NATIONAL GEOSPATIAL-INTELLIGENCE AGENCY

NGA.STND.0014_2.4

2019-08-21

NGA STANDARDIZATION DOCUMENT

Softcopy Image Processing Standard

(2019-08-21)

Version 2.4

NATIONAL CENTER FOR GEOSPATIAL INTELLIGENCE STANDARDS

This page intentionally includes only this statement.

IMPORTANT NOTICE

A hardcopy of this document may not be the version currently in effect. The current version can be obtained by contacting NIQU Link to http://t.nga.ic.gov/VVG/?org=tq (on classified website) for the most current contact information.

UNCLASSIFIED GOVERNMENT REFERENCES

National Support Data Extension (SDE) (Version 1.3/CN1) for the National Imagery Transmission Format (NITF), STDI-0001
Compendium of Controlled Extensions (CE) for the NITF, STDI-0002
National Geospatial-Intelligence Agency (NGA) Display Performance Standard (DPS), NGA.STND.0015_3.1
ICA Softcopy Image Processing Chain Baseline (Obsolete)
Interface Control Document for the ASARS-2 Improvement Program (AIP) Airborne Radar Set to Ground, ICD40042-166
National Geospatial-Intelligence Agency (NGA) National Imagery Transmission Format (NITF) Version 2.1 Commercial Dataset Requirements Document (NCDRD), STDI-60009
National Imagery Transmission Format (Version 2.0) for the National Imagery Transmission Format Standard, with Notices 1, 2, and 3, MIL-STD- 2500A
National Imagery Transmission Format (Version 2.1) for the National Imagery Transmission Format Standard, MIL-STD-2500C
NATO Secondary Imagery Format (NSIF), STANAG No. 4545
NGA Softcopy Exploitation Facility Standard (SEFS), NGA.STND.0016_2.0
NGA Softcopy Facilities Guidelines (SFG)
SQAP Ambient Light Measurement Procedure
SYERS-2 Quick Reference Guide, Version 1.1 Rev. J, Document No. VG-V20-0530
USIGS Interoperability Profile (UIP), N0102-G UIP SCN015

NON-GOVERNMENT REFERENCES

GeoTIFF Format Specification
GIF Graphics Interchange Format
International Color Consortium Specification
ISO/IEC 12087-5:1998(E) Basic Image Interchange Format (BIIF)
IEC 61966-2-1:1999, Multimedia systems and equipment – Colour measurement and management – Part 2-1: Colour management – Default RGB colour space – sRGB
ISO/IEC BIIF Profile for JPEG 2000 Version 01.00 (BPJ2K01.00), BPJ2K01.00
ISO/IEC BIIF Profile NATO Secondary Imagery Format Version 01.00 (NSIF01.00), NSIF01.00
ISO/IEC FDIS15444-1: 2002 JPEG 2000 Image Coding System – Part 1
JPEG File Interchange Format Version 1.02, C-Cube Microsystems
TIFF Revision 6.0, Adobe Developers Association

DOCUMENT CHANGE HISTORY

Date	Version	Description	DR/CA	Developer
2/25/05	0.1	First draft of Sections 1&2		SIPS Team
3/15/05	0.2	Updated Sections 1&2, added Section 3		SIPS Team
4/1/05	0.3	Added Section 4 contents		SIPS Team
5/12/05	0.4	Updated Section 3.x for review		SIPS Team
6/22/05	0.5	Updated Section 3 from comments at R&A meeting		SIPS Team
7/7/05	0.6	Updated Sections 3&4 for R&A meeting		Scott Bennett
7/28/05	0.7	Updated Section 5. All sections ready for review by authors.		Scott Bennett
9/9/05	0.8	Final Draft ready for review		Scott Bennett
12/29/05	0.9	Updated based on comments from 11/29 review meeting.		Scott Bennett
1/9/06	0.10	Updated based on comments from 1/4/06 review meeting		Scott Bennett
2/28/06	0.11	Made changes according to review meeting 1/24/06		Scott Bennett
3/9/06	0.12	Minor grammatical corrections, some rewording, and incorporated initial		Scott Bennett
3/23/06	0.13	Added Appendix A.		Scott Bennett
4/7/06	0.14	Fixed ADS formatting issue		Scott Bennett
5/15/06	1.0	Updated based on feedback from CCB review 4/28/06		Scott Bennett

