under each respective waveform.

For the first scenario with three returns which are well separated with moderate noise, noting that the second peak is close to the noise threshold, RJMCMC only recovering the first peak (c) and (e) FRI-MP recovers only the first and third peak, but the third is not well aligned with the true surface location. LiDARNet (g) retrieves 2 peaks accurately and misses only the second.

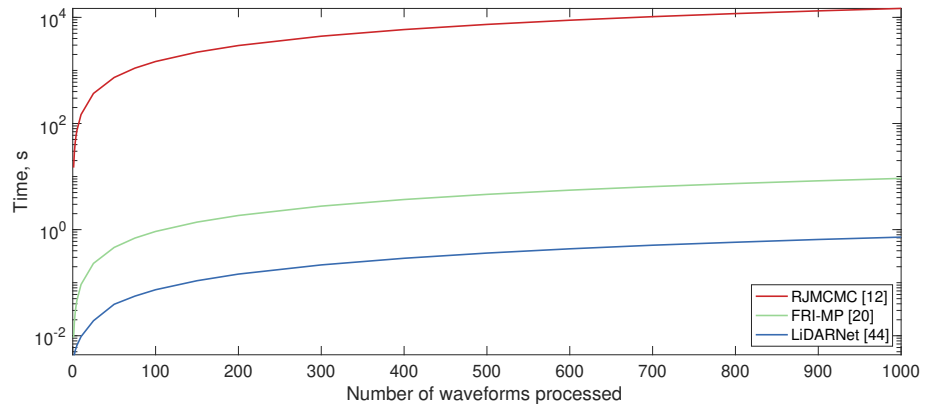
For the second scenario, a waveform with 2 close clusters is shown (b), with each cluster containing multiple surface returns (a total of 9 returns). RJMCMC (d) fits a peak under each cluster but does not resolve any of the individual surface returns. FRI-MP (f) only fits a single peak in the middle of the two clusters and LiDARNet (h) reconstructs multiple closely resolved surfaces under each cluster albeit at the wrong overall amplitude. This reconstruction does more accurately represent the surface distribution but RJMCMC’s singular peaks fit the the shape of the clusters

better and thus has a higher PSNR.

It is clear from this evaluation that LiDARNet can perform the same tasks as statistical models, without an explicit signal model nor any prior information on noise or number of surface returns. This is a proof-of-concept for using a data driven approach to enable LiDAR waveform processing using a single discrete processing approach for multiple scenarios. Here, it has been demonstrated that LiDARNet can also perform super-resolution, within the wider experiment of long range LiDAR applications in an automotive context in presence of significant background noise variation, which was previously only shown in the validation experiment for a short range experiment.

It should in theory be possible to augment the simulation approach and thus the training dataset to include other typical scenarios making surface proposals challenging in particular adverse weather conditions to improve the processing capabilities of LiDARNet continuously. Given that the simulation approach contains some form of signal model, LiDARNet is guided implicitly by such model assumptions and is therefore somewhat of a hybrid between model and data driven approaches.

So far the performance of LiDARNet has been evaluated in terms signal quality, however, the motivation to investigate deep learning for solid-state LiDAR was also the processing speed in presence of large concurrent detector arrays. To evaluate the potential processing speed for an array, the processing times for all 3 considered methods is evaluated in terms of number of waveforms. It assumed that RJMCMC and FRI-MP scale linearly with the number of waveforms. More recent work has been carried out to accelerate RJMCMC but primarily in terms of single waveform acceleration [103] rather than concurrent waveform processing. So the graph in Figure 4.14 can only be seen as a relative comparison and illustrates primarily the scaling potential of LiDARNet.



**Figure 4.14:** Scaling of processing times for multiple concurrent waveforms comparing statistical approach with proposed LiDARNet machine learning approach.



Even if either statistical approaches can be readily distributed across several compute units, LiDARNet demonstrates its inherent capability to scale with multiple inputs and outputs with near constant processing times until memory limitations of the GPU are reached as the number of concurrent waveforms increases. In other words, LiDARNet can readily process e.g. 10 waveforms at less than 10 times the speed of processing a single waveform. It also has a constant processing time for a particular number of waveforms, because a trained network has a constant runtime without any significant variation enabling dependable high throughput processing. This makes it predictable in terms of speed, allowing for efficient resource allocation and process scheduling, which is important for complex tasks such as autonomous driving with very limited time budget for decision making.

## 4.7 Summary and Conclusions

This chapter has briefly introduced the major components of modern deep neural networks and a popular architecture used for signal processing tasks, the convolutional auto-encoder. Some of the recent applications to LiDAR processing and 1D signal processing were also discussed. A distinct lack of applying this modern technique directly to full LiDAR waveforms with meaningful abstraction was worth investigating, as a trained neural network has the two advantages of being fast with constant processing times per waveform providing high throughput processing.

A novel neural network architecture for localising surface positions directly from LiDAR waveforms using deep learning was developed and makes the following contributions:

- A neural network architecture capable of localising surface locations discretely from LiDAR waveforms, called LiDARNet.
- A validated training-by-simulation approach, which allows to train LiDARNet with simulated photon data capable of learning generic LiDAR waveform features which readily transfer to real data applications of the network.

The simulation framework of this thesis was used to generate the required large scale datasets to train deep neural networks. An initial test with a standard convolutional auto-encoder architecture indicated that this approach was not capable of generalising features and thus was only successful in like-for-like data with specific boundary conditions. This also meant that the auto-encoder did not transfer over to real unseen data or any data which deviated from the training data e.g. different noise levels.

Motivated by the capability to extract waveform features for a limited scope (specific noise level or specific target count), a dedicated architecture was designed around the encoding stage of the auto-encoder and a traditional deep classification stage employing hidden layers i.e. many fully connected artificial neurons to decode the signal into its principle surface locations. Dropouts were introduced strategically to reduce training bias alongside the generous allocation for the encoded information in a relatively large fully connected layer derived from the full all available data from the convolutional stage. The architecture was named LiDARNet and not only performed well on simulated data but also readily processed real data successfully when only trained using simulated data.

The training-by-simulation approach was validated against a real LiDAR super-resolution dataset with comparable performance to prior art. This illustrates LiDARNet’s capability to learn significant features of LiDAR waveforms in a general fashion capable of transferring it to new and different data within the scope of the specifications. An illustration of the inner workings of LiDARNet was also presented illustrating some features resembling signal properties in line with the signal model considered but also very different unusual features directly derived from learning from the input data.

A challenging automotive use case was created using the simulator described in Chapter 3 with multiple returns and high noise variation. LiDARNet again performed very well with respect to the established statistical approaches, but did so much faster with per waveform processing times in the low millisecond range and processing times remain constant unless memory bottlenecks are encountered multiple concurrent waveforms. Although RJMCMC demonstrated similar performance to LiDARNet, it requires several seconds to process a single waveform. A finite-rate-of-innovation approach was also considered, but it faired poorly in high noise scenarios with a very high false-alarm rate.

The presented architecture is not without shortcomings though. All approaches considered suffered from poor false-alarm rates and LiDARNet can still exhibit some bias towards training data. This should be addressed in future work, ideally with an extensive real dataset in conjunction with simulated data to augment the training approach rather than being solely based upon simulation. Further, the network has been by no means optimised and could benefit from more dedicated optimisation towards the signal format, specific activation functions and potentially custom layer configurations to further improve performance.

In conclusion, LiDARNet demonstrates the feasibility of applying deep neural net-

works directly to raw LiDAR waveforms with useful information extraction in the form of direct surface locations and excellent reconstruction quality of the waveform if required. The performance is comparable to statistical approaches while outperforming them dramatically in terms of speed. Further, once trained, the network will operate reliably at a constant speed and scales well to allow for concurrent waveform processing as required for large scale solid-state SPAD arrays for automotive LiDAR applications.

# Chapter 5

## Concurrent Block Sparse Sensing LiDAR

### 5.1 Introduction

For ever higher image resolution the amount of information to be acquired, processed and ultimately stored increases accordingly. In modern 2D colour cameras images are captured with millions of pixels, which produces a significant amount of data to be stored. While storage capacities have increased dramatically and have enabled work with raw image data, in most cases, the raw image is acquired and then compressed for more efficient transfer and storage.

Image compression exploits image attributes in some other domain such as the frequency domain, where the information is much more concise or sparse. However, this process is not without compromises. The transformations from raw image data to a compressed image requires complex computations and to maintain high image quality, in some cases lossless, the computational complexity increases. [176, 177].

In recent years, compression and sampling theory have been investigated further to eliminate the need for acquiring large amounts of raw data. The question posed was if the principal information can be acquired directly instead of sampling a great amount of data which is later discarded, while still requiring a computationally intense reconstruction? The answer is, yes, and the field this research created has been called compressive sensing (CS).

In normal two dimensional colour imaging, each pixel captures an intensity value for the three primary colours and in some cases an alpha value. In direct time-of-flight (ToF) imaging using solid-state photon detector arrays, which is considered in

this work, each pixel can capture a histogram accumulating a photon count distribution with hundreds and even thousands of data points. However, in most cases the histogram is sparsely populated with true photon returns. Further, the resultant depth maps have comparable features to traditional images albeit encoding distance rather than colour intensities. This means that higher resolutions increase the amount of sampled data significantly for depth versus traditional intensity imaging and leads to even greater challenges in terms of bandwidth and storage capabilities for a depth imaging system.

While most colour imaging systems operate in a passive fashion, ToF operates on the principle of active illumination and as discussed in Chapter 2 the amount of power radiated has limits imposed by health and safety regulations to avoid any potential damage, in particular to eyes of humans and other animals as well as other devices. As ToF imagers increase their resolution and range, the desirable burst illumination for the fastest mode of operation outputs more and more laser power per unit area. This puts limiting factors on the system design and makes burst illumination for large array imagers infeasible. Compressed sensing requires some form of structured illumination, which provides design flexibility for the emission system. This means that emission density and therefore the laser output power per unit area can be managed.

Compressive sensing can both reduce the amount of data captured initially and can provide a way to control the laser power output density, although this does not come without a penalty. It incurs significant computational cost, i.e. the complexity of the system shifts from the emission/detection system to the signal processing stage of the system and can also require a longer sampling time.

This thesis makes the following contributions to the field of compressive depth recovery:

- Formalising and extending an existing depth reconstruction scheme to enable complex depth recovery for a novel blocked compressive depth recovery framework.
- A blocking scheme for arrayed detectors to enable small scale compressive sensing, improving depth reconstruction quality with orders of magnitude faster processing speeds.
- A system architecture which takes full advantage of an arrayed emitter and detector array, enabling the parallelisation of compressive depth sensing, providing a pathway to extremely fast compressive depth imaging at long range.

- A novel block architecture, which proposes a block sparse depth sensor with integrated reconstruction logic to enable high resolution high frame rate light detection and ranging (LiDAR) imaging for long range applications.

This chapter introduces the theory of compressive sensing and a literature review on blocked compressive sensing as well as compressive sensing frameworks for depth imaging in Section 5.2. The shortcomings of current schemes are discussed, namely long sampling times and expensive processing, when applied to arrayed ToF imagers of increasing array sizes. The contributions of this thesis are presented in detail from Section 5.3, onwards including a detailed signal and system formulation, a sparsity analysis of the compressive depth problem when applied to small block sizes. The small scale problem is then parallelised and scaled up for solid-state ToF arrays in Section 5.4 and results are presented in Section 5.5. The chapter concludes in Section 5.6.

## 5.2 Compressive Sensing

Many natural signals can be represented in a compact fashion, reducing their size dramatically. This is achieved by transforming a signal into another basis domain. The linear transformation operator,  $\theta$ , is in effect an instruction set translating a signal into another domain. The concept of compression arises as some signals are sparse in some domain while being dense in their natural domain. This means the transformed signal can be represented with fewer non-zero components,  $m$ , than the  $n$  components required in its natural domain, for example a natural RGB intensity image and its wavelet representation. For the remainder of this thesis a vectorised format is used unless stated otherwise. A two-dimensional signal  $X \in \mathbb{R}^{N_x \times N_y}$  is vectorised such that

$$x \in \mathbb{R}^n = \text{raster}(X), \quad (5.1)$$

where  $n := N_x \times N_y$ , with  $N_x$  being the number of rows in a matrix or image container array and  $N_y$  the number of columns. All non-zero components  $m$  can be stored more efficiently by means of only storing positions of significant signal components. The forward and backwards transformations incur significant computational cost, but the benefit of smaller data volume has been deemed worth the trade-off in digital communications. Most notably in image [177] and video [176] data transfer where bandwidth is limited but local processing power on host and/or client side is sufficient to perform the necessary transformations on an ad hoc basis.

A sparse signal,  $\chi$ , is a signal with less significant values (i.e. non-zero) than zero

(or close to zero) entries. In this work a signal is  $s$ -sparse if it has  $s$  non-zero entries. And the sparsity,  $\Psi$ , is a relative measure such that

$$\Psi = \frac{s}{n}. \quad (5.2)$$

In other words the smaller  $\Psi$  the sparser a signal.

Compressed sensing is the mathematical theory of directly capturing the significant components of a natural signal and thus eliminating the need for forward compression. A projection operator,  $\phi$ , is introduced which samples a signal in a structured fashion. It is noteworthy that the structure can be random, orthonormal or otherwise as long as it satisfies the restricted isometric property (RIP). Isometry is the transformation of two metric spaces, which preserves the distance measure between them. In the context of compressed sensing [178] introduces the restricted isometric constant (RIC),  $\delta_s$ , for a  $s$ -sparse signal and a matrix,  $F$ , as

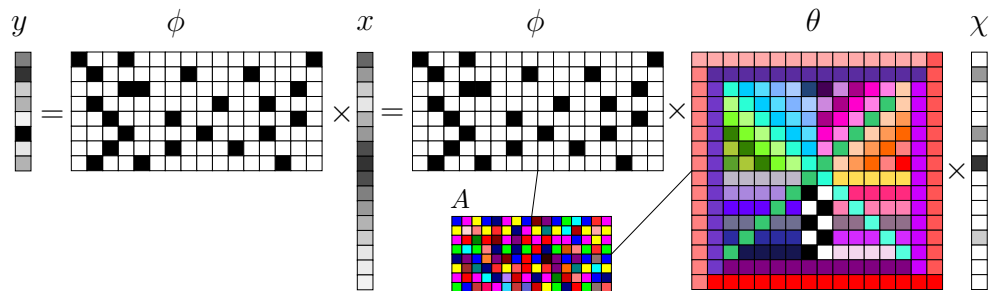
$$(1 - \delta_s)\|x\|^2 \leq \|Fx\|^2 \leq (1 + \delta_s)\|x\|^2. \quad (5.3)$$

If the RIP inequality holds, then  $F$  and all sub-vectors of  $F$  of a  $s$ -sparse signal approach isometry and thus approximate an orthonormal system. It has been shown that Gaussian random matrices provide the smallest upper bounds for rectangular matrices in terms of isometry. This means in the context of CS, that Gaussian projection matrices provide the most likely transformation into the sparsest basis of a signal [179].

The theory of compressive sensing was independently formulated by [76] and [75]. It provides a robust framework to recover a signal,  $x \in \mathbb{R}^n$  from a small subset of measurements  $y \in \mathbb{R}^m$  by means of linear programs of the form

$$y = \phi x + \epsilon = \phi \theta \chi + \epsilon, \quad (5.4)$$

where  $\phi$  is the projection transform,  $\theta$  is a linear basis transform as mentioned previously and  $\epsilon$  is noise which most often is i.i.d. Gaussian random noise but in imaging applications could also be Poisson random [180] for very low photon count scenarios. In literature a sensing matrix is often  $A = \phi \theta$ . For cases where reconstruction is performed on the signal directly or the sparse basis is also the projection,  $A = \phi$ . A visualisation of this problem is shown in Figure 5.1.



**Figure 5.1:** Compressive sensing visualisation [181]. The transform  $\theta$ , the projection  $\phi$  and the sensing matrix  $A$  are purely illustrative and do not necessarily depict a real representation thereof.

To reconstruct the data the problem is formulated as an optimisation problem

$$\begin{aligned} \min_x \quad & \alpha \|\theta x\|_1 \\ \text{s.t.} \quad & \phi x = y, \end{aligned} \tag{5.5}$$

where  $\|\cdot\|_1$  is the pseudo  $\ell_1$  norm and  $\alpha$  is a weighting parameter. This optimisation problem can be solved as a basis pursuit de-noising (BPDN) problem in the form of

$$\min_x \|\phi x - y\|_2^2 + \alpha \|\theta x\|_1, \tag{5.6}$$

where  $\|\cdot\|_2^2$  is the  $\ell_2$  or Euclidean norm.

The use of CS principles has seen significant popularity in signal and imaging applications which naturally operate in sparse domains for data analysis in some cases even acquisition. A lot of early CS imaging work has focused on magnetic-resonance-imaging (MRI) imaging which operates in the frequency domain [182] and is still a very active field of research [183, 184, 185, 186, 187, 188].

In this work we aim to reconstruct natural depth image signals, while the majority of CS focuses on two dimensional intensity and colour imaging most often following the principles of the breakthrough application of CS theory to imaging, the single pixel camera.

### The Single-Pixel Camera

One of the most impressive applications of CS theory is the single-pixel camera first introduced in [189]. It demonstrated a useful application of these principles to a very useful application; imaging. Modern image sensors can readily capture high resolution images in the visible domain and the challenges of reading out such large amounts of data have mostly been overcome. However, for imaging outside the



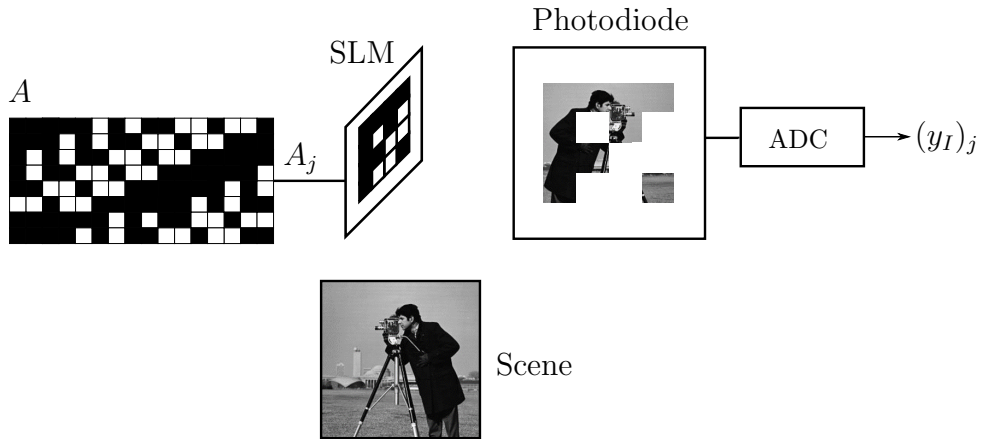
visible spectrum, detector costs are often prohibitive due to scarce materials and difficult manufacturing processes.

The single-pixel camera proposes a meaningful path for imaging at a high resolution by encoding the resolution with a passive device, a spatial light modulator (SLM). In most cases this is a digital-micromirror-device (DMD) which enables the user to structure a scene response via pattern projections from the scene onto a single detector. In the case of a DMD many switchable mirrors, can be toggled to form a pattern, where mirrors can attenuate scene elements by deflecting light in a controlled fashion. This structured sampling approach links the spatial information defined by the sensing pattern to the captured scene response from a single detector. CS enables a system designer to reconstruct full images with a resolution defined by the SLM at wavelengths for which sensing elements are either expensive to produce in arrayed form, or may not exist at all.

An intensity image vectorised as  $x_I \in \mathbb{R}^{n=N_x \times N_y}$  has an associated measurement defined as

$$(y_I)_j = A_j x_I. \quad (5.7)$$

which is the inner product of the scene  $x_I$  with the  $j^{\text{th}}$  projection or  $j^{\text{th}}$  row of  $A$ . This forms a cumulative intensity measurement  $(y_I)_j$  illustrated in Figure 5.2. There are drawbacks associated with the reduction in detector cost and those are



**Figure 5.2:** Illustration of the single pixel camera first presented in [189].

acquisition time and reconstruction complexity. For high resolutions with single detectors, acquisition times of several seconds and even minutes are not uncommon. This is still orders of magnitude faster compared to point scans of the scene, as the number of measurements,  $m \ll n$  in CS. Therefore, it accelerates and improves the

feasibility of non-visible imaging dramatically but is nonetheless a limiting factor.

Another trade-off is the costly reconstruction and often large projection patterns, making the reconstruction memory intensive. This has been addressed with transform based pattern generations, where the pattern is constructed on-the-fly rather than stored, which in of itself comes at a computational cost. However, one key advantage of compressive sensing is the theoretical link that a unifying projection is optimal to most basis functions, which counter-intuitively is a random projection [75, 190, 191, 179, 192]. This makes the single-pixel camera a future proof system, where the measurements can yield better results if captured with a random projection, as the basis assumptions can be imposed as a post-processing step and thus can be altered if a more suitable basis functions for the given signal are discovered.

To achieve CS imaging practically one particular sensing matrix structure has seen particular popularity based on the Walsh-Hadamard transform. One of the key reasons for its wide use in optical CS imaging is that the  $\pm 1$  structure can be readily implemented with optical binary modulators with alternating binary patterns. Due to its coherence to the Haar wavelet transform it is often randomly permuted to reduce coherence between the measurement and the wavelet basis [189, 193]. It is important to note though, that such Walsh-Hadamard patterns have by definition a relative sparsity of  $\Psi \approx 0.5$ . While this is perfectly acceptable for passive imaging systems, it means for active imaging systems that the pattern has a constant power output and is fairly dense. This can impose limits on the operating range similar to the restrictions of full flash illumination LiDAR as discussed in Chapter 2.3.3 for long range applications. To gain full flexibility, the projection pattern should be as sparse as possible, while maintaining good performance in the CS regime.

### 5.2.1 Linear transforms and projection patterns

One key element in compressive sensing is the projection pattern  $\phi$  and its relationship to the linear transform  $\theta$ . While there are many transforms to consider, this work focuses on transforms which can be readily implemented in a discrete fashion with simple matrix arithmetic. While some work has shown that Noiselets and Curvelets provide good support for imaging applications and robustness to noise [194, 77], their implementation is much more complex than Haar [195], Daubechies [196] and the wider family of wavelets as well as the discrete cosine transform [197], which all sparsify the signals considered. They also have straightforward discrete implementations, which can often be approximately computed with a single coefficient matrix and its inverse using straightforward linear algebra. The focus of this work is to demonstrate a way to achieve real-time operation from sampling

through to processing at high frame rates. This requires the ability to efficiently implement the proposed frameworks on hardware. Another important linear transform considered for compressive imaging is total-variation (TV) [198], which can be implemented discretely as difference matrices, making it also suitable for fast and efficient implementations [199].

A key finding of CS theory was the use of a random measurement operator for  $\phi$ . It was shown that Gaussian random matrices are near optimal i.e. are incoherent to most basis functions and satisfy the RIP [75, 190, 179, 200]. While in theory a fully random Gaussian matrix is achievable, it is more practical to use binary sampling operators. For such Gaussian random operators, bounds have been derived relating an  $s$ -sparse signal of size  $n$  to the number of measurements  $m$  [201, 202],

$$m \geq 2s \log(n/s) + 7s/5 + 1. \quad (5.8)$$

The compression ratio in this work is defined as  $\mathcal{C} = m/n$ . A popular choice in single-pixel camera systems is the Walsh-Hadamard transform,  $H_n \in \{\pm 1\}^{2^n}$ . It is a square size transform and is closely related to the Fourier transform but operates solely in the real domain [203]. It is also partially coherent with Haar wavelets [193]. Nonetheless, it has been shown to be a practical choice due to its simplicity of being a binary transform. To achieve incoherence to any basis a series of patterns are generated as random permutations of  $H_n$ . To encode the  $\pm 1$  nature, a measurement is made up of two projections from the same pattern, where  $H_{n+} = (H_n < 0) = 0$  and  $H_{n-} = \text{abs}((H_n > 0) = 0)$ . A measurement is the response from a single permutation  $y_j = (H_{n+})_j x - (H_{n-})_j x$ . The Walsh-Hadamard transform illuminates effectively the entire scene albeit in two steps. This, however, doubles the sampling time and thus makes it not the fastest choice in terms of sampling rate. It should be noted, though, that due to its structure and pseudo-random permutation, the transform can be defined implicitly, making it possible to generate the pattern on the fly while maintaining repeatability in the measurement sequence [200, 204].

While those fairly structured approaches have proven overall successful they provide little flexibility over the density of the projection pattern in relation to a  $s$ -sparse signal. Therefore they provide no way to align the projection density with the relative sparsity of a signal, which is particularly important if one also wants to minimise the active illumination output power as in this work.

To gain flexibility in the sampling operator design while also gaining the advantage of discrete generation various binary projection schemes have been proposed [205, 206, 207, 208]. It has been shown that such random binary matrices perform closely to

Gaussian random matrices and uphold the same properties making them suitable for applications in CS imaging, albeit they perform better for smaller signals [205]. This is an important finding as it allows for procedurally generated projection pattern which perform similarly to a Gaussian random binary matrix obviating the need the storage.

It would be ideal to utilise a near-optimal Gaussian random matrix over discrete matrices as they are more robust for a range of  $s$ -sparse signals and can be treated as universal with respect to most basis transforms. However, in most single-pixel systems a sequence of patterns for moderate image sizes of  $n \geq 64 \times 64$  can become a memory issue, as all  $m$  instances of the pattern have to be permanently stored and accessed throughout the optimisation. As image size increases this becomes an infeasible proposition. A typical approach to reduce the computational burden and memory requirements in these cases is the use of blocking schemes.

### 5.2.2 Blocked Compressive Sensing

As image sensors increase in size, the amount of data being sampled and processed rises dramatically and puts significant strain on bandwidth and computing resources. One of the common approaches to reduce the computational burden is to split a complex large scale problem into many smaller problems. This of course is only effective if computations can be executed in parallel and the incurred communication overhead does not significantly impact the savings made. Since CS problems considered thus far all operate on large image signals they often have very expensive reconstruction routines which can take many seconds or longer. So although the sampling is more efficient and in some cases faster than traditional image acquisition, the structure of recovering the significant components into natural signals can limit real-time processing alongside acquisition. In the single-pixel camera system, the problem size is inherently linked to the size of the projection matrix,  $\phi \in \mathbb{R}^{m \times n}$ . Splitting this large scale problem into many small scale problems can accelerate the computation of the solution dramatically.

As before an image signal is  $X$ . Now let the signal be divided into non-overlapping blocks of size  $n_B = N_B \times N_B$ . This results in  $N_B = n/n_B$  blocks. Splitting a natural image into small blocks of size  $n_B \lll n$  not only reduces the signal size per problem but also reduces the number of measurements  $m_B \lll m$  as the principal components of (5.8) are  $s$  and  $n$  with  $s \leq n$ , so if  $n_B < n$  decreases so does the the number of measurements  $m_B < m$  per block. This reduces the per block complexity but the total captured information  $m = n \cdot m_B / n_B$  is usually comparable to full-scale CS.

The first application of a blocking scheme to CS was proposed in [129] as block compressive sensing (BCS), which utilises effectively a 3-stage recovery, where  $m_B = m(n_B/n)$ . This scheme assumes an optimal block size of  $N_b = 32$ . Each block is sampled with an orthonormal i.i.d. Gaussian matrix,  $\phi_B \in \mathbb{R}^{m_B \times N_B^2}$ , such that the CS problem becomes

$$y_i = \phi_B x_i, \quad (5.9)$$

per block for measurement vector  $y_i \in \mathbb{R}^{m_B}$  and vectorised signal  $x_i \in \mathbb{R}^{B^2}$ . Performing block independent reconstruction yields an initial estimate for further more sophisticated global reconstruction. However, these initial estimates often result in undesirable blocking artefacts for the block sizes considered [129, 209]. To find an initial per block reconstruction, a solution could be estimated using the linear least square approach, such that  $\hat{x} = \phi^+ y$ , where  $+$  denotes the pseudo-inverse. This, however, performs poorly for an under-determined system ( $m \ll n$ ). The authors propose that each signal is estimated using minimum mean squared error linear estimation (MMSE) with an auxiliary reconstruction matrix,  $\hat{\phi}_B = R_{xx} \phi_B^\top (\phi_B R_{xx} \phi_B^\top)^{-1}$ . But it is not clear how the auto-correlation function,  $R_{xx}$  is estimated without prior signal knowledge. This yields an estimate  $x^{(0)} = [\hat{x}_1 \dots \hat{x}_{n/n_B}]$ . Next, iterative-thresholding is applied, a special case of a projected Landweber (PL) algorithm [210, 211, 212, 213], with  $x^{(0)}$  as the initial global estimate. The associated projection matrix is straightforward to construct from the individual block matrices with a diagonal block structure,

$$\phi = \begin{bmatrix} \phi_B & 0 & \dots & 0 \\ & \ddots & & \\ \vdots & & \phi_B & \vdots \\ & & & \ddots \\ 0 & \dots & & \phi_B \end{bmatrix}, \quad (5.10)$$

which is essentially a block raster scan. The full signal is then projected onto a convex set and hard-thresholded to enforce sparsity using the initial estimate  $x^{(0)}$  and  $\phi$ . In the first stage this is done in the Lapped Transform,  $\theta_{LT}$  discrete cosine transform (DCT) domain i.e. the basis transform extends beyond the block boundary, [214], followed by wavelet domain  $\theta_{WT}$  and the oversampled Lapped transform  $\theta_{OLT}$  hard-thresholding. This is illustrated in Algorithm 1.

These steps perform full-scale non-linear reconstruction with the only benefit being the simpler construction of  $\phi$ . This means that despite the savings made due to

---

**Algorithm 1** Block-compressive sensing BCS reconstruction [129]

---

**Input**  $y, \hat{\phi}_B, \phi, \theta_{LT}, \theta_{WT}, \theta_{OLT}$   
**Output**  $\hat{x}$

```

for every block  $i$  do
   $\hat{x}_i^{(0)} = \hat{\phi}_B y_i$ 
end for
for  $k = 0$  to  $K$  do
   $x_w = \text{Wiener}(x^{(k)})$ 
   $\hat{x}^{(k+1)} = \text{Transform Threshold}(\theta_{LT}, x_w^{(k)})$ 
end for
 $\hat{x}^{(0)} = \hat{x}^{(K)}$ 
for  $k = 0$  to  $K$  do
   $\hat{x}^{(k+1)} = \text{Transform Threshold}(\theta_{UWT}, \theta_{OLT}, \hat{x}^{(k)})$ 
end for
 $\hat{x} = x^{(K)}$ 

```

---

the block structure, the sparsity regularisation is still a full scale problem with run times well above 30 s and up to 5 min, meaning that real-time operation is still not achieved. In comparison to full scale processing methodologies at the time, the reconstruction is several orders of magnitude faster though.

The approach of using iterative-thresholding on a block estimates was further improved in [213, 209], where a Wiener filter was directly incorporated into the basis thresholding rather than performed in two separated steps leading to the smoothed projected Landweber (SPL) approach to BCS shown in Algorithm 2. This algorithm improves upon reconstruction quality further, but improves only marginally upon the processing times, which range between 1-5 minutes per  $512 \times 512$  image. While more recent work [215] has reduced processing times to  $< 10$  s, using a single-value decomposition scheme for the block operations estimations in an otherwise similar scheme to BCS-SPL [213].

The concept of block based problem distribution has been successfully applied to CS but processing times demonstrated are still  $> 1$  s, which is not desirable for applications where real-time decisions are to be made based on the desired output signal made from efficient low bandwidth CS measurements. For standard intensity imaging and global post-processing, the most common block size is  $n_B = 32 \times 32$ . In this thesis it is investigated if smaller block sizes can help to reduce the computational burden if savings made due to compressive signal acquisition are more significant than is the case with normal intensity measurements.

In contrast to most CS schmes including blocked variants just described, this work considers the final output to be depth information, which can either be represented as a point cloud, LiDAR cube or simply a two-dimensional depth map of first returns with a more costly signal acquisition for depth due to the added dimension and

---

**Algorithm 2** Block-compressive sensing with smoothed projected Landweber reconstruction BCS-SPL [213]

---

**Input**  $y, \phi_B, \theta$   
**Output**  $\hat{x}$

```

for every block  $i$  do
     $\hat{x}_i^{(0)} = \phi_B^\top y_i$ 
end for
 $k = 0$ 
repeat
     $x_w = \text{Wiener}(\hat{x}^{(k)})$ 
    for every block  $i$  do
         $\hat{\hat{x}}_i^{(k)} = x_w + \phi_B^\top (y_i - \phi_B \hat{x}_i^{(k)})$ 
    end for
     $\check{\hat{x}}^{(k)} = \theta \hat{\hat{x}}^{(k)}$ 
     $\check{\hat{x}}^{(k)} = \text{Threshold}(\check{\hat{x}}^{(k)})$ 
     $\bar{x}^k = \theta^{-1} \check{\hat{x}}^{(k)}$ 
    for every block  $i$  do
         $\hat{\hat{x}}_i^{(k+1)} = \bar{x}_i^k + \phi_B^\top (y_i - \phi_B \bar{x}_i^k)$ 
    end for
     $D^{(k+1)} = \|\hat{\hat{x}}^{k+1} - \hat{\hat{x}}^{(k)}\|_2$ 
     $k = k + 1$ 
until  $|D^{(k)} - D^{(k-1)}| < 10^{-4}$ 
 $\hat{x} = \hat{\hat{x}}^{(k)}$ 

```

---

scales dramatically with desired range precision over intensity imaging as discussed in Chapter 2.

Significant savings are possible by applying CS principles to such complex signals, but this further increases the complexity of the signal reconstruction. Blocking schemes can provide a pathway to decrease the processing time penalty. It is also noteworthy that most work considers the block independent solutions only as a prior to full-scale CS reconstructions and sparsity regularisation. While the construction of  $\theta$  at full-scale becomes trivial as only  $\theta_B$  must be stored, subsequent arithmetic operations still operate for  $n$ -length signals. Combined with basis transformations via  $\theta \in \mathbb{R}^{n^2}$  being appropriately sized for a  $n$ -length vector rather than a  $n_B$ -length block vector incurs significant complexity in mathematical operations, as these transforms often scale quadratically. Although fast and efficient transforms exist, this increase in complexity of current schemes may explain the still relatively long run times.

### 5.2.3 Compressive Depth Imaging

In this section a review of existing compressive depth imaging schemes is provided, which add the modality of range finding to the single pixel camera methodology by adding two key components; a pulsed laser source with timing and trigger sync to a photon detector. Time is added as an additional dimension into compressive sens-

ing, so two particular sparsity constraints can be considered; temporal and spatial sparsity. Both are summarised and discussed in the context of long range LiDAR applications for complex natural scenes.

### Temporal Depth Sparsity

Several compressive schemes have been proposed which exploit sparsity in the depth domain directly. They are based on modulated or coded illumination schemes, often referred to as indirect ToF.

For a continuous modulation scheme depth is derived from the phase difference between the transmitted and received waveforms,

$$F_{\text{in}}(\omega) = \rho e^{-j\omega\tau} F_{\text{out}}(\omega), \quad (5.11)$$

where  $\omega$  is frequency of the input and output wave  $F$ ,  $\rho$  the reflectivity of an object and  $\tau$  the time delay to be determined [216, 217, 218, 219, 220]. The frameworks described to sample the continuous return function in a compressive fashion reduces the complexity of the per pixel acquisition in time with sparsity in the frequency domain. This means that for high resolution depth imagers, the same limitations apply as with traditional scanning systems. Further, the sub-sample rate has the potential to degrade the actual depth precision. Additional constraints are also placed upon the compressive reconstruction, e.g. a single significant frequency, such that only a single depth is allowed in a return signal [217].

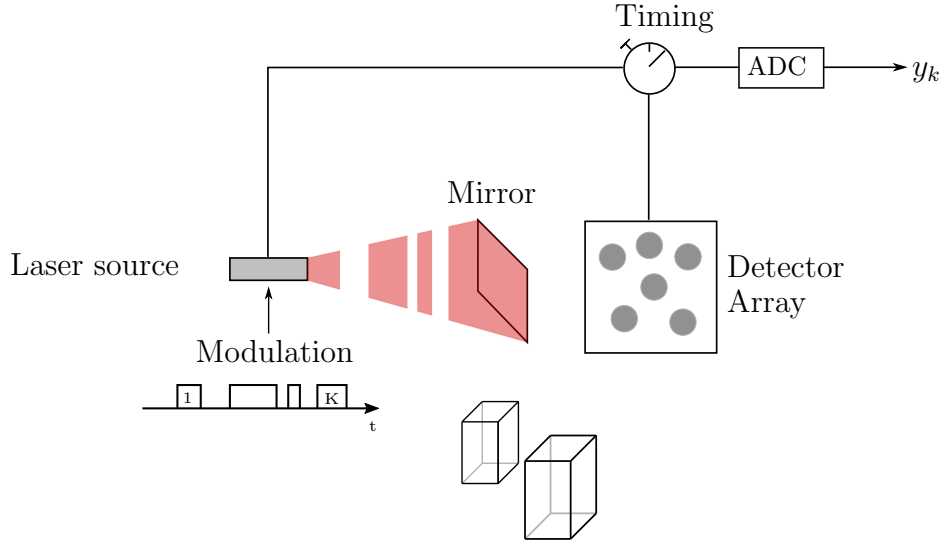
Other work limits the number of surfaces with geometric constraints [216, 218]. The complexity to increase the number of surfaces in these instances is non trivial or very expensive. While fair assumptions, it is not clear how, together with reduced depth precision, the compressive savings in the above cases improve overall system performance. Further, modulated ToF schemes are somewhat limited to shorter range applications, as the signal has varying amplitude and thus the minimal relevant amplitude needs to be at a high enough power level to receive a sufficient return signal, meaning that the maximum amplitude and thus power is higher than pulsed ToF for the same operating range.

The way this information content is compressively sampled is the key differentiator between the different works considered. So far sub-sampling was achieved in time at the sampling stage (analogue-to-digital converter (ADC) or time-to-digital converter (TDC)). It shall be noted here that while this indeed reduces the read-out data volume, it does not provide any time savings in the sampling stage, as this is bound by the desired operating range and its associated ToF,  $\delta t_{\text{max}}$ . Violating this principle



would make range measurements ambiguous.

Another scheme employs sparse encoding strategies prior to the sampling stage. For such coded schemes two approaches have been demonstrated, one based on a so called coded aperture [221] in the spatio-temporal domain, and another using a pulse coding scheme in the temporal domain only [222, 223] and is illustrated in Figure 5.3.



**Figure 5.3:** Illustration of coded compressive depth system as described in [223]. For  $K$  discrete pulses of varying pulse length measurements are taken, forming a compressive measurement vector.

These investigations aim to address the limiting assumption made thus far of single returns per pixel, by employing custom coding schemes to enable decoding of more than one return, while retaining compressive sensing advantages. This work relies heavily on sparse deconvolution, which can take the same form as BPDN (5.6) problem formulations commonly used in CS or similar linear program formulations utilising  $\ell_2$  or  $\ell_1$  norms and combinations thereof to enforce sparse reconstruction. In particular [223] proposes a more efficient computational scheme utilising a back-projection and thresholding scheme, but yet again it only considers short ranges with up to 3 surfaces.

It is becoming apparent that pulsed coding or continuous modulated coherent illumination are hard to extend to complex long range scenes due to their power requirements and range ambiguity. In particular, for the custom coding schemes, where pulses can be both longer and shorter than  $\delta t_{\max}$ , longer pulses contain more laser power which could afford longer ranges but shorter pulses with less power limit that maximum range. Further, this method often increases the time it takes to acquire a signal, which may be a reasonable trade-off in short range systems where

$\delta t_{\max}$  is fairly short, but can be prohibitive to fast frame rates for ranges of above e.g. 100 m.

The above schemes currently limit performance and seem unsuitable for long range LiDAR applications in automotive scenarios, but show potential in exploiting sparsity in the temporal domain for active coherent illumination systems such as LiDAR and radio detection and ranging (RADAR) in general. If they improve further and address some of the current shortcomings, they could be incorporated in spatial sparsity schemes to sense depth with compressive sampling in both the spatial and temporal domain.

In the context of this work, which considers long range depth imaging applications, exploiting temporal sparsity seems to impose too many limitations for complex scenes; low frame rates, increased laser power, while only providing marginal savings in the sampling stage at the cost of complex reconstruction, therefore the suitability of exploiting spatial depth sparsity is reviewed next.

### Spatial Depth Sparsity

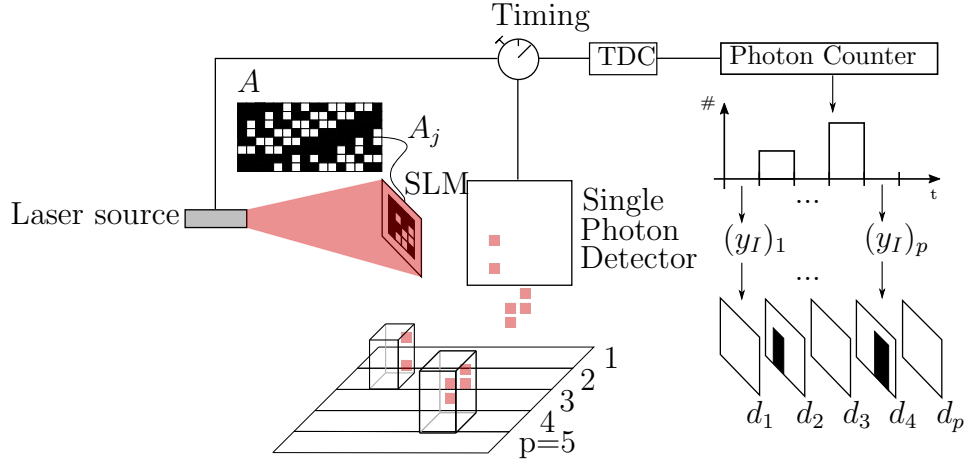
Another approach to compressive depth imaging is the exploitation of sparsity in the spatial domain. In [224] a hybrid system is presented which exploits both spatial and temporal sparsity and relies on a total of 4 phase images without compression using an array ToF camera. This enforces spatial constraints by linear combinations of pixel read-outs, to effectively utilise a passive sensing matrix in the read-out stage. This increase in phase images addresses another shortcoming of phase derived depth measurement as frequencies are periodic and thus can become ambiguous in the frequency domain. Each phase image can be treated as a normal single-pixel type reconstruction problems. The compressive scheme combines 2 phase images into a single measurement each and uses a blocking strategy to increase reconstruction efficiency.

One major downside of this scheme is the two-stage nature of depth recovery. First the phases have to be estimated for all pixels and then the spatial structure is recovered from linear combinations of pixels, which makes this approach very expensive in terms of reconstruction time and complexity despite employing a blocking scheme to accelerate the second stage with significant degradation in the final depth image. However, range and power limitations still apply as this is a frequency modulated scheme.

The system considered in this work utilises single photon avalanche detector (SPAD) detectors and a pulsed illumination scheme to allow for long range imaging, rather

than the indirect ToF frameworks discussed above. Such direct ToF systems have a finite depth sampling resolution with  $p$  steps and measurements are often accumulated in a histogram.

By exploiting spatial sparsity at every discrete depth step, depth sparsity is often implicitly assumed similar to the assumptions made in [217]. This means that very sparse intensity image slices are expected for every time step due to the general sparse nature of photon counting histograms and depth as illustrated in Figure 5.4.



**Figure 5.4:** Illustration of time-gated compressive single pixel depth system principles for the methodologies considered in e.g. [225, 204]. For  $p$  discrete time steps a normal single-pixel CS recovery is performed resulting binary depth masks forming depth slices.

This casts the compressive problem onto multiple 2D imaging problems and thus multiple instances of the standard CS problem formulation [225, 226, 204, 227]. They formulate the CS problem as

$$\begin{bmatrix} y_1^{(1)} & \dots & y_1^{(p)} \\ \vdots & \ddots & \vdots \\ y_m^{(1)} & \dots & y_m^{(p)} \end{bmatrix} = A \begin{bmatrix} x^{(1)} & \dots & x^{(p)} \end{bmatrix}, \quad (5.12)$$

where  $p$  is the number of time or depth steps and  $m$  the number of measurements. This transforms the single-pixel problem from one dimension to  $p$  dimensions. While the sensing matrix,  $A$ , remains of constant size and is shared among all  $p$  problem instances, it is not trivial to parallelise all instances at once, in particular for large  $p$ .

For applications where a high-dynamic range is desired, such as the envisioned task of long range automotive depth sensing with centimetre resolution,  $p$  can easily be in the thousands. In most cases each time-gated intensity slice is recovered sequentially, increasing the time it takes to compute results. Each intensity slice is then placed in a depth cube,  $D \in \mathbb{R}^{N_x \times N_y \times p}$  and a depth map,  $x_D \in \mathbb{R}^{n=N_x \times N_y}$  can be obtained

by forming a binary mask  $I_p = (x_I = 0)_p$ , which results in a depth map slice of  $(x_D)_p = I_p \circ d(p)$  and a flattened depth map image is then,  $x_D = \frac{\sum_p (x_D)_p}{\sum_p I_p}$ .

While real-time processing has been demonstrated in [227], it is not suitable for decision making at only 3 Hz for fairly short histograms of 512 bins. One common theme among these systems is the extremely fast sampling rates, theoretically enabling ultra-fast imaging, if the processing scheme can handle the high sampling rates. However, due to the problem formulation, processing of the acquired data despite compressive savings is still a challenging task for high spatial and high depth resolution.

To minimise the computational burden, mask priors are introduced in [228]. Here, the problem is constrained to two Lambertian surfaces. The surface positions are estimated from the scene response, i.e. a histogram,  $h \in \mathbb{N}^p$  in a parametric fashion similar to approaches presented in Chapter 2 and 4. The parametric depth estimation per pattern  $d_i$ , yields the respective measurement vector,  $(y_I)_i$ . The spatial shape is then recovered with two individual CS optimisations, where photon count information is discarded, yielding two binary masks describing the outline of either object,  $I_i \in 0, 1^{n=N_x \times N_y}$

$$[(y_I)^1, (y_I)^1] = A[I^1, I^2]. \quad (5.13)$$

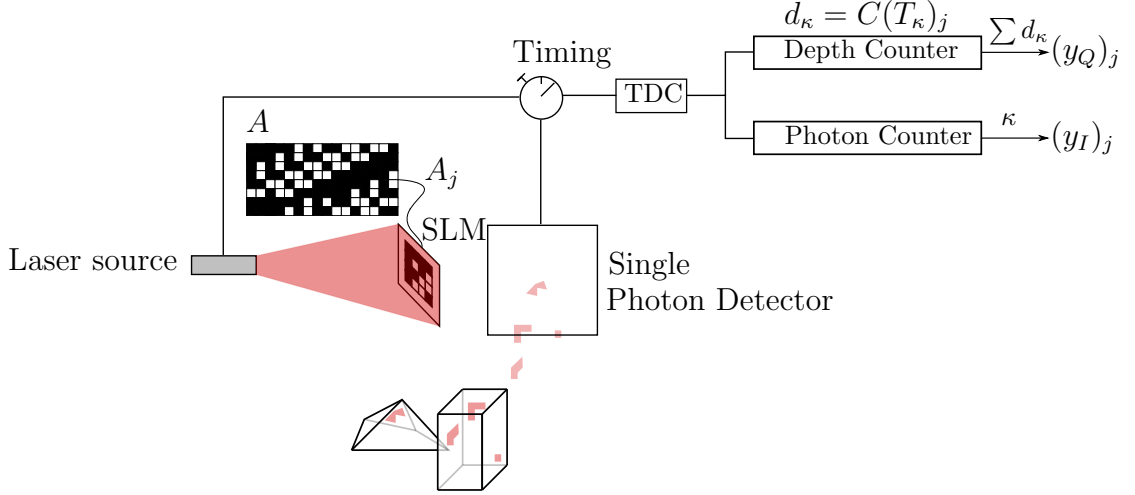
While much more efficient and more compact than full time-gating and complete depth cube recovery. It places severe constraints on the application scenario and cannot be considered a general imaging approach.

In related work, some key assumptions of the masking approach have been used to formulate a more general approach [229, 230, 231]. Here, a proxy, the so called *time-of-flight sum*, is introduced to split the depth recovery into two spatial image recoveries in vector notation  $x_Q = x_I \circ x_D \in \mathbb{R}^n$ , with the associated measurement vector

$$(y_Q)_j = A_j x_Q = C \sum_i (A_j)_i \eta_i T_i, \quad (5.14)$$

where  $C$  is the conversion constant from time to distance, using ToF principles (2.1),  $\eta_i$  is the number of photons reaching pixel  $i$  and  $T_i$  the ToF of each pixel [231]. This measurement definition is neither clear on how  $T_i$  and  $\eta_i$  are formed during sampling nor on how they are linked to the spatial regime, or that an implicit assumption is made that a single pixel in each row of  $A$  only ever records a single depth. This would be in conflict with a key assumption provided that depth is non-

linear meaning that photons with different ToF contribute to the same measurement. This work interprets, as the author specifies a time-correlated single photon counting (TCSPC) sampling device, that each photon event  $\kappa$  yields a time-stamp  $T_\kappa$  and that  $\kappa$  observations form an aggregated measurement for an active pattern as illustrated in Figure 5.5.



**Figure 5.5:** Illustration of the singl-pixel LiDAR system presented in [231] and the measurement acquisition for a pattern cycle using Hadamard patterns. Final returns are depicted in time. The system captures two cumulative measures associated with structured basis illumination patterns, the photon count,  $y_I$ , and the time-of-flight sum  $y_Q$ , which can be reconstructed using CS principles and used to compute depth.

Mathematically,

$$(y_Q)_j = A_j x_Q = C \sum_i ((A_j)_i \sum_{\kappa} (\eta_i T_i)_\kappa) \quad (5.15)$$

$$(y_I)_j = A_j x_I = \sum_i ((A_j)_i \sum_{\kappa} (\eta_i)_\kappa), \quad (5.16)$$

where for the  $j^{\text{th}}$  emission pattern and the  $i^{\text{th}}$  pixel of the emission pattern with  $(\eta_i)_\kappa \in \{0, 1\}$  indicates if pixel  $i$  has observed a pixel during observation  $\kappa$ .

This particular approach makes more implicit assumptions. The main recovery focuses on  $x_Q$  assuming small TV across the ToF-sum as indicated by the use of TVAL3 [232], a TV minimisation framework. The estimate for  $\bar{x}_Q$  is basis transformed and wavelet hard thresholded [233]. This effectively forces a large portion of the scene to zero, which limits the recovery to few surfaces and makes the implicit assumption of large continuous uniform Lambertian surfaces. Finally, a least-square recovery with a debiasing step is performed on the non-zero components of  $\bar{x}_Q$  using gradient projection for sparse reconstruction (GPSR) [234] to recover both  $x_I$  and  $x_Q$ , with the final depth recovery being an element-wise division  $x_D = x_Q./x_I$ .

It is quite obvious that this is a very costly recovery with many steps making real-time reconstruction at high frame rates infeasible. It constrains the problem to a few surfaces and also limits the recovery to significant foreground objects. Their approach is also very sensitive to background noise and their proposed mitigations are not very effective in the described sampling scheme, as measurements are aggregated immediately. As the author correctly identifies, many photons with different ToF contribute to the final measurement. However, they seem to ignore that background photons also arrive throughout the timing cycle, i.e. also have a particular time measurement attached, despite not coming from any surface in the scene.

Further, the acquisition times are limited by the SLM and further by the problem size, a limitation of all single-pixel systems with a hard limit defined by the desired final image size.

This work addresses these shortcomings by expanding the concept of the distance sum and formulates it more rigorously in Section 5.3.1. The concept of two image reconstructions to recover depth is intriguing for its simplicity. However, for practical applications with a wide range of scenarios across a high dynamic range with a long maximum operating range ( $> 100\text{ m}$ ) the hard thresholding step constraining the scene to few foreground objects makes their framework impracticable for the considered type of application. A more robust noise removal scheme is also required to deal with much higher background rates introduced by outdoor imaging, than envisioned by [231]. Further, the acquisition time has to be significantly reduced without limiting resolution and the processing time needs to be shortened by several orders of magnitude. This thesis therefore investigates block-independent sparsity regularisation for depth signals.

## 5.3 Small Scale Sparse Depth Sensing

The prior art described above has focused on the application of the single pixel camera methodology [189] and its use for depth recovery. However, most of the previous work assumes short range applications with high density patterns. Further, the direct depth estimation techniques often rely on solving depth for every single time step in the acquired pattern histogram. While this time-gating approach is a common technique to estimate LiDAR cubes, it is computationally not very efficient. Another approach requires heavy constraints due to the chosen basis and projection function and is also severely range limited, with long exposure sequences.

To address these shortcomings, a framework is presented in this work, which makes explicit use of a photon detector array. The nature of the array and available

resources allows for design flexibility of the system as well as the sampling and subsequent processing methodologies. In the simplest case, an array of photon detector elements can indeed be interpreted as a collection of single pixels, where the intrinsic resolution of the sensor is encoded in the emitter device.

This retains the advantage of smaller more efficient sensor arrays which can image at a higher resolution compared to linearly sampling them without a compressive illumination scheme. However, there is significant scope to allow for a detector to be resolution matched to the emitter, which can aggregate more information in a single measurement utilising a Super-Pixel [82]. The developed framework in this work addresses key issues identified above to enable higher frame-rates with low illumination densities for eye-safe long range imaging, while maintaining the key advantage of CS of low bandwidth sampling over traditional linear sampling. By introducing parallelism into CS depth imaging by utilising independent block compressive sensing, a particular focus is placed upon pattern size. The framework described herein proposes a pattern size significantly smaller than described in literature for most single pixel systems and further maintains a massively parallel processing approach to recover depth.

### 5.3.1 Observational Model

#### Signal Model

In most state-of-the-art solid-state LiDAR systems currently available, a laser array is scanned across line-wise in a sequential fashion. As discussed in Chapter 2, in most cases a full array illumination not only poses significant safety risk, but also introduces a significant energy demand.

In this work, the goal is to recover a vectorised depth map,  $x_D \in \mathbb{R}^n$  of  $X_D \in \mathbb{R}^{N_x \times N_y}$ , from a photon sensitive solid-state detector array. By applying CS principles depth can be estimated from a collection of  $m$  ToF measurements of a scene environment by means of sparse illumination and sampling. The framework can work in modes, firstly the fully compressive case, where  $m < n$ , and secondly using discrete sparse oversampling ( $d$ Sparse), where  $m > n$ . Both cases employ the same sparse sampling scheme, where each measurement is obtained from probing the environment with a sparse structured binary illumination scheme. This can reduce the average laser power density emitted to enhance eye-safety while also enabling a flexible laser power budget allocation to increase range.

**Assumption 1** *For a high resolution solid-state ToF imaging array, the field-of-view (FoV) of a single pixel is very narrow to avoid overlap between pixels and*

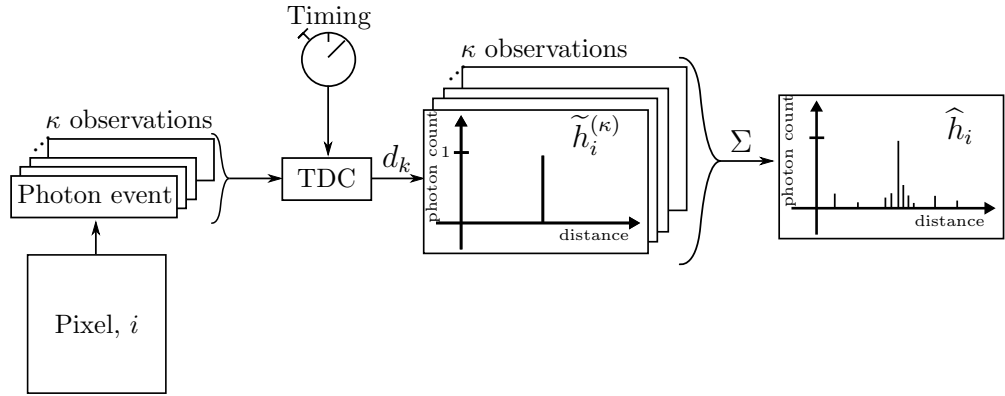
*captures in most cases only a single significant surface return.*

In the ideal noiseless case the histogram of a pixel  $i$ ,  $h_i$ , has a single surface return normalised to 1 with all other bins containing no counts and thus depth is simply

$$(x_D)_i = h_i \circ d, \quad (5.17)$$

where  $\circ$  is an element-wise multiplication and  $d \in \mathbb{R}^p$  encoding the distance value for each of the  $p$  bins.

However, a direct Time-of-Flight (dToF) system captures a histogram with a counting process which can be described as a Poisson random process [91, 31, 82],  $\mathcal{P}(c)$ , where  $c$  is the count rate distribution and thus the mean and variance as in equation (2.22). For every count a histogram  $(\tilde{h}_i)^\kappa$  is observed. Several  $\kappa$  Poisson observations are made i.e. time stamps from photon count events which form a histogram estimate  $\hat{h}_i$  shown in Figure 5.6.



**Figure 5.6:** Histogram acquisition for an  $i^{\text{th}}$  pixel, assuming one count per illumination cycle resulting in  $\kappa \leq \kappa_{\max}$  observations yielding a histogram estimate,  $\hat{h}_i$

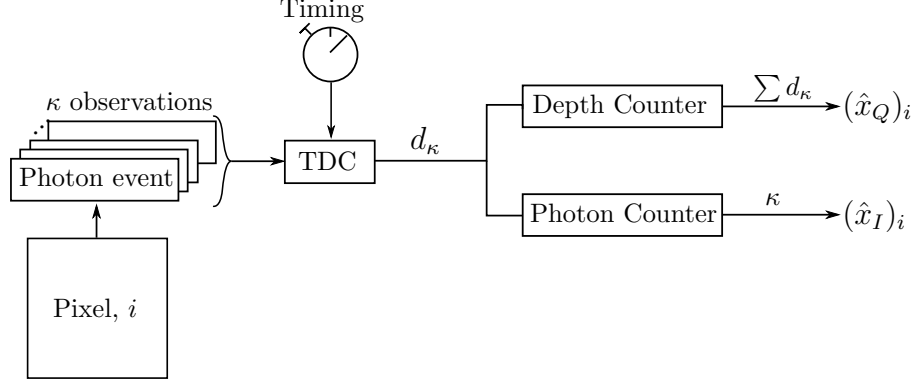
Now let the estimate for depth be

$$(\hat{x}_D)_i = \frac{\sum_{j=1}^p (\hat{h}_i(j) \cdot d(j))}{\sum_{j=1}^p (\hat{h}_i(j))} =: \frac{(\hat{x}_Q)_i}{(\hat{x}_I)_i} \quad (5.18)$$

where  $(\hat{x}_Q)_i$  is the depth sum (or ToF-sum [231]) and  $(\hat{x}_I)_i$  is the photon event sum for pixel  $i$  illustrated in Figure 5.7 for a sampling approach foregoing the histogram.

Although this works retains a single histogram to derive the compressive measurements, the two quantities can also be sampled as above. The total compression with





**Figure 5.7:** Measurement acquisition for an  $i^{\text{th}}$  pixel yielding the observed values for  $x_Q$  and  $x_I$ .

a single histogram versus full histogram sampling is therefore

$$\mathcal{C}_{\text{total}} = \frac{2m + mn + p}{np} = \mathcal{C} \frac{2 + n + \frac{p}{m}}{p}, \quad (5.19)$$

where,  $m$  is the number of measurements,  $n$  the number of pixels in the final reconstruction (greatest number of either the emitter or detector),  $mn$  is the size of the sensing matrix,  $p$  is the size of a histogram and  $\mathcal{C} = m/n$  is the spatial compression.

For the case shown in Figure 5.7,  $\mathcal{C}_{\text{total}} = \frac{2m+mn}{np}$ . These significant savings apply throughout this work, but for consistency the spatial compression  $\mathcal{C}$  is primarily quoted since histogram sizes can vary for different operating conditions and across approaches, while the spatial compression  $\mathcal{C}$  is comparable across CS approaches. The total compression versus full histogram processing improves dramatically if  $p$  is large as is the case for this work.

Following [231], this work relies on the following assumption to estimate these quantities individually.

**Assumption 2** Let  $x_Q \in \mathbb{R}^n$  be the image vector of total distance travelled by all collected photons at pixel  $i$  for a single most significant surface return and let  $x_I \in \mathbb{N}^n$  be the intensity image vector whose  $i^{\text{th}}$  entry is the associated photon count at pixel  $i$ . It is assumed that  $x_Q$  and  $x_I$  have sparse properties such as small TV or can be represented in a sparse basis [189, 231].

Therefore, once  $(\hat{x}_Q)_i$  and  $(\hat{x}_I)_i$  are estimated for each pixel  $i$ , the depth is obtained directly [231] as

$$\hat{x}_D = \frac{\hat{x}_Q}{\hat{x}_I}. \quad (5.20)$$

### System Model

In this work the focus is on depth estimation in small blocks with a measurement obtained by means of a sequence of patterns (a set of active pixels). The proposed sparse LiDAR system aggregates this structured information from a constrained set of pixels in a block defined by a sparse pattern in a pattern histogram,  $h_j^* \in \mathbb{R}^p$ .

A pseudo-random binary pattern,  $\phi$ , defines the subset of active pixels per measurement. This binary pattern is generated as outlined in Algorithm 3.

---

**Algorithm 3** Pseudo-random binary projection pattern generation

---

**Input**  $s, m, n$   
**Output**  $\phi$   
**for** every pattern  $j$  **do**  
     $\phi_j = \text{zeros}(n)$   
     $\text{Ind} = []$   
    **while**  $\text{length}(\text{Ind}) < s$  **do**  
         $\text{pixel} = \text{random}(1, n)$   
        **if**  $\text{pixel}$  not in  $\text{Ind}$  **then**  
            Add  $\text{pixel}$  to  $\text{Ind}$   
        **end if**  
    **end while**  
     $\phi_j(\text{Ind}) = 1$ ;  
**end for**

---

Just as with intensity imaging, where the total photon count is spatially related to the pattern stimulation, so is the total ToF recorded during a pattern exposure. However, because the ToF becomes non-linear i.e. a single measurement value contains multiple depths due to accumulation of multiple events from potentially different surfaces for the same time measurement, the scaling factor has to be retrieved to recover the true ToF at a particular spatial point mapped to the pattern projection to retrieve the correct spatial location [231].

Conveniently, this is the photon count image,  $x_I$ . For ease of illustration and to enable noise removal, measurements in this work solely derived from histograms.

Therefore,  $s$  pixel histograms are collected with an associated pattern sequence  $\phi_j$  to form

$$h_j^* = \sum_{i \in \phi_j} (\hat{h}_i). \quad (5.21)$$

The two proxy quantities,  $y_Q$  for the depth sum  $x_Q$  and  $y_I$  the photon count  $x_I$ , first defined in [231], are derived from each pattern histogram. Each quantity is recovered by solving a BPDN problem which exploit that natural images or a subset thereof have sparse representations or sparse properties. With the solution for both

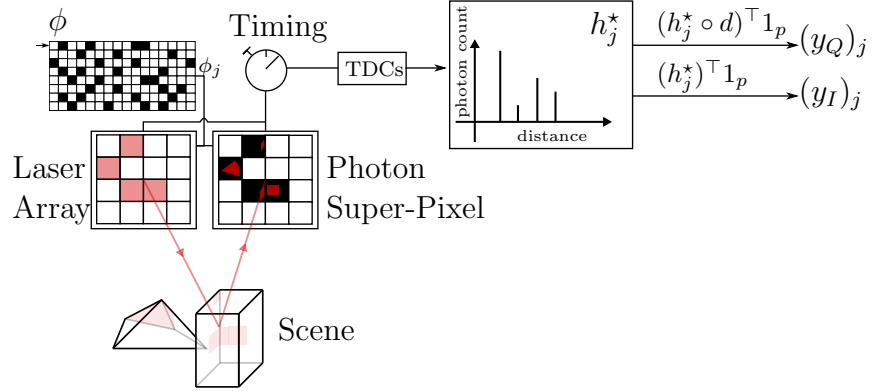
proxy quantities, depth can be retrieved by de-coupling ToF from the photon counts as defined in equation (5.20).

Every pattern  $\phi_j$  specifies the pixels contributing to the  $j^{\text{th}}$  compressive measurement of the depth sum and the photon count as

$$(y_Q)_j = \sum_{i \in \phi_j} (x_Q)_i + (\beta_Q)_i = \sum_{k=1}^p (h_j^*(k) \cdot d(k)) \quad (5.22)$$

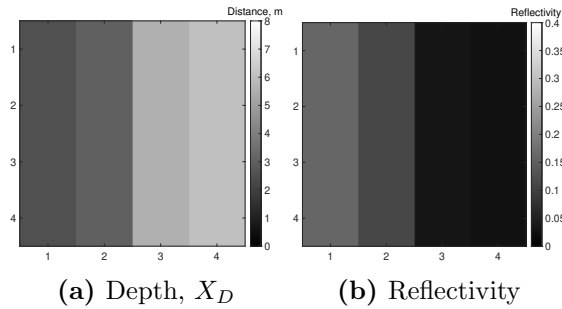
$$(y_I)_j = \sum_{i \in \phi_j} (x_I)_i + (\beta_I)_i = \sum_{k=1}^p (h_j^*(k)), \quad (5.23)$$

where  $(\beta_Q)_i$ ,  $(\beta_I)_i$  is Poisson noise with a mean and variance relative to the background count mean,  $\beta$ , with minor changes to the model presented in [82]. A system illustration is shown in Figure 5.8.



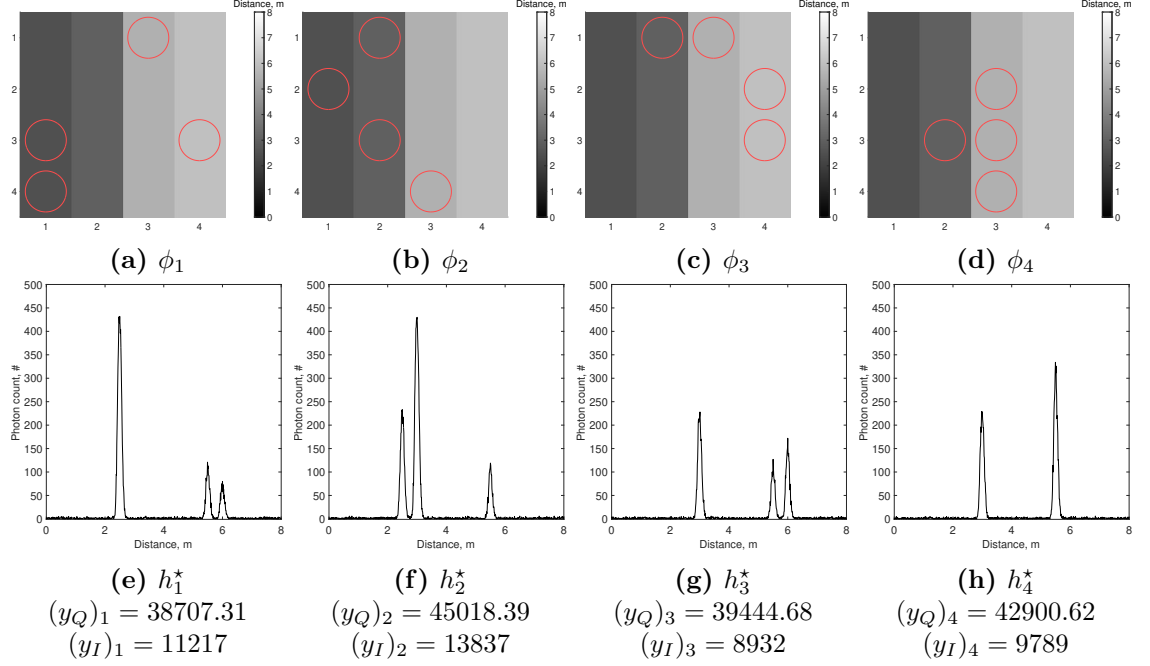
**Figure 5.8:** Compressive Super-Pixel LiDAR [82] sampling behaviour, where  $1_p$  is a vector of ones with length  $p$ . A super-pixel denotes a photon detector array, which can sample individual pixels in a structured manner.

To illustrate this measurement acquisition, an example for a single block scene is shown with four distinct surfaces in Figure 5.9.



**Figure 5.9:** Ground truth for block compressive depth sensing (a) with respective reflectivity (b).

This scene is compressively sampled with a target sparsity of  $s = 4$  or  $\Psi = 0.25$  and a spatial sampling compression rate of  $\mathcal{C} = 0.75$ . This means that there are 4 active pixels for a total of  $m = 12 < n = 16$  measurement patterns stored in  $\phi$  for a  $4 \times 4$  photon detector array. For this particular illustration, only the first 4 measurements are shown in Figure 5.10.



**Figure 5.10:** First four sample steps in a sequence of  $m = 12$  with  $s = 4$  for a  $n = 16$  ( $4 \times 4$ ) block. Illumination pattern  $\phi_i$  are overlaid atop the depth ground truth (a)-(d) and corresponding pattern response histograms are shown in (e)-(h).

In this example the background rate is very low compared to the signal rate, however this is often not the case. While for such minimal ambient noise, the regularisation in a sparse basis removes noise efficiently, when noise levels approach the signal level, the background mean count rate should be estimated as  $\hat{\beta}$ . Two practical approaches are proposed.

First, an additional detector area with the  $s$  pixels which are not in line with any emitter radiation, aggregates a background noise histogram,  $h_n$ , without active illumination for the same number of realisations as an instance of  $h^*$ . The active background compensation can then be estimated as

$$\hat{\beta}_{\text{active}} = \max(h_n) + \eta, \quad (5.24)$$

where  $\eta$  is an offset parameter.

Second, to estimate the background mean practically without additional hardware, a small section in  $h^*$  could be allocated where the emitter is idle, for example an

additional  $l_n$  bins beyond the operating range.

The passive background compensation is then

$$\hat{\beta}_{\text{passive}} = \max(h^*[p - l_n : p]) + \eta. \quad (5.25)$$

The noise compensation scheme is applied before measurements  $(y_Q)_j$  and  $(y_I)_j$  are stored and the noise compensated histogram for pattern  $j$  is  $(h_j^*)' = \max((h_j^* - \hat{\beta}), 0)$  which replaces  $h_j^*$  in equations (5.22) and (5.23).

While each individual pixel probes a single distance derived from the ToF of the reflected photon, a measurement is made up of up  $s$  contributions, which are not necessarily from the same surface, but have similar background rates. In particular the distance sum measurements reflect the pattern variation, while intensity can remain approximately constant for small FoV detectors.

**Assumption 3** *For the field-of-view of a small scale or block in a large scale photon array, the likelihood of active pixels probing the same surfaces is high and therefore the signal-to-noise ratio (SNR) across measurements increases.*

To reconstruct the depth map,  $\hat{x}_D$  by equation (5.20) from the measurements, equations (5.22)-(5.23) can be vectorised for all  $m$  patterns in  $\phi$  to estimate  $\hat{x}_Q$  and  $\hat{x}_I$  by re-formulating each problem in the form of equation (5.6) [82] such that

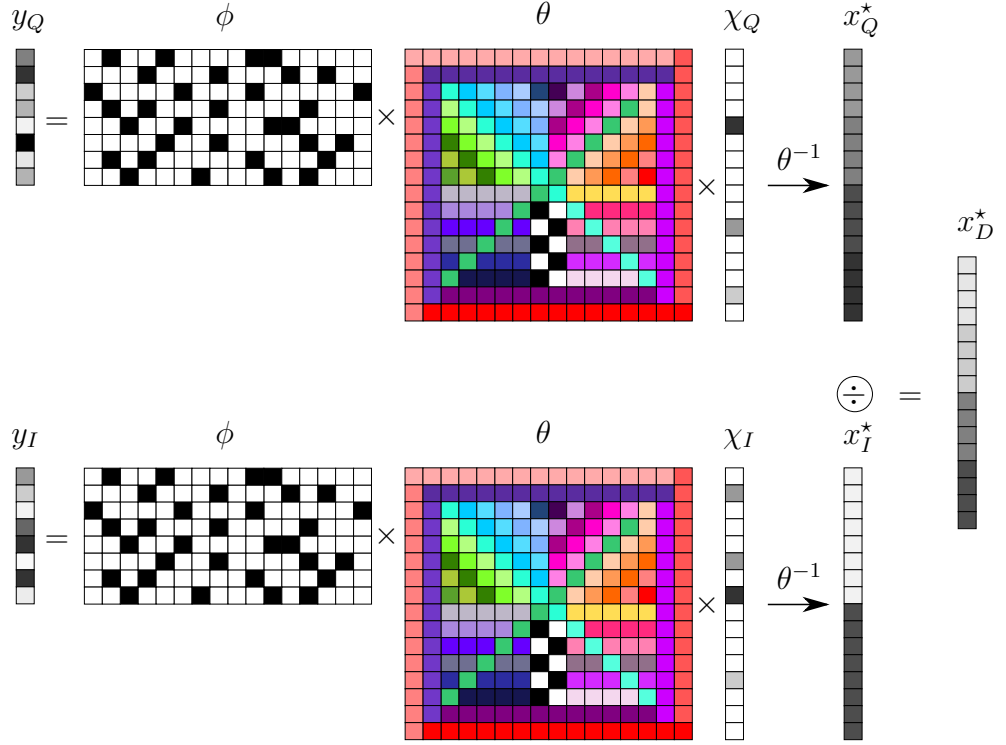
$$\min_{x_Q} \frac{1}{2} \|\phi x_Q - y_Q\|_2^2 + \alpha_Q \|\theta x_Q\|_1, \quad (5.26)$$

$$\min_{x_I} \frac{1}{2} \|\phi x_I - y_I\|_2^2 + \alpha_I \|\theta x_I\|_1, \quad (5.27)$$

where  $\alpha_Q, \alpha_I$  are weights to the  $\ell_1$ -regularisation terms with a linear transform  $\theta$ .

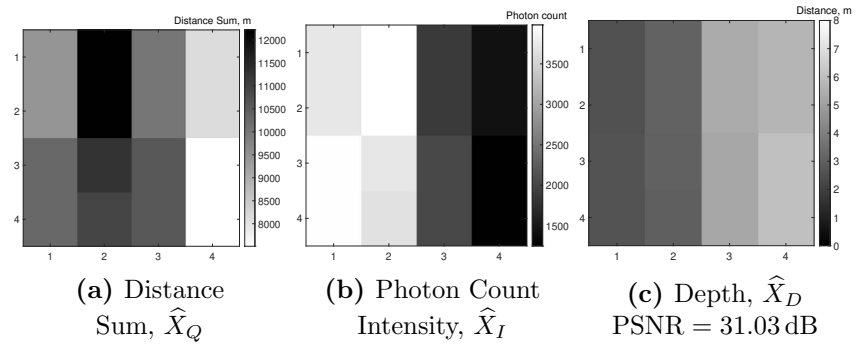
This is visualised in Figure 5.11. If the signal can be sparsely represented in another basis,  $\theta$  is a linear basis transform, e.g. a wavelet transform. For signals with sparse gradients, this can be an isotropic difference transform to minimise TV [75, 235, 231].

In this work, the linear transform is applied to a binary random projection pattern and as such is interchangeable in the optimisation stage. This means that the choice of basis is modular in the sense that if better basis functions are found in the future, they could be readily placed into the framework without any changes to the sampling scheme.



**Figure 5.11:** Compressive small scale depth sensing visualisation, the two proxy quantities for depth are randomly sampled randomly to obtain compressive measurements of their sparse basis, which are recovered as dense data of the photon count,  $x_I^*$  and the total distance  $x_Q^*$ . Note:  $\theta$  is purely illustrative and not a real transform (such as DCT), while  $\phi$  is representative of a real projection pattern matrix.

The reconstruction result for the sample block in Figure 5.10 uses  $\theta = \text{DWT}$  (Daubechies wavelet transform [196, 236]) with alternating direction method of multipliers (ADMM) [237] to solve equations (5.26)-(5.27) for both the proxy quantities without background noise compensation with the final depth map shown in Figure 5.12.



**Figure 5.12:** Single block reconstruction for compressive depth sensing. (a) Distance sum, (b) Photon count intensity and (c) depth using Daubechies Wavelets ( $l = 2$ ) for  $m = 12$  with  $s = 4$  for a  $n = 16$  ( $4 \times 4$ ).

### Discrete Sparse Oversampling

Considering very small block sizes opens up an interesting practical simplification of the reconstruction of both proxy quantities. The pattern matrix  $\phi$  is of very small  $m \times n$  size. For  $2 \leq s \leq n$  active pixels, the resulting system of equations (5.22)-(5.23) is under-determined like any other CS problem. The special case with target sparsity of  $s = 1$  would be a traditional sampling approach i.e.  $m = n$  measurements and loses the aggregation aspect of a CS measurement, where several pixels are combined, which reduces the number of realisations required to obtain a good estimate of depth. With modest  $m > n$  and  $s < n$  in this small scale problem, the size of  $\phi$  remains small enough for practical applications.

The compact measurement scheme reduces memory requirements as full histograms do not need to be stored for every pixel but can be acquired for ensembles of  $s$  pixels as with the compressive case (see equation (5.21)). The problem can be designed with  $s$  related to  $m$  in such a way that equations (5.22)-(5.23) are fully determined. This is in some way a sparse over-sampling approach but it retains all other reductions of the CS sensing approach such as lower emission power for each sparse measurement and compact measurement storage. To enforce a deterministic solution space, it is enforced that all pixels are active at least once across  $\phi$  making it possible to compute a pseudo-inverse. In other words, it is assumed that  $\phi$  is full column-rank [193].

The linear system of equations can then be readily solved with an approximated least square solution [238]. Let  $A^+ = (\phi^T \phi)^{-1} \phi^T \in \mathbb{R}^{n \times m}$  be the standard  $\ell_2$  pseudo-inverse, the problem simplifies to

$$\hat{x}_Q = (A^+) y_Q , \quad (5.28)$$

$$\hat{x}_I = (A^+) y_I \quad (5.29)$$

and depth is simply recovered via equation (5.20). Noise compensation can be applied in the same way as in the compressive case. While this approach in of itself is quite obvious, it becomes a viable alternative to linear sampling due to the small problem size considered in this work for depth sensing. This discrete depth recovery using sparse illumination was presented in [239] and is called *dSparse*. The exposure patterns are still  $s \ll n$  and thus require much less average radiation output power versus linear sampling schemes.

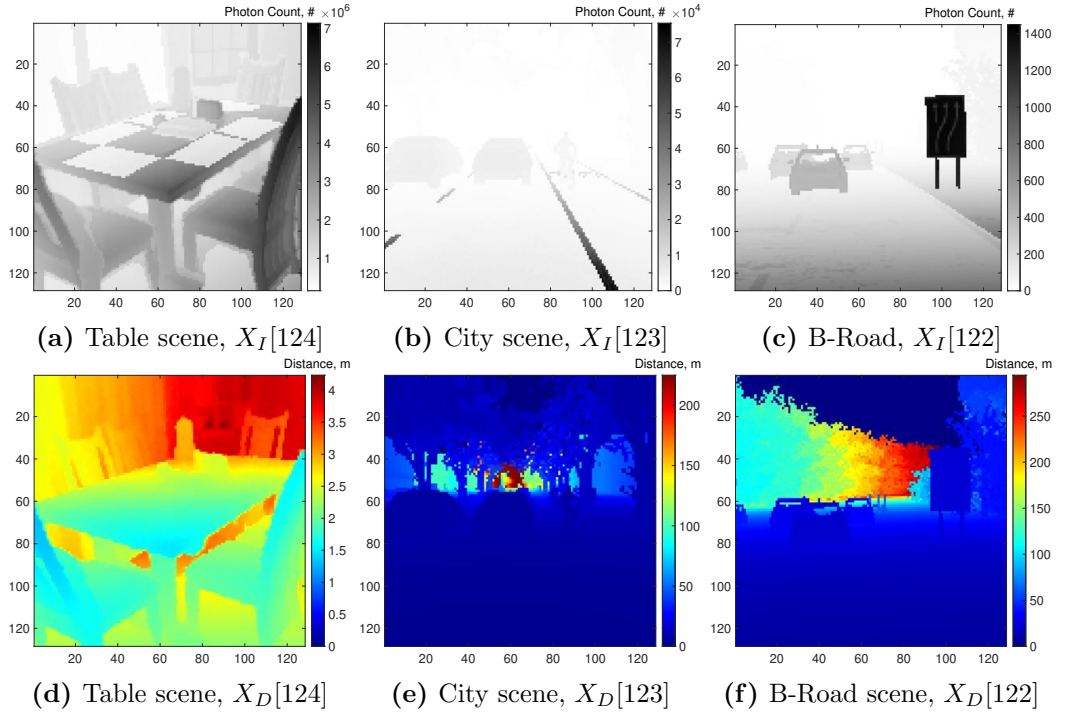
Although bandwidth savings made by the compressive depth sensing approach are obvious i.e. few  $m < n$  very compact measurements and significant savings due to foregoing histogram storage per pixel, there are benefits to this discrete methodology,

if the sampling budget permits this approach. Despite the sampling time being longer than in the compressive case, it is still shorter than a scanning LiDAR system, as the exposure patterns are still  $s$ -sparse pattern flash emissions. A linear scanning solid-state LiDAR requires  $p > N_x$  or  $p > N_y$  depending on line scan direction, where  $p$  is the number of illumination cycles.  $d$ Sparse performs a faster measurement cycle if  $\{N_x \vee N_y\} > m$ . Further, this approach retains the total compression afforded by not requiring full histogram storage on a per pixel basis.

### 5.3.2 Sparsity of Small Scale Signals

To analyse small scale sparsity for natural depth signals, the following three scenes shown in Figure 5.13 are considered. The total image size is  $128 \times 128$ , which is divided into various block sizes.

It is of particular interest how block sizes below  $32 \times 32$  perform, as the majority of literature chooses this block size. Further, the assumption that  $X_Q$  is less sparse than  $X_I$  for natural scenes in [231] may be justified for very simple scenes, but a more detailed analysis is carried out within this work's methodology and reconstruction framework, where there are no constraint on the number of surfaces in the scene unlike previous work.



**Figure 5.13:** Three different scenes - short (a,d), medium (b,e) and long range (c,f) - for sparsity analysis of depth proxies  $X_I$  and  $X_Q = X_D \circ X_I$ .

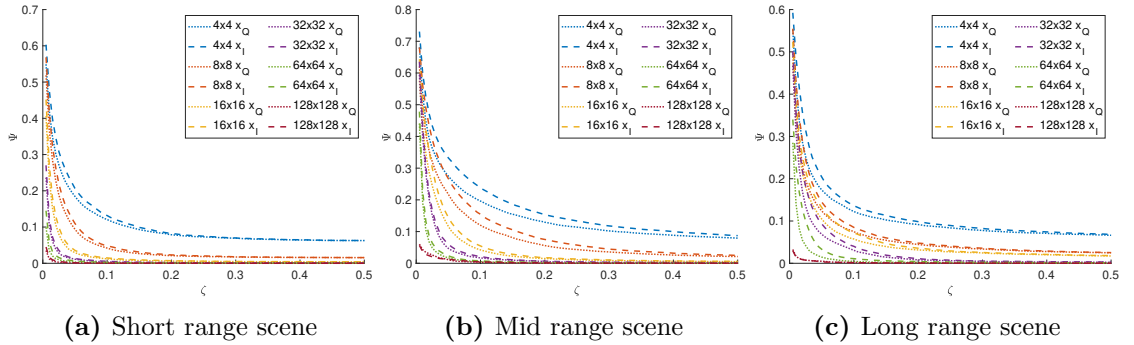


## Sparsity

To find out if the depth sum  $X_Q$  is more or less sparse than  $X_I$ , scenes in Figure 5.13 are used. As outlined in Chapter 3, an estimate for reflectivity is used to generate a photon count image using the photon count model from [78] and an ideal depth image,  $X_D$  is known from the datasets, then the depth sum is simply  $X_Q = X_I \circ X_D$ . For linear transforms, Daubechies wavelets ( $D(2)$ ), Haar wavelets, DCT, fast Fourier transform (FFT) magnitude and finally TV are considered. Both  $X_Q$  and  $X_I$  are transformed and hard-thresholded [233] to zero if smaller than the threshold value,  $\tau$ , derived from a relative threshold  $\zeta$ , such that

$$\tau = \zeta \max(X). \quad (5.30)$$

The relative threshold is swept across a range of 0.005 to 0.995 to determine the sparsity,  $\Psi$  of each signal with respect to the maximum values in each signal. This ensures that the energy content is accurately considered and is equivalent to retaining  $\zeta n$  components of an ordered signal  $X \in \mathbb{R}^{n=N_x \times N_y}$ . An illustrative selection for each scene type is provided in Tables 5.1-5.3 and for each scene the results for the DCT basis are shown in Figure 5.14 with sparsity as defined in equation (5.2).



**Figure 5.14:** Three different scenes - short, medium and long range - sparsity  $\Psi$  of depth proxies,  $X_Q$  and  $X_I$  in DCT basis for varying relative energy threshold  $\zeta$

It is obvious from these results that in most basis transforms  $X_Q$  is actually sparser than  $X_I$  or  $\Psi_Q < \Psi_I$  as block size decreases. Sparsity as a whole decreases as the block size shrinks i.e. more non-zero components are retained and therefore  $\Psi$  increases. It is also good to see that sparsity behaves very similar across scene types.

It is fortunate, however, that the sparsity of both signals are very well aligned and follows the same trends for a particular block size. This indicates that it is possible to apply many of the sparsity assumptions which have been validated for natural intensity images and by extension photon count images to the depth sum image.