(continued on next page)

DOCUMENT CHANGE HISTORY

Date	Version	Description	DR/CA	Developer
09/01/06	1.0.1	1-D LUT update, DRA update, DRA Verification, TTC/MTFC database organization, Data Remap- ping, 1-D LUT verification, convolution verification		SIPS Team
01/09/07	1.0.2	Removed Sparse Sam-pled LUTs, Removed TBR/TBD's from TBR/TBD History, Added PEDF and Lin-Log equations to Section 3, Added Scaling and Asymmetric Pixel		SIPS Team
02/16/07	1.1	Wordsmithing, document formatting, addressed comments from review for v1.1 release. Replaced TBR/TBD history table with information from v1.0 release and closeout date of v1.1 release.		SIPS Team
6/8/07	2.0D-1	Restructured document organization and started incorporating color management and multiband operations.		SIPS Team
7/9/07	2.0D-2	Moved Reference image chain to end of main document body. Completed first revision to include CMM documentation. Rewrote most of interactive processing section to incorporate new		SIPS Team

(continued on next page)

DOCUMENT CHANGE HISTORY

Date	Version	Description	DR/CA	Developer
7/20/07	2.0D-3	Add more detail about CMM. Add example and profiles to database. Sweep the document for ambiguities relating to application to single vs. multiband data.		SIPS Team
8/6/07	2.0D-4	Addressed comments from Review meeting		SIPS Team
9/28/07	2.0	Updated handling of Asymmetric pixels in JPEG2000 workflow. Fixed several wording issues, formatting issues, etc.		SIPS Team
5/15/08	2.1D-1	Updated Scope in sec.1.2, updated selected ADSs in sec. 2.0, refined wording in sec.3.0, expanded Verification Section 4.0, updated Appendix A (sec.6.0)		SIPS Team
6/30/08	2.1	Updated Document format for PDF delivery. Enhanced document navigation through bookmarks and links. Finalized verification section. Updated RRDS, 1DLUT, and CMM ADS's.		SIPS Team
5/26/09	2.2-d3	Updated ADS to replace LaGrange Interpolation with a Four-point interpolator. Added Compromise Coefficients. Modified section 4.3.		SIPS Team

(continued on next page)

DOCUMENT CHANGE HISTORY

Date	Version	Description	DR/CA	Developer
8/17/09	2.2-d4	Added Maximum Pixel Decimation ADS, included handling of colorized change composite image products. Added section to address AGI data. Revised and updated default settings tables in appendices. Updated scaling ADS for proper Intra-Octave scaling, included verification		SIPS Team
10/15/09	2.2-d5	Added glossary. Updated Convolution ADS to include a subsection describing correlation. Revised every section of the document referencing convolution and/or correlation to clearly distinguish between the two operations and specify correct usage.		SIPS Team
2/1/10	2.2	Addressed reviewer's comments. Updated color management/ICC profiles sections. Finalized changes (spelling, grammar, links, etc.) from draft updates. Revised section 4.3 to improve clarity and consistency with DPS.		SIPS Team
7/13/10	2.2	Changed cover page. Updated scope to address only raster data handling at this time.		SIPS Team

(continued on next page)