**Table 5.1:** Sparsity (lower is better) in compressive depth sensing for intensity and depth sum,  $X_I, X_Q \in \mathbb{R}^{128 \times 128}$  for short range

| $\zeta$ |            | Sparsity, $\Psi$ (s/n ratio at relative threshold, $\zeta$ ) |          |              |          |                |          |                |          |                |          |                  |          |
|---------|------------|--|----------|--------------|----------|----------------|----------|----------------|----------|----------------|----------|------------------|----------|
|         |            | $4 \times 4$   |          | $8 \times 8$ |          | $16 \times 16$ |          | $32 \times 32$ |          | $64 \times 64$ |          | $128 \times 128$ |          |
|         |            | $\Psi_Q$   | $\Psi_I$ | $\Psi_Q$     | $\Psi_I$ | $\Psi_Q$       | $\Psi_I$ | $\Psi_Q$       | $\Psi_I$ | $\Psi_Q$       | $\Psi_I$ | $\Psi_Q$         | $\Psi_I$ |
| 0.01    | Daubechies | 0.31   | 0.36     | 0.24         | 0.29     | 0.19           | 0.21     | 0.14           | 0.16     | 0.08           | 0.10     | 0.01             | 0.02     |
|         | Haar       | 1.00   | 1.00     | 0.98         | 0.98     | 0.98           | 0.98     | 0.97           | 0.97     | 0.97           | 0.97     | 0.55             | 0.54     |
|         | DCT        | 0.42   | 0.48     | 0.37         | 0.42     | 0.23           | 0.27     | 0.10           | 0.12     | 0.04           | 0.06     | 0.30             | 0.26     |
|         | FFT        | 0.46   | 0.52     | 0.43         | 0.49     | 0.28           | 0.31     | 0.12           | 0.14     | 0.04           | 0.06     | 0.34             | 0.30     |
|         | TV         | 0.91   | 0.92     | 0.79         | 0.82     | 0.65           | 0.70     | 0.53           | 0.58     | 0.44           | 0.46     | 0.41             | 0.35     |
| 0.05    | Daubechies | 0.16   | 0.18     | 0.10         | 0.12     | 0.05           | 0.06     | 0.01           | 0.02     | 0.00           | 0.01     | 0.00             | 0.00     |
|         | Haar       | 0.99   | 0.98     | 0.90         | 0.90     | 0.89           | 0.89     | 0.88           | 0.88     | 0.87           | 0.87     | 0.29             | 0.26     |
|         | DCT        | 0.18   | 0.21     | 0.09         | 0.11     | 0.03           | 0.04     | 0.01           | 0.01     | 0.00           | 0.01     | 0.05             | 0.06     |
|         | FFT        | 0.21   | 0.24     | 0.10         | 0.12     | 0.03           | 0.04     | 0.01           | 0.01     | 0.00           | 0.01     | 0.06             | 0.06     |
|         | TV         | 0.73   | 0.75     | 0.50         | 0.53     | 0.31           | 0.33     | 0.21           | 0.22     | 0.16           | 0.16     | 0.15             | 0.12     |
| 0.10    | Daubechies | 0.12   | 0.13     | 0.05         | 0.06     | 0.01           | 0.02     | 0.00           | 0.00     | 0.00           | 0.00     | 0.00             | 0.00     |
|         | Haar       | 0.97   | 0.96     | 0.82         | 0.82     | 0.77           | 0.77     | 0.75           | 0.75     | 0.74           | 0.74     | 0.22             | 0.18     |
|         | DCT        | 0.12   | 0.13     | 0.04         | 0.05     | 0.01           | 0.01     | 0.00           | 0.01     | 0.00           | 0.00     | 0.02             | 0.02     |
|         | FFT        | 0.13   | 0.14     | 0.04         | 0.05     | 0.01           | 0.02     | 0.00           | 0.01     | 0.00           | 0.00     | 0.02             | 0.03     |
|         | TV         | 0.63   | 0.65     | 0.38         | 0.40     | 0.21           | 0.22     | 0.15           | 0.15     | 0.12           | 0.11     | 0.11             | 0.08     |

**Table 5.2:** Sparsity (lower is better) in compressive depth sensing for intensity and depth sum,  $X_I, X_Q \in \mathbb{R}^{128 \times 128}$  for mid range

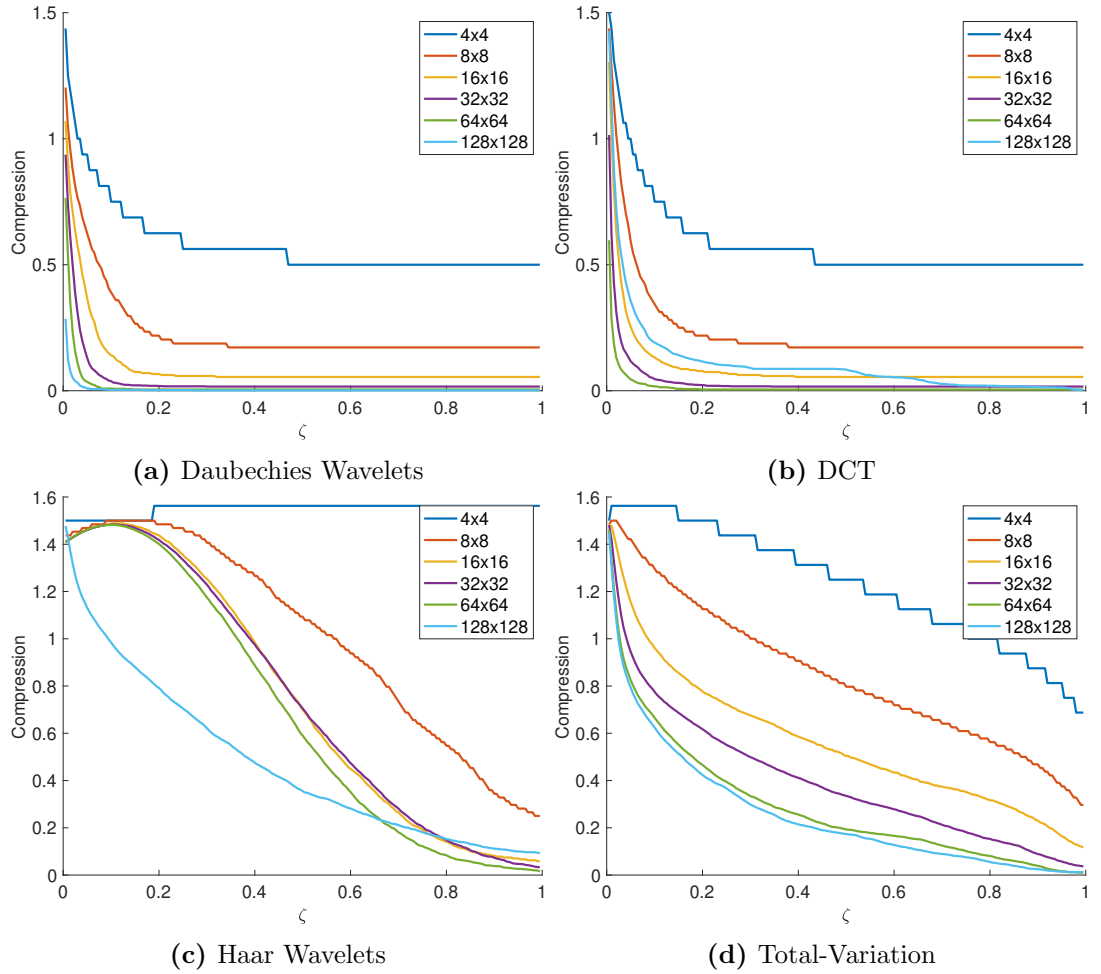
| $\zeta$ |            | Sparsity, $\Psi$ (s/n ratio at relative threshold, $\zeta$ ) |          |              |          |                |          |                |          |                |          |                  |          |
|---------|------------|--|----------|--------------|----------|----------------|----------|----------------|----------|----------------|----------|------------------|----------|
|         |            | $4 \times 4$   |          | $8 \times 8$ |          | $16 \times 16$ |          | $32 \times 32$ |          | $64 \times 64$ |          | $128 \times 128$ |          |
|         |            | $\Psi_Q$   | $\Psi_I$ | $\Psi_Q$     | $\Psi_I$ | $\Psi_Q$       | $\Psi_I$ | $\Psi_Q$       | $\Psi_I$ | $\Psi_Q$       | $\Psi_I$ | $\Psi_Q$         | $\Psi_I$ |
| 0.01    | Daubechies | 0.42   | 0.50     | 0.30         | 0.37     | 0.23           | 0.29     | 0.16           | 0.19     | 0.11           | 0.12     | 0.04             | 0.04     |
|         | Haar       | 0.98   | 0.98     | 0.98         | 0.98     | 0.98           | 0.98     | 0.98           | 0.98     | 0.97           | 0.97     | 0.19             | 0.15     |
|         | DCT        | 0.53   | 0.60     | 0.48         | 0.56     | 0.43           | 0.51     | 0.40           | 0.43     | 0.19           | 0.23     | 0.65             | 0.48     |
|         | FFT        | 0.55   | 0.60     | 0.53         | 0.59     | 0.48           | 0.57     | 0.49           | 0.52     | 0.20           | 0.24     | 0.74             | 0.65     |
|         | TV         | 0.86   | 0.85     | 0.75         | 0.74     | 0.62           | 0.60     | 0.37           | 0.35     | 0.20           | 0.20     | 0.09             | 0.05     |
| 0.05    | Daubechies | 0.20   | 0.23     | 0.14         | 0.17     | 0.09           | 0.11     | 0.06           | 0.07     | 0.03           | 0.03     | 0.01             | 0.01     |
|         | Haar       | 0.97   | 0.96     | 0.90         | 0.91     | 0.89           | 0.89     | 0.88           | 0.87     | 0.86           | 0.85     | 0.09             | 0.07     |
|         | DCT        | 0.27   | 0.33     | 0.22         | 0.27     | 0.12           | 0.14     | 0.05           | 0.06     | 0.02           | 0.02     | 0.21             | 0.14     |
|         | FFT        | 0.31   | 0.37     | 0.25         | 0.31     | 0.13           | 0.17     | 0.05           | 0.06     | 0.02           | 0.02     | 0.25             | 0.20     |
|         | TV         | 0.72   | 0.71     | 0.57         | 0.56     | 0.42           | 0.41     | 0.20           | 0.19     | 0.09           | 0.08     | 0.03             | 0.02     |
| 0.10    | Daubechies | 0.16   | 0.19     | 0.09         | 0.11     | 0.05           | 0.06     | 0.03           | 0.03     | 0.01           | 0.02     | 0.00             | 0.00     |
|         | Haar       | 0.94   | 0.94     | 0.82         | 0.82     | 0.77           | 0.77     | 0.75           | 0.75     | 0.72           | 0.71     | 0.06             | 0.04     |
|         | DCT        | 0.20   | 0.24     | 0.12         | 0.15     | 0.04           | 0.05     | 0.02           | 0.02     | 0.01           | 0.01     | 0.09             | 0.07     |
|         | FFT        | 0.22   | 0.27     | 0.14         | 0.18     | 0.04           | 0.06     | 0.02           | 0.02     | 0.00           | 0.01     | 0.11             | 0.10     |
|         | TV         | 0.63   | 0.62     | 0.47         | 0.47     | 0.33           | 0.32     | 0.14           | 0.13     | 0.05           | 0.04     | 0.02             | 0.01     |

**Table 5.3:** Sparsity (lower is better) in blocked compressive depth sensing for intensity and depth sum,  $X_I, X_Q \in \mathbb{R}^{128 \times 128}$  for long range

| $\zeta$ |            | Sparsity, $\Psi$ (s/n ratio at relative threshold, $\zeta$ ) |          |              |          |                |          |                |          |                |          |                  |          |
|---------|------------|--|----------|--------------|----------|----------------|----------|----------------|----------|----------------|----------|------------------|----------|
|         |            | $4 \times 4$   |          | $8 \times 8$ |          | $16 \times 16$ |          | $32 \times 32$ |          | $64 \times 64$ |          | $128 \times 128$ |          |
|         |            | $\Psi_Q$   | $\Psi_I$ | $\Psi_Q$     | $\Psi_I$ | $\Psi_Q$       | $\Psi_I$ | $\Psi_Q$       | $\Psi_I$ | $\Psi_Q$       | $\Psi_I$ | $\Psi_Q$         | $\Psi_I$ |
| 0.01    | Daubechies | 0.35   | 0.46     | 0.25         | 0.33     | 0.17           | 0.23     | 0.12           | 0.15     | 0.07           | 0.07     | 0.02             | 0.01     |
|         | Haar       | 0.88   | 0.87     | 0.89         | 0.89     | 0.93           | 0.93     | 0.97           | 0.97     | 0.97           | 0.97     | 0.28             | 0.42     |
|         | DCT        | 0.42   | 0.49     | 0.36         | 0.43     | 0.33           | 0.38     | 0.31           | 0.36     | 0.14           | 0.23     | 0.19             | 0.19     |
|         | FFT        | 0.44   | 0.47     | 0.38         | 0.43     | 0.37           | 0.41     | 0.35           | 0.40     | 0.15           | 0.27     | 0.26             | 0.41     |
|         | TV         | 0.77   | 0.64     | 0.70         | 0.61     | 0.60           | 0.55     | 0.46           | 0.43     | 0.26           | 0.28     | 0.11             | 0.19     |
| 0.05    | Daubechies | 0.13   | 0.16     | 0.08         | 0.10     | 0.06           | 0.07     | 0.04           | 0.05     | 0.02           | 0.03     | 0.00             | 0.00     |
|         | Haar       | 0.86   | 0.85     | 0.82         | 0.82     | 0.85           | 0.85     | 0.87           | 0.86     | 0.86           | 0.85     | 0.09             | 0.14     |
|         | DCT        | 0.17   | 0.20     | 0.12         | 0.14     | 0.10           | 0.13     | 0.07           | 0.09     | 0.01           | 0.03     | 0.04             | 0.05     |
|         | FFT        | 0.19   | 0.22     | 0.14         | 0.16     | 0.11           | 0.15     | 0.08           | 0.11     | 0.02           | 0.03     | 0.05             | 0.08     |
|         | TV         | 0.62   | 0.57     | 0.52         | 0.49     | 0.37           | 0.37     | 0.22           | 0.24     | 0.07           | 0.08     | 0.04             | 0.04     |
| 0.10    | Daubechies | 0.10   | 0.11     | 0.06         | 0.06     | 0.04           | 0.04     | 0.02           | 0.03     | 0.01           | 0.02     | 0.00             | 0.00     |
|         | Haar       | 0.84   | 0.84     | 0.75         | 0.75     | 0.73           | 0.74     | 0.74           | 0.73     | 0.73           | 0.71     | 0.05             | 0.08     |
|         | DCT        | 0.12   | 0.14     | 0.07         | 0.08     | 0.06           | 0.07     | 0.03           | 0.04     | 0.00           | 0.01     | 0.02             | 0.03     |
|         | FFT        | 0.13   | 0.14     | 0.08         | 0.10     | 0.06           | 0.08     | 0.05           | 0.06     | 0.00           | 0.01     | 0.02             | 0.04     |
|         | TV         | 0.54   | 0.49     | 0.43         | 0.42     | 0.28           | 0.30     | 0.14           | 0.18     | 0.04           | 0.04     | 0.02             | 0.02     |

Importantly, as  $X_Q$  is often sparser than  $X_I$  for small block sizes  $N_B < 128$ , the depth information contained in  $X_Q$  should be sufficiently sampled if bounds are set by the recovery of  $X_I$ . Referring to results in Tables 5.2-5.3 it is already quite clear that the Haar wavelet transform is unsuitable for the small scale problem considered in this work. As DCT and the magnitude of a FFT are quite similar, DCT is favoured over FFT. The remaining analysis therefore focuses on Daubechies wavelets, DCT and TV.

To get an idea of what savings can be achieved in bandwidth, the spatial compression,  $\mathcal{C} = m/n$ , of a signal for a given linear transform across the relative threshold,  $\zeta$  is shown in Figure 5.15. The potential savings are linked to the sparsity and the



**Figure 5.15:** Compression potential based on sparsity and measurement limits for various linear transforms.

density of  $A$ . For example, a  $s = 4$ -sparse signal of size  $n = 16$ , has a targeted sparsity of  $\Psi = 0.25$  as defined in equation (5.2). Measurements  $m$  are estimated using equation (5.8) and the following analysis considers the spatial compression potential of  $X_Q$  for the mid-range scene by varying the relative energy threshold  $\zeta$

as before. The respective sparsity  $\Psi$  determines  $s$  and thus  $m$  allowing to study the effects of the basis sparsity on the compression potential.

For Daubechies wavelets and DCT (Figure 5.15(a-b)) a consistent compressibility can be observed. Daubechies wavelets in particular scale very well across various block sizes, which is somewhat expected as it is a common choice for forward compression schemes such as JPEG2000. DCT performs very similar to Daubechies wavelets in particular for block sizes  $N_B < 128$ , but performs worse as block sizes increase.

Haar wavelets are considered in Figure 5.15(c), as they have been shown to be linked to Hadamard-Walsh matrices [193] in CS imaging. While they do approach similar compressibility for larger images, they are not suitable for block based CS at the very smallest size of  $4 \times 4$ , and while this transform does work as block size increases, it is vastly outperformed by Daubechies wavelets.

Finally, TV is considered in terms of compressibility in Figure 5.15(d). It follows the same trends as the other basis functions, meaning that as signal size increases, the potential for compression increases. In the case of the smallest considered block size of  $4 \times 4$ , the compressibility does not reduce quite as drastically as for larger scale signals. It should be noted that compressibility in this case is considered to be lossy, as values are hard thresholded to 0. This is done to emulate the behaviour of a typical BPDN-solver or similar inverse linear program with basis regularisation.

This analysis implies that although larger signals are generally more compressible, in the case of CS imaging, a trade-off must be made between a targeted  $s$ -sparse signal informing the density of  $\phi$  and the desired compression and thus reconstruction performance.

### Basis Proxy Effects on Depth

To evaluate the effects of compressive sampling with a targeted  $s$  value defined by  $\Psi$ , the scenes are restored using their respective inverse transforms for Daubechies wavelets and DCT for  $\tilde{X}_I$  and  $\tilde{X}_Q$  respectively and depth is the recovered  $\tilde{X}_D = \tilde{X}_Q / \tilde{X}_I$ . The recovered depth is compared to the available ground truth,  $X_D$ , of the signal provided by the synthetic scenes. Total-variation is omitted from this particular analysis as the compressive sensing reconstruction does not recover the basis signal but only uses it as a regularisation parameter in constructing the signal directly.

In this initial analysis a selection of standard signal and image quality metrics are used for quality evaluation in this work, namely mean squared error (MSE),

power signal-to-noise ratio (PSNR), reconstruction signal-to-noise ratio (RSNR) [112], structural similarity index measure (SSIM) [240] and standard deviation ( $\sigma$ ).

A brief overview of these metrics is provided below for the ground truth  $x$  and the estimate  $y$  both  $x, y \in \mathbb{R}^n$  to be compared as

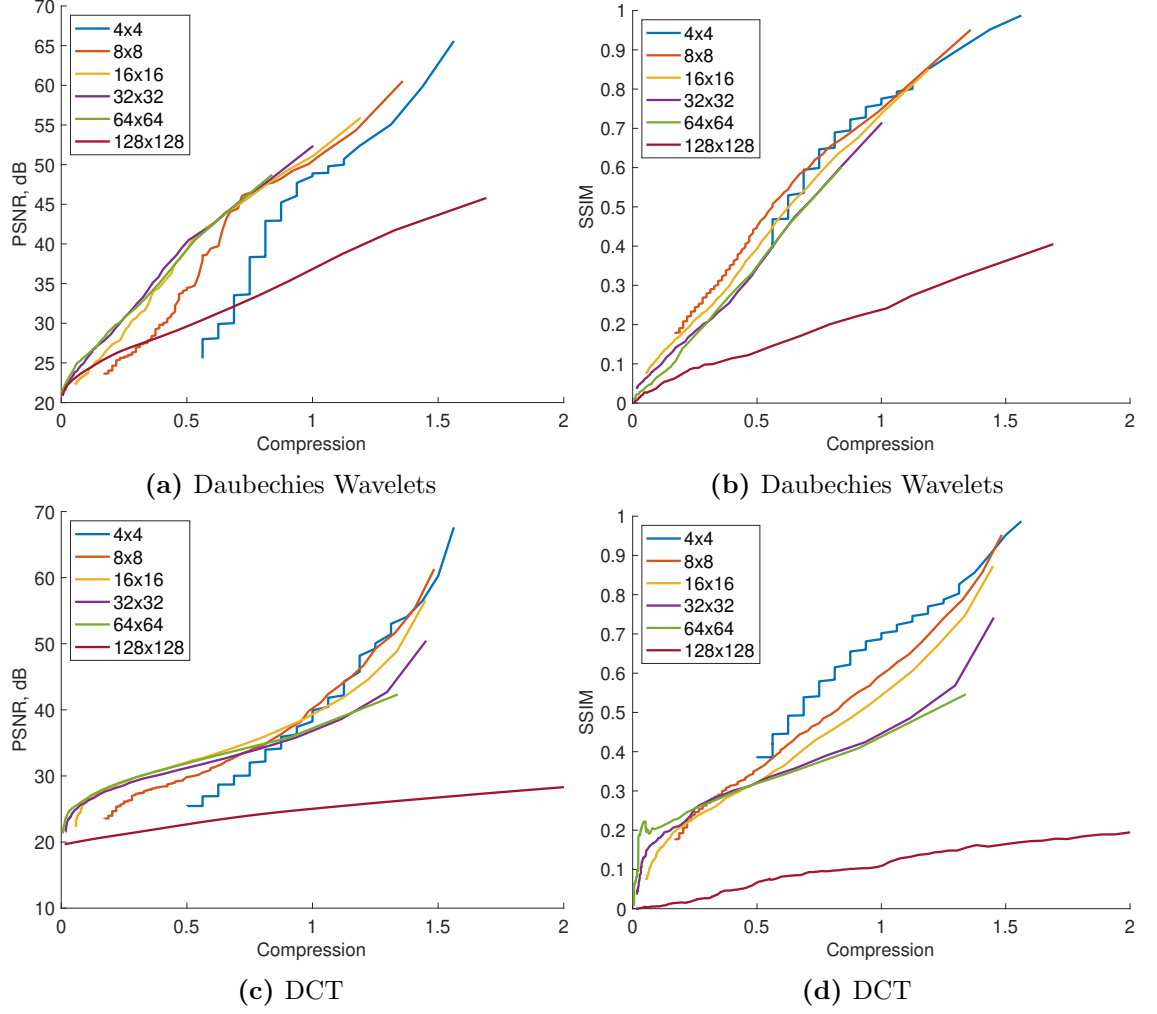
$$\begin{aligned}
 \text{MSE}(x, y) &= \frac{1}{n} \sum_{i=0}^n (x - y)^2 \\
 \text{PSNR}(x, y) &= 10 \log_{10} \frac{\max(x)}{\text{MSE}(x, y)} \\
 \text{RSNR}(x, y) &= 10 \log_{10} \frac{\sum_{i=0}^n (x)^2}{\sum_i^n (x - y)^2} \\
 \text{SSIM}(x, y) &= l(x, y)^\alpha \cdot c(x, y)^\beta \cdot s(x, y)^\gamma \\
 \sigma(x, y) &= \sqrt{\frac{1}{n} \sum_{i=0}^n |(|x - y|) - \mu(|x - y|)|^2},
 \end{aligned} \tag{5.31}$$

with  $l$  being luminance which in this case is depth,  $c$  a contrast comparator and  $s$  the structure comparison function with weighting parameters  $\alpha, \beta, \gamma$  with more details on their definition in [240] and  $\mu = \frac{1}{n} \sum_{i=0}^n (x)$  being the standard mean. The results for the mid-range scene are shown in Figure 5.16 for PSNR and SSIM as evaluation metrics for block sizes up to  $N_B = 32$  and with a full frame resolution of  $N_B = 128$ .

It is quite obvious that PSNR (Figure 5.16(a,c)) is a poor metric for such high dynamic signals, as the errors scale dramatically as range increases, further amplified due to photon count scaling in  $X_Q$ . This is often worsened in a real system by the fact that active imaging systems observe a lower SNR as range increases, meaning that the uncertainty across the operating range increases to the far end of the range spectrum observed.

Therefore, this work accompanies PSNR with at least another image quality metric such as SSIM or MSE. In particular SSIM provides more information about the structural integrity and continuity and is not as easily influenced by random large errors cancelling each other out. Judging the reconstruction quality solely on the PSNR would suggest that the  $4 \times 4$  tiling scheme is entirely unsuitable for block compressive sensing as it performs worse in the compression regime ( $\mathcal{C} < 1$ ), in particular in the wavelet basis (Figure 5.16(a)).

However, looking at the SSIM (Figure 5.16(b,d)), the smallest block size recovers the signal at a similar quality to all other block sizes and indeed due to its lower compressibility has a higher quality floor and also a higher quality ceiling. This



**Figure 5.16:** Reconstruction quality effects in Wavelet (Daubechies) for (a) PSNR and (b) SSIM as well as DCT basis (c) PSNR and (d) SSIM for various tile sizes and the respective compression ratio,  $\mathcal{C} = m/n$  based on sparsity of  $X_Q$ .

makes sense, because as more values are significant, less information is lost in the compression at the cost of slightly lower compressibility. As depth recovery is costlier than normal CS intensity recovery, the significant reduction in problem size could provide meaningful acceleration of the reconstruction framework to enable real-time performance.

Furthermore, the DCT basis shows better performance for the same level of compression as block size decreases and has a consistent compression curve throughout block sizes. This suggests that DCT is a very suitable basis for small scale block CS recovery. It is also interesting to see that with an increase in block size quality (SSIM) can significantly deteriorate. This may explain why most blocking schemes rely on full-scale steps to improve the overall quality.

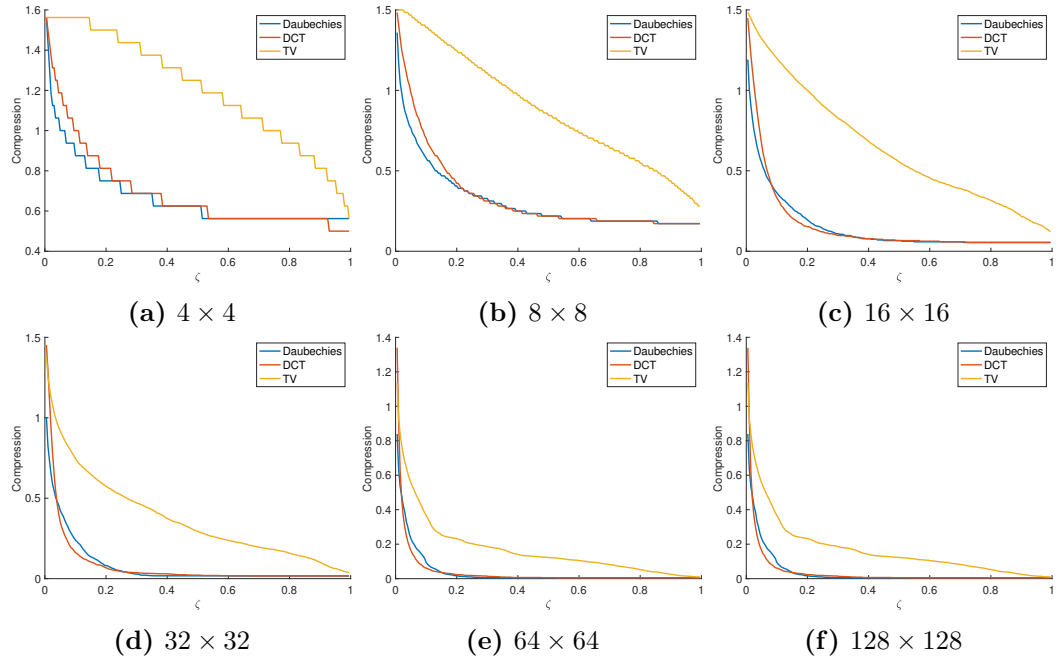
For the depth estimation with the two proxies,  $X_I$  and  $X_Q$ , with respect to com-

pressibility and associated overall quality it seems that block recovery is actually beneficial, due to the quality ceiling increasing as the block sizes decreases. This holds in particular for the DCT basis transform looking at Figure 5.16(d). This suggests, that independent compressive block recovery for this type of depth recovery may yield better results, than full scale CS reconstruction due to better per block reconstruction. As mentioned earlier, the subsequent reduction in problem size, could enable to exploit the compression at the sampling stage in real-time at very high frame rates as an independent block formulation has the potential to further benefit from massive parallelism.

### Total-Variation Block Compression

Total-variation is a popular regularisation function as it can be applied directly to recover the signal without prior basis transformations [199]. It has been extensively used in CS and inverse problems involving natural images [241, 189, 242, 243, 244, 245, 246, 247].

Applying blocking to a total-variation based signal recovery has its caveats. Total-variation operates on the entire signal, therefore separating the full-scale problem into small blocks without providing any connection to all neighbouring blocks should affect the compressibility dramatically. This is illustrated in Figure 5.17.



**Figure 5.17:** Spatial compression potential per block size for wavelets (Daubechies), DCT and TV based on relative energy threshold  $\zeta$  and the resultant sparsity of a signal.

The sparsity assumptions enforced by total-variation are clearly more appropriate as the block size increases and then approaches more traditional basis transformations

in particular when  $N_B = 128$  which in this analysis is the full-scale problem. This does not mean that TV is useless for such small scale blocking schemes. As a matter of fact it will be shown that it can perform quite well as it does approach the same sparsity levels as the other basis transforms at higher thresholds. It is obvious, however, that

$$\sum_b \text{TV}(X^b) \neq \text{TV}(X), \quad (5.32)$$

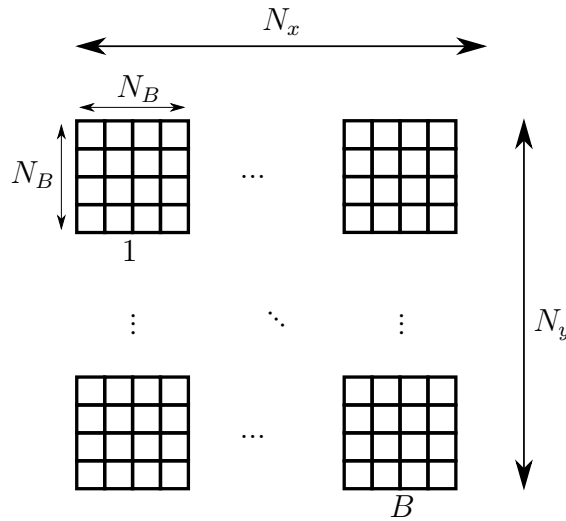
because the regularisation is not able to cascade through the entire image space and therefore independent blocking is not affected by gradient changes in another section of the image, while this is the case for the full image case.

It is investigated if TV can be used on its own or only as a full-scale post-processing step due to this inequality and how the two compare in section 5.4.3.

From the above small empirical study it would seem that TV is not the ideal linear transform for the compressive case of  $m < n$ , but may still be useful for sparse random illumination where  $m \geq n$  or indeed to improve a solution for the independent blocking scheme using the DCT or another basis in this work.

## 5.4 Time-of-Flight Solid-State Block Array

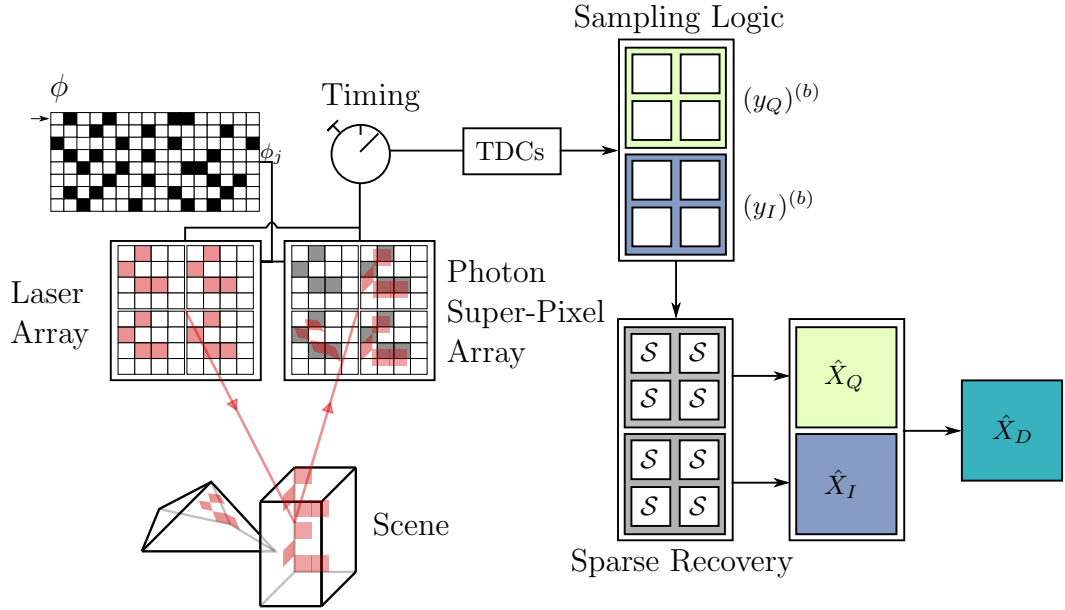
A time-of-flight solid-state detector array capable of capturing photon time stamp measurements can capture an image space  $X \in \mathbb{R}^{n=N_x \times N_y}$ . The imager can be tessellated into non-overlapping detector blocks of size  $n_B = N_B \times N_B$ . This results in  $B = n/n_B$  blocks as shown in Figure 5.18.



**Figure 5.18:** Checkerboard block structure for distribution of depth compressive sensing into smaller sub problems. Applicable to the emitter array, the detector array or both.



A proposed system architecture of this blocking methodology first presented in [82] is shown in Figure 5.19.



**Figure 5.19:** A full illustration of an arrayed Super-Pixel LiDAR system with resolution matched emitter and detector array. A Super-Pixel can be replaced with large single-pixel with the resolution defined by the emitter array. Conversely, a lower emitter array can be paired with a higher resolution super-pixel array.

The system can be configured in multiple ways making use of either an arrayed emitter, an arrayed detector or both. The resolution is defined by the highest resolution of either, whichever that may be. In the remainder of this chapter, the system is assumed to be configured as shown in Figure 5.19 with a resolution matched emitter and detector. The detector array is sectioned into  $B$  blocks and each block is sampled individually in its respective measurement vector  $(y_Q)^{(b)}$  and  $(y_I)^{(b)}$  each containing  $m_B$  measurements respectively with the sampling and signal model outlined in (5.22)-(5.23).

### 5.4.1 Signal Model

Now that the imager is an array, the problem can be distributed as many individual imaging blocks and the problem becomes

$$\begin{bmatrix} y_1^{(1)} \\ \vdots \\ y_m^{(B)} \end{bmatrix} = \begin{bmatrix} \phi^{(1)} \\ \vdots \\ \phi^{(B)} \end{bmatrix} \begin{bmatrix} x^{(1)} & \dots & x^{(B)} \end{bmatrix}, \quad (5.33)$$

where each block has its own projection sequence,  $\phi^{(b)}$ . For maximum efficiency the system is to operate on a single projection sequence, the problem simplifies to

$$\begin{bmatrix} y_1^{(1)} \\ \vdots \\ y_m^{(B)} \end{bmatrix} = \phi \begin{bmatrix} x^{(1)} & \dots & x^{(B)} \end{bmatrix}, \quad (5.34)$$

where  $\phi$  is applied to sample the scene in a structured fashion yielding a measurement vector on a per block basis, which results in a global measurement ensemble  $y_Q, y_I \in \mathbb{R}^{N_B \times m_B}$  with  $\phi \in \{0, 1\}^{N_B \times m_B \times n_B}$ , if  $\phi$  is identical for all blocks then  $\phi_B \in \{0, 1\}^{m_B \times n_B}$ . The transform  $\theta \in \mathbb{R}^{n_B \times n_B}$  is identical for all blocks. The full signal is again an ensemble of blocks such that  $x_Q, x_I, x_D \in \mathbb{R}^{N_B \times n}$ . Let  $\mathcal{S}(\cdot)$  be a generic solver to the problem formulated in (5.26) and (5.27) respectively.

Many algorithms exist to numerically find a solution to this inverse problem such as ADMM [237], fast iterative shrinkage thresholding algorithm (FISTA) [212], GPSR [234], orthogonal matching pursuit (OMP) [248]. A common solver for the TV formulation of these problems is TVAL3 [232], which is used in this work alongside ADMM for basis transforms to reconstruct depth images from a compressive measurement set.

These specific algorithms were chosen for their good performance, adaptability and reconfigurability and relatively low execution time. This does not mean they are the optimal solvers but they should provide a good idea of efficient solvers for these types of problems.

Now that the problem is separated into many small sub-problems, the depth reconstruction framework is described in Algorithm 4. The `unraster( $\cdot$ )` operator reshapes

---

**Algorithm 4** Depth Checkerboard Compressive Sensing (CBCS) Recovery [82]

---

**Input**  $y_Q, y_I, \phi, \theta$

**Output**  $\hat{X}_D$

**for** every block  $b$  **do**

$(\hat{x}_Q)^{(b)} = \mathcal{S}((y_Q)^{(b)}, \phi^{(b)}, \theta)$

$(\hat{x}_I)^{(b)} = \mathcal{S}((y_I)^{(b)}, \phi^{(b)}, \theta)$

**end for**

$\hat{x}_D(x_I > 0) = x_Q \cdot x_I$

$\hat{X}_D = \text{unraster}(\hat{x}_D)$

---

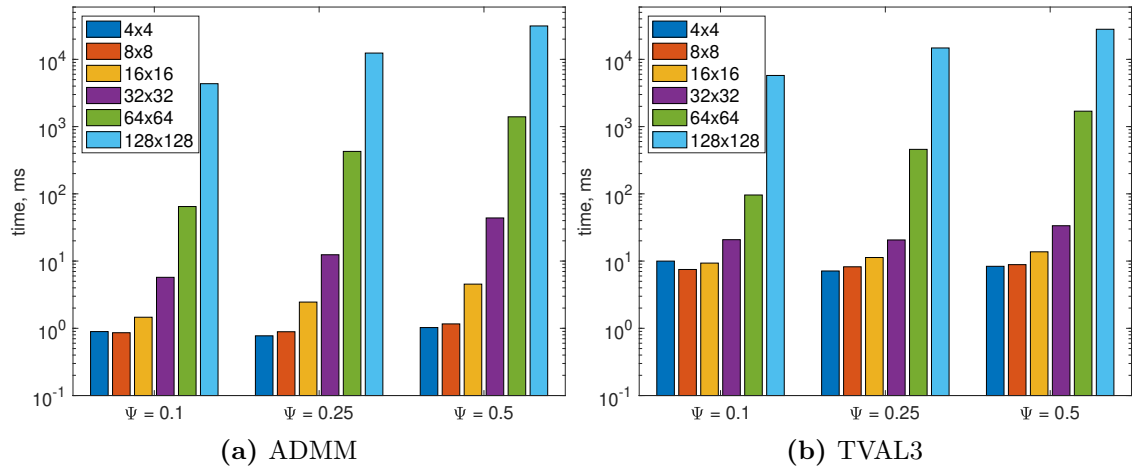
vectors into their array dimension and places them in their respective block location for each recovered image.

This framework removes the restrictive steps from the formulation in [231] and enforces basis sparsity directly, if applicable. The individual and independent blocking

structure further enables concurrent execution of each optimisation process unlike [213], which relies on full frame steps throughout a full iteration. Combined with a reduction in problem size related to the block size, this should provide a pathway to fast CS depth recovery.

### 5.4.2 Block Size Effects on Processing Speed

To illustrate the time savings for smaller block sizes, ADMM and TVAL3 are set for a constant number of iterations to remove that time variable and a random problem is generated for various block sizes shown in Figure 5.20.



**Figure 5.20:** Processing time scaling for increasing problem for BPDN using ADMM [237] and TV using [232] with a fixed number of iterations of  $i = 100$ .

Given that the time is shown on a logarithmic scale, it is quite obvious, that processing times rise exponentially with modest increases in block size, while  $N_B = 4$  and  $N_B = 8$  are comparable in terms of processing time for a fixed number of iterations, while all other sizes are dramatically slower and for  $N_B > 16$  recovery of images at real-time frame rates is unlikely. It needs to be noted, that forcing the iteration count to  $i = 100$  is beneficial to large scale problems and detrimental to small scale problems, as generally the the small scale problems requires fewer iterations  $i_{N_B \leq 16} \ll 100$  while  $i_{N_B > 16} \gg 100$ . Once again favouring smaller block sizes from a processing speed point of view.

### 5.4.3 Total Variation Extension

As stated in equation (5.32) TV is not separable in the same way a linear transform is, as DCT and discrete wavelet transform (DWT) can be readily adapted to a particular signal size and are often applied in a blocked fashion in their discrete implementations in particular in compressive applications [176]. To allow for TV

to take full advantage of the full scale problem, two methods are proposed, which utilise a fast parallel block solution as a prior, which in turn should either improve performance in speed, in quality or both. In particular full image regularisation should reduce any kind of blocking artefacts.

### TV with Block Prior

The utilisation of a block solution has been already discussed in the prior art of BCS with full scale processing steps using only basis sparsity optimisation steps. It is suggested in [202] that a prior in the optimisation problem reduces the number of required measurements for the full scale problem. Ideally the number of measurements can be the same as for the block ( $m_B$ ), which would result in a fairly small full-scale problem formulation. To construct such a measurement vector, let  $\tilde{y} \in \mathbb{R}^{m_B}$  be

$$\tilde{y} = \sum_b y^{(b)}. \quad (5.35)$$

The projection matrix for the full-scale problem,  $\tilde{\phi} \in \{0, 1\}^{m_B \times n}$ , with prior analysis for each pattern  $j$  is then

$$\tilde{\phi}_j = \begin{bmatrix} \phi_j^{(1)} & \dots & \phi_j^{(k)} \\ \vdots & \ddots & \vdots \\ \phi_j^{(q)} & \dots & \phi_j^{(B)} \end{bmatrix}. \quad (5.36)$$

Using the above full-scale formulations, the prior block solution is further optimised with a final TV optimisation routine. This results in the following optimisation problem

$$\begin{aligned} \min_x \quad & \beta \|x\|_{\text{TV}} \\ \text{s.t.} \quad & \tilde{\phi}x = \tilde{y} \text{ and } x_0 = \hat{x} \end{aligned} \quad (5.37)$$

where  $x$  is initialised as  $\hat{x}$  from the block solution. This can be implemented as shown in Algorithm 5 with depth recovery as before from the final full-scale estimates for the proxy quantities.

---

**Algorithm 5** Depth Block Compressive Sensing using TV recovery with block prior (CBCS-TVp).

---

**Input**  $y_Q, y_I, \phi, \theta, \tilde{y}_Q, \tilde{y}_I, \tilde{\phi}$   
**Output**  $\hat{X}_D$   
  **for** every block  $b$  **do**  
     $(\hat{x}_Q)^{(b)} = \mathcal{S}((y_Q)^{(b)}, \phi^{(b)}, \theta)$   
     $(\hat{x}_I)^{(b)} = \mathcal{S}((y_I)^{(b)}, \phi^{(b)}, \theta)$   
  **end for**  
   $\tilde{x}_Q = \text{TV}(\tilde{y}_Q, \tilde{\phi}, \hat{x}_Q)$   
   $\tilde{x}_I = \text{TV}(\tilde{y}_I, \tilde{\phi}, \hat{x}_I)$   
   $\tilde{X}_Q = \text{unraster}(\tilde{x}_Q)$   
   $\tilde{X}_I = \text{unraster}(\tilde{x}_I)$   
   $\hat{X}_D(\hat{X}_I > 0) = \tilde{X}_Q ./ \tilde{X}_I$

---

Alternatively, a block-diagonal sensing matrix,  $\bar{\phi} \in \{0, 1\}^{m \times n}$  and an ensemble of measurement vectors, rasterised as  $\bar{y} \in \mathbb{R}^{m=B \times m_B}$ , can be used as in (5.10) and (5.9). This can be used for a full-scale prior analysis such as presented in [246]. This approach uses the prior solution,  $\hat{x}$  in the regularisation directly, such that

$$\begin{aligned} \min_x \quad & \beta_1 \|x\|_{\text{TV}} + \beta_2 \|x - \hat{x}\|_{\text{TV}} \\ \text{s.t.} \quad & \bar{\phi}x = \bar{y}. \end{aligned} \tag{5.38}$$

The recovery is similar to the other TV extension using  $\hat{x}$  as a starting point and is shown below in Algorithm 6.

---

**Algorithm 6** Depth Block Compressive Sensing TV-TV [246] recovery with block prior (CBCS-TVTV).

---

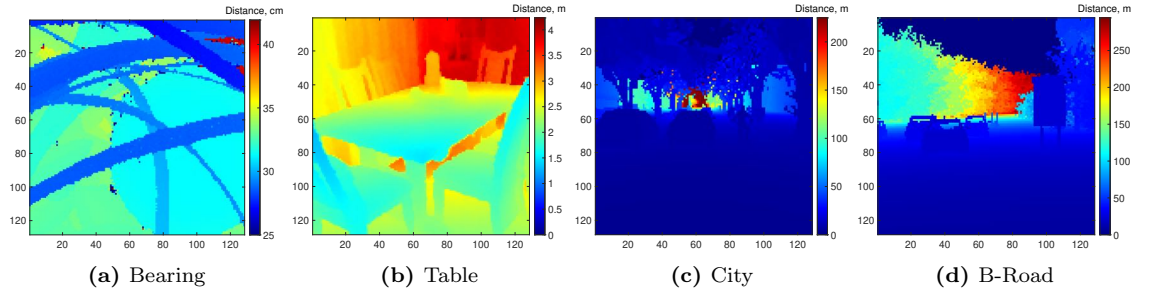
**Input**  $y_Q, y_I, \phi, \theta, \bar{y}_Q, \bar{y}_I, \bar{\phi}$   
**Output**  $\hat{X}_D$   
  **for** every block  $b$  **do**  
     $(\hat{x}_Q)^{(b)} = \mathcal{S}((y_Q)^{(b)}, \phi^{(b)}, \theta)$   
     $(\hat{x}_I)^{(b)} = \mathcal{S}((y_I)^{(b)}, \phi^{(b)}, \theta)$   
  **end for**  
   $\tilde{x}_Q = \text{TVTV}(\bar{y}_Q, \bar{\phi}, \theta, \hat{x}_Q)$   
   $\tilde{x}_I = \text{TVTV}(\bar{y}_I, \bar{\phi}, \theta, \hat{x}_I)$   
   $\tilde{X}_Q = \text{unraster}(\tilde{x}_Q)$   
   $\tilde{X}_I = \text{unraster}(\tilde{x}_I)$   
   $\hat{X}_D(\hat{X}_I > 0) = \tilde{X}_Q ./ \tilde{X}_I$

---

Both these approaches are empirically evaluated alongside individual CBCS modes for both basis transforms and TV regularisation.

## 5.5 Applications and Results

The goal in developing the checkerboard compressive sensing (CBCS) processing methodology described in this chapter is the formulation of a robust framework for efficient depth imaging at high resolution with high frame rates suitable for many applications without strict constraints on scene content or other restrictive assumptions often seen in previous work. The system presented in [82] and shown in Figure 5.19 is used as a simulation framework to re-sample real and synthetic photon count data for a variety of scenes shown in Figure 5.21.



**Figure 5.21:** Scenes for evaluation of CBCS methodology for arrayed CS LiDAR systems. (a) A real photon count scene depicting a bearing in underwater foliage [99]. (b) A real depth scene captured with a Kinect depth camera [124]. (c) A city scene from a synthetic depth data set [123]. (d) B-Road scene as a synthetic replica [122] of original KITTI benchmark [117]. (b)-(d) provide segmentation maps and RGB images from which photon counts are simulated.

The four scenes are chosen to emulate 4 different depth applications, the bearing scene from [99] represents a very detailed high-resolution ultra-short range application with a dynamic range of only about 30 cm with histogram sub-millimetre precision. The dining table scene from [124] represents a typical indoor short range application with a dynamic range of  $< 10$  m and millimetre precision. Next a mid-range city environment with many scene participants and objects such as trees and signs in the foreground and (0-50 m) and buildings in the background ( $\leq 250$ ) [123] and finally a long range scene depicting a typical B-Road scene with cars, traffic signs and tree foliage surrounding the street with a dynamic range of 0-300 m [122] with centimetre resolution in both mid and long range cases. The last two scenarios are typical scenes for automotive applications and demonstrate outdoor long-range applications which require fast decision making and thus call for high frame rates.

The reconstruction performance is compared with ground truth depth if available and a maximum cross-correlation approach for the real scene for a good depth estimate using the maximum of the correlation between return signal and the instrumental system response. An extended selection of quality metrics is used, because it is difficult to assess depth reconstruction quality, due to its dependence on max-

imum depth value for a high dynamic range. This means the absolute error can have a high variance and the structural integrity of shapes is no longer flattened as is the case for intensity images. This affects most error and image quality metrics. Using multiple metrics should allow to identify good solutions if all or most metrics consistently indicate a good reconstruction result relative to each other.

The framework presented in this work is compared to Howland et al. [231] representing a similar signal model and single pixel CS approach, BCS-SPL [213] for an established block CS approach which has primarily been applied to intensity and colour imaging and variants of the presented framework with various basis transforms, the TV extensions and the discrete pseudo-inverse approach *d*Sparse for sparse random imaging.

### 5.5.1 Practical Basis Transforms and Algorithms

To implement the basis transform  $\theta$  as efficiently as possible it is desirable to use a quantised discrete matrix transform. The DCT transform is defined in matrix form and can therefore be readily quantised into matrix form for the appropriate signal size. The other basis transform under investigation uses Daubechies wavelets [196]. These are often defined as continuous functions. While efficient implementations exist for this family of wavelets, recent work also suggests that they can be implemented in a single matrix transformation [236]. This work adopts this approach to generate a discrete wavelet transform (DWT) matrix with 2 filter coefficients. All projection matrices are generated using Algorithm 3 for the appropriate image or block size and are of pseudo-random binary nature.

The framework presented in [231] has been faithfully recreated from their description and considerable work has been carried out to retrieve the best possible results with extensive manual parameter tuning using the solvers TVAL3 for the total-variation estimates and GPSR [234] for the debiasing stages as outlined in their manuscript.

The source code for BCS-SPL has been published and was straightforward to apply to the dual-stage depth recovery via the two proxy units. The algorithm is operated in two modes using the DCT basis transform, with the prescribed optimal block size as described in [213] of  $N_B = 32$ , which shall be denoted as BCS-SPL<sub>32</sub> and for a fair comparison with the blocking scheme in this work, the block size was altered to  $N_B = 4$  as well (BCS-SPL<sub>4</sub>). Neither of these approaches are easily parallelisable, as they rely on full-scale reconstruction at one point or throughout the algorithm. The processing time is quoted for a full sequential run  $t_{\text{seq}}$ . All algorithms are measured in a full sequential mode for comparison.

The primary framework of this work, CBCS, is evaluated for block sizes as  $\text{CBCS}_{N_B}$  and similarly  $d\text{Sparse}_{N_B}$ , the chosen basis and optional extension are added onto this main naming scheme if applicable. All CBCS and  $d\text{Sparse}$  can be either fully parallelised i.e. are block independent or partially parallelisable in case of the two TV extensions where the block estimate is fully parallelisable and is followed by two full-scale problems.

An estimate for a full parallel implementation is provided as  $t_{\text{par}}$  which is derived from the slowest block solution for either  $x_Q$  or  $x_I$ . The TV extensions take the slowest block estimate time value and the slowest full-scale solution time is added for the parallel estimate for those approaches. The solver for BPDN using linear basis transforms chosen for this work is the popular ADMM [237] algorithm, due to its speed, flexibility and relative simplicity.

The TV based BPDN CS formulation uses the popular TVAL3 [232] algorithm as it is a fast approach and produces reliably good results. The first TV extension (5.37) exclusively uses TVAL3 for all TV regularisation based recoveries. The second TV extension (5.38) adopts a  $\text{CBCS}_4\text{-DCT}$  approach using ADMM for the block estimate and the algorithm from [246] to perform the full-scale recovery with prior using a block diagonal CS system.

### 5.5.2 Arrayed Sparse LiDAR

To illustrate a standard approach to determine the operational mode of a arrayed sparse LiDAR using the CBCS framework, a typical analysis of block size and sparsity is carried out to find good operational parameters followed by the sampling and processing protocol. While this analysis only includes three linear transforms, namely DWT, DCT and TV. Other basis functions can be readily implemented if more suitable for a particular system and its intended application scenario. A typical analysis to find good operational parameters is shown in Table 5.4, where compression ratio and target sparsity  $\Psi$  is swept across and compared with various block sizes.

This analysis uses a small selection of quality metrics (PSNR, SSIM, MSE) to determine a good operating regime. It is desirable to find the lowest compression ratio  $\mathcal{C}$  determining the number of measurements with the lowest sparsity,  $\Psi$  to reduce the projection pattern density and thus laser exposure per measurement. It is quite obvious again that PSNR is an unreliable metric when errors become large, however, SSIM tends to indicate when a reconstruction was successful with values usually greater than  $\text{SSIM} > 0.2$ . This metric seems quite low for the medium and long



**Table 5.4:** CBCS operating parameters for sparsity and compression illustrated on the *mid-range scene*. Operating point for examples in this chapter are highlighted.

|                |     | PSNR, dB      |       |              |       | SSIM          |      |             |      | MSE           |        |               |        |
|----------------|-----|---------------|-------|--------------|-------|---------------|------|-------------|------|---------------|--------|---------------|--------|
|                |     | $\mathcal{C}$ |       |              |       | $\mathcal{C}$ |      |             |      | $\mathcal{C}$ |        |               |        |
| $\Psi = 0.125$ |     | 0.125         | 0.25  | <b>0.5</b>   | 0.75  | 0.125         | 0.25 | <b>0.5</b>  | 0.75 | 0.125         | 0.25   | <b>0.5</b>    | 0.75   |
| $4 \times 4$   | DWT | 18.23         | 18.75 | 19.53        | 21.44 | 0.03          | 0.05 | 0.07        | 0.11 | 759.95        | 672.96 | 562.41        | 362.49 |
|                | DCT | 20.19         | 20.58 | 19.69        | 22.92 | 0.14          | 0.14 | 0.23        | 0.37 | 483.33        | 441.72 | 541.94        | 257.96 |
|                | TV  | 16.14         | 19.03 | 22.08        | 25.26 | 0.07          | 0.14 | 0.22        | 0.29 | 1227.62       | 631.58 | 312.86        | 150.29 |
| $8 \times 8$   | DWT | 18.15         | 18.43 | 19.41        | 21.84 | 0.03          | 0.05 | 0.09        | 0.41 | 772.87        | 724.57 | 578.93        | 330.37 |
|                | DCT | 20.75         | 20.63 | 21.78        | 22.74 | 0.17          | 0.20 | 0.27        | 0.32 | 425.46        | 437.10 | 335.23        | 268.70 |
|                | TV  | 21.31         | 23.85 | 24.50        | 24.32 | 0.15          | 0.22 | 0.31        | 0.44 | 373.53        | 208.00 | 179.26        | 186.70 |
| $16 \times 16$ | DWT | 17.74         | 17.98 | 18.40        | 19.70 | 0.03          | 0.04 | 0.07        | 0.38 | 850.18        | 804.61 | 730.47        | 540.67 |
|                | DCT | 18.86         | 19.93 | 20.72        | 20.56 | 0.12          | 0.17 | 0.22        | 0.24 | 656.91        | 513.34 | 427.64        | 444.14 |
|                | TV  | 21.85         | 21.19 | 20.45        | 20.88 | 0.20          | 0.22 | 0.30        | 0.38 | 330.21        | 383.62 | 455.87        | 412.15 |
| $32 \times 32$ | DWT | 17.64         | 17.56 | 17.97        | 17.80 | 0.02          | 0.03 | 0.06        | 0.26 | 868.79        | 885.67 | 806.27        | 838.87 |
|                | DCT | 18.12         | 19.76 | 19.13        | 19.54 | 0.19          | 0.21 | 0.23        | 0.24 | 779.56        | 534.41 | 617.11        | 561.78 |
|                | TV  | 19.23         | 18.43 | 18.44        | 18.47 | 0.18          | 0.19 | 0.23        | 0.27 | 602.68        | 725.27 | 722.93        | 717.81 |
| $\Psi = 0.25$  |     |               |       |              |       |               |      |             |      |               |        |               |        |
| $4 \times 4$   | DWT | 18.52         | 18.84 | 20.69        | 22.97 | 0.04          | 0.05 | 0.10        | 0.22 | 710.65        | 659.16 | 430.51        | 255.06 |
|                | DCT | 21.77         | 21.57 | <b>22.64</b> | 22.98 | 0.29          | 0.29 | <b>0.38</b> | 0.45 | 336.15        | 351.48 | <b>275.24</b> | 254.59 |
|                | TV  | 17.87         | 22.07 | <b>24.78</b> | 25.51 | 0.08          | 0.17 | <b>0.26</b> | 0.40 | 824.39        | 313.86 | <b>168.20</b> | 141.92 |
| $8 \times 8$   | DWT | 18.21         | 18.88 | 20.04        | 21.42 | 0.03          | 0.05 | 0.10        | 0.49 | 763.21        | 653.45 | 500.44        | 363.92 |
|                | DCT | 21.06         | 20.38 | 21.52        | 22.28 | 0.18          | 0.21 | 0.28        | 0.32 | 396.09        | 462.35 | 356.08        | 299.12 |
|                | TV  | 22.66         | 23.49 | 23.40        | 22.88 | 0.17          | 0.22 | 0.34        | 0.49 | 273.82        | 226.19 | 230.93        | 259.96 |
| $16 \times 16$ | DWT | 17.67         | 17.80 | 18.62        | 20.01 | 0.02          | 0.04 | 0.07        | 0.40 | 863.05        | 838.52 | 693.28        | 504.44 |
|                | DCT | 18.48         | 20.32 | 20.35        | 20.55 | 0.13          | 0.19 | 0.22        | 0.24 | 716.97        | 469.41 | 465.95        | 444.61 |
|                | TV  | 20.96         | 20.97 | 19.94        | 20.67 | 0.18          | 0.22 | 0.31        | 0.41 | 404.52        | 403.74 | 511.99        | 432.90 |
| $32 \times 32$ | DWT | 17.56         | 17.29 | 18.03        | 18.14 | 0.02          | 0.03 | 0.06        | 0.29 | 886.06        | 942.70 | 795.23        | 774.35 |
|                | DCT | 18.58         | 19.25 | 19.32        | 19.17 | 0.18          | 0.20 | 0.20        | 0.20 | 699.92        | 600.59 | 590.19        | 611.54 |
|                | TV  | 19.13         | 18.64 | 18.57        | 18.54 | 0.16          | 0.18 | 0.21        | 0.21 | 616.49        | 690.70 | 702.71        | 707.23 |
| $\Psi = 0.5$   |     |               |       |              |       |               |      |             |      |               |        |               |        |
| $4 \times 4$   | DWT | 19.21         | 19.68 | 21.05        | 23.90 | 0.04          | 0.07 | 0.13        | 0.49 | 606.56        | 543.87 | 396.52        | 205.79 |
|                | DCT | 23.30         | 22.69 | 22.83        | 23.80 | 0.44          | 0.41 | 0.45        | 0.50 | 236.08        | 272.17 | 263.07        | 210.50 |
|                | TV  | 19.82         | 23.89 | 24.97        | 25.63 | 0.09          | 0.23 | 0.47        | 0.62 | 526.90        | 206.33 | 160.66        | 138.21 |
| $8 \times 8$   | DWT | 18.24         | 18.30 | 19.90        | 21.71 | 0.03          | 0.05 | 0.13        | 0.51 | 758.11        | 747.05 | 516.68        | 340.59 |
|                | DCT | 21.41         | 21.18 | 22.48        | 22.84 | 0.20          | 0.23 | 0.32        | 0.34 | 567.11        | 385.32 | 285.59        | 262.78 |
|                | TV  | 23.26         | 23.82 | 23.23        | 23.07 | 0.21          | 0.30 | 0.43        | 0.53 | 238.26        | 209.71 | 239.91        | 249.31 |
| $16 \times 16$ | DWT | 17.69         | 17.64 | 19.23        | 20.01 | 0.03          | 0.04 | 0.09        | 0.42 | 860.55        | 870.03 | 602.41        | 503.89 |
|                | DCT | 19.50         | 20.88 | 20.88        | 21.58 | 0.15          | 0.20 | 0.21        | 0.22 | 567.11        | 412.09 | 412.27        | 350.98 |
|                | TV  | 20.62         | 20.57 | 20.77        | 20.53 | 0.18          | 0.23 | 0.32        | 0.36 | 437.72        | 442.57 | 423.30        | 446.71 |
| $32 \times 32$ | DWT | 17.62         | 17.84 | 18.05        | 18.15 | 0.02          | 0.03 | 0.06        | 0.28 | 873.46        | 830.86 | 791.52        | 772.95 |
|                | DCT | 19.18         | 19.87 | 19.83        | 18.75 | 0.12          | 0.16 | 0.14        | 0.16 | 610.36        | 520.13 | 525.34        | 673.93 |
|                | TV  | 19.22         | 18.63 | 18.36        | 18.37 | 0.16          | 0.14 | 0.14        | 0.16 | 604.98        | 692.53 | 736.65        | 735.95 |

range scene under investigation due to errors scaling as range increases. However, there is a clear distinction between complete failure at  $\text{SSIM} < 0.1$  and a chance for good reconstruction otherwise.

The MSE provides a good relative measure, but large values can indicate either a complete failure or just large errors at long ranges while the foreground depth returns are well reconstructed. It is quite clear that DCT generally outperforms DWT in most cases, in particular for lower compression ratios. Generally, reconstruction quality drops for larger tile sizes, likely due an increase in surface count in the FoV of the block imager in contrast to assumption 3 and thus  $n_B = 4 \times 4$  is treated as the optimal tile size for the remainder of this work.

The compression is chosen to be a good compromise of number of measurements and quality at  $\mathcal{C} = 0.5$  as a significant drop-off in quality is observed if fewer measurements are used, while more measurements only provide marginal increases. Choosing  $\Psi$  is a trade-off between pattern density and thus total laser output and quality. We choose  $\Psi = 0.25$  for a relatively low sparsity target of  $s = 4$  for  $4 \times 4$  blocks

as the quality is only marginally lower than at  $\Psi = 0.5$  but significantly higher than  $\Psi = 0.125$ . These results are in agreement with the previous basis sparsity investigations in Chapter 5.3.2.

Setting the block size to  $N_B = 4$  has also favourable effects on general processing times as well as sample times as it defines the individual block problem size for equations (5.22)-(5.23) which also affects sampling time. The sampling time can be very system specific, but this work defines the total sampling time for a pattern (and equivalently a single pixel or line of pixels in a scanning system) as

$$t_{\text{samp}} = \eta_P n_P \delta t, \quad (5.39)$$

where  $\eta_P$  is the number of pattern exposure cycles,  $n_P$  is the number of laser pulses and  $\delta t$  the time-of-flight as defined by its maximum operating range (here 300 m). In the following analysis the photon count simulation assumes a macropixel with 16 SPADs as outlined in Chapter 2 and Chapter 3. A single pulse can yield therefore up to 16 counts per bin in a histogram. It is assumed that one can record multiple events using multi-event TDC devices [249] and if all events can be added together.

For the chosen operating parameters, the processing and sampling times are presented in Table 5.5, where  $t_Q^{(1)}$  and  $t_I^{(1)}$  are the reconstruction times for a single block for each proxy respectively,  $t_{\text{par}}$  the slowest computation time per block per proxy for a fully parallel time estimation,  $t_{\text{seq}}$  the total computation time if all blocks are computed in sequential fashion. The sampling time for the defined compression is  $t_{\text{samp}}$  (equation (5.39)), leading to a frame time of  $T_{\text{par}}$  assuming full parallelism and  $T_{\text{seq}}$  for a fully sequential approach.

**Table 5.5:** CBCS block size processing and sampling time illustration for  $\Psi = 0.25$ ,  $\mathcal{C} = 0.5$ , and number of laser pulses,  $n_P = 100$  with a ToF of  $2 \mu\text{s}$  each. All times are in ms.

|                |            | $t_Q^{(1)}$ | $t_I^{(1)}$ | $t_{\text{par}}$ | $t_{\text{seq}}$ | $t_{\text{samp}}$ | $T_{\text{par}}$ | $T_{\text{seq}}$ | SSIM        |
|----------------|------------|-------------|-------------|------------------|------------------|-------------------|------------------|------------------|-------------|
| $4 \times 4$   | DWT        | 1.46        | 1.25        | 1.46             | 2790.00          | 1.60              | 3.06             | 2791.60          | 0.10        |
|                | <b>DCT</b> | <b>0.95</b> | <b>0.65</b> | <b>0.95</b>      | <b>1650.00</b>   | <b>1.60</b>       | <b>2.55</b>      | <b>1651.60</b>   | <b>0.38</b> |
|                | TV         | 9.84        | 10.71       | 10.71            | 21060.00         | 1.60              | 12.31            | 21061.60         | 0.26        |
| $8 \times 8$   | DWT        | 4.96        | 5.00        | 5.00             | 2550.00          | 6.40              | 11.40            | 2556.40          | 0.10        |
|                | DCT        | 3.59        | 3.59        | 3.59             | 1850.00          | 6.40              | 9.99             | 1856.40          | 0.28        |
|                | TV         | 11.48       | 9.92        | 11.48            | 5490.00          | 6.40              | 17.88            | 5496.40          | 0.34        |
| $16 \times 16$ | DWT        | 23.12       | 24.69       | 24.69            | 3080.00          | 25.60             | 50.29            | 3105.60          | 0.07        |
|                | DCT        | 45.78       | 48.28       | 48.28            | 6030.00          | 25.60             | 73.88            | 6055.60          | 0.22        |
|                | TV         | 248.28      | 190.78      | 248.28           | 28110.00         | 25.60             | 273.88           | 28135.60         | 0.31        |
| $32 \times 32$ | DWT        | 336.87      | 341.88      | 341.88           | 10870.00         | 102.40            | 444.28           | 10972.40         | 0.06        |
|                | DCT        | 1620.63     | 1651.25     | 1651.25          | 52370.00         | 102.40            | 1753.65          | 52472.40         | 0.20        |
|                | TV         | 764.37      | 637.50      | 764.37           | 22430.00         | 102.40            | 866.77           | 22532.40         | 0.21        |

From this comparison it should become quite obvious that the smaller the tile size, the faster the reconstruction can be. If this approach is to be implemented on a single system on a chip though, this would result in more communication overhead

as more parallel problems are required to be solved. This issue is addressed in Section 5.5.3. Assuming that the communication can be kept at a minimum, the chosen block size produces the best reconstruction quality with the fastest per block reconstruction time. If parallelism could not be exploited at all, then  $N_B = 4$  seems to be still the fastest option in this case with a forced number of iterations in the algorithm. This ignores the fact that the number of iterations generally also scales with the problem size, therefore this analysis only provides an indication in terms of logic scaling rather than absolute time. It nonetheless provides an indication of the good performance potential for a parallel implementation of CBCS<sub>4</sub>-DCT.

### Sampling and Processing Protocol

Before analysing the various scenes and reconstruction performance for various algorithms, an outline of the two sampling and processing protocols are provided. The protocol for CS imaging outlines the protocol for  $B$  blocks. The single-pixel case is the special case where the number of blocks is  $B = 1$ , so that the number of block measurements is  $m_B = m$  and the block size is the full frame size i.e.  $n_B = n$ . The sampling and processing methodologies are as follows.

1. The scene is illuminated with structured light-source and a binary pattern from a sequence of projection patterns  $\Phi$  for each block.
2. Each of all  $B$  blocks acquire  $m_B < n_B$  histograms.
3. Each of the  $m_B$  histograms per block,  $b$  yields two measurements,  $y_Q^{(b)}$  and  $y_I^{(b)}$
4. Recover both  $\hat{x}_Q^{(b)}$  and  $(\hat{x}_I)^{(b)}$  by solving BPDN for both quantities in parallel across all blocks.
5. OPTIONAL: Apply second stage using  $\hat{x}_Q$  and  $\hat{x}_I$  as priors for full scale L2-TV or TV-TV optimisation for full-scale TV regularisation.
6. Gather  $\hat{X}_Q$  and  $\hat{X}_I$  from block solutions or directly via TV extensions.
7. Compute depth as  $\hat{X}_D = \hat{X}_Q / \hat{X}_I$ .

All data is sampled this way for all algorithms throughout with the appropriate block sizing and number of measurements  $m_B$ .

A variation of above protocol is the sampling and processing procedure for  $d$ Sparse where  $\mathcal{C} > 1$  i.e. the scene is technically oversampled, but in a sparse random fashion.

1. The scene is illuminated with structured light-source and a binary pattern from a sequence of projection patterns  $\Phi$  for each block.

2. Each of all  $B$  blocks acquire  $m_B > n_B$  histograms.
3. Each of the  $m_B$  histograms per block,  $b$  yields two measurements,  $y_Q^{(b)}$  and  $y_I^{(b)}$
4. Each measurement sequence has a unique  $A^*$  applicable to all blocks.
5. Recover both  $\hat{x}_I^{(b)}$  and  $\hat{x}_Q^b$  as an approximate least-square solution  $x = A^*y$  per block in parallel.
6. Compute depth per block as  $\hat{x}_D = \hat{x}_Q / \hat{x}_I$ .
7. Gather all blocks and assemble  $\hat{X}_D$ .

Having discussed the the protocols and a typical procedure of how to choose typical operational parameters for CBCS, the performance of the presented methodology can be evaluated against the prior art.

## Results

The evaluation of the proposed framework are assessed on two major factors, which in most signal and image processing application is a trade-off; quality and processing speed. The goal of this work is to demonstrate that it is possible to reconstruct a depth map from compressive samples at  $> 30$  Hz i.e. a frame time of 33 ms.

The four scenes represent a range of use cases from short to long range LiDAR applications and should highlight if the framework can handle various scene types and operating ranges. For all CS methods the target sparsity and compression rate are kept constant at  $\Psi = 0.25$  and  $\mathcal{C} = 0.5$ , which was chosen as a good compromise between quality and compression. It should be noted that the single-pixel approach presented in [231] does not claim to be able to perform in applications with high dynamic range at long distances and the authors only demonstrate indoor short range applications, which are included in the selection of scenes.

Since [129, 213] specify an optimal block size of  $N_B = 32$  as BCS-SPL<sub>32</sub>, the framework was run in that way, but to be fair to this blocked approach, it is also run with the same block size as CBCS being  $N_B = 4$  as BCS-SPL<sub>4</sub>.

All synthetic histograms are generated with a modest ambient photon rate derived from the incident sunlight at 1 klux, which is roughly equivalent to a mean count rate of about 0.3 photons per bin for  $\eta_P = 3$ .

All measurements are sampled as defined in equations (5.22) and (5.23) respectively and noise compensation schemes are used throughout. For the real scene, the passive approach is used, while the synthetic versions use the active approach to estimate  $\hat{\beta}$  for each block as defined in equations (5.24)-(5.25).

It is later shown that this noise removal scheme works well even at high noise rates of above 100 photons per bin per laser cycle, however the simulated exposure time has to increase for this, which would also be true for real systems. The topic of sampling time is briefly discussed with respect to these approaches, but should only be seen as trends, because the final numbers always depend on the system specifications, the maximum photon count rate and how many events per laser cycle the sampling device can record.

As noted earlier, normal signal and image evaluation metrics tend to be inconsistent for depth images and therefore additional metrics to the ones in (5.31) are introduced which have been widely used in the depth estimation deep learning community e.g. [250, 251, 252, 152, 253, 254].

This should allow a more rigorous assessment of the various algorithms considered. The 3-pixel-accuracy metric, which returns the percentage of good pixels according to a set of thresholds based on a relative error, this should be a good metric for depth as it includes error scaling typical in depth imaging, where errors tend to become larger as range increases. The absolute relative difference (ARD) aims to capture the absolute error, the root mean square error with log scale (RMSE-LS) performs error analysis with a logarithmic scale and finally a log-scale invariant mean squared error (MSE-LSI). All of these metrics are formerly described as

$$\begin{aligned}
 \text{ACC3P}(x, y) &= \frac{\sum_{i=0}^n (\max(\frac{y}{x}, \frac{x}{y}) = \delta < \text{thr})}{n} \\
 \text{ARD}(x, y) &= \frac{1}{n} \sum_{i=0}^n \text{abs}(x - y) \\
 \text{RMSE} - \text{LS}(x, y) &= \sqrt{\frac{1}{n} \sum_{i=0}^n (\log(x) - \log(y))^2} \\
 \text{MSE} - \text{LSI}(x, y) &= \frac{1}{2n} \sum_{i=0}^n (\log(x) - \log(y) + \alpha(x, y))^2,
 \end{aligned} \tag{5.40}$$

for  $\delta < \{1.25, 1.25^2, 1.25^3\}$ ,  $\alpha = \frac{1}{n} \sum_{i=0}^n (\log(x) - \log(y))^2$  and with  $x$  being the ground truth and  $y$  being the estimate as before.

Each scene was reconstructed over 5 runs for each framework and the results are averaged across all runs. The overall performance of all algorithms is presented in Table 5.6 for the average across all 4 scenes across all computation runs.

Of all compressive schemes CBCS<sub>4</sub>-DCT outperforms most algorithms in particular in terms of speed but also in most cases in terms of depth map quality. It tends to recover over 88% of pixels with a relatively low error shown with the  $\delta_1$  good pixel

**Table 5.6:** Performance evaluation across various scenes for sparse depth frameworks, best results are highlighted for **best overall** and *best CS*. All scenes are of size  $128 \times 128$ . For all compressive schemes  $\Psi = 0.25$  and  $\mathcal{C} = 0.5$ , while for  $d\text{Sparse}$ ,  $\mathcal{C}_{dS} = 1.5$ .

|   | $m$  | $B$  | <i>higher is better</i> |              |              |              | <i>lower is better</i> |                |              |              |                       |                       |
|---|------|------|-------------------------|--------------|--------------|--------------|------------------------|----------------|--------------|--------------|-----------------------|-----------------------|
|   |      |      | PSNR<br>dB              | RSNR<br>dB   | SSIM         | $\delta_1$   | ARD                    | MSE            | RMSE-LS      | MSE-LSI      | $t_{\text{par}}$<br>s | $t_{\text{seq}}$<br>s |
| Howland [231]                             | 8192 | 1    | 13.76                   | 4.77         | 0.069        | 0.434        | 0.456                  | 1827.367       | 0.937        | 0.284        | -                     | 946.44                |
| BCS-SPL <sub>32</sub> [213]               | 512  | 16   | 21.00                   | 11.99        | 0.163        | 0.749        | 0.184                  | 740.831        | 0.257        | 0.059        | -                     | 1.72                  |
| BCS-SPL <sub>4</sub> [213]                | 8    | 1024 | 18.68                   | 9.67         | 0.157        | 0.818        | 0.150                  | 476.818        | 0.345        | 0.078        | -                     | <u>0.43</u>           |
| <b>CBCS<sub>4</sub>-DWT</b>               | 8    | 1024 | 14.00                   | 4.99         | 0.088        | 0.661        | 0.311                  | 674.637        | 0.569        | 0.149        | 0.00084               | 1.73                  |
| <b>CBCS<sub>4</sub>-DCT</b>               | 8    | 1024 | 22.20                   | 13.19        | <u>0.433</u> | <u>0.881</u> | <u>0.090</u>           | 326.922        | <u>0.219</u> | <u>0.033</u> | <u>0.00073</u>        | 1.49                  |
| <b>CBCS<sub>4</sub>-TV</b>                | 8    | 1024 | 21.56                   | 12.55        | 0.284        | 0.860        | 0.109                  | <b>229.305</b> | 0.246        | 0.037        | 0.00943               | 19.32                 |
| <b>CBCS<sub>4</sub>-TV-TV<sub>p</sub></b> | 8    | 1024 | <u>23.78</u>            | <u>14.78</u> | 0.371        | 0.751        | 0.185                  | 404.536        | 0.278        | 0.065        | 1.33295               | 22.01                 |
| <b>CBCS<sub>4</sub>-DCT-TVTV</b>          | 8    | 1024 | 18.78                   | 9.78         | 0.303        | 0.810        | 0.165                  | 943.242        | 0.318        | 0.071        | 90.96944              | 183.36                |
| <b><math>d\text{Sparse}_4</math></b>      | 24   | 1024 | <b>25.78</b>            | <b>16.77</b> | <b>0.657</b> | <b>0.914</b> | <b>0.046</b>           | 253.972        | <b>0.196</b> | <b>0.027</b> | <b>0.00017</b>        | <b>0.35</b>           |
| $d\text{Sparse}_8$                        | 96   | 256  | 23.43                   | 14.43        | 0.596        | 0.881        | 0.076                  | 563.622        | 0.275        | 0.057        | 0.00182               | 0.93                  |
| $d\text{Sparse}_{16}$                     | 384  | 64   | 21.95                   | 12.94        | 0.534        | 0.846        | 0.119                  | 975.344        | 0.336        | 0.094        | 0.01405               | 1.80                  |

accuracy and has low average error metrics. In some cases the total-variation scheme tends to outperform the simple CBCS scheme, but at a higher computational cost.

The prior-art tend to perform overall quite poorly, although BCS-SPL<sub>4</sub> is relatively close in performance to CBCS and is sequentially the fastest CS framework. However, CBCS is highly parallelisable, making it a faster option.

The overall best performer is  $d\text{Sparse}_4$ . This is not surprising as it has the most information available for reconstruction due to the slight oversampling scheme resulting in good reconstruction quality and the fastest possible reconstruction time due to its straightforward computation. However, this comes at the expense of longer sampling times shown later in Figures 5.25-5.26, which means CBCS<sub>4</sub>-DCT remains the fastest overall in terms of total frame time.

The detailed individual results for all 4 scenes are shown in Table 5.7 for the various frameworks and discussed in more detail.

The single-pixel approach (Howland [231]) can retrieve the short range scenes to some extent but fails for long range scenes. This is particular apparent in a selection of depth maps shown in Figure 5.22(b,g,l,q). The key failing seems to be due to the implicit masking assumption enforced by hard-thresholding in the wavelet domain, which aims to reduce the problem space to few relevant surfaces. This results in often large removals of the scene and excessive zeroing in non-smooth areas of the scene.

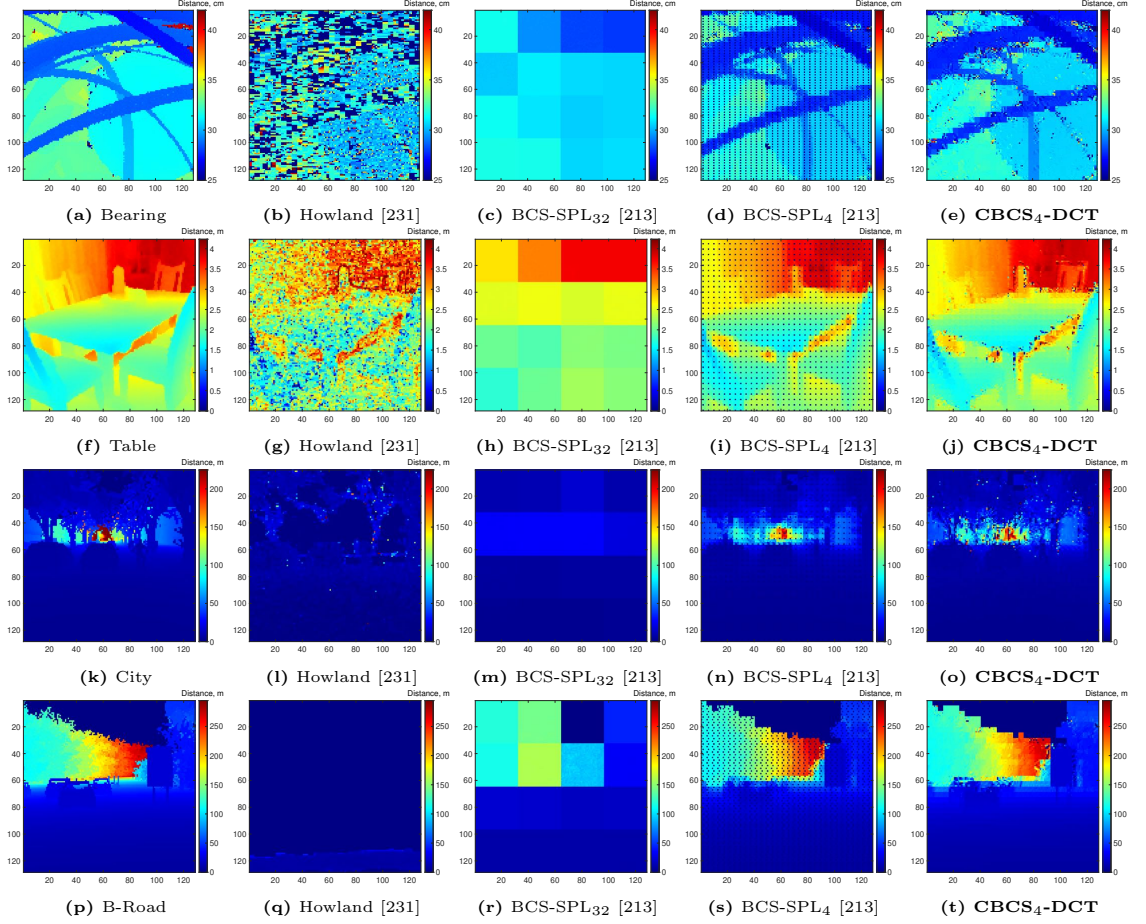
This explains the poor performance in the complex short range underwater scene with the ball bearing, as many angled objects are present. Further, not only is the number of measurements the highest. Although this can be reduced, it would likely reduce performance further. The processing time for the final image size in all cases of  $128 \times 128$  is 100s of seconds, in [82] smaller image sizes were considered for

**Table 5.7:** Depth compressive sensing comparison using various approaches broken down into various scenes of size  $128 \times 128$ . Ball bearing scene is real photon count data and the other data is synthetic with simulated photon counts and all are sampled in CS fashion using binary Gaussian random projection matrices with  $\Psi = 0.25$  at a compression rate of  $\mathcal{C} = 0.5$  for all compressive approaches and  $\mathcal{C}_{dS} = 1.5$ , best results are highlighted for best overall and best CS

| Scene   |                              | higher is better |               |              |              |              |              | lower is better |                |              |              |
|---------|------------------------------|------------------|---------------|--------------|--------------|--------------|--------------|-----------------|----------------|--------------|--------------|
|         |                              | PSNR<br>dB       | RSNR<br>dB    | SSIM         | $\delta_1$   | $\delta_2$   | $\delta_3$   | ARD             | MSE            | RMSE-LS      | MSE-LSI      |
| Bearing | Howland [231]                | 9.984            | 7.356         | 0.028        | 0.720        | 0.795        | 0.813        | 0.255           | 180.143        | 0.654        | 0.176        |
|         | BCS-SPL <sub>32</sub> [213]  | <b>27.092</b>    | <b>24.464</b> | 0.176        | <b>0.994</b> | <b>1.000</b> | <b>1.000</b> | <u>0.045</u>    | <u>3.417</u>   | <b>0.026</b> | <b>0.000</b> |
|         | BCS-SPL <sub>4</sub> [213]   | 19.266           | 16.638        | <u>0.251</u> | 0.942        | 0.946        | 0.946        | 0.085           | 53.422         | 0.272        | 0.052        |
|         | CBBCS <sub>4</sub> -DWT      | 7.433            | 4.804         | 0.014        | 0.644        | 0.657        | 0.660        | 0.367           | 325.789        | 0.857        | 0.236        |
|         | CBBCS <sub>4</sub> -DCT      | 21.832           | 19.204        | 0.211        | 0.974        | 0.989        | 0.991        | 0.047           | 11.575         | 0.128        | 0.008        |
|         | CBBCS <sub>4</sub> -TV       | 14.226           | 11.598        | 0.061        | 0.917        | 0.931        | 0.933        | 0.103           | 66.196         | 0.377        | 0.065        |
|         | CBBCS <sub>4</sub> -TV-TVp   | 26.057           | 23.429        | 0.171        | 0.992        | <b>1.000</b> | <b>1.000</b> | 0.051           | 4.496          | 0.029        | <b>0.000</b> |
|         | CBBCS <sub>4</sub> -DCT-TVTV | 16.351           | 13.723        | 0.106        | 0.880        | 0.930        | 0.948        | 0.105           | 41.475         | 0.237        | 0.026        |
|         | dSparse <sub>4</sub>         | 20.461           | 17.832        | 0.147        | 0.965        | 0.983        | 0.986        | 0.053           | 15.767         | 0.158        | 0.012        |
|         | dSparse <sub>8</sub>         | 19.594           | 16.966        | 0.216        | 0.954        | 0.977        | 0.982        | 0.057           | 19.214         | 0.178        | 0.015        |
|         | dSparse <sub>16</sub>        | 22.195           | 19.566        | <b>0.339</b> | 0.977        | 0.989        | 0.992        | <b>0.041</b>    | 10.566         | 0.121        | 0.007        |
| Table   | Howland [231]                | 15.758           | 11.672        | 0.084        | 0.654        | 0.876        | 0.936        | 0.210           | 0.460          | 0.606        | 0.193        |
|         | BCS-SPL <sub>32</sub> [213]  | 21.124           | 17.038        | 0.300        | 0.859        | 0.986        | <u>1.000</u> | 0.113           | 0.132          | 0.066        | 0.002        |
|         | BCS-SPL <sub>4</sub> [213]   | 14.243           | 10.157        | 0.042        | 0.890        | 0.900        | 0.900        | 0.122           | 0.677          | 0.126        | 0.007        |
|         | CBBCS <sub>4</sub> -DWT      | 11.390           | 7.303         | 0.034        | 0.782        | 0.806        | 0.814        | 0.216           | 1.262          | 0.192        | 0.016        |
|         | CBBCS <sub>4</sub> -DCT      | 25.008           | 20.922        | <u>0.582</u> | 0.972        | 0.991        | 0.994        | <u>0.037</u>    | 0.056          | 0.049        | <u>0.001</u> |
|         | CBBCS <sub>4</sub> -TV       | 26.216           | 22.130        | 0.475        | 0.974        | 0.994        | 0.997        | 0.038           | 0.043          | 0.038        | 0.001        |
|         | CBBCS <sub>4</sub> -TV-TVp   | <u>26.569</u>    | <u>22.483</u> | 0.544        | <u>0.978</u> | <u>0.999</u> | <u>1.000</u> | 0.061           | <u>0.039</u>   | 0.036        | 0.001        |
|         | CBBCS <sub>4</sub> -DCT-TVTV | 23.912           | 19.826        | 0.530        | <u>0.966</u> | 0.987        | 0.991        | 0.042           | 0.078          | 0.062        | 0.002        |
|         | dSparse <sub>4</sub>         | <b>36.427</b>    | <b>32.341</b> | <b>0.948</b> | <b>0.998</b> | <b>1.000</b> | <b>1.000</b> | <b>0.010</b>    | <b>0.004</b>   | <b>0.011</b> | <b>0.000</b> |
|         | dSparse <sub>8</sub>         | 34.513           | 30.427        | 0.912        | 0.996        | 1.000        | 1.000        | 0.014           | 0.006          | 0.016        | 0.000        |
|         | dSparse <sub>16</sub>        | 31.339           | 27.252        | 0.858        | 0.989        | 0.998        | 0.999        | 0.022           | 0.013          | 0.027        | 0.000        |
| City    | Howland [231]                | 17.943           | 0.062         | 0.066        | 0.264        | 0.402        | 0.472        | 0.599           | 811.260        | 1.068        | 0.453        |
|         | BCS-SPL <sub>32</sub> [213]  | 20.366           | 2.419         | 0.086        | 0.574        | 0.761        | 0.836        | 0.349           | 464.273        | 0.305        | 0.046        |
|         | BCS-SPL <sub>4</sub> [213]   | 23.695           | 5.748         | 0.115        | 0.740        | 0.804        | 0.837        | 0.230           | 216.574        | 0.364        | 0.066        |
|         | CBBCS <sub>4</sub> -DWT      | 20.281           | 2.334         | 0.083        | 0.577        | 0.635        | 0.664        | 0.433           | 484.632        | 0.567        | 0.138        |
|         | CBBCS <sub>4</sub> -DCT      | 22.658           | 4.711         | <u>0.389</u> | 0.775        | 0.850        | 0.886        | 0.217           | 276.210        | 0.300        | 0.045        |
|         | CBBCS <sub>4</sub> -TV       | <u>24.750</u>    | <u>6.803</u>  | 0.266        | <u>0.776</u> | <u>0.865</u> | <u>0.904</u> | <u>0.205</u>    | <u>169.403</u> | <u>0.240</u> | <u>0.028</u> |
|         | CBBCS <sub>4</sub> -TV-TVp   | 24.507           | 6.561         | 0.374        | 0.333        | 0.632        | 0.803        | 0.492           | 178.990        | 0.311        | 0.035        |
|         | CBBCS <sub>4</sub> -DCT-TVTV | 19.681           | 1.734         | 0.236        | 0.670        | 0.782        | 0.834        | 0.390           | 555.765        | 0.357        | 0.064        |
|         | dSparse <sub>4</sub>         | <b>26.315</b>    | <b>8.368</b>  | <b>0.753</b> | <b>0.861</b> | <b>0.912</b> | <b>0.933</b> | <b>0.098</b>    | <b>118.072</b> | <b>0.231</b> | <b>0.026</b> |
|         | dSparse <sub>8</sub>         | 23.260           | 5.313         | 0.646        | 0.789        | 0.855        | 0.885        | 0.170           | 238.935        | 0.343        | 0.059        |
|         | dSparse <sub>16</sub>        | 20.145           | 2.198         | 0.538        | 0.698        | 0.773        | 0.808        | 0.282           | 488.565        | 0.460        | 0.103        |
| B-Road  | Howland [231]                | 11.366           | 0.005         | 0.100        | 0.096        | 0.102        | 0.105        | 0.759           | 6317.606       | 1.422        | 0.315        |
|         | BCS-SPL <sub>32</sub> [213]  | 15.400           | 4.037         | 0.091        | 0.569        | 0.728        | 0.762        | 0.228           | 2495.500       | 0.629        | 0.187        |
|         | BCS-SPL <sub>4</sub> [213]   | 17.520           | 6.156         | 0.220        | 0.698        | 0.718        | 0.726        | 0.163           | 1636.599       | 0.619        | 0.188        |
|         | CBBCS <sub>4</sub> -DWT      | 16.892           | 5.528         | 0.220        | 0.640        | 0.669        | 0.682        | 0.228           | 1886.865       | 0.661        | 0.208        |
|         | CBBCS <sub>4</sub> -DCT      | 19.288           | 7.924         | <u>0.550</u> | <u>0.804</u> | <u>0.828</u> | <u>0.836</u> | <u>0.062</u>    | 1019.846       | 0.398        | 0.077        |
|         | CBBCS <sub>4</sub> -TV       | <b>21.038</b>    | <b>9.674</b>  | 0.333        | 0.774        | 0.814        | 0.829        | 0.089           | <b>681.578</b> | <b>0.328</b> | <b>0.052</b> |
|         | CBBCS <sub>4</sub> -TV-TVp   | 17.995           | 6.631         | 0.394        | 0.702        | 0.804        | 0.835        | 0.137           | 1434.618       | 0.734        | 0.222        |
|         | CBBCS <sub>4</sub> -DCT-TVTV | 15.183           | 3.819         | 0.341        | 0.725        | 0.790        | 0.812        | 0.121           | 3175.652       | 0.617        | 0.192        |
|         | dSparse <sub>4</sub>         | 19.920           | 8.556         | <b>0.782</b> | <b>0.830</b> | <b>0.842</b> | <b>0.847</b> | <b>0.024</b>    | 882.045        | 0.384        | 0.072        |
|         | dSparse <sub>8</sub>         | 16.371           | 5.007         | 0.607        | 0.785        | 0.810        | 0.820        | 0.064           | 1996.334       | 0.564        | 0.156        |
|         | dSparse <sub>16</sub>        | 14.114           | 2.751         | 0.400        | 0.718        | 0.765        | 0.784        | 0.131           | 3402.231       | 0.737        | 0.266        |

this framework but still requiring several 10s of seconds to reconstruct depth from compressive measurements. This makes it unsuitable to perform reconstruction in real-time even if reconstruction quality would be better.

Next, the state-of-the art in blocked compressive sensing (BCS-SPL) with the optimal sizing described in [129, 213] with a block size of  $N_B = 32$  completely fails for the problem at hand despite the metrics indicating otherwise, the visual results show dramatic blocking effects where all detail is lost Figure 5.22(c,h,m,r). This could be due to the division of the two proxy results, or the change in projection



**Figure 5.22:** Depth compressive sensing comparison for compressive depth recovery schemes

matrix from dense random projections to sparse random binary patterns and the application to photon count data, but it should be noted that we see dramatic speed improvements from the single-pixel approach to about 1.7 s per depth recovery.