DOCUMENT CHANGE HISTORY

Date	Version	Description	DR/CA	Developer
1/18/13	2.3	Added Band Dependent Linearization (BDL) Algorithm Description. Updated reference image chain for multi-band data. Added a Piecewise mathematical formula based LUT definition and corresponding ADS. Delivered new ICC Profiles and family of TTCs for use with color managed workflows.		Scott Bennett
3/31/14	2.4	Added sections and corresponding sub-sections Verification Strategies, ELT Component Level, and System Level sub-sections to all Reference Image Chain Descriptions; Sections		Jay Berman
4/11/14	2.4	Moved Unclassified SYERS and Global Hawk sections from classified appendix to main document."		Jay Berman
4/11/14	2.4	Added Figure 7 "SYERS Band Composites."		Jay Berman

(continued on next page)

DOCUMENT CHANGE HISTORY

Date	Version	Description	DR/CA	Developer
4/12/14	2.4	Copied unclassified "Default Parameter Ranges and Precisions" table from Appendix B.1 to Appendix A.3 for ease of reading. In previous versions reader would have to go back and forth between the two documents to comprehend image		Jay Berman
4/18/14	2.4	Corrected, edited, and updated Tables A.3, A.4, and A.5 to include unclassified airborne data sources and other unclassified sources; and add new /update existing de- fault display		Jay Berman
4/30/14	2.4	Created new glossary Sect 5.1 (terms.txt) and acronyms.txt Sect 5.2 files. Updated both listings; added new terms and acronyms introduced during general updates and edits to the SIPS main		Jay Berman
2/10/15	2.4	General edits and updates to the SIPS main text.		Jay Berman
8/21/19	2.4	Annual review conducted. No content change required.		Dhruv Mehta

TBR/TBD HISTORY

Date	Description	Developer	Closed
04/27/06	Complete Verification of RRDS generation	SIPS Team	02/16/07
04/27/06	Complete Verification of LaGrange Interpolation	SIPS Team	02/16/07
04/27/06	Complete Verification of Convolution	SIPS Team	02/16/07
04/27/06	Complete Verification of One Dimensional Look-Up-Tables	SIPS Team	02/16/07
04/27/06	Complete Verification of Dynamic Range Adjustment	SIPS Team	02/16/07
09/20/07	Update default band equalization recommendation	SIPS Team	05/15/08

This page intentionally includes only this statement.

Table of Contents

Introduction **1**
1.1 Purpose ... 1
1.2 Scope ... 1

Reference Image Chain Algorithm Descriptions **3**
2.1 Overview ... 3
2.2 Reduced Resolution Dataset Generation ... 5
2.2.1 Overview ... 5
2.2.2 Implementation Options ... 6
2.2.3 Inputs Required ... 6
2.2.4 Flow Diagram ... 7
2.2.5 Algorithm Details ... 7
2.2.5.1 Asymmetric Pixel Handling ... 9
2.2.5.2 JPEG2000 ... 10
2.2.6 Expected Output ... 14
2.2.7 Verification Strategies ... 15
2.2.7.1 ELT Component Level ... 15
2.2.7.1.1 Algorithm Inputs ... 15
2.2.7.1.2 Computation Details ... 18
2.2.7.1.3 Algorithm Output ... 18
2.2.7.2 System Level ... 18
2.3 Four-Point Interpolation ... 21
2.3.1 Overview ... 21
2.3.2 Implementation Options ... 22
2.3.3 Inputs Required ... 22
2.3.4 Flow Diagram ... 22
2.3.5 Algorithm Details ... 23
2.3.6 Expected Output ... 29
2.3.7 Verification Strategies ... 29
2.3.7.1 ELT Component Level ... 29
2.3.7.1.1 Algorithm Inputs ... 29
2.3.7.1.2 Computation Details ... 31
2.3.7.1.3 Algorithm Output ... 33