With the smaller block size of  $N_B = 4$ , the approach performs much better than before but still produces dramatic block artefacts Figure 5.22(d,i,n,s), which may appear constant, but vary with every computation and are not basis related. Further, each tile is still only a mean value i.e. lacks detail. In terms of processing speed it is the fastest sequential compressive approach at around 0.5 s. And despite having a per block stage, which could be parallelised, it is interlaced with full frame steps which can not be readily parallelised to the same extent as the presented CBCS and *d*Sparse frameworks.

For CBCS the best results are obtained using the DCT basis transform, it outperforms both DWT and TV transforms in terms of quality and processing speed in most cases. The sequential times are about 1 – 2 s but this approach can be readily parallelised and the slowest block reconstruction is always faster than 1 ms making it



extremely fast with theoretical processing rates of up to 1 kHz. Running the reconstruction without any optimization and paging through the blocks on a GPU results in real frame rates of  $> 5 \text{ Hz}$ , which demonstrates the bottleneck introduced by the large number of blocks for a single processing device approach due to communication overhead.

To exploit the full potential, a specialised processing pipeline is required, which is proposed in Section 5.5.3. The visual quality is also the best out of all compressive schemes Figure 5.22(e,j,o,t).

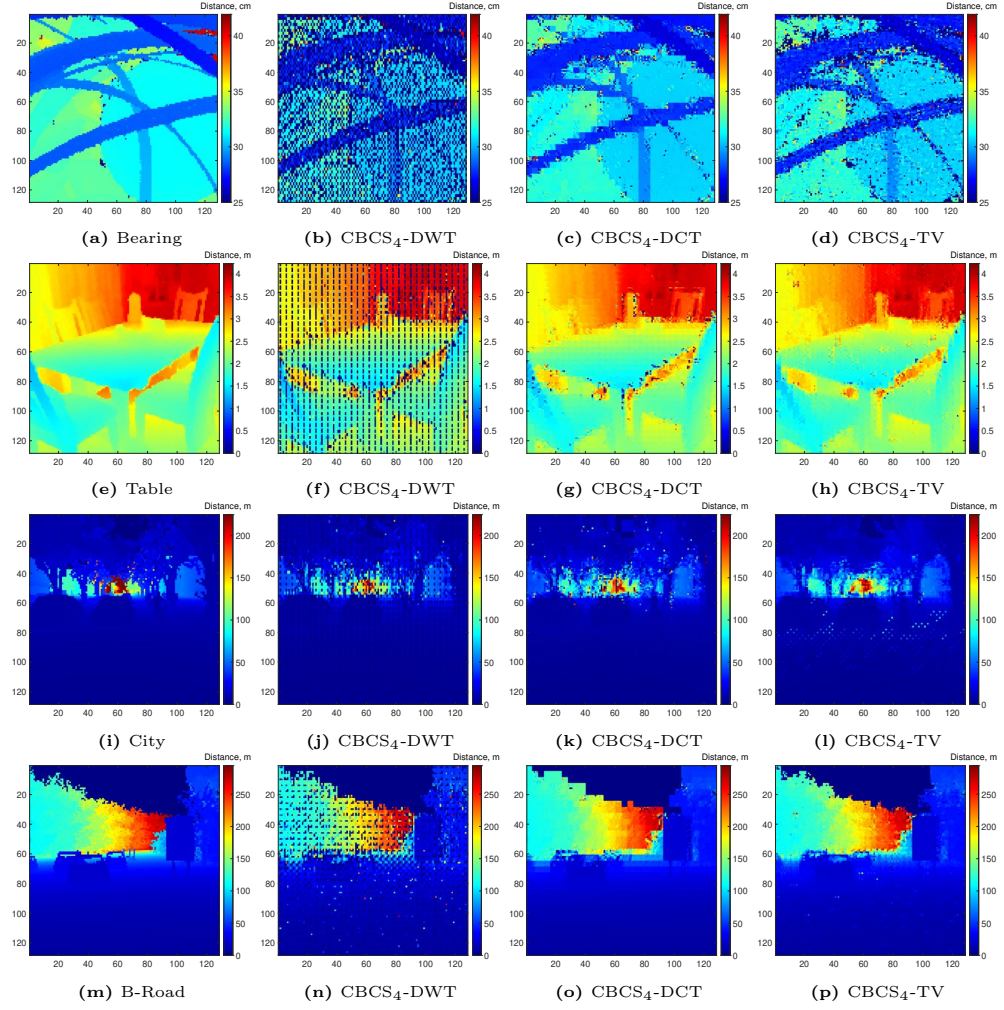
The other basis functions for CBCS perform better than the single-pixel approach and BCS-SPL in most cases, although it should be noted that DWT performs quite poorly overall in these particular operating conditions, despite its ability to perform well in many other CS applications. As already discussed CBCS has scope to easily substitute the basis transform in the reconstruction, applied to the same raw measurements.

CBCS-TV in terms of quality metrics performs slightly worse than CBCS-DCT as expected from the sparsity analysis, as TV should generally favour a lower compression rate closer to 1 (see Figure 5.15) for this block size. Upon visual inspection in Figure 5.23, TV actually produces overall good reconstructions with less blocking artefacts at large step edges.

The CBCS-DWT results demonstrate that an inappropriate basis for the block size results in major blocking artefacts similar to results from BCS-SPL<sub>4</sub>. CBCS-DCT is also not free of blocking artefacts, but they are very minimal overall and mostly present at step edges.

In the longer range scenes, some ringing effects can be observed where a lot of depth variation is present in a single block e.g in Figure 5.23(o). This was expected and the reason why TV extension were proposed, in equations (5.37) and (5.38) respectively, to take advantage of TV minimisation globally, which should increase smoothness in the final reconstruction result.

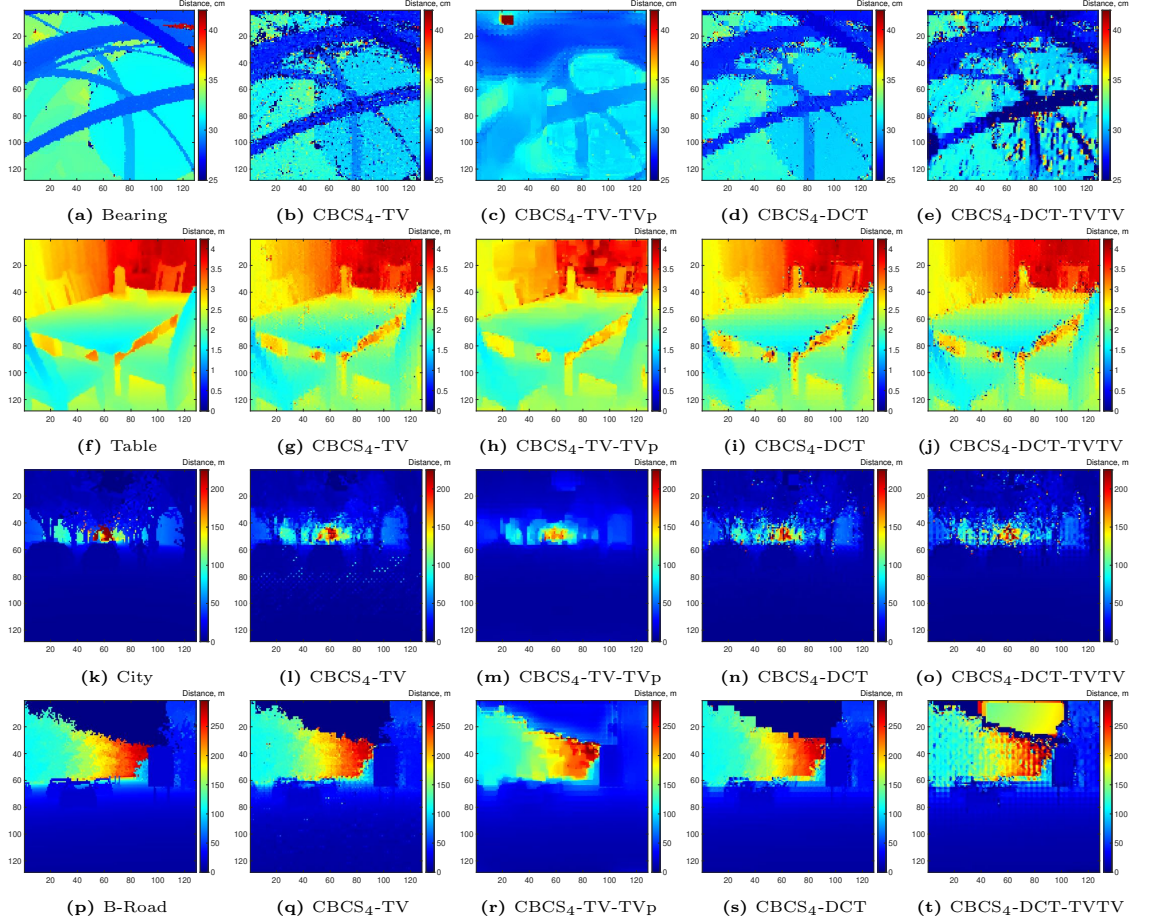
In particular the CBCS-TV-TVp approach practically eliminates all blocking artefacts but does so by aggressively averaging sections of images, resulting in loss of detail as shown in Figure 5.24. It also increases processing times, as the full-scale update step is quite expensive. In terms of total sequential time it only adds about 1 s but the minimum parallel execution time increases from a few ms to one second. A more efficient TV minimization algorithm may be able to reduce the time penalty.



**Figure 5.23:** Comparison of CBCS approaches with different basis transforms across the considered scenes.

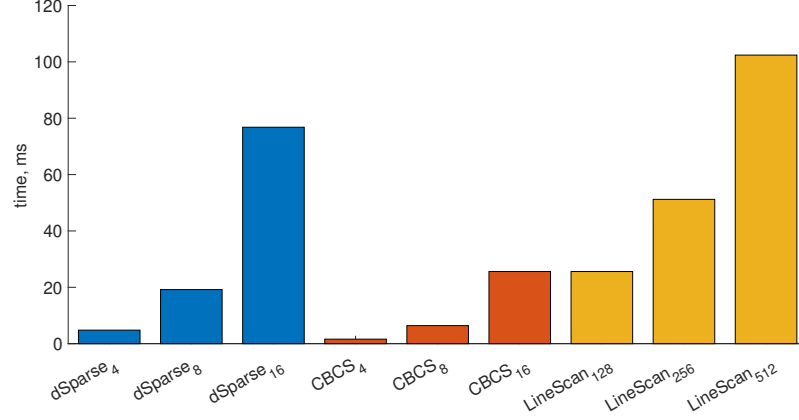
The TVTV approach presented in [246] using the all information available from block sampling using the block diagonal formulation of the sensing matrix and the measurement vector ensemble as with most other BCS approaches, shows promising results. It seems to improve some parts of the recovery for example the top leaf in Figure 5.24(e) but introduces new artefacts in other cases. It shows potential but even if the results can be dramatically improved it is unsuitable for real-time reconstructions.

All the above approaches are compressive approaches, but since the block size has become very small, the problem size becomes suitable to be oversampled without a severe sampling time penalty illustrated in Figure 5.25. The exposure time is set to  $200 \mu\text{s}$ , i.e 100 laser pulses imaging up to 300 m.



**Figure 5.24:** Comparison of CBCS TV extensions. CBCS-TV-TVp approach using CBCS-TV as a starting point and a cumulative measurement vector and global pattern to reconstruct a full scale problem using TV regularisation overall reduces block artefacts but averages out detail in the scene. Meanwhile the block diagonal approach and a CBCS-DCT prior in a TVTV approach seems to improve some areas of reconstruction while also introducing new errors and artefacts.

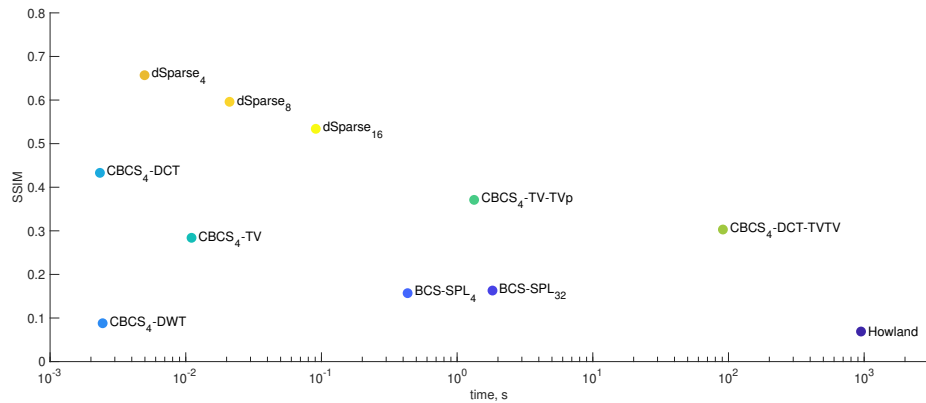
The bar chart in Figure 5.25 highlights that for  $N_B = 4$  and  $N_B = 8$  the total sample time is lower for an equivalent line scan for a  $128 \times 128$  pixel LiDAR. It is also obvious that for any larger block-sizes this trade-off does not necessarily make sense, as sampling times increase dramatically for  $N_B > 8$  for  $\mathcal{C} = 1.5$  in this case. Both  $d$ Sparse and CBCS benefit from a constant sampling time as the number of LiDAR pixels increases, which can not be said for normal line scanning systems, which scales linearly.



**Figure 5.25:** Sample time comparison between compressive schemes and linear line scans. For  $\eta_P = 1$  pattern exposure cycles and  $n_P = 100$  laser pulses per line and pattern exposure. Both  $d$ Sparse and CBCS schemes remain constant for chosen block size, while the line scan approach scales up as image size increases.

All considered examples use histogram sizes of  $p > 1000$ , which is in the long range cases  $p = 7500$ . For this case the total compression (see equation 5.19) for the  $n = 4 \times 4$  with  $m = 8$  block cases is  $\mathcal{C}_{\text{CBCS}} = 0.0637$  and for the discrete least square approach with  $m = 24$  is  $\mathcal{C}_{\text{CBCS}} = 0.0661$ . This demonstrates a key advantage of this and other spatial compressive depth approaches, as it allows to reduce memory requirements significantly, while the approach presented in this work also increases the depth reconstruction performance compared to other compressive frameworks.

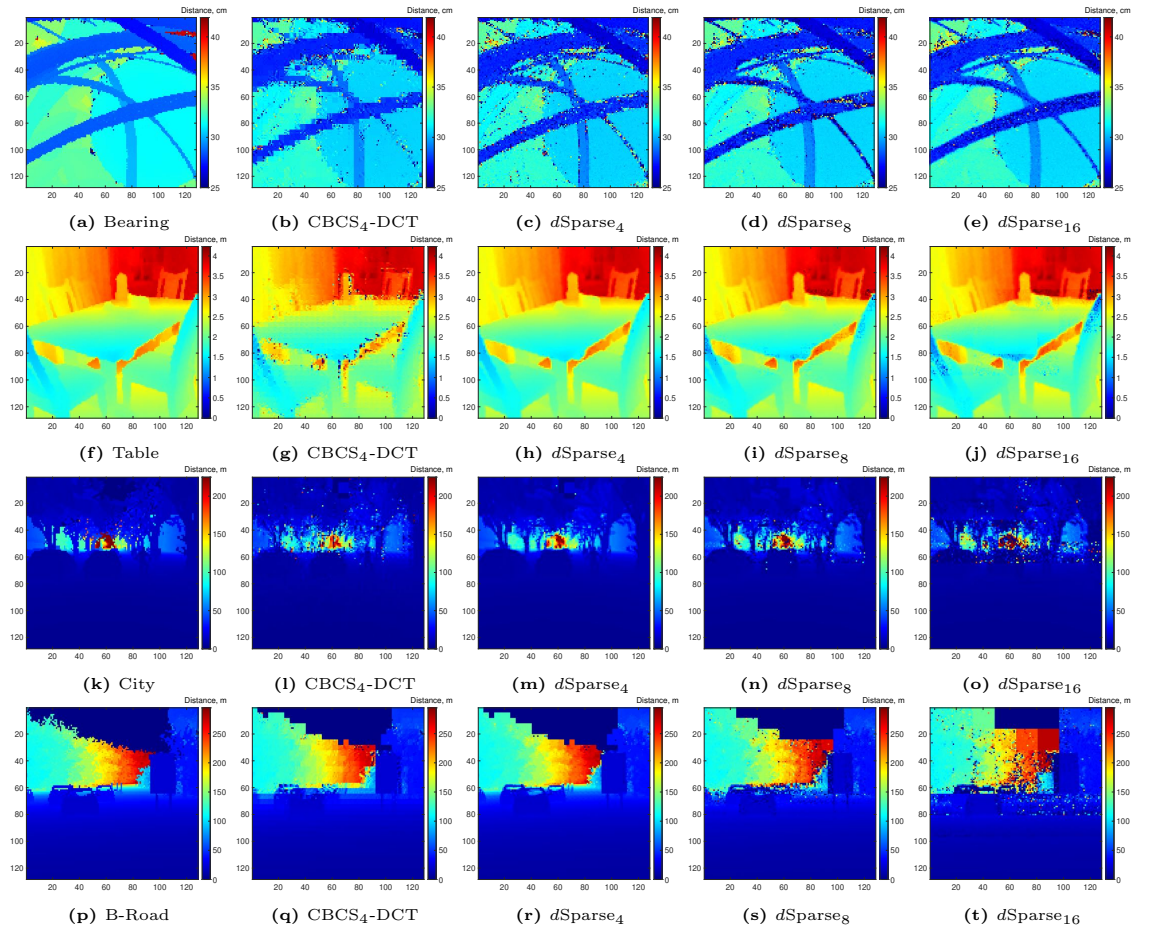
To illustrate this further, a quality metric is compared to the total frame time for all considered approaches and is presented in Figure 5.26. It is clear from this comparison, that this work's contributions outperform prior art both in quality and speed.



**Figure 5.26:** Quality (SSIM) and frame time comparison for compressive depth reconstruction. The frame time is sample time and total processing time combined. For CBCS and  $d$ Sparse processing is assumed to be parallel.

This figure shows the potential for frame rates well above 200 Hz for CBCS<sub>4</sub>-DCT and above 100 Hz for  $d\text{Sparse}_4$ , while outperforming all prior approaches in depth reconstruction quality.

The results for  $d\text{Sparse}$  with  $\mathcal{C} = 1.5$  and the various block sizes shown in Figure 5.27 are very good and also have by far the fastest reconstruction times at about 0.2 ms and even a fast sequential processing times of only 0.35 s for  $N_B = 4$ . The reconstruction quality is overall excellent, but blocking artefacts become more apparent as block-size increases and reconstructions become more noisy. With a  $4 \times 4$  block-size reconstruction is almost ideal with very little blocking other than the extreme step edges in the B-Road scene from tree foliage to 0 m (sky).



**Figure 5.27:** Depth sparse random sensing using a discrete pseudo-inverse  $d\text{Sparse}$  performs extremely well for  $N_B = 4$ , while as block size increases blocking artefacts and noise appears.

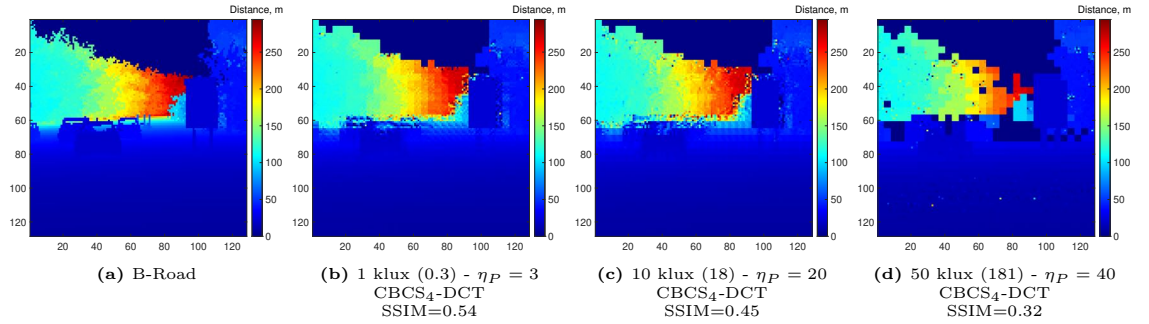
$d\text{Sparse}$  is clearly a very practical approach to reduce illumination density and sampling time in a very efficient fashion but at the cost of additional sampling time. There are benefits to this, which may not be directly obvious. However, if the same sparse random illumination was to be used to sample the entire scene pixel by pixel,



the cost of memory and processing of individual histograms for each pixel would be still dramatically higher than for the sparse reconstruction approaches presented in this chapter.

Further, CBCS and *d*Sparse can sample the scene more rapidly, which in turn can allow for longer exposure times if required e.g. in high noise scenarios or in general improve SNR, with the compressive scheme allowing for the longest exposure times. It further means that the entirety of the scene is sampled only marginally slower than a flash illumination. Meanwhile line-scans as resolution increases could introduce dramatic motion artefacts due to the rolling shutter type imaging issues in this case.

In terms of noise, the framework can maintain sufficient reconstruction quality for up to 50 klux with modest increases in exposure time illustrated in Figure 5.28.



**Figure 5.28:** Noise comparison of sparse block depth sensing schemes. A single exposure pattern exposure  $\eta_P$  is equivalent to a pattern sequence sample time of 1.6 ms. Values in brackets after illuminance indicate mean ambient photon count per bin.

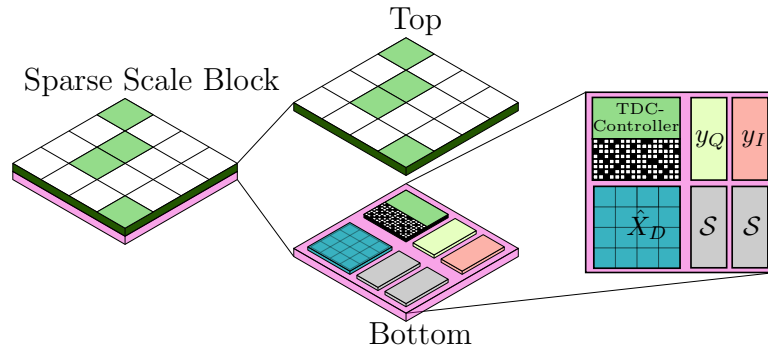
For longer exposures the reconstruction quality is likely to increase. Further the noise compensation scheme could be improved or even replaced with a peak finding algorithm such as LiDARNet in Chapter 4 to only process surface locations associated with a projection pattern. Regardless, the framework can deal with noise, as long as exposure and/or laser power is sufficiently adjusted, which is a common compromise for LiDAR systems.

The results in Table 5.6 show that the proposed frameworks can theoretically reach extremely high processing rates of  $> 1$  kHz and certainly camera compatible frame rates of  $> 30$  Hz with long exposure times enabling robust performance even with high ambient photon counts from the sun or indeed other LiDAR systems. The framework works well in all presented scenarios, without any obvious degradation even at extremely long ranges. The reduction in quality metrics is often due to the range error scaling as range increases. However, due to the extreme parallelism it is not straightforward to exploit this approach to the full potential without some dedicated hardware as standard multi-core CPU systems require a lot communication

to schedule and gather all blocks.

### 5.5.3 Scalable Imaging Arrays

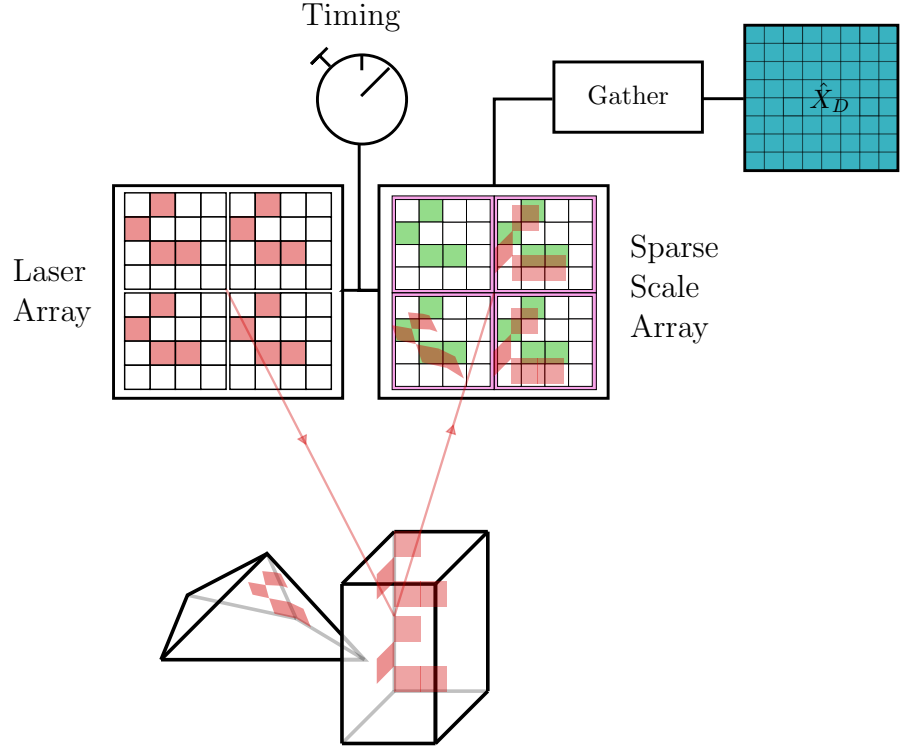
The compressive and sparse sampling and processing methodology proposed in this Chapter enables a plethora of parallelism exploitations, which in turn can enable extremely fast frame times. However, to fully take advantage of the parallelism, ideally the sampling and processing scheme is accompanied with dedicated hardware to facilitate the parallelism. As each block is treated independently in normal CBCS (without any TV extensions), no communication between blocks is required for processing. This makes the framework suitable to integrate logic into a single block and scale up to large imaging arrays. An integration proposal for such a detector block is shown in Figure 5.29.



**Figure 5.29:** An integrated Sparse Scale Block, performing structured sensing with sampling logic as well as sparse reconstruction integrated into an independent block device. Top layer consists of programmable photon detectors, controlled by the bottom layer, which contains all necessary components to perform depth reconstruction by sparse reconstruction,  $\mathcal{S}$  (CBCS or  $d$ Sparse).

Here, the detector block device is made up of two stackable layers. The top layer is a photon sensitive detector array or in its simplest form a single photon sensitive pixel. The bottom layer houses the sampling logic and pattern control, which can both address the pixel array and the emitter array as shown in Figure 5.30. Each block has storage vectors for both proxy quantities and dedicated logic to solve the sparse reconstruction for either, which then forms the block solution for depth. This is further aided by the extremely small problem size, which keeps memory requirements and bandwidth to a minimum.

It is assumed that it is possible to fit this logic under the detector block size. This means that high resolution arrays can be realised by distributing multiple blocks with an associated emitter device capable of structured illumination. The sizing and resource logistics are to be further investigated in Chapter 6.



**Figure 5.30:** Scalable Sparse Depth Imaging Array using integrated Sparse Scale Blocks with integrated sampling and reconstruction logic to exploit the parallelism enabled by the methodology outlined in this work. Illustrated is a small  $2 \times 2$  Sparse Scale array, but the architecture can scale readily to many more Sparse Scale blocks for higher resolutions.

Such a system could provide an efficient way for large scale high resolution imagers exploiting the parallelism enabled by the proposed framework. Further, because the sampling rate and processing times are block defined, the final image resolution have little effect on frame time.

## 5.6 Summary and conclusions

This chapter has introduced the concepts of compressive sensing (CS) and its applications to depth sensing exploiting sparsity in time and space. While the key advantage of CS is the reduction in sampling bandwidth and more compact measurement storage, it does so by increasing the processing cost. As resolution increases, this processing cost is prohibitive to real-time applications such as autonomous driving, where fast scene mapping, object recognition and decision making are required. The concept of blocking schemes for CS was introduced which reduce the computational burden and memory cost of structured illumination and reconstruction. However, these schemes employ full-scale steps in their reconstruction to reduce block artefacts, limiting scope for parallelism dramatically. The depth CS approaches in the



past have heavily relied on scene constraints to reduce the problem complexity, which limits their use to short range and simple scenes. In particular the direct exploitation of time or depth sparsity often results in a large problem formulation with very large problem spaces, as processing is performed on each time slice. The exploitation of spatial sparsity for depth is by contrast more efficient but required major constraints on the number of retrievable surfaces in the scene.

By addressing the above issues, this thesis makes the following contributions:

- A signal formulation for small scale compressive depth reconstruction
- An independent blocked compressive depth recovery framework for high-resolution and long range LiDAR applications at high frame rates and
- A system architecture proposal to exploit independent blocks for high-resolution sparse LiDAR systems.

More specifically, the small block size enables the assumption of few surfaces for a single block problem, while not adversely affecting the reconstruction of large scale depth scenes with multiple blocks. As each block is independent, it can also be processed independently and uses two proxy quantities, the depth sum and photon count for a given projection pattern, which can also be reconstructed in parallel. This makes it possible to reconstruct depth in a compressive fashion in a matter of milliseconds. The reconstruction quality of this methodology performs well using the discrete cosine transform (DCT) for a sparse representation, but is also modular in the sense that other basis transforms can be readily used if more suitable for the application. While this approach in theory can provide efficient and extremely fast sampling and reconstruction times, it is not necessarily straightforward to implement such massive parallelism with off-the-shelf components and multi-core systems. A system architecture to address this, was also presented to showcase an efficient way to enable fast and efficient high resolution 3D LiDAR solid-state imaging using independent block imagers with integrated logic further explored in Chapter 6.

Due to the small problem size within each block, the sample time is dramatically reduced in comparison to linear scans, and also versus other proposed compressive schemes. This may allow the system to operate in a slight oversampling mode, which in turn solves the inverse problem with a discrete least square approximation step. This framework using sparse random reconstruction, *dSparse*, further showcases an extremely fast and efficient mode of this framework but at the cost of more pattern exposures. However, the quality of reconstruction generally improves markedly, making this trade-off a worthwhile consideration.

This work demonstrates a proof-of-concept for ultra-fast 3D imaging at high resolution suitable for autonomous driving and other complex tasks. Good performance was demonstrated for high precision in both short range and long range scenarios. Although the current operational parameters yield good results, the question of optimal projection patterns is an open question in the CS field and could also enhance the performance of this framework. In particular a structured projected pattern may be designed by means of machine learning taking randomness into consideration but introducing some structure to aid reconstruction. This direction is certainly worthy of future investigations. Another key component of CS work is the chosen basis function. Since the framework allows for linear transforms to be plug-and-play, a more detailed investigation into the optimal basis function for small scale blocks might yield improved performance. Finally, the system design provides many opportunities for future work, both in terms of logic reduction, which is investigated in the next chapter, as well as dedicated sampling and processing logic, which would have implications on sample times, e.g. if the TDC can capture more events in a single exposure with a high-dynamic range and dedicated application specific integrated circuits (ASIC) to enable the small package proposed in this work.

# Chapter 6

## Small Footprint Sparse Sensing Logic

### 6.1 Introduction

The sparse depth imaging approach presented in Chapter 5 requires a parallel hardware implementation to take full advantage of the proposed architecture. It has been demonstrated that independent blocks can recover a scene, which suggests that the processing can also be done on a per block basis. This mitigates the overhead introduced by parallel computing such as task scheduling, data distribution and gathering of results.

However, thus far all computations and simulations have been performed on powerful general compute hardware in high-level processing languages, which often hide complexity and memory bottlenecks due to the abundance of resources. In this particular case, the main constraints are physical size and power. The size is limited by the detector array size for a particular imaging block, as the proposed system-on-a-chip (SoC) solution is a two-layered stacked approach, while power should always be kept to a minimum for efficiency and in this case to further prevent thermal effects, which can cause false photon events and decrease the signal-to-noise ratio (SNR) [255].

There are many more resource constrained applications in particular for battery powered systems such as drones, electric cars and mobile phones. This calls for strategies to optimise a processing pipeline. One branch of these strategies is approximate computing (AC) [256, 257, 258]. One added benefit to approximations is that they can also speed up overall processing, since algorithm approximations may allow fewer processing steps and smaller numerical representations reduce read and write

operations, further improving efficiency of the overall system.

This chapter introduces a common approximate computing technique being precision scaling alongside a brief overview of other branches of the field. This technique is applied to the sparse depth recovery problem presented in Chapter 5 for two configurations: The compressive sensing case with an un-rolled version of the optimisation algorithm alternating direction method of multipliers (ADMM) [237]) and the discrete sparse depth recovery using a discrete least square solution, *dSparse*. This work expands upon the initial findings presented in [239], where a resource and power analysis was performed in the context of reconstruction quality, where the field-programmable gate array (FPGA) synthesis was performed by a collaborator, who also provided a custom data type library for C. Although, the initial findings demonstrated that  $\ell$ ADMM and *dSparse* can tolerate precision scaling with minimal quality loss, it lacked a detailed analysis of the respective precision boundaries.

This thesis therefore provides a formal analysis of expected minimum precision in the context of quality degradation alongside a thorough analysis of breaking points for both approaches when precision scaling is applied. Further, extended resource requirements from [239] are translated into estimates for application specific integrated circuit (ASIC) logic core implementation size with comparisons to real single photon avalanche detector (SPAD) array sizes appropriate for the imaging block for a full block system implementation, finding that the considered sparse depth frameworks' complexity are appropriate for SoC systems constrained by the detector array size.

## 6.2 Approximate Computing

The demand for ever more capable computing systems is growing, initially supported by a density increase of transistors driven by decreasing manufacturing nodes as small as 5 nm [259]. Unfortunately, transistor sizes can only shrink so much, at least for silicon semi-conductors, due to physical limits [260, 261, 262]. To enable future advancements of computing systems when this limit is reached, other techniques have to be employed to increase functional density by minimising the complexity of circuits, rather than relying on the continued miniaturisation of transistors.

To address the above issues a new paradigm has emerged, *approximate computing* (AC), which utilises approximations across algorithms and circuits to reduce the number of transistors for arithmetic operations and by finding the smallest possible numerical representations, the memory required to store information can also be significantly reduced.

A notable approximation technique is loop perforation, which aims to streamline aspects of a particular iteration of a loop or altogether jump ahead in the iteration flow. This can aid execution times and decreases the number of operations at runtime and thus reduces power consumption [263, 258, 264, 265].

Many applications have a well defined precision for their inputs and outputs, which are often lower than the standard number precision found in general purpose computing; the double precision floating point precision popularised by the widespread use of 64 bit processing architectures for general compute hardware in recent years. The reduction of bit width means that fewer bit operations are required, this reduces the size of shift registers, arithmetic units, memory and many if not all connected components. This approximation technique is called *precision scaling* and has been demonstrated successfully for a wide range of applications [266, 267, 268, 264, 269, 258].

Memory in particular is usually a costly component in image and signal processing and could further benefit from optimised memory allocation techniques [270] if precision scaling is not sufficient to reduce the overall logic core size or memoization techniques which enable efficient usage of recurrent data [271, 272]. Approximate computing is a very active area of research [273, 265, 274, 275] with many more branches and techniques under investigation, however, this work only considers precision scaling in the first instance to assess potential savings and performance when approximations are made. If the analysis carried out in this work would not achieve a sufficient size reduction, additional targeted approximations could be made for example in the form of inexact hardware e.g. [276, 277].

Addressing the applications considered in this work, solid-state time-of-flight (ToF) imaging is resource constrained in two ways in the context of an integrated stacked processing solution. First, the size of the detector array limits the size of the logic to be the same size or less. Second, photon detector arrays are sensitive to thermal energy and SPAD detectors can trigger if subjected to excess heat and can cause false photon counts [255]. It is therefore advantageous to reduce power consumption by improving the efficiency of any logic placed underneath a photon counting device. It was shown in [239] that resources can be significantly reduced for the proposed system resulting in power draw reductions of above 70% versus full double floating point precision.

This chapter is primarily concerned with fitting a moderately complex optimisation framework and a discrete approximate method in the same space occupied by a single photon detector array. Precision scaling is applied to the processing methodologies proposed in Chapter 5 to assesses the feasibility of small scale sparse imaging

blocks. The initial findings presented in [239] are expanded upon and a more detailed analysis of the experiment is provided. Additionally, a case study is presented to determine if the proposed architecture can fit under a modern SPAD array [58] using the aforementioned precision scaling techniques.

### 6.2.1 Reduced Precision

The goal of reducing precision is to find the smallest possible bit width configuration for a given numerical data type to improve power and resource efficiency with a specific application in mind. The concept of precision scaling generally involves the reduction of bit width required to represent any value throughout processing stacks without any loss to final precision or with loss up to an acceptable level. For image reconstruction a SNR of above 30 dB is often considered of good enough [258] and any marginal gains in quality might incur a disproportional increase in resources.

This work will consider precision scaling for two common number formats, the floating point [278] using the FloatX library [279] and the fixed point format [280], with illustrations of both provided in Figure 6.1.

$$\begin{array}{c}
 \begin{array}{ccc}
 \pm \underbrace{10}_{2^1 \times 2^{\text{Exponent}}} \times & \underbrace{123456789}_{2^{\text{Significand}}} & \rightarrow \overbrace{12345.6789}^{\text{Mantissa}} \\
 & & \underbrace{\hspace{1.5cm}}_{10^{-4} = \text{Exponent}}
 \end{array} \\
 \text{(a) Floating Point (FP) [278]} \\
 \\
 \begin{array}{ccc}
 \pm & \underbrace{12345}_{2^1 \times 2^{\text{Integer}}} & \cdot \underbrace{6789}_{2^{\text{Fraction}}}
 \end{array} \\
 \text{(b) Fixed Point (FXP)}
 \end{array}$$

**Figure 6.1:** Illustration of floating and fixed point number formats.

Many digital signal processing (DSP) applications use floating point (FP) representations as it offers more flexibility due to the floating nature of the decimal point and, if properly optimised, the operational complexity can be reduced significantly by reducing shift operations [266, 268]. It is straightforward to switch between large numbers with low precision and small numbers with high precision i.e. to allow for more significant figures. The principal structure is illustrated above in Figure 6.1(a). The main component in the decimal representation is called the *Mantissa*, which is an integer number. To obtain the significant figures, this is accompanied with an exponent, which shifts the decimal point according to the exponent value. The bits required to encode this form the *Exponent*. For the *Mantissa* the bits form the

*Significand* [278]. The total bit width of a floating point number is

$$b_{\text{FP}} = \log_2(2^{\text{Sign}} \times 2^{\text{Exponent}} \times 2^{\text{Mantissa}}). \quad (6.1)$$

The fixed point (FXP) number representation has an implied stationary decimal point with two separate integer values encoding the *Integer* part of the number and another integer for the significant figures or the *Fraction*. This is useful when the value range and total precision is known a priori and remains constant [258]. The total bit width for a fixed point number is

$$b_{\text{FXP}} = \log_2(2^{\text{Sign}} \times 2^{\text{Integer}} \times 2^{\text{Fraction}}). \quad (6.2)$$

By minimising the bit width for either representation, several components, in particular memory, will reduce in size, arithmetic becomes less expensive as bit width decreases and thus allows for more compact and faster logic designs. This work provides a theoretical estimate based on system specifications. These are compared to an empirical study of precision breaking points with respect to depth reconstruction quality. Further, efficiencies afforded by approximations made to the iterative algorithm used to solve the compressive inverse problem are quantified.

## 6.3 Algorithm Hardware Optimisations

The majority of the work carried out in this work thus far was performed in software packages for advanced arithmetic such as Matlab and Python with feature-rich libraries. They are ideal for fast prototyping of many complex tasks, as they allow for a plethora of complex arithmetic operations to be performed in a few lines of code on general purpose computer hardware. However, those few lines of code can often hide a large number of sub-routines performing various tasks which, when rolled out, can turn seemingly a simple operation into a costly task on embedded hardware, where memory and compute capabilities are very limited.

In order to make the presented algorithms for the proposed sparse depth reconstruction system architecture compatible with embedded systems to assess its feasibility as an ASIC component in the form of a SoC, the optimiser used in the relevant chapter has been rolled out and simplified to only allow for vector and matrix arithmetic without any nested sub-routines. This allows the algorithm to be readily implemented in C/C++, which can then be converted to a hardware description language (HDL) to assess its complexity on a FPGA platform.

### 6.3.1 Lean ADMM

The compressive sensing reconstructions considered in this work take the normal basis pursuit de-noising (BPDN) form and can be solved using many available optimisation algorithms for this popular problem type. After careful consideration in Chapter 5, ADMM was chosen as the main algorithm and its processing flow is shown in Algorithm 7 for a standard  $\ell_2\ell_1$  minimisation.

---

**Algorithm 7** General ADMM least absolute shrinkage and selection operator (*lasso*) [237] suitable for BPDN

---

**minimise**  $\frac{1}{2}\|y - Ax\|_2^2 + \lambda\|z\|_1$

**subject to**  $Fx - z \leq t$

*Iterate :*

- 1:  $x^{k+1} = (A^T A + \rho F^T F)^{-1}(A^T y + \rho F^T (z^k - u^k))$
  - 2:  $z^{k+1} = S_{\lambda/\rho}(Fx^{k+1} + u^k)$
  - 3:  $u^{k+1} = u^k + Fx^{k+1} - z^{k+1}$
- 

Here,  $z \in \mathbb{R}^n$  is the basis representation of  $x \in \mathbb{R}^n$  with a linear transform  $F$  and  $u \in \mathbb{R}^n$  are the intermediate solutions, while  $A$  is the sensing matrix in a compressive sensing (CS) problem and  $y$  the corresponding measurement vector. To enforce sparsity a soft-threshold operation  $S$  is used [281]. Parameters playing a critical role are the tuning parameters  $\lambda$  and  $\rho$ .  $\lambda$  in particular, setting the threshold for significant components in the sparse signal, is often derived from a set of measurements, while  $\rho$  is a constant tuning parameter scaling value changes per iteration. They interact to determine the sparsity threshold and thus play a major role in the final performance and convergence of the algorithm. The algorithm formulation suggests as many iterations,  $k$ , to satisfy the inequality with a set threshold,  $t$ .

Although the above algorithm formulation can be readily implemented in high-level software packages such as Matlab, many operations, in particular computing  $x^{k+1}$ , are challenging to perform on embedded systems from first principles and would require often costly additional packages such as Eigen [282] or custom libraries to perform matrix inversion and other advanced matrix arithmetic. While performing such operations at runtime can be useful for general purpose approaches, particular application specific considerations can provide meaningful simplifications of the algorithm.

Such complications are identified for ADMM in the context of checkerboard compressive sensing (CBCS). To further reduce arithmetic complexity, parameters are identified which can be readily pre-computed as they remain constant throughout the operation.



The simplifications made to the algorithm were first presented in [239] and the resulting algorithm variant was called  $\ell$ ADMM (short for lean ADMM) as shown in Algorithm 8. In this section, the considerations made to arrive at this more efficient optimiser will be expanded upon. The changes were made to the Matlab implementation of ADMM [237]. The resultant algorithm was then ported to C/C++ for FPGA synthesis as presented in [239], while the custom data formats were integrated collaboratively for [239]. The base Matlab implementation uses a more efficient LU-decomposition to replace the matrix inversion in Algorithm 7, but for embedded systems even a LU-decomposition can be considered expensive. Luckily, the majority of complex arithmetic operations are based around the sensing matrix,  $A \in \mathbb{R}^{m \times n}$  for an individual sparse imaging block, which is constant throughout the operation of a sparse depth imaging system as described in Chapter 5 for a pre-computed pattern sequence.

---

**Algorithm 8** *leanADMM lasso* for  $m < n$  [239]

---

**Input:**  $A, A^T y, L^{-1}, U^{-1}, y, \lambda$   
**Output:**  $x$

*Initialization* :  $\text{const}\{\alpha, \rho, \kappa = \lambda/\rho\}, \text{zeros}\{z, q, u\}$

- 1: **for**  $k = 0$  to  $k_{max}$  **do**
- 2:  $q^{k+1} = A^T y + \rho(z^k - u^k)$
- 3:  $x^{k+1} = q^{k+1}/\rho - 1/\rho^2 * A^T(U^{-1}(L^{-1}(Aq^{k+1})))$
- 4:  $\hat{x}^{k+1} = \alpha x^{k+1} + (1 - \alpha)z^k$
- 5:  $xu^{k+1} = \hat{x}^{k+1} + u^k$
- 6:  $z_1^{k+1} = \max\{0, xu^{k+1} - \kappa\}; z_2^{k+1} = \max\{0, -xu^{k+1} - \kappa\}$
- 7:  $z^{k+1} = z_1^{k+1} - z_2^{k+1}$
- 8:  $u^{k+1} = u^k + (\hat{x}^{k+1} - z^{k+1})$
- 9: **end for**
- 10:  $x = z^{k_{max}}$
- 11: **return**  $x$

---

There are a few components which can be pre-computed as they remain constant throughout the iterations. To determine the threshold to enforce a sparse solution, a scaling parameter  $\tau$  is used to determine the threshold parameter  $\lambda$ . For the proposed measurement methodology in CBCS one can readily estimate  $\tau$  from one of the measurement quantities such that  $\tau = \frac{\nu}{\max(y_I)}$ , where  $\nu$  is a scaling parameter, which can be applied to both the reconstruction of  $x_I$  and  $x_Q$  alike. The threshold parameter is unique to either problem and is computed as  $\lambda_I = \tau \max(|Ay_I|)$  and  $\lambda_Q = \tau \max(|Ay_Q|)$  respectively. The  $A^T y \in \mathbb{R}^n$  term is assumed to be pre-computed after each measurement cycle alongside the respective  $\lambda$  parameter. The auxiliary matrices  $\{L, U\} \in \mathbb{R}^{m \times m}$  are constant matrices linked to  $A$  and thus can all be stored efficiently in read-only memory. The original implementation calls for a left matrix division, which in this work is simplified to normal matrix multiplications by pre-computing  $L^{-1}$  and  $U^{-1}$ . The sensing matrix contains both the projection and

the linear transform i.e.  $A = \phi\theta$  as presented in Chapter 5.