2.3.7.2 System Level 34
2.4 Convolution and Correlation 35
2.4.1 Overview 35
2.4.2 Implementation Options 35
2.4.3 Inputs Required 35
2.4.4 Flow Diagram 36
2.4.5 Algorithm Details 37
2.4.5.1 Convolution 37
2.4.5.2 Correlation 38
2.4.5.3 Image Boundary Handling 39
2.4.6 Expected Output 39
2.4.7 Verification Strategies 40
2.4.7.1 ELT Component Level 40
2.4.7.1.1 Algorithm Inputs 40
2.4.7.1.2 Computation Details 42
2.4.7.1.3 Algorithm Output 44
2.4.7.2 System Level 45
2.5 One-Dimensional Look-Up Table Application 47
2.5.1 Overview 47
2.5.2 Implementation Options 47
2.5.3 Inputs Required 47
2.5.4 Flow Diagram 48
2.5.5 Algorithm Details 48
2.5.6 Expected Output 48
2.5.7 Verification Strategies 48
2.5.7.1 ELT Component Level 49
2.5.7.1.1 Algorithm Inputs 49
2.5.7.1.2 Computation Details 49
2.5.7.1.3 Algorithm Outputs 51
2.5.7.2 System Level 51
2.6 Dynamic Range Adjustment 53
2.6.1 Overview 53
2.6.2 Implementation Options 53
2.6.3 Inputs Required 53
2.6.4 Flow Diagram 54
2.6.5 Algorithm Details 54
2.6.6 Expected Output 56
2.6.7 Verification Strategies 56
2.6.7.1 ELT Component Level 57
2.6.7.1.1 Algorithm Inputs 57
2.6.7.1.2 Computation Details 57
2.6.7.1.3 Algorithm Output 59
2.6.7.2 System Level 60

2.7 Scaling 61
2.7.1 Overview 61
2.7.2 Implementation Options 61
2.7.3 Inputs Required 61
2.7.4 Flow Diagram 62
2.7.5 Algorithm Details 63
2.7.5.1 Scale Factors < 1.0 63
2.7.5.2 Scale Factors > 1.0 64
2.7.6 Expected Output 65
2.7.7 Verification Strategies 65
2.7.7.1 ELT Component Level 65
2.7.7.1.1 Algorithm Inputs 65
2.7.7.1.2 Computation Details 66
2.7.7.1.3 Derive 1-D Intra-Octave Kernels 66
2.7.7.2 System Level 70
2.8 Asymmetric Pixel Correction 71
2.8.1 Overview 71
2.8.2 Implementation Options 71
2.8.3 Inputs Required 72
2.8.4 Flow Diagram 73
2.8.5 Algorithm Details 73
2.8.6 Expected Output 75
2.9 Color Management Module 77
2.9.1 Overview 77
2.9.2 Implementation Options 77
2.9.3 Inputs Required 77
2.9.4 Flow Diagram 78
2.9.5 Algorithm Details 78
2.9.6 Expected Output 80
2.9.7 Verification Strategy 80
2.10 Max-Pixel Decimation 81
2.10.1 Overview 81
2.10.2 Implementation Options 81
2.10.3 Inputs Required 81
2.10.4 Flow Diagram 82
2.10.5 Algorithm Details 83
2.10.5.1 In-Gamut Example 83
2.10.5.2 Out-of-Gamut Example 84
2.10.5.2.1 Out-of-Gamut Mapping 84
2.10.6 Expected Output 86
2.10.7 Verification Strategies 86
2.10.7.1 ELT Component Level 86
2.10.7.1.1 Algorithm Inputs 86

2.10.7.1.2 Computation Details ... 88
2.10.7.1.3 Algorithm Output ... 90
2.10.7.2 System Level ... 91
2.11 Band Dependent Linearization ... 93
2.11.1 Overview ... 93
2.11.2 Implementation Options ... 94
2.11.3 Inputs Required ... 94
2.11.4 Flow Diagram ... 95
2.11.5 Algorithm Details ... 95
2.11.5.1 Implementation Using All Image Pixels ... 95
2.11.5.2 Implementation Using Reduced Resolution Data Sets for Statistics Calculation ... 97
2.11.5.3 Implementation Using Only Displayed Image Pixels 98
2.11.6 Expected Output ... 98
2.11.7 Verification Strategies ... 98
2.11.7.1 ELT Component Level ... 99
2.11.7.1.1 Algorithm Inputs ... 99
2.11.7.1.2 Computation Details ... 100
2.11.7.1.3 Algorithm Output ... 102
2.11.7.2 System Level ... 103
2.12 LUT Generation from Piecewise Mathematical Formulae ... 105
2.12.1 Overview ... 105
2.12.2 Implementation Options ... 105
2.12.3 Inputs Required ... 105
2.12.4 Flow Diagram ... 106
2.12.5 Algorithm Details ... 106
2.12.6 Expected Output ... 107
2.12.7 Verification Strategies ... 107
2.12.7.1 ELT Component Level ... 107
2.12.7.1.1 Algorithm Inputs ... 107
2.12.7.1.2 Computation Details ... 108
2.12.7.1.3 Algorithm Output ... 108
2.12.7.2 System Level ... 109
Reference Implementation 111
3.1 Overview ... 111
3.2 Data Acquisition and Archiving ... 113
3.2.1 Dissemination and Storage Element ... 114
3.2.2 JPEG2000 Workflow ... 117
3.2.2.1 JPEG2000 Decoding ... 117
3.2.2.2 JPEG2000 Transcoding ... 118
3.2.2.3 JPEG2000 Encoding ... 118
3.2.3 Image Chipping ... 118

3.2.4 Bandwidth Expansion 118
3.2.5 Bandwidth Compression 118
3.2.6 BWC/BWE Database 118
3.3 Non-Interactive Processing 119
3.3.1 Product Generation Options 119
3.3.1.1 Band Equalization 120
3.3.1.2 Modulation Transfer Function Restoration 122
3.3.1.3 Atmospheric Correction 123
3.3.1.4 Data Remapping 123
3.3.1.4.1 PEDF Encoding 123
3.3.1.4.2 LinLog Encoding 124
3.3.1.5 Asymmetric Pixel Correction 124
3.3.2 Product Generation Options (PGO) Databases 125
3.3.2.1 Modulation Transfer Function Restoration (MTFR) Database 125
3.3.2.2 Asymmetric Pixel Correction Database 125
3.3.3 Reduced Resolution Dataset (RRDS) Generation 126
3.3.4 Reduced Resolution Dataset (RRDS) Database 126
3.4 Interactive Processing 126
3.4.1 Processing Order 126
3.4.2 Geometric Transform 126
3.4.3 Interpolator Database 128
3.4.3.1 ICC Profile Database 128
3.4.3.2 Embedded Profiles 128
3.4.4 Band Operations 128
3.4.5 Color Space Transform 128
3.4.6 Alternate Color Space Operations 129
3.4.7 Output Preparation 130
3.4.8 Reference Implementations 131
3.4.8.1 Reference Interactive Chain for Single-Band Imagery 131
3.4.8.1.1 Sharpness Enhancement 131
3.4.8.1.2 Sharpening Database 132
3.4.8.1.3 Histogram 132
3.4.8.1.4 Dynamic Range Adjustment (DRA) 133
3.4.8.1.5 Dynamic Range Adjustment (DRA) Database 133
3.4.8.1.6 One-Dimensional Look-Up Table 133
3.4.8.1.7 Tone Transfer Curve (TTC) Database 133
3.4.8.2 Reference Interactive Chain for Multiband Imagery . 133
3.4.8.2.1 DRA 134
3.4.8.2.2 DRA d/b 135
3.4.8.2.3 CST 135
3.4.8.2.4 ICC Profile d/b 135
3.4.8.2.5 Sharpness Enhancement 135