Additionally, through empirical evaluation, values for the regularisation parameters  $\alpha, \rho$  were determined alongside an appropriate number of iterations. For the considered  $4 \times 4$  imaging block and set of experiments, this was hand-tuned to be  $k = 5$ ,  $\nu = 200$ ,  $\alpha = 1.7$  and  $\rho = 1.1$  for double precision floating point reconstruction quality. All simplifications above make it possible to implement the algorithm in a discrete fashion with constant execution times only involving simple matrix arithmetic, which can be readily implemented on embedded hardware. Next, an investigation is presented which analyses the effects of precision scaling on above algorithm as well as  $d$ Sparse with respect to depth reconstruction quality and total transistor count derived from logic primitives utilised in FPGA synthesis.

## 6.4 Precision Scaling Effects

Conceptually, a discrete algorithm will occupy a fixed amount of space if implemented as an ASIC. However, translating an algorithm from a high-end language to a transistor count and ultimately physical space is not a trivial task. This section explores resource utilisation of  $\ell$ ADMM and  $d$ Sparse using a prototyping platform in form of the Xilinx FPGA UltraScale+ ZCU106 [283]. By converting C/C++ code into HDL, logic can be synthesised for use on a target FPGA platform. Synthesis is the process of converting principal operations to be executed in algorithms to logic primitives which are then replicated in the FPGA structure by matching the available logic primitives available. The full FPGA synthesis reports from [239] were provided for this work by a collaborator alongside an arithmetic library for the two data formats considered. This chapter expands upon the initial findings and further provides an estimation of real term transistor usage in terms of approximate NAND gate logic cost for the logic primitives used on the FPGA platform. The transistor count should allow to find a general idea of the size of an ASIC implementation of said logic. The areas for all logic configurations are compared against SPAD array imager sizes in Section 6.5.2. An overview of the used logic primitives is provided in Table 6.1.

Using the documentation for the UltraScale architecture of the test platform [284, 285, 286], transistor counts were estimated from the specifications of each logic component. Look-up tables (LUT) provide basic routing and arithmetic operations using carry logic and multiplexing to facilitate summing arithmetic. It is assumed that a look-up table acts as a routing device where each input bit is connected to all output bits and thus its total bit count is  $b^2$ .

**Table 6.1:** A full list of FPGA logic primitives utilised in the synthesis for the experiments in [239]. This work provides NAND gate count estimates for the utilised logic building blocks based on information in the FPGA documentation [284, 285, 286].

|        |        | Type            | Bit Width | NAND Gates |
|--------|--------|-----------------|-----------|------------|
| Logic  | LUT1   | Look-up table   | 1         | 1          |
|        | LUT2   | Look-up table   | 2         | 4          |
|        | LUT3   | Look-up table   | 3         | 9          |
|        | LUT4   | Look-up table   | 4         | 16         |
|        | LUT5   | Look-up table   | 5         | 25         |
|        | LUT6   | Look-up table   | 6         | 36         |
|        | DSP48  | Arithmetic Unit | 48        | 2304       |
|        | MUX7   | Multiplexer     | 13        | 26         |
|        | MUX8   | Multiplexer     | 24        | 40         |
|        | CARRY8 | Carry logic     | 8         | 8          |
| Memory | FFD    | Flip-Flop       | 1         | 6          |
|        | FFS    | Flip-Flop       | 1         | 6          |
|        | BRAM18 | Block RAM       | 18000     | 18000      |
|        | BRAM36 | Block RAM       | 36000     | 36000      |
|        | SR16   | Shift Register  | 16        | 16         |
|        | SR32   | Shift Register  | 32        | 32         |
|        | RAMD32 | Memory          | 32        | 32         |
|        | RAMS32 | Memory          | 32        | 32         |

The multiplexers (MUX) are estimated based on their input and output bit widths, for example a MUX7 has two 6 bit inputs and combines them to a single 13 bit output with an additional carry bit, this results in a total of 26 bits, which is assumed to simplistically translate to the same number of NAND gates.

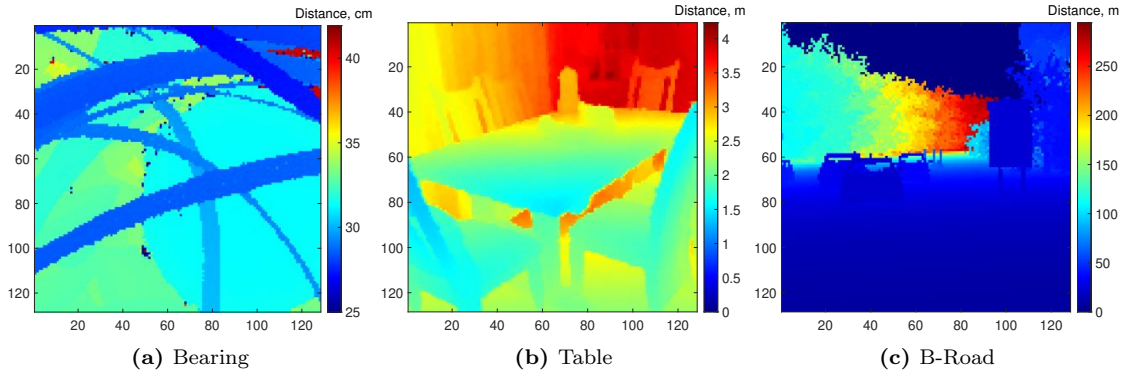
The logic primitive with the highest complexity is the DSP slice, providing a 48 bit output. This is a complex component with various arithmetic functionality. Here it is assumed that it performs the most expensive arithmetic operation being multiplication for a 48 bit number assuming  $\mathcal{O}(b^2)$  operations, resulting in a total of 2304 NAND gates.

Finally, and probably most intuitively, are memory components which in the case of shift registers, random access memory (RAM) and simple memory registers are assumed to have the same number of NAND gates as the number of bits they store. In the case of a flip-flop (in synchronous (FFS) and asynchronous (FFD) configurations) the total complexity for a single bit input is 6 NAND gates to facilitate the storage of two bit states in a stable fashion.

These are simplistic assumptions about the total NAND gate count and can therefore only provide a rough approximation to real logic size when implemented as an ASIC. Due to the nature of FPGA prototyping there is plenty of overhead versus a application specific implementation due to its configurable nature. Similarly, some of the complex logic primitives provided in a prototype platform, may be more expensive to implement in an ASIC while other savings and optimisations can be made

for the particular application. However, the NAND gate count estimates should provide a good idea of feasibility and can indicate if the logic complexity is close to fit within constraints of a sensor area, is simply too complex and would require further approximations and optimisations or is outright impossible to be matched to a typical detector array.

To assess the impact of precision scaling on depth reconstruction quality using the presented sparse reconstruction approaches, three scenes are considered as shown in Figure 6.2. The selection comprises real photon count data in the *Bearing* scene and two scenes with simulated photon counts from synthetic data using the simulation tools presented in Chapter 3.



**Figure 6.2:** Three different scenes - short (a), medium (b) and long range (c) for precision scaling effects on sparse depth reconstruction using CBCS.

The initial results presented [239] focused mainly on resource use and power draw with empirically found bit widths around an arbitrary breaking point, this work will try to elaborate upon these initial results by providing a more structured approach to finding the respective breaking points for a generic light detection and ranging (LiDAR) system with known specifications and with specific applications in mind, represented by the three scenes; ultra short range millimetre resolution (Figure 6.2(a), short range centimetre resolution 6.2(b) and long range centimetre resolution 6.2(c).

To find the minimum precision required for  $\ell$ ADMM and  $d$ Sparse for the considered application types, several specifications should be considered as outlined in Table 6.2 for a  $4 \times 4$  Sparse Scale Block (Figure 5.29). The primary goal is to estimate the largest possible value for both the integer part and fractional part to find the minimum bit width required to encapsulate all intermediate results without loss in precision. If some loss in precision is acceptable, the bit width can be reduced further. However, this work considers a lossy compressive imaging scheme and therefore the aim is to reduce precision before further losses occur.

**Table 6.2:** Properties affecting precision limits for  $\ell$ ADMM and  $d$ Sparse.

| Property                    | Variable         | System                                    |  |          |
|-----------------------------|------------------|---|--|----------|
| Block length                | $n$              | 16  |  |          |
| Max photon count per sample | $c_P$            | 16  |  |          |
| Sample steps                | $\eta_P$         | 3   |  |          |
|                             |                  | Scenes                                    |  |          |
|                             |                  | Floating                                  | Table  | B-Road   |
| Max range                   | $r_{\max}$       | 45 cm                                     | 10 m   | 300 m    |
| Depth resolution            | $r_{\min}$       | 0.01 cm                                   | 0.01 m   | 0.04 m   |
| Bins                        | $n_{\text{bin}}$ | 4500                                      | 1000   | 7500     |
| Largest photon count        | $(Y_I)_{\max}$   | $\sim 4000$                               | Eq.(6.3)   | Eq.(6.3) |
| Largest value               | $(Y_Q)_{\max}$   | $\sim 12000$                              | Eq.(6.4)   | Eq.(6.4) |
|                             |                  | $\ell$ ADMM                               | $d$ Sparse   |          |
| Measurements                | $m$              | 8   | 48   |          |
| Iterations                  | $k$              | 5   | 1  |          |
| Multiplications             | $p$              | $10k$                                     | $2nm$  |          |
| Decimal exponent            | $e$              | $\log_{10}(\frac{200}{(Y_I)_{\max}}) - 1$ | $\log_{10}(\frac{1}{p} \frac{1}{r_{\min} n_{\text{bin}}})$ |          |
| Fraction length             | $f$              | $10^e$                                    |  |          |

With the considered LiDAR system (Figure 5.30) using independent small block imagers for compressive or sparse sampling, the system specifications inform the upper limit of photon counts per sample per pixel per block. Several scenes representative of different application scenarios are considered. They provide the maximum range and sampling resolution, which informs the number of bins in a histogram and consequently the matching distance vector encoding the bin number as depth, **d**. Ultimately, the largest values are expected to be intermediate values, in particular since the final depth estimate is a division in the CBCS framework as outlined in Chapter 5. Those two proxy quantities might therefore require higher precision than the final result. Maxima for both measurement quantities can be estimated as follows. The maximum photon count can be estimated by the multiplication chain,

$$(Y_I)_{\max} = \eta_P c_P m n_{\text{bin}}, \quad (6.3)$$

where  $\eta_P$  is the number of illumination cycles per pattern exposure,  $m$  the total number of patterns exposed and  $c_P$  the maximum photon count per bin per sample step across all histogram bins  $n_{\text{bin}}$ . This is a theoretical limit, which is unlikely to occur in practice.

Next, to estimate the limit for the depth sum, the following is considered

$$(Y_Q)_{\max} = \sum (\eta_P c_P m \circ ((\mathbf{d} + \mathbf{1})^{-1}) \circ \mathbf{d}), \quad (6.4)$$

where  $\mathbf{d} \in \mathbb{R}^{n_{\text{bin}}}$  is the distance vector spanning across the range  $r_{\max}$  with step-size  $r_{\min}$ ,  $\mathbf{1}$  is a vector of ones matching the prior dimension and  $\circ$  represents element-

wise multiplication. Here, the maximum photon count is inversely scaled across the range under ideal conditions. These two values should provide worst cases for the bit width,  $b$ , required to maintain precision throughout the depth computation.

To estimate the integer part of the fixed point numerical representation, the largest binary representation is found for  $Y_Q$  with an additional bit to allow for overflow. The fractional parts should scale based on multiplications. In Table 6.2 the exponent is estimated for both algorithms based on the most demanding fractional component of each algorithm. In the case of  $\ell$ ADMM, the most convenient value to find a limit seems to be the tuning parameter  $\tau$ , which is assumed to be pre-computed at double floating point precision as it sets the threshold parameter  $\lambda$  and is thus involved the sparsity regularisation affecting precision. For  $d$ Sparse, the cumulative nature of matrix multiplication is used to estimate the largest fractional error. With this information each minimum bit width can be estimated for the fractional parts of each algorithm, leading to the full limits for fixed point numbers as

$$b_{\text{FXPInt}} = \text{ceil}(\log_2((Y_Q)_{\max})) + 1, \quad (6.5)$$

$$b_{\text{FXPFrac-}\ell} = \text{ceil}(\log_2(p \log_{10}(\frac{1}{f}))) + 1 \text{ and} \quad (6.6)$$

$$b_{\text{FXPFrac-d}} = \text{ceil}(\log_2(p|e|)) + 1. \quad (6.7)$$

The  $\text{ceil}(\cdot)$  operator is forcing the number to be rounded to the next bigger integer value. A bit is added as a safety net in case of overflow.

For floating point, a slightly different approach is used to estimate the largest possible value for the mantissa by scaling the fraction of the two maximum estimates by the largest estimated exponent, while the bit width of the exponent is simply the exponent estimate for each respective algorithm, which leads to

$$b_{\text{FPSig}} = \text{ceil}(\log_2(10^{|e|} \frac{(Y_Q)_{\max}}{(Y_I)_{\max}})) + 1 \text{ and} \quad (6.8)$$

$$b_{\text{FPExp}} = |e| + 1 \quad (6.9)$$

with an additional bit as before. Using all values as specified in Table 6.2 the estimated minimal bit widths for each respective scenario are presented in Table 6.3. These values will be compared against a comprehensive bit width analysis for all three scenes in the next two sections for each considered reconstruction framework.

In the next part, the logic primitive usage from [239] is expanded from the full synthesis reports for all logic primitives used and shown in Table 6.1 for a small

**Table 6.3:** Estimates for minimum bit width,  $b$ , for various scenes and algorithms.

| Scene    | Floating Point |          |             |          |
|----------|----------------|----------|-------------|----------|
|          | $\ell$ ADMM    |          | $d$ Sparse  |          |
|          | Significand    | Exponent | Significand | Exponent |
| Floating | 13             | 7        | 23          | 7        |
| Table    | 11             | 6        | 11          | 6        |
| B-Road   | 11             | 7        | 11          | 7        |
|          | Fixed Point    |          |             |          |
|          | $\ell$ ADMM    |          | $d$ Sparse  |          |
|          | Integer        | Fraction | Integer     | Fraction |
| Floating | 15             | 14       | 22          | 15       |
| Table    | 20             | 14       | 22          | 15       |
| B-Road   | 23             | 14       | 26          | 15       |

selection of bit widths for floating and fixed point arithmetic.

### 6.4.1 Compressive Depth Reconstruction

In this section, the analysis from [239] is expanded and tackled from the perspective of potential transistor count. Further, a more thorough breaking point analysis is provided, which is compared with minimum bit width estimates as presented in Table 6.3 from the accompanying equations (6.9) for floating point and (6.7) for fixed point representations. The full resource utilisation of  $\ell$ ADMM for a  $4 \times 4$  sparse scale block is provided in Table 6.4. The problem is set up with  $m = 8$  measurements and a pattern density of  $\psi = 0.25$  (i.e. 4 active pixels per pattern exposure).

The overall resource usage is compared to the double floating point precision. The extended analysis from NAND gate counts for particular logic primitives (Table 6.1) provides a slightly different weighting than the more high-level analysis provided in [239]. In particular memory units are much larger components than most other logic primitives with significant numbers for flip-flops and shift registers not considered in earlier work.

The potential for logic savings is significant for both float and fixed point data representations. A sharp drop is observed by dropping from double to single precision with over 70% in logic reduction, overall it can be observed that floating point is slightly more efficient than fixed point, primarily due to the added arithmetic cost incurred by fixed point operations. It should be stressed that the logic resources are of illustrative nature as the breaking points for this data were chosen empirically until quality dropped by visual inspection and/or when quality metrics dropped by more than 10% versus double precision results. More importantly the breaking point was only found for a single short range scene.

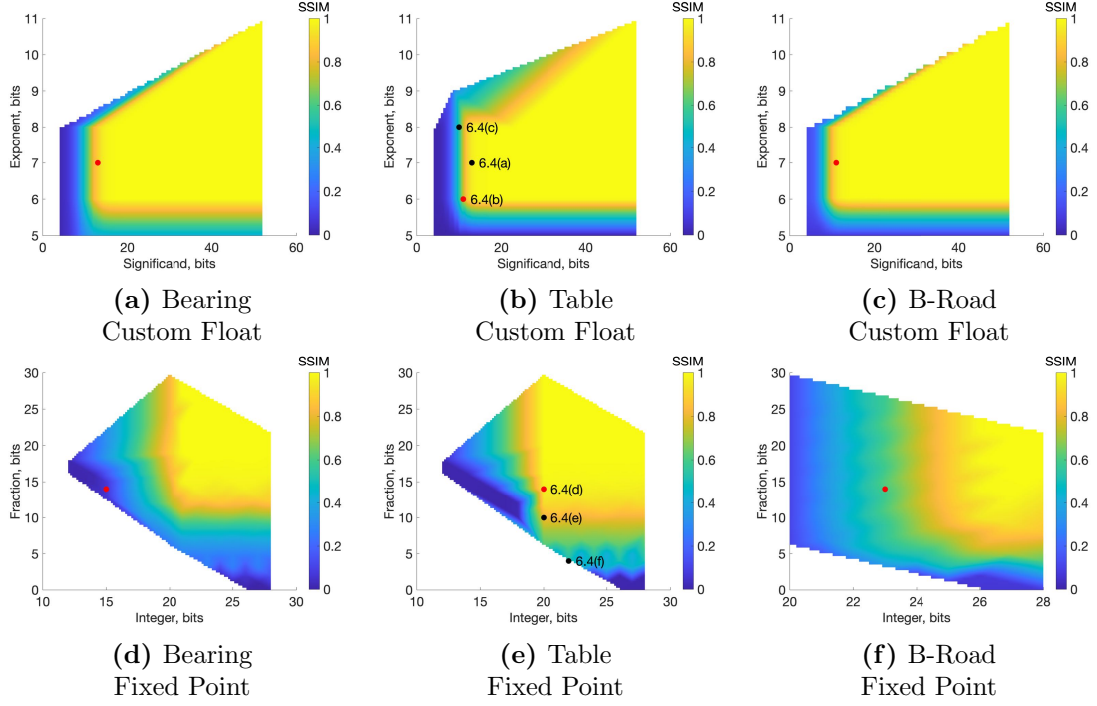
**Table 6.4:** Resource utilisation on FPGA for  $\ell$ ADMM in terms of hardware primitives in the context of custom data types extended from [239] with estimates of NAND gates extrapolated from Table 6.1.

|        |            | Custom Float |         |         |         | Fixed Point |         |         |
|--------|------------|--------------|---------|---------|---------|-------------|---------|---------|
|        |            | 64 bits      | 32 bits | 18 bits | 12 bits | 28 bits     | 23 bits | 20 bits |
| Logic  | LUT1       | 110          | 46      | 50      | 46      | 36          | 31      | 33      |
|        | LUT2       | 928          | 545     | 433     | 322     | 930         | 363     | 372     |
|        | LUT3       | 2572         | 1608    | 620     | 559     | 641         | 548     | 492     |
|        | LUT4       | 2146         | 1146    | 893     | 689     | 386         | 407     | 333     |
|        | LUT5       | 2065         | 1241    | 573     | 508     | 327         | 289     | 273     |
|        | LUT6       | 5851         | 1713    | 1573    | 985     | 560         | 396     | 386     |
|        | DSP48      | 82           | 36      | 8       | 8       | 81          | 57      | 32      |
|        | MUX7       | 119          | 0       | 11      | 0       | 0           | 0       | 0       |
| Memory | MUX8       | 11           | 0       | 0       | 0       | 0           | 0       | 0       |
|        | CARRY8     | 438          | 177     | 188     | 154     | 147         | 77      | 97      |
|        | FFD        | 24558        | 10631   | 7132    | 5036    | 2764        | 2147    | 1523    |
|        | FFS        | 14           | 36      | 20      | 20      | 14          | 14      | 14      |
|        | BRAM18     | 24           | 0       | 0       | 0       | 0           | 0       | 0       |
|        | BRAM36     | 7            | 2       | 0       | 0       | 2           | 2       | 2       |
|        | SR16       | 2222         | 912     | 607     | 445     | 0           | 0       | 0       |
|        | SR32       | 522          | 42      | 27      | 21      | 0           | 0       | 0       |
|        | RAMD32     | 512          | 390     | 170     | 110     | 280         | 230     | 198     |
|        | RAMS32     | 0            | 320     | 258     | 168     | 364         | 299     | 261     |
|        | NAND Gates | 1419605      | 386745  | 180009  | 132237  | 341112      | 268246  | 202412  |
|        | Usage      | 100.0%       | 27.2%   | 12.7%   | 9.3%    | 24.0 %      | 18.9%   | 14.3%   |

To provide a more analytical evaluation of breaking points for precision scaling on sparse depth imaging, a thorough manual bit width sweep was carried out using the C/C++ code created for [239] to capture quality degradation across various bit width combinations for the respective numerical formats. Double floating point (64 bit) precision is chosen as the baseline and all subsequent reduced precisions and resulting depth reconstructions are compared against it. The results for  $\ell$ ADMM are mapped in Figure 6.3 with the estimated minima from Table 6.3 highlighted. All considered metrics (equations in 5.31) behave very similar in this experiment and structural similarity index measure (SSIM) was chosen for illustration purposes due to its intuitive normalised scaling.

What stands out in the graphs in Figure 6.3 is the consistent breaking point behaviour when certain values for the two data types are reached for  $\ell$ ADMM across all scenes and the excellent tolerance to precision scaling before quality degradation occurs. As expected the quality boundaries are not the same for each of the considered scenarios due to value magnitude and overall precision variations in the problem definitions as discussed previously. The floating point estimates for minimum bit width seem to be reasonable, while the fixed point estimate is in the general area for the two simulated systems, where all system specifications were known. For the real data scene, the estimate is off by about 5 bits for the *Integer* part. A potential explanation for this could be the windowed nature of this data, which might affect





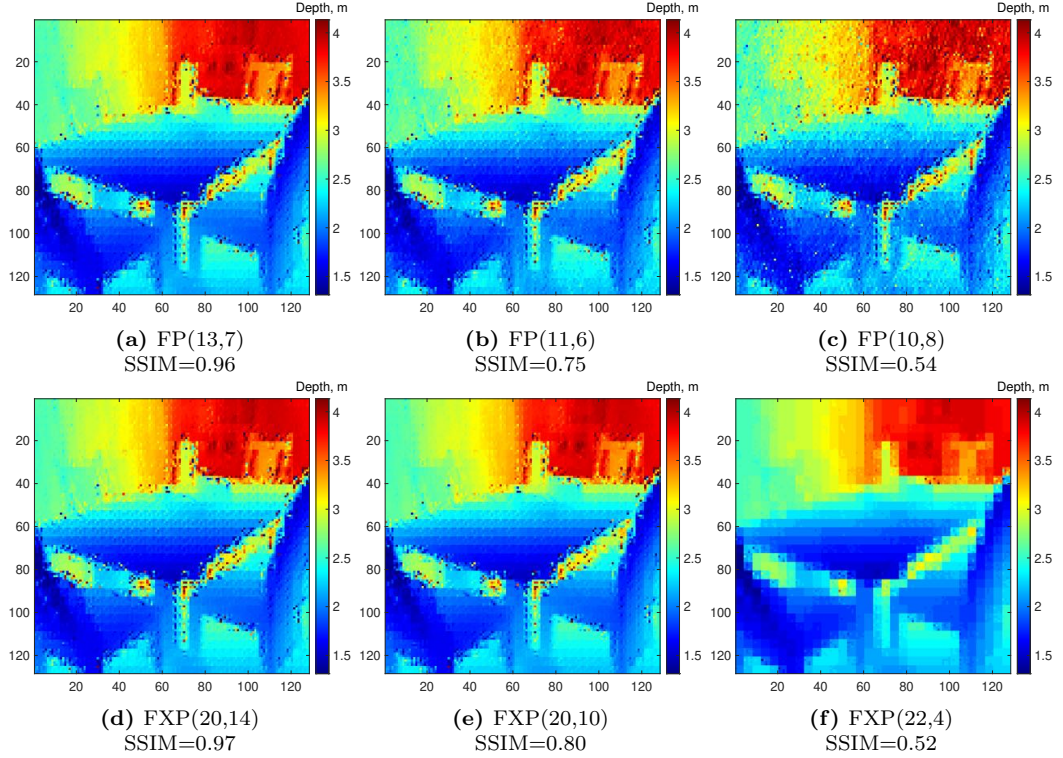
**Figure 6.3:** Precision effects on reconstruction quality for CBCS and  $\ell$ ADMM versus full double float precision. A consistent plane of no degradation can be observed, which ends in a smooth but pronounced quality boundary for all scenarios. Red points indicate the precision limits determined in Table 6.3 and labelled points are examples shown in Figure 6.4

the overall value distribution in comparison to the full range scenes considered for the simulated system approach.

To illustrate how each algorithm breaks when precision is reduced beyond the quality boundary, a representative selection is provided for the short range Table scene in Figure 6.4, but similar behaviours were observed for all scenes.

The overall performance of  $\ell$ ADMM in this chapter is limited to a single pattern sequence for memory efficiency of sharing a pattern sequence across all block. This can introduce additional blocking artefacts in the compressive case as can be seen. Therefore, the quality degradation is compared versus double floating point precision to only assess the effects of precision scaling. Floating point degrades once the *Significand* drops below 13 bits, which results in more noise across the reconstructed image.

Fixed point on the other hand seems to be primarily affected by the number of *Fraction* bits and fails by losing detail and averaging the block depth value or rather losing the values after the decimal point.



**Figure 6.4:** Illustration of quality degradation for precision scaling of  $\ell$ ADMM evaluated against double floating point precision. Floating point configuration is shortened to FP(Significand, Exponent) and fixed point representation to FXP(Integer, Fraction).

### 6.4.2 Sparse Depth Reconstruction

Another approach presented in this work is the usage of a pseudo-inverse to find the least-square solution to the inverse problem of recovering values from sparse structured illumination. In this case the length of the pattern sequence increases, such that the system of equations becomes determined i.e.  $m \geq n$ . This incurs a small cost in larger measurement vectors. However, measurement vectors still remain smaller than full per pixel histogram storage shown in Section 6.5 and the benefit of pattern pixel accumulation to increase SNR per single measurement still allows for fast sampling rates.

In this approach, the solutions to the depth sum  $x_Q$  and the intensity photon count image  $x_i$  are found directly by applying the pseudo-inverse to each measurement vector  $y_Q$  and  $y_I$  with a full parallel program illustration shown in Algorithm 9.

It was shown in Chapter 5 that this approach performs quite well if the added number of patterns is not an issue. The scalable block array architecture proposed in this work can further extend its application to long and short range applications alike. In [239] it was shown that this approach is extremely resource efficient with a total resource usage using double precision comparable to single precision  $\ell$ ADMM.

**Algorithm 9** Discrete Depth Sparse recovery (*dSparse*) [239]**Input**  $y_Q, y_I, A^+$ **Output**  $\hat{X}_D$ **for** each block  $b$  concurrently **do**

$$(\hat{x}_Q)^{(b)} = A^+ y_Q$$

$$(\hat{x}_I)^{(b)} = A^+ y_I$$

**end for**

$$\hat{x}_D(x_I > 0) = x_Q ./ x_I$$

$$\hat{X}_D = \text{unraster}(\hat{x}_D)$$

It was also shown that the power reduces in line with resource reduction and is not discussed in this chapter further. A full extended resource utilisation of FPGA primitives is shown in Table 6.5 for  $m = 48$  patterns,  $\psi = 0.25$  for a  $4 \times 4$  sparse scale block as before.

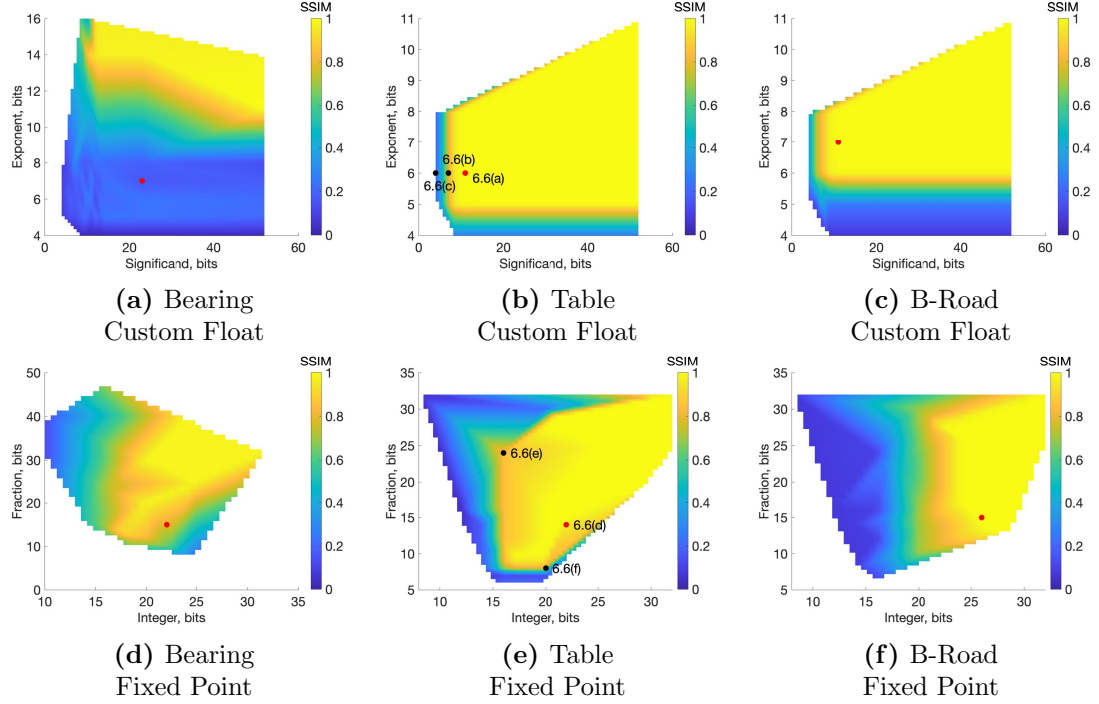
**Table 6.5:** Resource utilisation on FPGA for *dSparse* in terms of hardware primitives in the context of custom data types extended from [239] with estimates of NAND gates extrapolated from Table 6.1.

|        |            | Custom Float |         |         |         | Fixed Point |         |         |
|--------|------------|--------------|---------|---------|---------|-------------|---------|---------|
|        |            | 64 bits      | 32 bits | 18 bits | 12 bits | 28 bits     | 23 bits | 20 bits |
|        |            |              |         |         |         |             |         |         |
| Logic  | LUT1       | 23           | 9       | 11      | 14      | 60          | 51      | 44      |
|        | LUT2       | 541          | 242     | 113     | 83      | 320         | 292     | 282     |
|        | LUT3       | 3568         | 948     | 211     | 177     | 327         | 273     | 243     |
|        | LUT4       | 263          | 154     | 190     | 151     | 260         | 270     | 265     |
|        | LUT5       | 246          | 90      | 99      | 91      | 200         | 206     | 181     |
|        | LUT6       | 734          | 147     | 231     | 129     | 531         | 495     | 546     |
|        | DSP48      | 20           | 8       | 2       | 2       | 1           | 1       | 1       |
|        | MUX7       | 20           | 0       | 3       | 0       | 2           | 1       | 1       |
|        | MUX8       | 2            | 0       | 0       | 0       | 0           | 0       | 0       |
|        | CARRY8     | 493          | 150     | 56      | 45      | 27          | 22      | 21      |
| Memory | FFD        | 10191        | 2778    | 1449    | 1035    | 912         | 833     | 776     |
|        | FFS        | 1            | 6       | 1       | 1       | 1           | 1       | 1       |
|        | BRAM18     | 5            | 2       | 2       | 2       | 3           | 3       | 1       |
|        | BRAM36     | 0            | 0       | 0       | 0       | 0           | 0       | 0       |
|        | SR16       | 173          | 39      | 46      | 41      | 0           | 0       | 0       |
|        | SR32       | 74           | 20      | 0       | 0       | 0           | 0       | 0       |
|        | RAMD32     | 0            | 0       | 0       | 0       | 0           | 0       | 0       |
|        | RAMS32     | 0            | 2       | 2       | 2       | 2           | 2       | 50      |
|        | NAND Gates | 277993       | 93179   | 66827   | 59178   | 94673       | 92540   | 58540   |
|        | Usage      | 100.0%       | 33.5%   | 24.0%   | 21.3%   | 34.1%       | 33.3%   | 21.1%   |

The overall savings achieved by precision scaling for a single block are similar to the savings with respect to double precision as were observed for  $\ell$ ADMM, with the exception of RAM usage, which is slightly higher even for much smaller bit widths. This is probably due to the large matrix multiplication required to find the solution the linear system of equations, in this case  $x \in \mathbb{R}^{16} = A^+ \in \mathbb{R}^{16 \times 48} \times y \in \mathbb{R}^{48 \times 1}$ . This fairly large multiplication has to be performed twice per block in comparison to many smaller multiplications in  $\ell$ ADMM. However, the overall complexity is lower and despite the larger matrix multiplication, this approach can enable extremely

small and power efficient logic to reconstruct depth using a sparse scale block.

Further, similar to  $\ell$ ADMM, a comprehensive analysis of the effects of precision scaling on the reconstruction quality in comparison to double floating point precision for the three scenarios is shown in Figure 6.5.

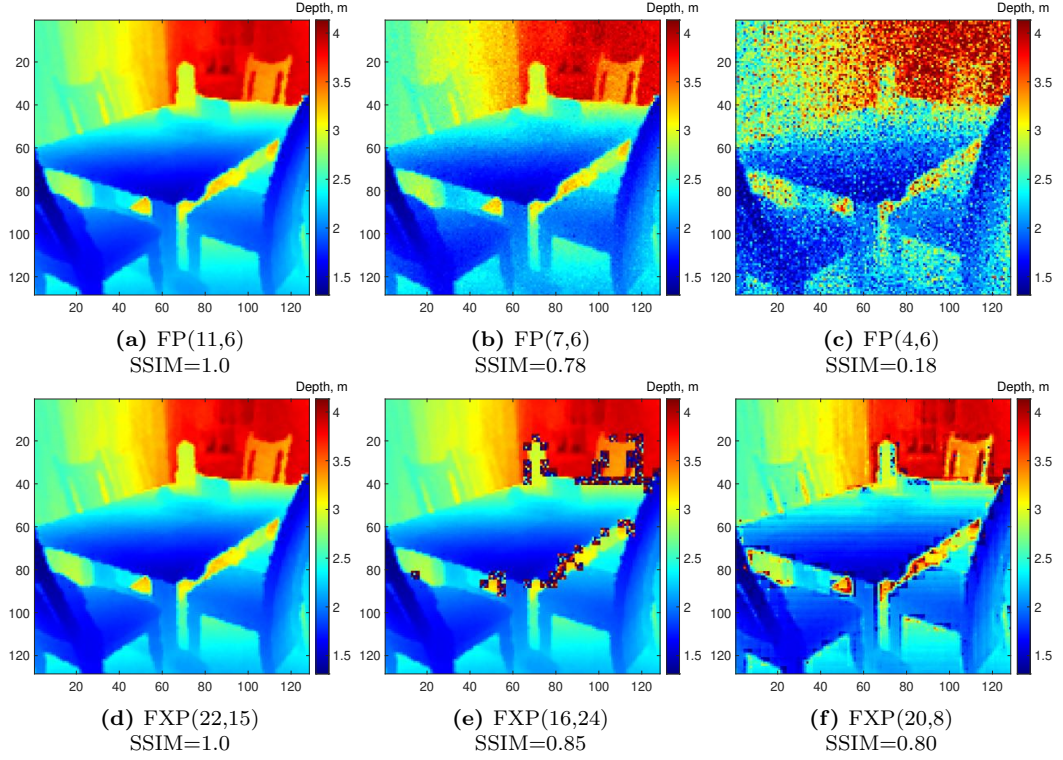


**Figure 6.5:** Precision effects on reconstruction quality for CBCS using  $d$ Sparse versus full double float precision. A clear quality boundary can be observed for all cases. Red points indicate the precision limits determined in Table 6.3 and labelled points are examples shown in Figure 6.6

Similar to the results for  $\ell$ ADMM in Figure 6.3, the predictions for minimum bit width for the simulated system are close to the breaking point boundaries for both numerical representations. It is curious, however, that the fixed point boundaries are not consistent as before. This is likely due to more multiplications in a single step and the summations in matrix multiplication.

Even more striking is the quality degradation for the real scene using  $d$ Sparse and floating point in Figure 6.5(a), which looks very different to the simulated data, despite behaving exactly the same for  $\ell$ ADMM. This again could be a combination of the algorithm itself and the windowed nature of the real data. Nonetheless, the overall quality degradation behaves as expected, albeit with much higher bit widths compared to the simulated system specifications for the other two scenes. The range resolution of the real data is significantly higher, which may also contribute to this particular degradation map.

It was shown in Chapter 5 that  $d$ Sparse can provide excellent depth reconstructions and likewise very good reconstruction performance is observed even for significant bit width reductions. To demonstrate how  $d$ Sparse fails when precision is reduced beyond the quality boundary a representative selection of reconstructions is provided in Figure 6.6. For consistency the table scene is shown, but failure behaviour is very similar across all scenes considered.



**Figure 6.6:** Illustration of quality degradation for precision scaling of  $d$ Sparse evaluated against double floating point precision. Floating point configuration is shortened to FP(Significand, Exponent) and fixed point representation to FXP(Integer, Fraction).

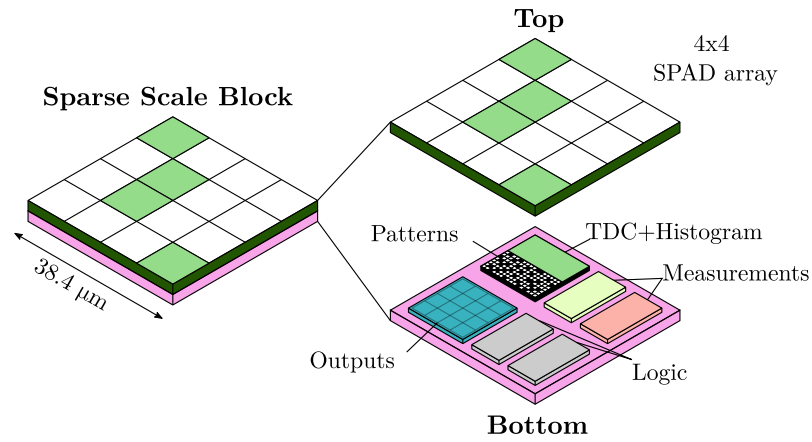
Figure 6.6(a)-(c) illustrates the quality degradation as the *Significand* bits are reduced with the reconstruction becoming noisier with grainy depth maps in the worst case. If the *Exponent* is dropped below 6 bits, no reconstruction occurs with a blank depth map.

For the fixed point breaking point analysis for  $d$ Sparse, blocking artefacts are observed once the *Integer* is reduced below a certain value, while edge artefacts occur when the *Fraction* bit width is insufficient.

Overall,  $d$ Sparse behaves more consistently compared to  $\ell$ ADMM in terms of failure behaviour, likely due to its non-iterative operation. Regardless, the presented results of precision scaling have demonstrated options to significantly reduce resource cost for the two frameworks without sacrificing reconstruction quality, if the bit width is carefully chosen for a specific application.

## 6.5 Case Study: Logic Footprint

It has been shown that both compressive sparse depth reconstruction and a sparse discrete depth reconstruction schemes can be synthesised for a FPGA and could thus be implemented as an ASIC. As both approaches can be implemented with a specific complexity using a constant number of logic primitives with potential to reduce resources using precision scaling, this section demonstrates how each particular number of logic resources would translate into a physical size in comparison to typical values for a SPAD imaging array as envisioned for a sparse scale block as shown in Figure 6.7.



**Figure 6.7:** Sparse Scale Block Components

In particular, a recently presented SPAD array [58] provides a good size estimate for two variants of  $4 \times 4$  photon detector arrays providing size constraints to the logic, which is either the same size or ideally smaller than the detector array. The analysis provided should only be seen as a feasibility study and a full ASIC implementation of the logic may require more space to facilitate additional memory, wiring, read-out and other design constraints. The case study should, nonetheless, be able to answer the question, if the proposed processing stacks can be realistically placed underneath a photon detector array to form a small independent block sensor.

### 6.5.1 SPAD Array Size

The dimensions for a photon detector array are taken from a recent  $256 \times 256$  SPAD array [58], which provides real dimensions for both single SPADs and a typical macro-pixel in a  $4 \times 4$  configuration. This case study compares the size of the logic to both the macro-pixel and a  $4 \times 4$  standard pixel array. The dimensions for both considered  $4 \times 4$  detector block configurations are shown in Table 6.6. This SPAD array was also chosen as a reference point because it stacks logic underneath the array



**Table 6.6:** Physical dimensions of a  $4 \times 4$  SPAD arrays with and without macro-pixels configuration based on dimensions of SPAD in [58].

|                   | Unit            | 40 nm SPAD Array |              |              |
|-------------------|-----------------|------------------|--------------|--------------|
|                   |                 | $1 \times 1$     | $4 \times 4$ |              |
|                   |                 | $1 \times 1$     | $1 \times 1$ | $4 \times 4$ |
| Macro-pixel SPADs |                 | 1                | 16           | 256          |
| Size (length)     | $\mu\text{m}$   | 9.6              | 38.4         | 153.6        |
| Size (area)       | $\mu\text{m}^2$ | 92.16            | 1474.56      | 23592.96     |

(manufactured in a 40 nm process) for time-to-digital converter (TDC) histogram logic manufactured in a 90 nm process, which is used as the largest technology node in the following size analysis for the overall sparse depth reconstruction logic.

### 6.5.2 Sparse Logic Component Size

Two size constraints have been established. This constitutes the maximum permissible area any required processing and control logic can occupy. The analysis carried out in this section provides baseline estimates to assess feasibility for a single sparse scale block implementation as conceptually shown in Figure 6.7. To find an approximate size for the logic and memory required to perform sparse depth recovery, three technology nodes are considered. A technology node refers the smallest feature size which can be achieved in semi-conductor manufacturing. For this case study, it is assumed that a NAND gate in a complementary metal-oxide semiconductor (CMOS) implementation occupies a square space for simplicity i.e. the node size squared. The three manufacturing nodes are summarised in Table 6.7.

**Table 6.7:** Physical dimensions for transistors in various technology nodes with square size assumption.

|               | Unit                           | 28 nm | 40 nm | 90 nm |
|---------------|--------------------------------|-------|-------|-------|
| Size (length) | $\mu\text{m}$                  | 0.028 | 0.04  | 0.09  |
| Size (area)   | $\mu\text{m}^2 \times 10^{-3}$ | 0.784 | 1.6   | 8.1   |

The three nodes were chosen for various reasons. The 90 nm node is considered because a logic stack in this node for the considered detector array has been presented in [58]. Further, a like-for-like size comparison of the SPAD technology node [39] is considered with the 40 nm node and finally a 28 nm node is considered for a smaller but reasonable near future technology node, considering that many modern SoC designs are manufactured in sizes as small as 5 nm [259].

Although the arithmetic logic is expected to be the largest component in this analysis, in particular since two are required for the two concurrent recovery processes of  $x_Q$  and  $x_I$ , a simple full system DSP logic estimate is provided by considering

additional memory for measurements, pattern projection, sensing matrix and final outputs, which will scale with bit width,  $b$ , as shown in Table 6.8.