3.4.8.2.6 Modulation Transfer Function Enhancement (MTFE) Database ... 135
3.4.8.2.7 One Dimensional Look-Up Table ... 136
3.4.8.2.8 Tone Transfer Curve (TTC) Database ... 136
3.4.8.2.9 Output Preparation ... 136
3.5 Rendering ... 137
3.5.1 Printer Look-Up Table ... 138
3.5.2 Dissemination & Storage ... 138
3.5.3 Video Look-Up Table ... 138
3.6 Output ... 138
3.6.1 Device Calibration ... 138
3.6.2 Environmental Effects ... 140
3.7 Advanced Geospatial-Intelligence Products ... 140
3.7.1 Product Utility ... 141
3.7.2 Processing History ... 142
3.7.3 Display Processing ... 143
Verification **145**
4.1 Test Imagery ... 145
4.2 Pan Imagery ... 146
4.3 Multiband Imagery ... 147
4.3.1 Color Appearance Models ... 147
4.3.1.1 Modeling Reference Display Output ... 149
4.3.1.1.1 IDEX Aim ... 150
4.3.1.2 Convert to CIEXYZ ... 152
4.3.1.3 Calculate CIELab Quantities ... 152
4.3.2 Multiband Error Metrics ... 153
4.3.2.1 Comparison of Pan and Multiband Metrics ... 153
Glossary and Acronyms **157**
5.1 Glossary ... 158
5.2 Abbreviations/Acronyms ... 164
A Databases and Imagery **167**
A.1 Databases ... 167
A.1.1 Database Contents ... 167
A.1.2 Database Formats ... 168
A.1.2.1 Common Kernel File Format ... 168
A.1.2.2 Common Look-Up Table (LUT) Format ... 171
A.2 Test Imagery ... 176
A.3 Default Settings for Various Sensors ... 178
A.4 Data Remapping ... 182
A.4.1 Global Hawk VIS, IR, and SAR Bit-Depth Re-Mapping ... 183

A.4.2 SYERS-2A Visible and Infrared Bit-Depth Re-Mapping 183

B Reserved **185**

This page intentionally includes only this statement.

List of Figures

2.1 Graphical Representation of a 1-D LaGrange Interpolator 23
2.2 Auto-DRA Algorithm Using Penetration Points 55
2.3 Histogram of Original 10x10, 11-bit Matrix 58
2.4 Histogram of DRA Processed 10x10, 11-bit Image 59
2.5 Mapping of Scaled Image Pixel into the Original Image 63
2.6 Mapping of Corrected Image Pixel into the Original Image 74
2.7 Spatial Quantization Example Based on Quarter-pixel Sampling Grid 75
2.8 In-Gamut Max-Pixel Decimation on a Single Block of Data 84
2.9 Out-of-Gamut Max-Pixel Decimation on a Single Block of Data . . . 84
2.10 Max-Pixel Input 11x11, 8-bit Matrix 87
2.11 Max-Pixel Input LUT 87
2.12 11x11, 24-bit Matrix for Max-Pixel Example 89
2.13 Decimated 6x6, 24-bit Max-Pixel Output Matrix 90
2.14 Decimated 6x6, 8-bit Max-Pixel Output Matrix 90
2.15 Graphical representation of calculating BP_b 96
2.16 Graphical representation of data used for calculating $WP_{b,f}$ 97
2.17 Black Point Histograms for Bands 0-2 of the input matrices 101
3.1 High Level Reference Image Chain 112
3.2 Data Acquisition and Archiving Component 113
3.3 Non-Interactive Processing Component 119
3.4 Product Generation Example 1 121
3.5 Product Generation Example 2 122
3.6 Interactive Processing Component 127
3.7 Sample Alternate Color Space Operations 130
3.8 Reference Interactive Chain for Single Band Imagery 132
3.9 Reference Interactive Chain for Multiband Imagery 134
3.10 Rendering Component 137
3.11 Output Component 139
3.12 Standard Processing vs. AGI processing 141
3.13 SIPS Display Processing Categories 144
4.1 Color Modeling in SIPS 149
4.2 Comparison of L^* and Inverse IDEX 154
A.1 Common Kernel Format Layout 169

A.2 Sample Common Kernel Files 171
A.3 LUT File Format Layout 172
A.4 Sample LUT Files 173
A.5 Sample Math Formula LUT 174
A.6 Scalar Math Functions Based on "C" Programming Language 175
A.7 SYERS Band Composites 181

List of Tables

2.1 Positions for Application of Kernels to Generate R_1 Values 10
2.2 7x7 Anti-Aliasing Kernel 15
2.3 4x4 LaGrange Interpolation Kernel for RRDS Generation 16
2.4 R_0 Matrix 17
2.5 R_1 Matrix 19
2.6 R_2 Matrix 19
2.7 Sample LaGrange Coefficients for Various Spacings 24
2.8 Sample Compromise Coefficients for Various Spacings 26
2.9 4x4 LaGrange Interpolation Kernel ($^{16}/_{32}$ x $^{8}/_{32}$) 29
2.10 Input 4x4 Matrix Segment for Single Point Verification 30
2.11 Input 10x10 Matrix for Matrix Verification 30
2.12 LaGrange Single Point Verification Visualization 31
2.13 10x10 Matrix with Edge Mirroring 32
2.14 Preliminary LaGrange Interpolation Results 33
2.15 Output 10x10 LaGrange Interpolation Matrix 33
2.16 Boundary Handling for Convolution and Correlation 39
2.17 5x5 Convolution Kernel 40
2.18 5x5 Convolution Input Matrix 41
2.19 4x4 Convolution Kernel 41
2.20 4x4 Input Convolution Matrix 42
2.21 5x5 Convolution Matrix with Edge Mirroring Applied 42
2.22 5x5 Convolution Interim Output Matrix 43
2.23 4x4 Convolution Example Input Matrix with Edge Mirroring 44
2.24 4x4 Convolution Example Interim Output 44
2.25 5x5 Convolution Example Output Matrix 45
2.26 4x4 Convolution Example Output Matrix 45
2.27 TTC Verification Input Matrix Data 49
2.28 TTC Family 0 Member 0 Sparsely Sampled Examples 50
2.29 TTC Numerical Output Verification 51
2.30 Parameters of the DRA Algorithm 56
2.31 DRA Algorithm Parameters 57
2.32 DRA Input 10x10, 11-bit Matrix 57
2.33 DRA Example Interim Calculated Values 58

2.34 DRA Processed 10x10, 11-bit matrix 60
2.35 7x7 Anti-Aliasing Kernel 65
2.36 11x11 R_1 Input Matrix 66
2.37 RRDS Anti-aliasing Filter LSF 66
2.38 Truncated, Scaled 3-Element LSF 66
2.39 1-D Intra-Octave Anti-Aliasing Kernels 67
2.40 Intermediate Anti-Alias Filtered R_1 Matrix 68
2.41 LaGrange Pixel Spacings for Rows and Columns of R_1 68
2.42 Filtered R_1 Matrix with LaGrange Interpolators Applied 69
2.43 Final 9x9 Scaled Matrix at 0.4x Absolute Magnification 70
2.44 CMM Example Converting Between sRGB and ProPhoto RGB 79
2.45 Subset of LUT used with Distance and Angle from (155, 255, 255) . . . 86
2.46 Subset of Max-Pixel LUT illustrated in Figure 2.11 88
2.47 BDL Input 100x100, 11-bit Matrix for Band 0 99
2.48 BDL Input 100x100, 11-bit Matrix for Band 1 99
2.49 BDL Input 100x100, 11-bit Matrix for Band 2 100
2.50 Initial White Point Calculation Values 101
2.51 Final Black Point and White Point Values 102
2.52 BDL Output 100x100, 11-bit Matrix for Band 0 102
2.53 BDL Output 100x100, 11-bit Matrix for Band 1 103
2.54 BDL Output 100x100, 11-bit Matrix for Band 2 103
2.55 Piecewise Mathematical Formula output LUT 108
3.1 Supported Imagery and Compression Formats 114
4.1 Error Metric Tolerances for Various Bit Depths 147
A.1 Database Layout 167
A.2 Number of Kernel Values 170
A.3 Default Initial Display Settings 178
A.4 Default Non-Interactive Image Chain Settings 181
A.5 Default Interactive Image Chain Settings 182
A.6 Default Parameter Ranges and Precisions 182