**Table 6.8:** Full DSP logic components for sparse imaging block in terms of estimated NAND gates for varying bit widths,  $b$ .

|                 | Type   | Data dimension  | NAND gate count     |
|-----------------|--------|-----------------|---------------------|
| Pattern storage | Memory | $48 \times 16$  | $768 \times b$      |
| Histogram       | Memory | $1 \times 7500$ | $7500 \times b$     |
| Measurements    | Memory | $2 \times 48$   | $96 \times b$       |
| Reconstruction  | Logic  | $2 \times 16$   | see Table 6.4 & 6.5 |
| Output          | Memory | $2 \times 16$   | $32 \times b$       |

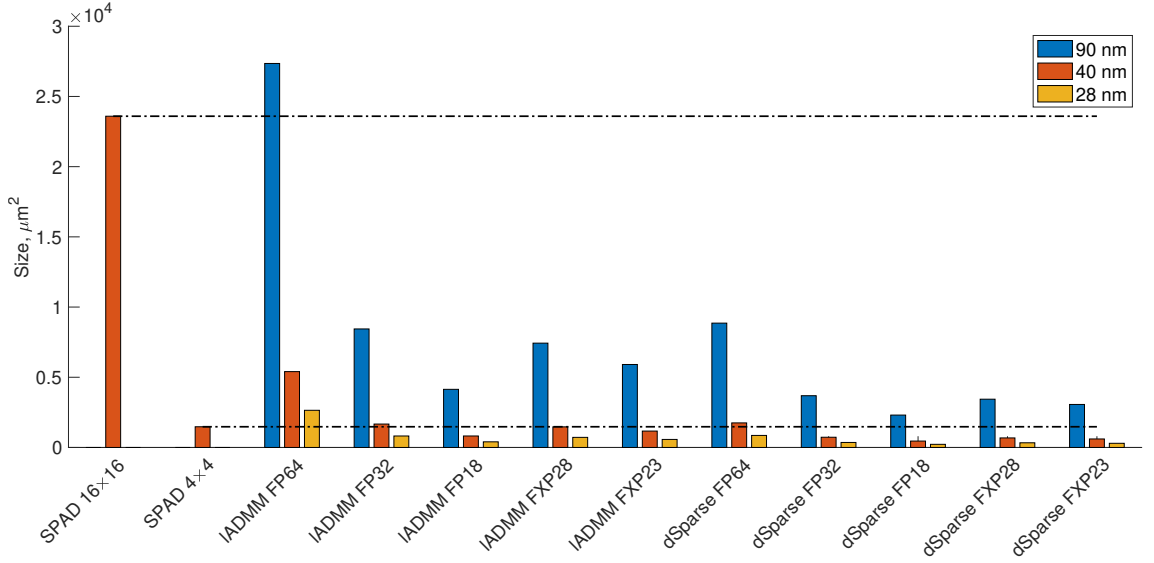
It should be noted that the pattern storage is a catch-all container, which is by design chosen to be larger than its typical individual components. For the compressive case, the actual projection,  $\phi$  is binary and thus more compact, and the the final sensing matrix,  $A$ , with associated basis transform,  $\theta$ , is smaller compared to the larger pseudo inverse,  $A^+$  for  $d$ Sparse, with an associated binary projection matrix. It was shown in Chapter 5 that  $d$ Sparse can operate well with only 24 patterns, and therefore the considered 48 patterns should be a good estimate for a full system memory for this component which may require a bit of extra space. A single histogram container is considered to enable noise removal per pattern-measurement cycle for the largest histogram case being the long range automotive application. The remaining components are the measurement, reconstruction logic and output containers, of which two are required to make the most of the reconstruction. This means a depth and intensity image are reconstructed for each block in a larger scalable block array. All components together form a simple representation of a full SoC for the sparse scale block.

### 6.5.3 Block Logic Core Size

With a rudimentary system component list, manufacturing nodes and the size constraints imposed by the SPAD detector array, the full system size is estimated by combining results from Tables 6.4, 6.5 and 6.8 as summarised in Table 6.9.

The above should provide a good estimate for the various system sizes across various bit widths afforded by precision scaling. Using the total NAND gate or transistor count estimations for each approach and precision alongside the various technologies node, the size is readily computed. All sizes of the resulting SoC are compared against both the single  $4 \times 4$  (i.e. 16 individual SPADs) and the macro-pixel  $4 \times 4$  detector array (i.e. 16 pixels arranged in a  $4 \times 4$  configuration where each pixel is a grouping of 16 SPADs for higher SNR) across all technology nodes as shown in Figure 6.8.





**Figure 6.8:** SPAD and logic size comparison chart across various precisions and technology nodes.

What is interesting in this figure, is that a sparse depth recovery system in a 90 nm node would not fit under a macro-pixel array when utilising double floating point precision for a compressive configuration but dropping the precision to single and below allows all frameworks to fit comfortably underneath a macro-pixel array. It stands out, that when looking specifically at the single pixel array, the analysis is not as straightforward.

The 90 nm node is simply too large to fit any framework under the smaller detector array even with precision scaling savings. In the 40 nm node, the compressive  $\ell$ ADMM framework fits when precision is reduced below single floating point precision, and for  $d$ Sparse, the precision needs to be at least dropped to single floating point to comfortably fit. Even at 28 nm, double precision would not fit for  $\ell$ ADMM, although it would accommodate the  $d$ Sparse approach. Overall though, in this smaller manufacturing process, other than double floating point  $\ell$ ADMM all

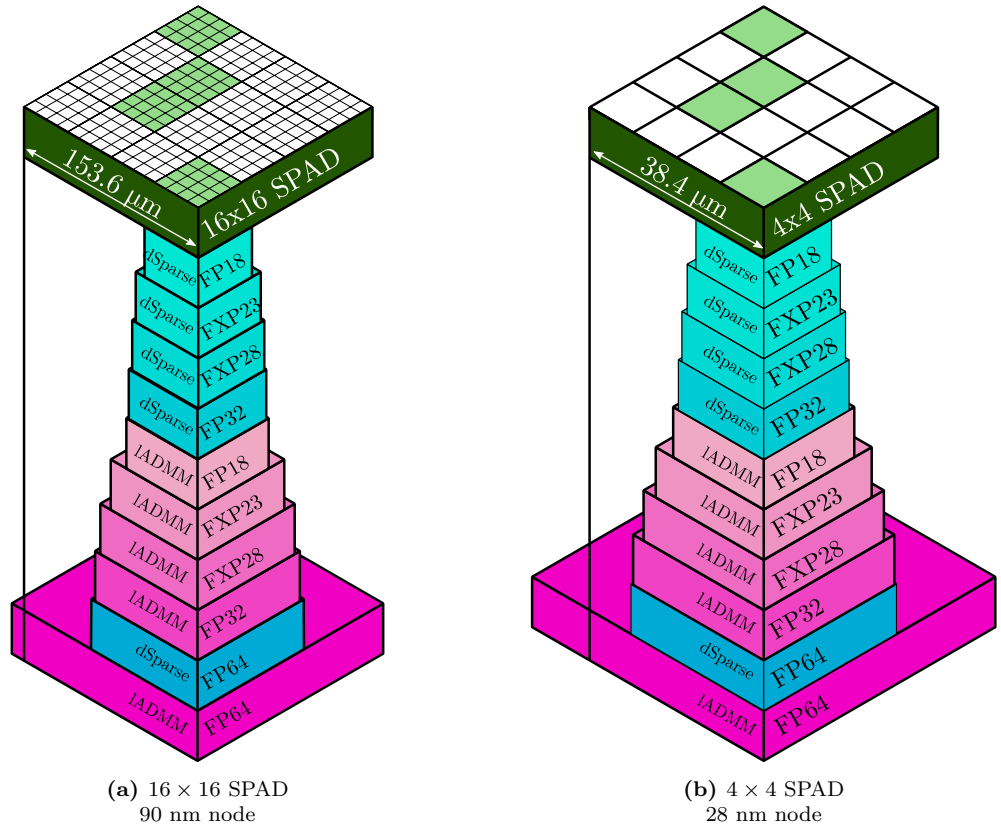
**Table 6.9:** NAND gate estimation for full sparse computation logic and associated storage.

|             |             | NAND Gate Count Estimate |                 |            |              |         |                |
|-------------|-------------|--------------------------|-----------------|------------|--------------|---------|----------------|
|             | Datatype    | Bit Width , b            | Pattern Storage | Histograms | Measurements | Logic   | Output         |
| $\ell$ ADMM | Float       | 64                       | 49152           | 480000     | 6144         | 2839210 | 2048           |
|             | Float       | 32                       | 24576           | 240000     | 3072         | 773490  | 1024           |
|             | Float       | 18                       | 13824           | 135000     | 1728         | 360018  | 576            |
|             | Fixed Point | 28                       | 21504           | 210000     | 2688         | 682224  | 896            |
|             | Fixed Point | 23                       | 17664           | 172500     | 2208         | 536492  | 736            |
| $d$ Sparse  | Float       | 64                       | 49152           | 480000     | 6144         | 555986  | 2048           |
|             | Float       | 32                       | 24576           | 240000     | 3072         | 186358  | 1024           |
|             | Float       | 18                       | 13824           | 135000     | 1728         | 133654  | 576            |
|             | Fixed Point | 28                       | 21504           | 210000     | 2688         | 189346  | 896            |
|             | Fixed Point | 23                       | 17664           | 172500     | 2208         | 185080  | 736            |
|             |             |                          |                 |            |              |         | <b>Total</b>   |
|             |             |                          |                 |            |              |         | <b>3376554</b> |
|             |             |                          |                 |            |              |         | <b>1042162</b> |
|             |             |                          |                 |            |              |         | <b>511146</b>  |
|             |             |                          |                 |            |              |         | <b>917312</b>  |
|             |             |                          |                 |            |              |         | <b>729600</b>  |
|             |             |                          |                 |            |              |         | <b>1093330</b> |
|             |             |                          |                 |            |              |         | <b>455030</b>  |
|             |             |                          |                 |            |              |         | <b>284782</b>  |
|             |             |                          |                 |            |              |         | <b>424434</b>  |
|             |             |                          |                 |            |              |         | <b>378188</b>  |

other frameworks would comfortably fit under the detector block array.

This demonstrates that the complexity of the framework presented in this work is not only suitable for a SoC implementation but should also fit comfortably under a typical SPAD detector array. In many cases the estimates are significantly smaller, which would further decrease cost and thermal/power budget and allows for additional components to either increase the overall capability of the system or provides design flexibility to accommodate wiring and any other required logic for the full system.

To visualise the sizing further, an illustration is provided in Figure 6.9. Here two particular nodes are chosen, 90 nm and 28 nm, and the full system size is compared to a macro-pixel and single pixel array size respectively.



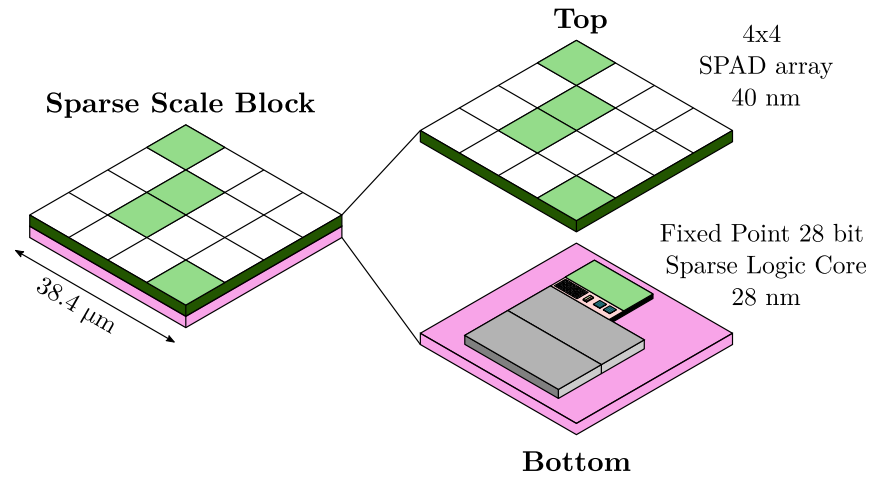
**Figure 6.9:** Size of logic core versus SPAD arrays. *dSparse* in blue tones and *lADMM* in magenta tones. A macro-pixel array is paired with 90 nm SoC size estimates while a single pixel array is compared to 28 nm SoC area approximations.

This figure demonstrates the significant savings afforded by precision scaling and algorithm optimisations. More importantly, it further confirms that the system proposal of this work is feasible to be implemented as an efficient small scale SoC for a resource constrained imaging system.

## 6.6 Summary and Conclusions

This chapter has introduced the concept of approximate computing (AC) to improve resource and power efficiency in embedded systems with limited resources. In particular the concept of precision scaling was discussed in detail, as it provides an opportunity to reduce logic resource requirements and therefore logic size by improving the computational efficiency due to the reduction of bit width, which reduces both arithmetic operations, memory requirements and power draw.

Earlier results applying precision scaling to the presented sparse depth reconstruction framework were expanded and a thorough analysis of precision scaling on the quality of reconstruction was added with an attempt to provide an initial estimate based on system specifications. In most cases the respective estimate provided a good starting point for further precision scaling optimisation fairly close to the quality boundaries for both considered reconstruction methods.



**Figure 6.10:** Sparse Scale Block with a 28 bit fixed point logic core with component scaling for 28 nm transistor sizing.

Finally, a case study was presented which considers a recent SPAD array in terms of area size for two pixel configuration. The logic primitives used in the FPGA synthesis were translated into transistor counts which in turn were used to compute the logic area for three technology nodes. The results from the precision scaling were combined with a simple full system logic estimation to provide area estimates for all synthesised precision logic configuration and compared to the photon detector array sizes. For the largest technology node, double precision floating point was too large, but depending on configuration and technology node, the logic should comfortably fit under a detector array and in some cases with room to spare to allow for more design flexibility. An illustration of such a SoC and detector block combination is shown in Figure 6.10 with appropriate component scaling.

This chapter has shown that the proposed system architecture and associated frameworks are not only ASIC compatible, but can in theory also fit under a detector array with minor precision scaling optimisation. Further savings could be achieved with more approximations, if required, to reduce power and size even more.

The feasibility of the Sparse Scale block demonstrates a potential system capable of LiDAR imaging, which exploits hardware parallelism for a wide scale of resolutions. The key performance metrics are dictated by the block performance rather than the full system array, which provides a reliable constant high frame rate as resolution is scaled up.

# Chapter 7

## Conclusion and Future Work

### 7.1 Summary of Main Findings

The work presented in this thesis addresses challenges associated with the processing of raw measurements from solid-state time-of-flight (ToF) arrays for light detection and ranging (LiDAR) systems with an emphasis on long range use cases as required for automotive applications.

The concepts of LiDAR and the enabling technology for solid-state high resolution LiDAR were introduced in Chapter 2, with particular emphasis on emerging large scale single photon avalanche detector (SPAD) arrays. A clear trend of increasing array sizes was observed, yet many of the demonstration systems with integrated sampling circuitry have reduced depth resolution and/or limited range, due to the complexity associated with the large volumes of data as range increases and depth resolution improves. Further, laser power for the considered systems is limited due to the operational window afforded by silicon, making long range applications more difficult to achieve at high frame rates. The concepts and advantages of the full LiDAR waveform were discussed in the context of recent processing methodologies, noting distinct compromises between speed, dynamic range and resolution.

To investigate novel processing frameworks, a simulation tool chain was developed and presented in Chapter 3. The focus of this framework was to enable waveform simulation from scenes with semantic content to enable visual inspection with ground truth for evaluation and the ability to sample the data in a non-linear fashion to enable investigation of novel system architectures. The ability to generate large LiDAR waveform datasets from synthetic depth datasets motivated the investigation into deep learning (DL) methods to process LiDAR data, as they provide an efficient way to facilitate high throughput processing. The ability to sample both real and

simulated LiDAR data in a random sparse fashion enabled the exploration of novel compressive sensing schemes applied to long range LiDAR depth imaging.

Using modern DL techniques, a novel convolutional neural network (CNN) architecture was developed to localise returns from LiDAR waveforms in Chapter 4.

This work addresses the challenges of processing large data volumes generated by future solid-state high-resolution LiDAR detector arrays, with the following contributions:

- A novel neural network architecture for LiDAR waveform processing, LiDAR-Net. It has the capability of learning significant waveform features to discretely localise surface returns in LiDAR waveforms with multiple returns in both low and high ambient noise scenarios in a unified network architecture.
- Validation of a training-by-simulation approach and feature tracing for the presented network architecture, confirming relevant features are learned and generalised. This provides a meaningful way to augment real datasets and pre-train a network for faster deployment on simulated data.
- Peak localisation performance is comparable to state-of-the-art model based approaches, while being twice as fast as a prior approach at 0.44 ms per histogram providing an extremely fast and high throughput approach to LiDAR waveform processing for solid-state arrays at high frame rates.

Compressive sensing was investigated to address the issue of high laser power density for long range LiDAR and ever increasing data bandwidth scaling with photon detector array size in Chapter 5. Although structured sparse illumination reduces laser power density, the sampling and processing times for high resolution compressive depth reconstruction were prohibitive to long range depth imaging in real-time.

A novel compressive depth reconstruction framework was presented, with major outcomes being:

- The formulation of a small-scale compressive depth reconstruction signal model for a narrow field-of-view imaging block with excellent depth reconstruction performance even at long range and scope for noise suppression, while the prior art only was limited to short range and low noise applications.
- A novel parallel block sparse LiDAR architecture was proposed to distribute the small-scale compressive depth reconstruction across a large solid-state photon detector array with a parallel processing architecture, which was shown to be capable of depth image frame rates of well above 200 Hz.

- A practical device for scaling up LiDAR resolution was proposed in the form of a block imaging platform with integrated sparse depth reconstruction logic. This exploits the independent nature of the proposed framework and enables scalable high frame rate depth imagers with performance bounds set by the block size rather than array size.

The inception of the sparse scale block device led to an investigation in Chapter 6 into the feasibility of a stacked detector device, with a full system-on-a-chip (SoC) occupying the same or less space than the detector to perform depth and intensity reconstruction using sparse principles. An introduction to approximate computing was provided and precision scaling was applied to an optimised compressive framework suitable for hardware implementation and a discrete sparse reconstruction framework with the key findings of this case study being:

- A theoretical estimate and an empirical analysis of the precision boundaries for the compressive depth reconstruction framework was presented in Chapter 5 for floating and fixed point representations. This demonstrated significant tolerances for both sparse depth reconstruction methods before reconstruction quality suffers, allowing for hardware approximations reducing resource and power requirements.
- A case study translated resource usage of the considered sparse reconstruction frameworks to transistor count which were compared to the area of a modern SPAD array for the considered imaging system architecture. It was shown that with moderate precision reduction, either framework could theoretically fit directly under a SPAD detector array in form of a stacked SoC demonstrating that the complexity of the proposed system is feasible to implement in the future.

The work presented in this thesis has addressed the challenges identified for solid-state arrayed LiDAR for automotive application with a novel approach for high-throughput LiDAR waveform processing using deep learning in the form of LiDAR-Net and a novel compressive depth reconstruction framework enabling sparse illumination, fast compressive sampling with low memory requirements and fast processing times due to an independent parallel block signal formulation. With a focus on efficiency throughout the development of this work, both approaches provide practical approaches for effective future LiDAR systems utilising solid-state photon detector arrays.

## 7.2 Future Work

Significant effort has been afforded to mitigate the absence of a real solid-state arrayed light detection and ranging (LiDAR) sensor by means of comprehensive simulation tools complemented by small amounts of real LiDAR waveforms. Although both major contributions; an efficient high throughput neural network, LiDARNet, which can localise surface returns directly from the LiDAR waveform and a blocked compressive depth sensing framework, checkerboard compressive sensing (CBCS), which enables fast sampling via structured random sampling and real-time processing at high speed using an independent block parallel processing approach have been successfully applied to real data, both approaches should be demonstrated in conjunction with a real sensor system and further examples in the future.

In particular, the deep learning approach might see improvements if trained using a comprehensive real LiDAR waveform dataset with a more diverse set of operating conditions. This could still be augmented using the presented simulation tools to add additional variance and robustness for training LiDARNet. Further, the presented convolutional neural network (CNN) architecture could benefit from optimisations to find the most suitable parameters for each network layer alongside an expanded intermediate signal analysis to improve learning rates by means of different activation functions, training strategies and more advanced deep learning (DL) techniques not considered in this work. The parameter space could also be reduced by means of weight pruning or other size reduction techniques to improve speed and resource requirements, which may enable an embedded implementation of this architecture.

The compressive sensing depth framework makes the major assumption of few surface returns in a small field-of-view and only returns a single depth value per pixel. While this work has shown to recover depth well in this framework, it would be beneficial to quantify the uncertainty in this reconstruction framework. In particular, how uncertainty is affected across the full range scale and the influence of random pattern generation in various noise conditions. Furthermore, it would be beneficial to extend this sparse depth framework to multi-return recovery similar to processing full waveforms with LiDARNet or other waveform processing strategies, e.g. by pre-processing the data by methods such as LiDARNet to work with surface proposals rather than cumulative measurements derived from the histogram directly. The basis transforms considered in this work were chosen primarily for speed and efficiency, but other more complex basis functions may perform better and should be explored alongside efforts to reduce their computational and resource impact on the proposed system. Additionally, although a thorough feasibility study for sparse reconstruction on hardware was presented, the processing chain could



also be implemented discretely using DL techniques directly from raw compressive measurements. Another worthwhile investigation would be to use machine learning techniques to post-process and de-noise the depth proposals with the compressive information available. Deep learning could also be leveraged to determine custom sensing patterns tailored to compressive depth recovery to improve performance further.

Finally, since two sparse reconstruction frameworks were demonstrated to be suitable for hardware implementations, work could be carried out to create a full demonstration system by linking a photon detector array with a field-programmable gate array (FPGA) platform in the first instance, which could lead to a full investigation and development of a system-on-a-chip (SoC) chip for a full or partial photon detector array for sparse depth sensing in an integrated fashion.

# Bibliography

- [1] Jeff Hecht, “Lidar for self-driving cars,” *Optics and Photonics News*, vol. 29, no. 1, pp. 26–33, 2018.
- [2] Mial E. Warren, “Automotive lidar technology,” in *IEEE Symposium on VLSI Circuits, Digest of Technical Papers*. June 2019, vol. 2019-June, pp. C254–C255, Institute of Electrical and Electronics Engineers Inc.
- [3] You Li and Javier Ibanez-Guzman, “Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems,” *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, July 2020.
- [4] SAE, “J3016b taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles,” Tech. Rep., 2018.
- [5] Philip E. Ross, “The audi a8: the world’s first production car to achieve level 3 autonomy,” *IEEE Spectrum: Technology, Engineering, and Science News*, pp. 3–4, July 2017.
- [6] John Krafcik, “Waymo is opening its fully driverless service to the general public in phoenix,” 2020.
- [7] Julius Ziegler, Philipp Bender, Markus Schreiber, Henning Lategahn, Tobias Strauss, Christoph Stiller, Thao Dang, Uwe Franke, Nils Appenrodt, Christoph G. Keller, Eberhard Kaus, Ralf G. Herrtwich, Clemens Rabe, David Pfeiffer, Frank Lindner, Fridtjof Stein, Friedrich Erbs, Markus Enzweiler, Carsten Knoppel, Jochen Hipp, Martin Haueis, Maximilian Trepte, Carsten Brenk, Andreas Tamke, Mohammad Ghanaat, Markus Braun, Armin Joos, Hans Fritz, Horst Mock, Martin Hein, and Eberhard Zeeb, “Making bertha drive-an autonomous journey on a historic route,” *IEEE Intelligent Transportation Systems Magazine*, 2014.
- [8] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel

- Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars Erik Jendrossek, Christian Koenen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney, “Stanley: The robot that won the darpa grand challenge,” *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
- [9] R H Rasshofer and K Gresser, “Automotive radar and lidar systems for next generation driver assistance functions,” *Advances in Radio Science*, vol. 3, pp. 205–209, 2005.
- [10] Alberto Broggi, Paolo Grisleri, and Paolo Zani, “Sensors technologies for intelligent vehicles perception systems: A comparison between vision and 3d-lidar,” in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. Oct. 2013, pp. 887–892, IEEE.
- [11] W. L. Sjogren and W. R. Wollenhaupt, “Lunar shape via the apollo laser altimeter,” *Science*, vol. 179, no. 4070, pp. 275–278, 1973.
- [12] Henry W. Jr. Brandhorst, “Terrestrial photovoltaic measurement procedures,” *Proceedings ERDA/NASA Photovoltaic Measurements Workshop*, pp. 1–17, 1977.
- [13] Thomas J.T.P. Van Den Berg and Henk Spekreijse, “Near infrared light absorption in the human eye media,” *Vision Research*, vol. 37, no. 2, pp. 249–253, Jan. 1997.
- [14] K. B. Cooper, R. J. Dengler, G. Chattopadhyay, E. Schlecht, J. Gill, A. Skalare, I. Mehdi, and P. H. Siegel, “A high-resolution imaging radar at 580 ghz,” *IEEE Microwave and Wireless Components Letters*, vol. 18, no. 1, pp. 64–66, 2008.
- [15] Jesus Grajal, Alejandro Badolato, Gorka Rubio-Cidre, Luis Ubeda-Medina, Beatriz Mencia-Oliva, Antonio Garcia-Pino, Borja Gonzalez-Valdes, and Oscar Rubinos, “3-d high-resolution imaging radar at 300 ghz with enhanced fov,” *IEEE Transactions on Microwave Theory and Techniques*, 2015.
- [16] Jesus Grajal, Gorka Rubio-Cidre, Alejandro Badolato, Luis Ubeda-Medina, Federico Garcia-Rial, Antonio Garcia-Pino, and Oscar Rubinos, “Compact radar front-end for an imaging radar at 300 ghz,” *IEEE Transactions on Terahertz Science and Technology*, vol. 7, no. 3, pp. 268–273, May 2017.

- 
- [17] Andrew M. Wallace, Abderrahim Halimi, and Gerald S. Buller, “Full waveform lidar for adverse weather conditions,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7064–7077, July 2020.
  - [18] E. Charbon, “Single-photon imaging in complementary metal oxide semiconductor processes,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 372, no. 2012, pp. 20130100, Mar. 2014.
  - [19] Samuel Rigault, Nicolas Moeneclaey, Lioua Labrak, and Ian O’connor, “A low-voltage sub-ns pulse integrated cmos laser diode driver for spad-based time-of-flight rangefinding in mobile applications,” in *International System on Chip Conference*. Sept. 2019, vol. 2019-Sept, pp. 5–10, IEEE Computer Society.
  - [20] S. Rigault, N. Moeneclaey, L. Labrak, and I. Orconnor, “Cmos vcsel driver dedicated for sub-nanosecond laser pulses generation in spad-based time-of-flight rangefinder,” in *Proceedings - 33rd Conference on Design of Circuits and Integrated Systems, DCIS 2018*. Apr. 2019, Institute of Electrical and Electronics Engineers Inc.
  - [21] Christopher V. Poulton, Ami Yaacobi, David B. Cole, Matthew J. Byrd, Manan Raval, Diedrik Vermeulen, and Michael R. Watts, “Coherent solid-state lidar with silicon photonic optical phased arrays,” *Optics Letters*, vol. 42, no. 20, 2017.
  - [22] Neale A.W. Dutton, Tarek Al Abbas, Istvan Gyongy, Francescopaolo Mattioli Della Rocca, and Robert K. Henderson, “High dynamic range imaging at the quantum limit with single photon avalanche diode-based image sensors,” *Sensors (Switzerland)*, vol. 18, no. 4, 2018.
  - [23] Joshua Rapp, Julian Tachella, Yoann Altmann, Stephen McLaughlin, and Vivek K. Goyal, “Advances in single-photon lidar for autonomous vehicles: Working principles, challenges, and recent advances,” *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 62–71, July 2020.
  - [24] Edoardo Charbon, “Single-photon imaging in cmos,” *Imaging*, vol. 7780, pp. 77801D–77801D–15, 2010.
  - [25] J. Wilson and J. F. B. Hawkes, *Optoelectronics: An Introduction*, Prentice Hall International Series in Optoelectronics, first edition, 1983.

- 
- [26] Bahram Nabet, *Photodetectors: Materials, Devices and Applications*, Elsevier Inc., Oct. 2015.
- [27] Henri Dautet, Pierre Deschamps, Bruno Dion, Andrew D. MacGregor, Darleene MacSween, Robert J. McIntyre, Claude Trottier, and Paul P. Webb, “Photon counting techniques with silicon avalanche photodiodes,” *Applied Optics*, vol. 32, no. 21, pp. 3894, July 1993.
- [28] A. Lacaita, M. Ghioni, F. Zappa, G. Ripamonti, and S. Cova, “Recent advances in the detection of optical photons with silicon photodiodes,” *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 326, no. 1-2, pp. 290–294, Mar. 1993.
- [29] S Cova, M Ghioni, A Lacaita, C Samori, and F Zappa, “Avalanche photodiodes and quenching circuits for single-photon detection,” *Applied Optics*, vol. 35, no. 12, pp. 1956, 1996.
- [30] Danilo Bronzi, Federica Villa, Simone Tisa, Alberto Tosi, and Franco Zappa, “Spad figures of merit for photon-counting, photon-timing, and imaging applications: A review,” 2016.
- [31] Sergio Hernández-Marín, Andrew M. Wallace, and Gavin J. Gibson, “Bayesian analysis of lidar signals with multiple returns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2170–2180, 2007.
- [32] Yoann Altmann, Ximing Ren, Aongus McCarthy, Gerald S. Buller, and Steve McLaughlin, “Robust bayesian target detection algorithm for depth imaging from sparse single-photon data,” *IEEE Transactions on Computational Imaging*, vol. 2, no. 4, pp. 1–1, 2016.
- [33] Yoann Altmann, Ximing Ren, Aongus McCarthy, Gerald S. Buller, and Steve McLaughlin, “Lidar waveform-based analysis of depth images constructed using sparse single-photon data,” *IEEE Transactions on Image Processing*, vol. 25, no. 5, pp. 1935–1946, May 2016.
- [34] Agata M. Pawlikowska, Abderrahim Halimi, Robert A. Lamb, and Gerald S. Buller, “Single-photon three-dimensional imaging at up to 10 kilometers range,” *Optics Express*, vol. 25, no. 10, pp. 11919, May 2017.
- [35] Marek Gersbach, Cristiano Niclass, Edoardo Charbon, Justin Richardson, Robert Henderson, and Lindsay Grant, “A single photon detector implemented in a 130nm cmos imaging process,” in *ESSDERC 2008 - Proceedings*

- of the 38th European Solid-State Device Research Conference. 2008, pp. 270–273, IEEE Computer Society.
- [36] Justin A. Richardson, Lindsay A. Grant, and Robert K. Henderson, “Low dark count single-photon avalanche diode structure compatible with standard nanometer scale cmos technology,” *IEEE Photonics Technology Letters*, vol. 21, no. 14, pp. 1020–1022, July 2009.
- [37] Eric A.G. Webster, Justin A. Richardson, Lindsay A. Grant, David Renshaw, and Robert K. Henderson, “A single-photon avalanche diode in 90-nm cmos imaging technology with 44% photon detection efficiency at 690 nm,” *IEEE Electron Device Letters*, vol. 33, no. 5, pp. 694–696, May 2012.
- [38] Edoardo Charbon, “Introduction to time-of-flight imaging,” in *Proceedings of IEEE Sensors*. Nov. 2014, vol. 2014-Decem, pp. 610–613, IEEE.
- [39] S. Pellegrini, B. Rae, A. Pingault, D. Golanski, S. Jouan, C. Lapeyre, and B. Mamdy, “Industrialised spad in 40 nm technology,” in *2017 IEEE International Electron Devices Meeting (IEDM)*. Dec. 2017, pp. 16.5.1–16.5.4, IEEE.
- [40] Carsten Degenhardt, Gordian Prescher, Thomas Frach, Andreas Thon, Rik De Gruyter, Anja Schmitz, and Rob Ballizany, “The digital silicon photomultiplier - a novel sensor for the detection of scintillation light,” in *IEEE Nuclear Science Symposium Conference Record*, 2009, pp. 2383–2386.
- [41] Salvatore Gnechi, Neale A.W. Dutton, Luca Parmesan, Bruce R. Rae, Sara Pellegrini, Stuart J. McLeod, Lindsay A. Grant, and Robert K. Henderson, “Digital silicon photomultipliers with or/xor pulse combining techniques,” *IEEE Transactions on Electron Devices*, vol. 63, no. 3, pp. 1105–1110, 2016.
- [42] Cristiano Niclass, Mineki Soga, Hiroyuki Matsubara, and Satoru Kato, “A 100m-range 10-frame/s 340x96-pixel time-of-flight depth sensor in 0.18um cmos,” in *European Solid-State Circuits Conference*, 2011, vol. 48, pp. 107–110.
- [43] Sarrah. M. Patanwala, Istvan Gyongy, Neale A. W. Dutton, Bruce. R. Rae, and Robert. K. Henderson, “A reconfigurable 40nm cmos spad array for lidar receiver validation,” in *International Image Sensor Workshop*, 2019.
- [44] F. Zappa, M. Ghioni, S. Cova, L. Varisco, B. Sinnis, A. Morrison, and A. Mathewson, “Integrated array of avalanche photodiodes for single-photon count-

- ing,” in *27th European Solid-State Device Research Conference*, Stuttgart, Germany, 1997, IEEE.
- [45] Marius A. Albota, Richard M. Heinrichs, David G. Kocher, Daniel G. Fouche, Brian E. Player, Michael E. O’Brien, Brian F. Aull, John J. Zayhowski, James Mooney, Berton C. Willard, and Robert R. Carlson, “Three-dimensional imaging laser radar with a photon-counting avalanche photodiode array and microchip laser,” *Applied Optics*, vol. 41, no. 36, pp. 7671, Dec. 2002.
  - [46] Cristiano Niclass, Mineki Soga, Hiroyuki Matsubara, Satoru Kato, and Manabu Kagami, “A 100-m range 10-frame/s 340x96-pixel time-of-flight depth sensor in 0.18  $\mu\text{m}$  cmos,” *IEEE Journal of Solid-State Circuits*, vol. 48, no. 2, pp. 559–572, Feb. 2013.
  - [47] I Gyongy, N Calder, A Davies, N A W Dutton, P Dalgarno, R Duncan, C Rickman, and R K Henderson, “256x256, 100kfps, 6% fill-factor time-resolved spad image sensor for microscopy applications,” in *Electron Devices Meeting (IEDM), 2016 IEEE International*, 2016, number 3-7 Dec. 2016, pp. 200–203.
  - [48] F. Villa, R. Lussana, D. Bronzi, F. Zappa, and A. Giudice, “3d spad camera for advanced driver assistance,” in *2017 International Conference of Electrical and Electronic Technologies for Automotive*. 2017, Institute of Electrical and Electronics Engineers Inc.
  - [49] Salvatore Gnechchi and Carl Jackson, “A  $1 \times 16$  sipm array for automotive 3d imaging lidar systems,” in *2017 International Image Sensor Workshop (IISW), Hiroshima, Japan*, 2017, pp. 133–136.
  - [50] Davide Portaluppi, Enrico Conca, and Federica Villa, “ $32 \times 32$  cmos spad imager for gated imaging, photon timing, and photon coincidence,” *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 24, no. 2, 2018.
  - [51] Luca Parmesan, Neale A.W. Dutton, Neil J. Calder, Lindsay A. Grant, and Robert. K. Henderson, “A 256x256 spad array with in-pixel time to amplitude conversion for fluorescence lifetime imaging microscopy,” in *International Image Sensor Workshop*, 2015, pp. 8–11.
  - [52] T. Al Abbas, N. A.W. Dutton, O. Almer, S. Pellegrini, Y. Henrion, and R. K. Henderson, “Backside illuminated spad image sensor with  $7.83\mu\text{m}$  pitch in 3d-stacked cmos technology,” in *Technical Digest - International Electron Devices Meeting, IEDM*. Jan. 2017, pp. 8.1.1–8.1.4, Institute of Electrical and Electronics Engineers Inc.

- 
- [53] Tarek Al Abbas, Neale A.W. Dutton, Oscar Almer, Neil Finlayson, Francesco-paolo Mattioli Della Rocca, and Robert Henderson, “A cmos spad sensor with a multi-event folded flash time-to-digital converter for ultra-fast optical transient capture,” *IEEE Sensors Journal*, vol. 18, no. 8, 2018.
- [54] Sahba Jahromi, Jussi-Pekka Jansson, and Juha Kostamovaara, “Solid-state 3d imaging using a 1nj/100ps laser diode transmitter and a single photon receiver matrix,” *Optics Express*, vol. 24, no. 19, pp. 21619, 2016.
- [55] Scott Lindner, Chao Zhang, Ivan Michel Antolovic, Martin Wolf, and Edoardo Charbon, “A  $252 \times 144$  spad pixel flash lidar with 1728 dual-clock 48.8 ps tdc, integrated histogramming and 14.9-to-1 compression in 180nm cmos technology,” in *IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, 2018, vol. 2018-June.
- [56] Juan Mata Pavia, Martin Wolf, and Edoardo Charbon, “Measurement and modeling of microlenses fabricated on single-photon avalanche diode arrays for fill factor recovery,” *Optics Express*, vol. 22, no. 4, pp. 4202, Feb. 2014.
- [57] Giuseppe Intermite, Aongus McCarthy, Ryan E. Warburton, Ximing Ren, Federica Villa, Rudi Lussana, Andrew J. Waddie, Mohammad R. Taghizadeh, Alberto Tosi, Franco Zappa, and Gerald S. Buller, “Fill-factor improvement of si cmos single-photon avalanche diode detector arrays by integration of diffractive microlens arrays,” *Optics Express*, vol. 23, no. 26, pp. 33777, Dec. 2015.
- [58] Robert K. Henderson, Nick Johnston, Sam W. Hutchings, Istvan Gyongy, Tarek Al Abbas, Neale Dutton, Max Tyler, Susan Chan, and Jonathan Leach, “A  $256 \times 256$  40nm/90nm cmos 3d-stacked 120db dynamic-range reconfigurable time-resolved spad imager,” in *IEEE International Solid-State Circuits Conference*. Mar. 2019, vol. 2019-Febru, pp. 106–108, Institute of Electrical and Electronics Engineers Inc.
- [59] Istvan Gyongy, Sam W. Hutchings, Max Tyler, Susan Chan, Feng Zhu, Robert K. Henderson, and Jonathan Leach, “1kfps time-of-flight imaging with a 3d-stacked cmos spad sensor,” in *International Image Sensor Workshop*, 2019.
- [60] Istvan Gyongy, Sam W. Hutchings, Abderrahim Halimi, Max Tyler, Susan Chan, Feng Zhu, Stephen McLaughlin, Robert K. Henderson, and Jonathan Leach, “High-speed 3d sensing via hybrid-mode imaging and guided upsampling,” *Optica*, vol. 7, no. 10, pp. 1253, Oct. 2020.



- [61] Francesco Mattioli Della Rocca, Hanning Mai, Sam W. Hutchings, Tarek Al Abbas, Andreas Tsiamis, Peter Lomax, Istvan Gyongy, Neale A. W. Dutton, and Robert K. Henderson, “A  $128 \times 128$  spad dynamic vision-triggered time of flight imager,” in *ESSCIRC 2019 - IEEE 45th European Solid State Circuits Conference (ESSCIRC)*. Sept. 2019, pp. 93–96, IEEE.
- [62] Vinit H. Dhulla, Sapna S. Mukherjee, Adam O. Lee, Nanditha Dissanayake, Booshik Ryu, and Charles Myers, “256 x 256 dual-mode cmos spad image sensor,” in *Advanced Photon Counting Techniques XIII*, Baltimore, Maryland, US, 2019, vol. 10978, p. 26.
- [63] Chao Zhang, Scott Lindner, Ivan Michel Antolovic, Juan Mata Pavia, Martin Wolf, and Edoardo Charbon, “A 30-frames/s,  $252 \times 144$  spad flash lidar with 1728 dual-clock 48.8-ps tdcs, and pixel-wise integrated histogramming,” *IEEE Journal of Solid-State Circuits*, pp. 1–15, 2019.
- [64] Matteo Perenzoni, Nicola Massari, Leonardo Gasparini, Manuel Moreno Garcia, Daniele Perenzoni, and David Stoppa, “A fast  $50 \times 40$ -pixels single-point dtof spad sensor with photon counting and programmable roi tdcs, with  $\sigma < 4$  mm at 3 m up to 18 klux of background light,” *IEEE Solid-State Circuits Letters*, vol. 3, pp. 86–89, 2020.
- [65] Sahba Jahromi, Jussi-Pekka Jansson, Pekka Keranen, and Juha Kostamovaara, “A  $32 \times 128$  spad-257 tdc receiver ic for pulsed tof solid-state 3-d imaging,” *IEEE Journal of Solid-State Circuits*, pp. 1–11, 2020.
- [66] Arin Can Ulku, Claudio Bruschini, Ivan Michel Antolovic, Yung Kuo, Rinat Ankri, Shimon Weiss, Xavier Michalet, and Edoardo Charbon, “A  $512 \times 512$  spad image sensor with integrated gating for widefield flim,” *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 25, no. 1, 2019.
- [67] Kazuhiro Morimoto, Andrei Ardelean, Ming-Lo Wu, Arin Can Ulku, Ivan Michel Antolovic, Claudio Bruschini, and Edoardo Charbon, “Megapixel time-gated spad image sensor for 2d and 3d imaging applications,” *Optica*, vol. 7, no. 4, 2020.
- [68] Carsten Degenhardt, Ben Zwaans, Thomas Frach, and Rik De Gruyter, “Arrays of digital silicon photomultipliers -intrinsic performance and application to scintillator readout,” *IEEE Nuclear Science Symposium Conference Record*, pp. 1954–1956, 2010.
- [69] Mark A. Itzler, Mark Entwistle, Mark Owens, Xudong Jiang, Ketan Patel, Krystyna Slomkowski, Tim Koch, Sabbir Rangwala, Peter F. Zalud, Young

- Yu, John Tower, and Joseph Ferraro, “Inp-based geiger-mode avalanche photodiode arrays for three-dimensional imaging at  $1.06\ \mu\text{m}$ ,” in *Advanced Photon Counting Techniques III*, Mark A. Itzler and Joe C. Campbell, Eds. May 2009, vol. 7320, p. 73200O, SPIE.
- [70] Mark A. Itzler, Mark Entwistle, Mark Owens, Ketan Patel, Xudong Jiang, Krystyna Slomkowski, Sabbir Rangwala, Peter F. Zalud, Tom Senko, John Tower, and Joseph Ferraro, “Geiger-mode avalanche photodiode focal plane arrays for three-dimensional imaging lidar,” in *Infrared Remote Sensing and Instrumentation XVIII*, Marija Strojnik and Gonzalo Paez, Eds. Aug. 2010, vol. 7808, p. 78080C, SPIE.
- [71] Philip A. Hiskett, Karen J. Gordon, Jeremy W. Copley, and Robert A. Lamb, “Long range 3d imaging with a  $32\times 32$  geiger mode ingaas/inp camera,” in *Advanced Photon Counting Techniques VIII*. May 2014, vol. 9114, p. 91140I, SPIE.
- [72] Amir Sammak, Mahdi Aminian, Lis K. Nanver, and Edoardo Charbon, “Cmos-compatible puregab ge-on-si apd pixel arrays,” *IEEE Transactions on Electron Devices*, vol. 63, no. 1, pp. 92–99, Jan. 2016.
- [73] Kenji Makino, Takuya Fujita, Tatsuya Hashi, Shunsuke Adachi, Shigeyuki Nakamura, Koei Yamamoto, Takashi Baba, and Yoshihito Suzuki, “Development of an ingaas spad 2d array for flash lidar,” in *Quantum Sensing and Nano Electronics and Photonics XV*, 2018, p. 19.
- [74] Peter Vines, Kateryna Kuzmenko, Jarosław Kirdoda, Derek C. S. Dumas, Muhammad M. Mirza, Ross W. Millar, Douglas J. Paul, and Gerald S. Buller, “High performance planar germanium-on-silicon single-photon avalanche diode detectors,” *Nature Communications*, vol. 10, no. 1, pp. 1086, Dec. 2019.
- [75] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao, “Stable signal recovery from incomplete and inaccurate measurements,” *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, Aug. 2006.
- [76] David L. Donoho, “Compressed sensing,” *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.
- [77] J. Romberg, “Imaging via compressive sampling,” *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14–20, Mar. 2008.

- [78] Wolfgang Wagner, Andreas Ullrich, Vesna Ducic, Thomas Melzer, and Nick Studnicka, “Gaussian decomposition and calibration of a novel small-footprint full-waveform digitising airborne laser scanner,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 60, no. 2, pp. 100–112, 2006.
- [79] Davide Maria Perfetto, Roberto Opromolla, Michele Grassi, and Christoph Schmitt, “Lidar-based model reconstruction for spacecraft pose determination,” in *2019 IEEE International Workshop on Metrology for AeroSpace, MetroAeroSpace 2019 - Proceedings*. June 2019, pp. 1–6, Institute of Electrical and Electronics Engineers Inc.
- [80] A M Wallace, S Buller, and A C Walker, “3d imaging and ranging by time-correlated single photon counting,” *Computing & Control Engineering Journal*, vol. 12, no. August, pp. 157–168, 2001.
- [81] Velodyne LiDAR, “Velodyne hdl-64e s3,” 2017.
- [82] Andreas Aßmann, Brian Stewart, João F.C. Mota, and Andrew M. Wallace, “Compressive super-pixel lidar for high-framerate 3d depth imaging,” in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Ottawa, Canada, 2019.
- [83] Alexander Carballo, Jacob Lambert, Abraham Monrroy, David Wong, Patiphon Narksri, Yuki Kitsukawa, Eijiro Takeuchi, Shinpei Kato, and Kazuya Takeda, “Libre: The multiple 3d lidar dataset,” *arXiv*, Mar. 2020.
- [84] Hall David, “128 lasers on the car go round and round: David hall on velodyne’s new sensor - velodyne lidar 360 blog,” 2017.
- [85] Velodyne LiDAR, “Velodyne lidar puck vlp-16 spec sheet,” Tech. Rep., 2017.
- [86] Ibeo Automotive Systems GmbH, “Lux datasheet,” Tech. Rep., 2017.
- [87] Ouster Inc., “Os2 long-range high-resolution imaging lidar datasheet,” Tech. Rep., 2020.
- [88] Riegl Laser Measurement Systems GmbH, “Riegl vux-1lr datasheet,” Tech. Rep.
- [89] Ibeo Automotive Systems GmbH, “Next solid-state lidar datasheet,” Tech. Rep., 2019.
- [90] Ouster Inc., “Es2 datasheet,” Tech. Rep., 2020.
- [91] Sara Pellegrini, Gerald S Buller, Jason M Smith, Andrew M Wallace, and Sergio Cova, “Laser-based distance measurement using picosecond resolution

- time-correlated single-photon counting,” *Meas. Sci. Technol.*, vol. 11, no. 00, pp. 712–716, 2000.
- [92] Wolfgang Wagner, Andreas Ullrich, T. Melzer, Christian Brieke, and K. Kraus, “From single-pulse to full-waveform scanners: Potential and practical challenges,” *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 35, pp. 201–206, 2004.
- [93] W Wagner, A Roncat, T Melzer, and A Ullrich, “Waveform analysis techniques in airborne laser scanning,” in *International Archives of Photogrammetry and Remote Sensing*, 2007, vol. XXXVI, pp. 413–418.
- [94] Martin Pfennigbauer, Clifford Wolf, Josef Weindopf, and Andreas Ullrich, “Online waveform processing for demanding target situations,” *Proc. SPIE 9080: Laser Radar Technology and Applications XIX and Atmospheric Propagation XI*, 2014.
- [95] C Mallet, U Soergel, and F Bretar, “Analysis of full-waveform lidar data for classification of urban areas,” *Photogrammetrie - Fernerkundung - Geoinformation*, vol. XXXVII, no. Part B3a, pp. 2–4, 2008.
- [96] Ayush Bhandari and Ramesh Raskar, “Signal processing for time-of-flight imaging sensors: An introduction to inverse problems in computational 3-d imaging,” *IEEE Signal Processing Magazine*, vol. 33, no. 5, pp. 45–58, Sept. 2016.
- [97] Andreas Aßmann, Brian Stewart, and Andrew M. Wallace, “Deep learning for lidar waveforms with multiple returns,” in *2020 28th European Signal Processing Conference (EUSIPCO)*, Amsterdam, 2020, p. 1571.
- [98] Karolina D Fieber, Ian J Davenport, James M Ferryman, Robert J Gurney, Jeffrey P Walker, and Jorg M Hacker, “Analysis of full-waveform lidar data for classification of an orange orchard scene,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 82, pp. 63–82, 2013.
- [99] Puneet Chhabra, Aurora Maccarone, Aongus McCarthy, Gerald Buller, and Andrew Wallace, “Discriminating underwater lidar target signatures using sparse multi-spectral depth codes,” in *2016 Sensor Signal Processing for Defence, SSPD 2016*, 2016.
- [100] Sergio Hernandez-Marin, Andrew M. Wallace, and Gavin J. Gibson, “Multi-layered 3d lidar image construction using spatial models in a bayesian frame-

- work,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 1028–1040, 2008.
- [101] Jing Ye, Andrew Wallace, and John Thompson, “Parallel markov chain monte carlo computation for varying-dimension signal analysis,” in *European Signal Processing Conference*, 2009, pp. 2673–2677.
- [102] Andrew M. Wallace, Jing Ye, Nils J. Krichel, Aongus McCarthy, Robert J. Collins, and Gerald S. Buller, “Full waveform analysis for long-range 3d imaging laser radar,” *Eurasip Journal on Advances in Signal Processing*, vol. 2010, 2010.
- [103] Jing Ye, Andrew M. Wallace, Abdallah Al Zain, and John Thompson, “Parallel bayesian inference of range and reflectance from ladar profiles,” *Journal of Parallel and Distributed Computing*, vol. 73, no. 4, pp. 383–399, Apr. 2013.
- [104] Dongeek Shin, Feihu Xu, Franco N. C. Wong, Jeffrey H. Shapiro, and Vivek K Goyal, “Computational multi-depth single-photon imaging,” *Optics Express*, vol. 24, no. 3, pp. 1873, Feb. 2016.
- [105] Yoann Altmann, Reuben Aspden, Miles Padgett, and Steve McLaughlin, “A bayesian approach to denoising of single-photon binary images,” *IEEE Transactions on Computational Imaging*, vol. 3, no. 3, pp. 460–471, Sept. 2017.
- [106] Julián Tachella, Yoann Altmann, Ximing Ren, Aongus McCarthy, Gerald S. Buller, Stephen McLaughlin, and Jean Yves Tourneret, “Bayesian 3d reconstruction of complex scenes from single-photon lidar data,” *SIAM Journal on Imaging Sciences*, vol. 12, no. 1, 2019.
- [107] J. Castorena, C.D. Creusere, and D. Voelz, “Using finite moment rate of innovation for lidar waveform complexity estimation,” in *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*. Nov. 2010, pp. 608–612, IEEE.
- [108] Juan Castorena and Charles D Creusere, “Compressive sampling of lidar: Full-waveforms as signals of finite rate of innovation,” in *20th European Signal Processing Conference (EUSIPCO 2012)*, 2012.
- [109] Yingbo Hua and Tapan K. Sarkar, “Matrix pencil method for estimating parameters of exponentially damped/undamped sinusoids in noise,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 5, pp. 814–824, May 1990.

- 
- [110] Ayush Bhandari, Andrew M Wallace, and Ramesh Raskar, “Super-resolved time-of-flight sensing via fri sampling theory,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 4009–4013, IEEE.
- [111] Donggeek Shin, Ahmed Kirmani, Vivek K Goyal, and Jeffrey H. Shapiro, “Photon-efficient computational 3-d and reflectivity imaging with single-photon detectors,” *IEEE Transactions on Computational Imaging*, vol. 1, no. 2, pp. 112–125, June 2015.
- [112] Abderrahim Halimi, Yoann Altmann, Aongus McCarthy, Ximing Ren, Rachael Tobin, Gerald Stuart Buller, and Stephen McLaughlin, “Restoration of intensity and depth images constructed using sparse single-photons data,” *24th European Signal Processing Conference 2016*, pp. 86–90, Aug. 2016.
- [113] Joshua Rapp and Vivek K Goyal, “A few photons among many: Unmixing signal and noise for photon-efficient active imaging,” *IEEE Transactions on Computational Imaging*, vol. 3, no. 3, pp. 445–459, May 2017.
- [114] Duan Li, Lijun Xu, and Xiaolu Li, “Full-waveform lidar echo decomposition based on wavelet decomposition and particle swarm optimization,” *Measurement Science and Technology*, vol. 28, no. 4, 2017.
- [115] Abderrahim Halimi, Rachael Tobin, Aongus McCarthy, Jose Bioucas-Dias, Stephen McLaughlin, and Gerald S. Buller, “Robust restoration of sparse multidimensional single-photon lidar images,” *IEEE Transactions on Computational Imaging*, vol. 6, pp. 138–152, July 2019.
- [116] Julián Tachella, Yoann Altmann, Nicolas Mellado, Aongus McCarthy, Rachael Tobin, Gerald S. Buller, Jean-Yves Tournet, and Stephen McLaughlin, “Real-time 3d reconstruction from single-photon lidar data using plug-and-play point cloud denoisers,” *Nature Communications*, vol. 10, no. 1, pp. 4984, Dec. 2019.
- [117] Andreas Geiger, Philip Lenz, and Raquel Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” *Computer Vision and Pattern Recognition*, pp. 3354–3361, 2012.
- [118] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov,

- “Scalability in perception for autonomous driving: Waymo open dataset,” *arXiv*, Dec. 2019.
- [119] A. Hovi and I. Korpela, “Real and simulated waveform-recording lidar data in juvenile boreal forest vegetation,” *Remote Sensing of Environment*, vol. 140, pp. 665–678, Jan. 2014.
  - [120] Jean-Philippe Gastellu-Etchegorry, Tiangang Yin, Nicolas Lauret, Eloi Grau, Jeremy Rubio, Bruce D. Cook, Douglas C. Morton, and Guoqing Sun, “Simulation of satellite, airborne and terrestrial lidar with dart (i): Waveform simulation with quasi-monte carlo ray tracing,” *Remote Sensing of Environment*, vol. 184, pp. 418–435, Oct. 2016.
  - [121] Steven Hancock, John Armston, Michelle Hofton, Xiaoli Sun, Hao Tang, Laura I. Duncanson, James R. Kellner, and Ralph Dubayah, “The gedi simulator: A large-footprint waveform lidar simulator for calibration and validation of spaceborne missions,” *Earth and Space Science*, vol. 6, no. 2, pp. 294–310, Feb. 2019.
  - [122] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig, “Virtual worlds as proxy for multi-object tracking analysis,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4340–4349.
  - [123] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez, “The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3234–3243.
  - [124] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus, “Indoor segmentation and support inference from rgb-d images,” in *European Conference on Computer Vision*, Florence, Italy, 2012, pp. 746–760, Springer, Berlin, Heidelberg.
  - [125] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun, “Carla: An open urban driving simulator,” *arXiv*, , no. CoRL, pp. 1–16, 2017.
  - [126] Wolfgang Wagner, “Radiometric calibration of small-footprint full-waveform airborne laser scanner measurements: Basic physical concepts,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 65, no. 6, pp. 505–513, 2010.
  - [127] A M Baldridge, S J Hook, C I Grove, and G Rivera, “The aster spectral library version 2.0,” *Remote Sensing of Environment*, vol. 113, no. 4, pp. 711–715,

- 2009.
- [128] Susan K. Meerdink, Simon J. Hook, Dar A. Roberts, and Elsa A. Abbott, “The ecostress spectral library version 1.0,” *Remote Sensing of Environment*, vol. 230, pp. 111196, Sept. 2019.
  - [129] Lu Gan, “Block compressed sensing of natural images,” in *2007 15th International Conference on Digital Signal Processing*. July 2007, pp. 403–406, IEEE.
  - [130] F Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.
  - [131] D. H. Hubel and T. N. Wiesel, “Receptive fields of single neurones in the cat’s striate cortex,” *The Journal of Physiology*, vol. 148, no. 3, pp. 574–591, Oct. 1959.
  - [132] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *The Journal of Physiology*, vol. 160, no. 1, pp. 106–154, Jan. 1962.
  - [133] Kunihiro Fukushima, “Cognitron: A self-organizing multilayered neural network,” *Biological Cybernetics*, vol. 20, no. 3-4, pp. 121–136, Sept. 1975.
  - [134] Marvin Minsky and Seymour Papert, *Perceptrons*, MIT Press, Cambridge, MA, USA, 2017 edition, 1969.
  - [135] Robert R. Schaller, “Moore’s law: past, present, and future,” *IEEE Spectrum*, vol. 34, no. 6, pp. 52–55, 57, June 1997.
  - [136] Elie Track, Nancy Forbes, and George Strawn, “The end of moore’s law,” *Computing in Science and Engineering*, vol. 19, no. 2, pp. 4–6, Mar. 2017.
  - [137] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, “Imagenet classification with deep convolutional neural networks,” in *IEEE Conference on Neural Information Systems (NIPS)*, 2012.
  - [138] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas,



- Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, “Tensorflow: Large-scale machine learning on heterogeneous systems,” 2015.
- [139] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [140] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1989.
- [141] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.
- [142] Yoshua Bengio, “Learning deep architectures for ai,” *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
- [143] Kunihiro Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, Apr. 1980.
- [144] L. D. Le Cun Jackel, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, Bb Le Cun, Js Denker, and D. Henderson, “Handwritten digit recognition with a back-propagation network,” *Advances in Neural Information Processing Systems*, pp. 396–404, 1990.
- [145] Vinod Nair and Geoffrey E Hinton, “Rectified linear units improve restricted boltzmann machines,” in *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, 2010, pp. 807–814.
- [146] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, “On rectified linear units for speech processing,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Oct. 2013, pp. 3517–3521.
- [147] George E. Dahl, Tara N. Sainath, and Geoffrey E. Hinton, “Improving deep neural networks for lvcsr using rectified linear units and dropout,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Oct. 2013, pp. 8609–8613.

- 
- [148] J. Weng, N. Ahuja, and T.S. Huang, “Cresceptron: a self-organizing neural network which grows adaptively,” in *International Joint Conference on Neural Networks (IJCNN)*. Jan. 1992, pp. 576–581, Institute of Electrical and Electronics Engineers (IEEE).
- [149] John Juyang Weng, Narendra Ahuja, and Thomas S Huang, “Learning recognition and segmentation using the cresceptron,” *International Journal of Computer Vision*, vol. 25, no. 2, pp. 109–143, 1997.
- [150] Dominik Scherer, Andreas Müller, and Sven Behnke, “Evaluation of pooling operations in convolutional architectures for object recognition,” *IEEE International Conference on Artificial Neural Networks (ICANN)*, vol. 20, 2010.
- [151] Dan C Ciresan, Ueli Meier, Jonathan Masci, Luca M Gambardella, and J Urgan Schmidhuber, “Flexible, high performance convolutional neural networks for image classification,” in *22nd International joint conference on Artificial Intelligence*, 2011.
- [152] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab, “Deeper depth prediction with fully convolutional residual networks,” in *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016*, 2016, pp. 239–248.
- [153] Christopher A. Metzler, Ali Mousavi, and Richard G. Baraniuk, “Learned d-amp: A principled cnn-based compressive image recovery algorithm,” in *arXiv*, Apr. 2017, pp. 1–18.
- [154] Laurie Bose, Jianing Chen, Stephen J. Carey, Piotr Dudek, and Walterio Mayol-Cuevas, “A camera that cnns: Towards embedded neural networks on pixel processor arrays,” *ArXiv*, Sept. 2019.
- [155] Corneliu T.C. Arsene, Richard Hankins, and Hujun Yin, “Deep learning models for denoising ecg signals,” in *2019 27th European Signal Processing Conference (EUSIPCO)*. Sept. 2019, pp. 1–5, IEEE.
- [156] Geoffrey E. Hinton and Ruslan R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, July 2006.
- [157] Leslie Casas, Nassir Navab, and Vasileios Belagiannis, “Adversarial signal denoising with encoder-decoder networks,” *arXiv*, Dec. 2018.

- 
- [158] Antonia Creswell and Anil Anthony Bharath, “Denoising adversarial autoencoders,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 4, pp. 968–984, Apr. 2019.
- [159] Yongtao Yu, Jonathan Li, Haiyan Guan, Fukai Jia, and Cheng Wang, “Learning hierarchical features for automated extraction of road markings from 3-d mobile lidar point clouds,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 2, pp. 709–726, Feb. 2015.
- [160] Yin Zhou and Oncel Tuzel, “Voxelnet: End-to-end learning for point cloud based 3d object detection,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Dec. 2018, pp. 4490–4499, IEEE Computer Society.
- [161] Luca Caltagirone, Samuel Scheidegger, Lennart Svensson, and Mattias Wahde, “Fast lidar-based road detection using fully convolutional neural networks,” in *IEEE Intelligent Vehicles Symposium, Proceedings*. July 2017, pp. 1019–1024, Institute of Electrical and Electronics Engineers Inc.
- [162] Zhenfeng Shao, Linjing Zhang, and Lei Wang, “Stacked sparse autoencoder modeling using the synergy of airborne lidar and satellite optical and sar data to map forest above-ground biomass,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 12, pp. 5569–5582, Dec. 2017.
- [163] Benzhang Wang, Yiliu Feng, and Hengzhu Liu, “Multi-scale features fusion from sparse lidar data and single image for depth completion,” *Electronics Letters*, vol. 54, no. 24, pp. 1375–1377, Nov. 2018.
- [164] Narada Warakagoda, Johann Dirdal, and Erlend Faxvaag, “Fusion of lidar and camera images in end-to-end deep learning for steering an off-road unmanned ground vehicle,” in *FUSION 2019 - 22nd International Conference on Information Fusion*, 2019.
- [165] Gledson Melotti, Cristiano Premebida, and Nuno Goncalves, “Multimodal deep-learning for object recognition combining camera and lidar data,” in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2020*. Apr. 2020, pp. 177–182, Institute of Electrical and Electronics Engineers Inc.
- [166] Serkan Kiranyaz, Turker Ince, Ridha Hamila, and Moncef Gabbouj, “Convolutional neural networks for patient-specific ecg classification,” in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine*

- and Biology Society, EMBS*. Nov. 2015, vol. 2015-Novem, pp. 2608–2611, Institute of Electrical and Electronics Engineers Inc.
- [167] Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj, “Real-time patient-specific ecg classification by 1-d convolutional neural networks,” *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 3, pp. 664–675, Mar. 2016.
  - [168] Takayuki Shinohara, Haoyi Xiu, and Masashi Matsuoka, “Fwnetae: Spatial representation learning for full waveform data using deep learning,” in *Proceedings - 2019 IEEE International Symposium on Multimedia, ISM 2019*. Dec. 2019, pp. 259–266, Institute of Electrical and Electronics Engineers Inc.
  - [169] Gangping Liu and Jun Ke, “Deep-learning for super-resolution full-waveform lidar,” in *Optoelectronic Imaging and Multimedia Technology VI*, Qionghai Dai, Tsutomu Shimura, and Zhenrong Zheng, Eds. Nov. 2019, vol. 11187, p. 38, SPIE.
  - [170] Alex Turpin, Gabriella Musarra, Valentin Kapitany, Francesco Tonolini, Ashley Lyons, Ilya Starshynov, Federica Villa, Enrico Conca, Francesco Fioranelli, Roderick Murray-Smith, and Daniele Faccio, “Spatial images from temporal data,” *Optica*, vol. 7, no. 8, pp. 900, Aug. 2020.
  - [171] Paul Kirkland, Valentin Kapitany, Ashley Lyons, John Soraghan, Alex Turpin, Daniele Faccio, and Gaetano Di Caterina, “Imaging from temporal data via spiking convolutional neural networks,” in *Emerging Imaging and Sensing Technologies for Security and Defence V; and Advanced Manufacturing Technologies for Micro- and Nanosystems in Security and Defence III*. Sept. 2020, vol. 11540, p. 15, SPIE.
  - [172] Stefano Zorzi, Eleonora Maset, Andrea Fusiello, and Fabio Crosilla, “Full-waveform airborne lidar data classification using convolutional neural networks,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 10, pp. 8255–8261, Oct. 2019.
  - [173] Ryan E. Warburton, Aongus McCarthy, Andrew M. Wallace, Sergio Hernandez-Marin, Robert H. Hadfield, Sae Woo Nam, and Gerald S. Buller, “Subcentimeter depth resolution using a single-photon counting time-of-flight laser ranging system at 1550 nm wavelength,” *Optics Letters*, vol. 32, no. 15, pp. 2266, 2007.
  - [174] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv*, July 2012.

- 
- [175] Ralph Neuneier and Hans Georg Zimmermann, *How to Train Neural Networks*, Springer, Berlin, Heidelberg, 2012.
- [176] Didier Le Gall, “Mpeg: a video compression standard for multimedia applications,” *Communications of the ACM*, vol. 34, no. 4, pp. 46–58, 1991.
- [177] G.K. Wallace, “The jpeg still picture compression standard,” *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.
- [178] Emmanuel J. Candes and Terence Tao, “Decoding by linear programming,” *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.
- [179] Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin, “A simple proof of the restricted isometry property for random matrices,” *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, Dec. 2008.
- [180] Zachary T Harmany, Roummel F Marcia, and Rebecca M Willett, “This is spiral-tap: Sparse poisson intensity reconstruction algorithms-theory and practice,” *IEEE Transactions on Image Processing*, vol. 21, no. 3, pp. 1084–1096, 2012.
- [181] Richard Baraniuk, “Compressive sensing [lecture notes],” *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118–121, July 2007.
- [182] Michael Lustig, David L. Donoho, Juan M. Santos, and John M. Pauly, “Compressed sensing mri,” *IEEE Signal Processing Magazine*, vol. 25, no. March 2008, pp. 72–82, Mar. 2008.
- [183] M. Murphy, M. Alley, J. Demmel, K. Keutzer, S. Vasanawala, and M. Lustig, “Fast l1-spirit compressed sensing parallel imaging mri: Scalable parallel implementation and clinically feasible runtime,” *IEEE Transactions on Medical Imaging*, vol. 31, no. 6, pp. 1250–1262, June 2012.
- [184] Shahrooz Faghih Roohi, Dornoosh Zonoobi, Ashraf A. Kassim, and Jacob L. Jaremko, “Dynamic mri reconstruction using low rank plus sparse tensor decomposition,” in *Proceedings - International Conference on Image Processing, ICIP*. Sept. 2016, vol. 2016-Augus, pp. 1769–1773, IEEE.
- [185] Loubna El Gueddari, C. Lazarus, H. Carrie, A. Vignaud, and Ph Ciuciu, “Self-calibrating nonlinear reconstruction algorithms for variable density sampling and parallel reception mri,” in *Proceedings of the IEEE Sensor Array and Multichannel Signal Processing Workshop*, 2018, vol. 2018-July, pp. 415–419.

- 
- [186] Baran Gözcü, Rabeeh Karimi Mahabadi, Yen Huan Li, Efe Ilicak, Tolga Çukur, Jonathan Scarlett, and Volkan Cevher, “Learning-based compressive mri,” *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1394–1406, June 2018.
  - [187] Morteza Mardani, Enhao Gong, Joseph Y. Cheng, Shreyas S. Vasanawala, Greg Zaharchuk, Lei Xing, and John M. Pauly, “Deep generative adversarial neural networks for compressive sensing mri,” *IEEE Transactions on Medical Imaging*, vol. 38, no. 1, pp. 167–179, Jan. 2019.
  - [188] Dong Du, Zhibin Pan, Penghui Zhang, Yuxin Li, and Weiping Ku, “Compressive sensing image recovery using dictionary learning and shape-adaptive dct thresholding,” *Magnetic Resonance Imaging*, vol. 55, pp. 60–71, Jan. 2019.
  - [189] Marco F. Duarte, Mark A. Davenport, Dharmpal Takhar, Jason N. Laska, Ting Sun, Kevin F. Kelly, and Richard G. Baraniuk, “Single-pixel imaging via compressive sampling,” *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 83–91, Mar. 2008.
  - [190] Emmanuel J. Candès and Terence Tao, “Near-optimal signal recovery from random projections: Universal encoding strategies?,” *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, Dec. 2006.
  - [191] Emmanuel J. Candès and Justin Romberg, “Sparsity and incoherence in compressive sampling,” *Inverse Problems*, vol. 23, pp. 969–985, 2007.
  - [192] Justin Romberg, “Compressive sensing by random convolution,” *SIAM Journal on Imaging Sciences*, vol. 2, no. 4, pp. 1098–1128, Jan. 2009.
  - [193] Marco F. Duarte and Yonina C. Eldar, “Structured compressed sensing: From theory to applications,” *IEEE Transactions on Signal Processing*, 2011.
  - [194] R. Coifman, F. Geshwind, and Y. Meyer, “Noiselets,” *Applied and Computational Harmonic Analysis*, vol. 10, no. 1, pp. 27–44, Jan. 2001.
  - [195] Alfred Haar, “Zur theorie der orthogonalen funktionensysteme,” *Mathematische Annalen*, vol. 69, no. 3, pp. 331–371, Sept. 1910.
  - [196] Ingrid Daubechies, “Orthonormal bases of compactly supported wavelets,” *Communications on Pure and Applied Mathematics*, vol. 41, no. 7, pp. 909–996, 1988.
  - [197] N. Ahmed, T. Natarajan, and K.R. Rao, “Discrete cosine transform,” *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, Jan. 1974.

- 
- [198] Leonid I. Rudin, Stanley Osher, and Emad Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D: Nonlinear Phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.
- [199] Antonin Chambolle, “An algorithm for total variation minimization and applications,” *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1-2, pp. 89–97, 2004.
- [200] Alfred M. Bruckstein, David L. Donoho, and Michael Elad, “From sparse solutions of systems of equations to sparse modeling of signals and images,” *SIAM Review*, vol. 51, no. 1, pp. 34–81, Feb. 2009.
- [201] Venkat Chandrasekaran, Benjamin Recht, Pablo A. Parrilo, and Alan S. Willsky, “The convex geometry of linear inverse problems,” *Foundations of Computational Mathematics*, vol. 12, no. 6, pp. 805–849, Dec. 2012.
- [202] João F.C. Mota, Nikos Deligiannis, and Miguel R.D. Rodrigues, “Compressed sensing with prior information: Strategies, geometry, and bounds,” *IEEE Transactions on Information Theory*, vol. 63, no. 7, pp. 4472–4496, 2017.
- [203] William K. Pratt, Harry C. Andrews, and Julius Kane, “Hadamard transform image coding,” *Proceedings of the IEEE*, vol. 57, no. 1, pp. 58–68, 1969.
- [204] Ming-Jie Sun, Matthew. P. Edgar, Graham M. Gibson, Baoqing Sun, Neal Radwell, Robert Lamb, and Miles J. Padgett, “Single-pixel 3d imaging with time-based depth resolution,” *Nature Communications*, vol. 7, no. May, pp. 1–10, 2016.
- [205] Ronald A. DeVore, “Deterministic constructions of compressed sensing matrices,” *Journal of Complexity*, vol. 23, no. 4-6, pp. 918–925, Aug. 2007.
- [206] Weizhi Lu, Weiyu Li, Kidiyo Kpalma, and Joseph Ronsin, “Near-optimal binary compressed sensing matrix,” *arXiv*, 2013.
- [207] Pradip Sasmal, R. Ramu Naidu, Challa S. Sastry, and Phanindra Jampana, “Composition of binary compressed sensing matrices,” *IEEE Signal Processing Letters*, vol. 23, no. 8, pp. 1096–1100, Aug. 2016.
- [208] Mahsa Lotfi and Mathukumalli Vidyasagar, “Compressed sensing using binary matrices of nearly optimal dimensions,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 3008–3021, Aug. 2020.
- [209] James E. Fowler, Sungkwang Mun, and Eric W. Tramel, “Block-based compressed sensing of images and video,” *Foundations and Trends in Signal Processing*, vol. 4, no. 4, pp. 297–416, 2012.

- 
- [210] Jarvis Haupt and Robert Nowak, “Signal reconstruction from noisy random projections,” *IEEE Transactions on Information Theory*, vol. 52, no. 9, pp. 4036–4048, Sept. 2006.
- [211] José M. Bioucas-Dias and Mário A.T. Figueiredo, “A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration,” *IEEE Transactions on Image Processing*, vol. 16, no. 12, pp. 2992–3004, Dec. 2007.
- [212] Amir Beck and Marc Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.
- [213] Sungkwang Mun and James E. Fowler, “Block compressed sensing of images using directional transforms,” in *Proceedings - International Conference on Image Processing, ICIP*. Nov. 2009, pp. 3021–3024, IEEE.
- [214] Trac D. Tran, Jie Liang, and Chengjie Tu, “Lapped transform via time-domain pre- and post-filtering,” *IEEE Transactions on Signal Processing*, vol. 51, no. 6, pp. 1557–1571, June 2003.
- [215] Ali Akbari and Maria Trocan, “Robust image reconstruction for block-based compressed sensing using a binary measurement matrix,” in *2018 25th IEEE International Conference on Image Processing (ICIP)*. Oct. 2018, pp. 1832–1836, IEEE.
- [216] Ahmed Kirmani, Andrea Colaço, Franco N. C. Wong, and Vivek K. Goyal, “Exploiting sparsity in time-of-flight range acquisition using a single time-resolved sensor,” *Optics Express*, vol. 19, no. 22, pp. 21485, Oct. 2011.
- [217] Petros T. Boufounos, “Depth sensing using active coherent illumination,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2012, pp. 5417–5420, IEEE.
- [218] Ahmed Kirmani, Andrea Colaco, Franco N. C. Wong, and Vivek K Goyal, “Co-dac: A compressive depth acquisition camera framework,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2012, pp. 5425–5428, IEEE.
- [219] Fengqiang Li, Huaijin Chen, Adithya Pediredla, Chiakai Yeh, Kuan He, Ashok Veeraraghavan, and Oliver Cossairt, “Cs-tof: High-resolution compressive time-of-flight imaging,” *Optics Express*, vol. 25, no. 25, pp. 31096, Dec. 2017.



- 
- [220] Daniel J. Lum, Samuel H. Knarr, and John C. Howell, “Frequency-modulated continuous-wave lidar compressive depth-mapping,” *Optics Express*, vol. 26, no. 12, 2018.
- [221] Achuta Kadambi and Petros T. Boufounos, “Coded aperture compressive 3-d lidar,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. Apr. 2015, vol. Kadambi201, pp. 1166–1170, IEEE.
- [222] Achuta Kadambi, Refael Whyte, Ayush Bhandari, Lee Streeter, Christopher Barsi, Adrian Dorrington, and Ramesh Raskar, “Coded time of flight cameras : Sparse deconvolution to address multipath interference and recover time profiles,” *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 1–10, 2013.
- [223] Petros T Boufounos, “High-resolution lidar using random demodulation,” *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 36–40, 2018.
- [224] Stephan Antholzer, Christoph Wolf, Michael Sandbichler, Markus Dielacher, and Markus Haltmeier, “Compressive time-of-flight 3d imaging using block-structured sensing matrices,” *Inverse Problems*, vol. 35, no. 4, pp. 045004, Apr. 2019.
- [225] Darryl Sale, Christopher J. Rozell, Justin K. Romberg, and Aaron D. Lanterman, “Compressive lidar in realistic environments,” in *2012 IEEE Statistical Signal Processing Workshop, SSP 2012*. Aug. 2012, pp. 720–723, IEEE.
- [226] H Yu, E Li, W Gong, and S Han, “Structured image reconstruction for three-dimensional ghost imaging lidar,” *Opt Express*, vol. 23, no. 11, pp. 14541–14551, 2015.
- [227] Matthew Edgar, Steven Johnson, David Phillips, and Miles Padgett, “Real-time computational photon-counting lidar,” *Optical Engineering*, vol. 57, no. 03, pp. 1, Dec. 2017.
- [228] Andrea Colaco, Ahmed Kirmani, Gregory A. Howland, John C. Howell, and Vivek K. Goyal, “Compressive depth map acquisition using a single photon-counting detector: Parametric signal processing meets sparsity,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. June 2012, pp. 96–102, IEEE.
- [229] Gregory Howland, Petros Zerom, Robert W Boyd, and John C Howell, “Compressive sensing lidar for 3d imaging,” in *CLEO:2011 - Laser Applications to*

- Photonic Applications*, Washington, D.C., 2011, number 2, p. CMG3, OSA.
- [230] G A Howland, P B Dixon, and J C Howell, “Photon-counting compressive sensing laser radar for 3d imaging,” *Applied Optics*, vol. 50, no. 31, pp. 5917, 2011.
  - [231] Gregory A. Howland, Daniel J. Lum, Matthew R. Ware, and John C. Howell, “Photon counting compressive depth mapping,” *Optics Express*, vol. 21, no. 20, pp. 23822, Sept. 2013.
  - [232] Chengbo Li, Wotao Yin, and Yin Zhang, “Tval3: Tv minimization by augmented lagrangian and alternating direction algorithms,” 2010.
  - [233] David L. Donoho and Iain M. Johnstone, “Threshold selection for wavelet shrinkage of noisy data,” *Engineering in Medicine and Biology Society, 1994. Engineering Advances: New Opportunities for Biomedical Engineers. Proceedings of the 16th Annual International Conference of the IEEE*, pp. A24—A25, 1994.
  - [234] Mario A. T. Figueiredo, Robert D. Nowak, and Stephen J. Wright, “Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems, iee j,” *Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 586–597, 2007.
  - [235] Rebecca M. Willet, Roummel F. Marcia, and Jonathan M. Nichols, “Compressed sensing for practical optical imaging systems: a tutorial,” *Optical Engineering*, vol. 50, no. 7, pp. 072601, July 2011.
  - [236] Dmitry Popov, Artem Gapochkin, and Alexey Nekrasov, “An algorithm of daubechies wavelet transform in the final field when processing speech signals,” *Electronics (Switzerland)*, vol. 7, no. 7, pp. 120, July 2018.
  - [237] Stephen Boyd, Neil Parikh, Eric Chu, and Borja Peleato, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.
  - [238] Emmanuel J. Candes and Justin K. Romberg, “Signal recovery from random projections,” in *Computational Imaging III*, Charles A. Bouman and Eric L. Miller, Eds. Mar. 2005, vol. 5674, p. 76, International Society for Optics and Photonics.
  - [239] Andreas Aßmann, Yun Wu, Brian Stewart, and Andrew M. Wallace, “Accelerated 3d image reconstruction for resource constrained systems,” in *2020*

- 28th European Signal Processing Conference (EUSIPCO)*, Amsterdam, 2020, p. 565.
- [240] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
  - [241] Emmanuel J. Candès, Justin Romberg, and Terence Tao, “Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information,” *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.
  - [242] Tao Wan, Nishan Canagarajah, and Alin Achim, “Compressive image fusion,” in *2008 15th IEEE International Conference on Image Processing*. 2008, pp. 1308–1311, IEEE.
  - [243] Mário A.T. Figueiredo and José M Bioucas-Dias, “Restoration of poissonian images using alternating direction optimization,” *IEEE Transactions on Image Processing*, vol. 19, no. 12, pp. 3133–3145, Dec. 2010.
  - [244] Jie Xu, Jianwei Ma, Dongming Zhang, Yongdong Zhang, and Shouxun Lin, “Improved total variation minimization method for compressive sensing by intra-prediction,” *Signal Processing*, vol. 92, no. 11, pp. 2614–2623, Nov. 2012.
  - [245] S. Vishnukumar and M. Wilscy, “Single image super-resolution based on compressive sensing and improved tv minimization sparse recovery,” *Optics Communications*, vol. 404, pp. 80–93, Dec. 2017.
  - [246] Marija Vella and João F. C. Mota, “Single image super-resolution via cnn architectures and tv-tv minimization,” in *British Machine Vision Conference (BMVC)*, Birmingham, United Kingdom, July 2019.
  - [247] Marija Vella and João F. C. Mota, “Robust single-image super-resolution via cnns and tv-tv minimization,” *ArXiv*, Apr. 2020.
  - [248] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad, “Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition,” in *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*. 1993, pp. 40–44, IEEE Comput. Soc. Press.
  - [249] Neale Dutton, Johannes Vergote, Salvatore Gnechhi, Lindsay Grant, David Lee, Sara Pellegrini, Bruce Rae, and Robert Henderson, “Multiple-event direct

- to histogram tdc in 65nm fpga technology,” in *2014 Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, Grenoble, France, 2014.
- [250] David F Fouhey, Abhinav Gupta, and Martial Hebert, “Data-driven 3d primitives for single image understanding,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 3392–3399.
  - [251] David Eigen, Christian Puhrsch, and Rob Fergus, “Depth map prediction from a single image using a multi-scale deep network,” in *Advances in Neural Information Processing Systems*, 2014, vol. 3, pp. 2366–2374.
  - [252] David Eigen and Rob Fergus, “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, vol. 2015 Inter, pp. 2650–2658.
  - [253] Dan Xu, Wei Wang, Hao Tang, Hong Liu, Nicu Sebe, and Elisa Ricci, “Structured attention guided convolutional neural fields for monocular depth estimation,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Mar. 2018, pp. 3917–3925.
  - [254] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova, “Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 8001–8008, 2019.
  - [255] Eric A.G. Webster, Lindsay A. Grant, and Robert K. Henderson, “A high-performance single-photon avalanche diode in 130-nm cmos imaging technology,” *IEEE Electron Device Letters*, vol. 33, no. 11, pp. 1589–1591, Nov. 2012.
  - [256] Jie Han, “Introduction to approximate computing,” in *Proceedings of the IEEE VLSI Test Symposium*. May 2016, vol. 2016-May, IEEE Computer Society.
  - [257] Ankur Agrawal, Jungwook Choi, Kailash Gopalakrishnan, Suyog Gupta, Ravi Nair, Jinwook Oh, Daniel A. Prener, Sunil Shukla, Vijayalakshmi Srinivasan, and Zehra Sura, “Approximate computing: Challenges and opportunities,” in *2016 IEEE International Conference on Rebooting Computing, ICRC 2016 - Conference Proceedings*. Nov. 2016, Institute of Electrical and Electronics Engineers Inc.

- [258] Sparsh Mittal, “A survey of techniques for approximate computing,” *ACM Computing Surveys*, vol. 48, no. 4, 2016.
- [259] Geoffrey Yeap, X. Chen, B. R. Yang, C. P. Lin, F. C. Yang, Y. K. Leung, D. W. Lin, C. P. Chen, K. F. Yu, D. H. Chen, C. Y. Chang, S. S. Lin, H. K. Chen, P. Hung, C. S. Hou, Y. K. Cheng, J. Chang, L. Yuan, C. K. Lin, C. C. Chen, Y. C. Yeo, M. H. Tsai, Y. M. Chen, H. T. Lin, C. O. Chui, K. B. Huang, W. Chang, H. J. Lin, K. W. Chen, R. Chen, S. H. Sun, Q. Fu, H. T. Yang, H. L. Shang, H. T. Chiang, C. C. Yeh, T. L. Lee, C. H. Wang, S. L. Shue, C. W. Wu, R. Lu, W. R. Lin, J. Wu, F. Lai, P. W. Wang, Y. H. Wu, B. Z. Tien, Y. C. Huang, L. C. Lu, Jun He, Y. Ku, J. Lin, M. Cao, T. S. Chang, S. M. Jang, H. C. Lin, Y. C. Peng, J. Y. Sheu, and M. Wang, “5nm cmos production technology platform featuring full-fledged euv, and high mobility channel finfets with densest 0.021 $\mu\text{m}^2$  sram cells for mobile soc and high performance computing applications,” in *Technical Digest - International Electron Devices Meeting, IEDM*. Dec. 2019, vol. 2019-Decem, Institute of Electrical and Electronics Engineers Inc.
- [260] J. D. Meindl, Q. Chen, and J. A. Davis, “Limits on silicon nanoelectronics for terascale integration,” *Science*, vol. 293, no. 5537, pp. 2044–2049, 2001.
- [261] Robert W. Keyes, “Physical limits of silicon transistors and circuits,” *Reports on Progress in Physics*, vol. 68, no. 12, pp. 2701–2746, 2005.
- [262] J. P. Colinge, “Multigate transistors: Pushing moore’s law to the limit,” in *International Conference on Simulation of Semiconductor Processes and Devices, SISPAD*. Oct. 2014, pp. 313–316, Institute of Electrical and Electronics Engineers Inc.
- [263] Stelios Sidiroglou, Sasa Misailovic, Henry Hoffmann, and Martin Rinard, “Managing performance vs. accuracy trade-offs with loop perforation,” in *SIGSOFT/FSE 2011 - Proceedings of the 19th ACM SIGSOFT Symposium on Foundations of Software Engineering*, New York, New York, USA, 2011, pp. 124–134, ACM Press.
- [264] Qiang Xu, Todd Mytkowicz, and Nam Sung Kim, “Approximate computing: A survey,” *IEEE Design and Test*, vol. 33, no. 1, pp. 8–22, Feb. 2016.
- [265] Antonio G.M. Strollo and Darjn Esposito, “Approximate computing in the nanoscale era,” in *ICICDT 2018 - International Conference on IC Design and Technology, Proceedings*. June 2018, pp. 21–24, Institute of Electrical and Electronics Engineers Inc.

- 
- [266] Jonathan Ying Fai Tong, David Nagle, and Rob A. Rutenbar, “Reducing power by optimizing the necessary precision/range of floating-point arithmetic,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 273–286, 2000.
- [267] Thomas Y. Yeh, Petros Faloutsos, Milos Ercegovac, Sanjay J. Patel, and Glenn Reinman, “The art of deception: Adaptive precision reduction for area efficient physics acceleration,” in *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, 2007, pp. 394–406.
- [268] AW Brown, PHJ Kelly, and Wayne Luk, “Profiling floating point value ranges for reconfigurable implementation,” ... of the 1st HiPEAC Workshop on ..., pp. 1–11, 2007.
- [269] Renee St Amant, Amir Yazdanbakhsh, Jongse Park, Bradley Thwaites, Hadi Esmailzadeh, Arjang Hassibi, Luis Ceze, and Doug Burger, “General-purpose code acceleration with limited-precision analog computation,” in *Proceedings - International Symposium on Computer Architecture*, 2014, pp. 505–516.
- [270] Paulo Garcia, Deepayan Bhowmik, Robert Stewart, Greg Michaelson, and Andrew Wallace, “Optimized memory allocation and power minimization for fpga-based image processing,” *Journal of Imaging*, vol. 5, no. 1, pp. 7, Jan. 2019.
- [271] Abbas Rahimi, Luca Benini, and Rajesh K. Gupta, “Spatial memoization: Concurrent instruction reuse to correct timing errors in simd architectures,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 12, pp. 847–851, Dec. 2013.
- [272] Tuaha Nomani, Mujahid Mohsin, Zahid Pervaiz, and Muhammad Shafique, “Xuavs: Towards efficient approximate computing for uavs - low power approximate adders with single lut delay for fpga-based aerial imaging optimization,” *IEEE Access*, vol. 8, pp. 102982–102996, 2020.
- [273] Michael Lass, Thomas D. Kuhne, and Christian Plessl, “Using approximate computing for the calculation of inverse matrix inline-formula tex-math notation=“ $\sqrt[n]{x}$ ”,” *IEEE Embedded Systems Letters*, vol. 10, no. 2, pp. 33–36, June 2018.
- [274] Jorge Echavarria, Katja Schutz, Andreas Becher, Stefan Wildermann, and Jurgen Teich, “Can approximate computing reduce power consumption on fpgas?,” *2018 25th IEEE International Conference on Electronics Circuits and Systems, ICECS 2018*, pp. 841–844, 2019.

- [275] Yun Wu, F C Mota, and Andrew M Wallace, “Approximate lasso model predictive control for resource constrained systems,” in *International Conference in Sensor Signal Processing for Defence*, Edinburgh, 2020, number 1, pp. 1–6, IEEE.
- [276] Jie Han and Michael Orshansky, “Approximate computing: An emerging paradigm for energy-efficient design,” in *Proceedings - 2013 18th IEEE European Test Symposium, ETS 2013*, 2013.
- [277] James Bornholt, Todd Mytkowicz, and Kathryn S. McKinley, “Uncertain<t>: Abstractions for uncertain hardware and software,” *IEEE Micro*, vol. 35, no. 3, pp. 132–143, May 2015.
- [278] Microprocessor Standards Committee of the IEEE Computer Society, “Ieee standard for floating-point arithmetic,” *IEEE Standards Association*, vol. IEEE Std 7, June 2019.
- [279] Goran Flegar, Florian Scheidegger, Vedran Novaković, Giovanni Mariani, Andrés E. Tomás, A. Cristiano I. Malossi, and Enrique S. Quintana-Ortí, “Floatx: A c++ library for customized floating-point arithmetic,” *ACM Transactions on Mathematical Software*, vol. 45, no. 4, 2019.
- [280] Ambrose Finnerty and Herve Ratigner, “Reduce power and cost by converting from floating point to fixed point introduction,” *White Paper: Floating vs Fixed Point*, vol. 491, pp. 1–14, 2017.
- [281] David L. Donoho, “De-noising by soft-thresholding,” *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, 1995.
- [282] Gaël Guennebaud and Jacob Benoît, “Eigen v3,” 2010.
- [283] Xilinx, “Zcu106 evaluation board,” *Xilinx.com*, vol. UG1224, no. v1.4, pp. 1–134, Oct. 2019.
- [284] Xilinx, “Ultrascale architecture libraries guide,” *Xilinx.com*, vol. UG974, no. v2014.1, pp. 1–422, Apr. 2014.
- [285] Xilinx, “Ultrascale architecture configurable logic block user guide (ug574),” *Xilinx.com*, vol. UG574, no. v1.5, Feb. 2017.
- [286] Xilinx, “Ultrascale architecture: Dsp slice user guide (ug579),” *Xilinx.com*, vol. UG579, no. v1.10, pp. 1–75, Sept. 2020.