Introduction

The NGA Softcopy Image Processing Standard (SIPS) establishes and documents key processes necessary for NGA image processing and exploitation. These key processes are provided for the purpose of improving the consistency, aiding in the verification and testing, and optimizing the image quality of displayed imagery in the National System for Geospatial-Intelligence (NSG) environment. The processes and procedures documented in this standard draw upon established practices for image handling, as well as advanced image processing concepts, to provide a strong basis for future advancement of exploitation architecture.

1.1 Purpose

The intent of this standard is to document the processes and procedures that are necessary for the NSG to be in compliance with NGA Operational Requirements Document (NORD) Key Performance Parameter (KPP) 3, which states that the NSG will not impart any image quality degradation to the image except those dictated by the user as a result of product selection. This standard will aid in the development of Government Off-The-Shelf (GOTS) and Commercial Off-The-Shelf (COTS) image handling software by providing unambiguous specification of key image processing algorithms so that the image quality derived from the implementation of those algorithms using the reference image chain provided will be consistent throughout the NSG. The standard will also aid in the verification and testing of the NSG by providing clearly defined test cases to ensure that a specific image chain implementation produces image quality consistent with other implementations used by the NSG.

1.2 Scope

This standard is intended for use in the development and testing of NSG image handling and subsequent display processing. It provides standard image handling processes for integer-valued raster image data and covers every image handling and processing function, including compression, expansion, enhancement and preparation of the raster image data for final softcopy display. This standard also provides algorithm descriptions that may be utilized in image handling and display applications to achieve the required image quality standard

Image quality metrics are provided and are the bases for assessing compliance of GOTS and COTS image handling and display software for processing still imagery at roaming rates < 480 pixels per second (pps). Sample imagery is also available with this standard for validating image quality using the metrics provided. Input and processed test imagery is listed in the appropriate appendix along with processing parameters used to produce the output by the reference image chain given in this document.

Note that this standard is currently applicable only to the handling of the raster image data. While software developers are encouraged to preserve the metadata of an image as much as possible, there are a number of factors regarding metadata handling that need to be investigated or resolved before being addressed as part of this standard.

Reference Image Chain Algorithm

Descriptions

2.1 Overview

Preserving image quality through an image processing architecture requires a suite of image processing algorithms. Each algorithm takes one or more input images and a set of parameters defining the operation to be performed. Multiple algorithms are then linked together forming what is commonly referred to as an image chain. This section of the Softcopy Image Processing Standard details the algorithms used for constructing a reference image chain suitable for library or Electronic Light Table (ELT) applications. A particular application (e.g. an ELT package) may utilize these or other appropriate algorithms assembled in a manner consistent with this document to minimize image quality degradation.

This section is devoted to the algorithms used by the reference image chain to perform image processing for the NSG. Each algorithm is given its own subsection for easy reference. The individual algorithms can be utilized for building an image chain that minimizes image quality degradation during image exploitation.

Because each algorithm modifies the data in some manner, errors may easily propagate through the system. The reference image chain presented in Section 3 utilizes the algorithms in this section in an order that minimizes error propagation and preserves image quality. Alternate implementations may be designed for performance or other reasons, but image quality standards will be verified against the reference implementation. Further information on image quality metrics, procedures, and tolerances required for compliance may be found in Section 4.

The following are the Algorithm Description Sheet (ADS) for each algorithm provided in this standard. The ADS's are specifically designed to work as stand- alone documents for reference during implementation. Only algorithms used by the reference implementation have an ADS included in this document.

This page intentionally includes only this statement.

2.2 Reduced Resolution Dataset Generation

Algorithm Name	Reduced Resolution Data Set Generation
Databases Required	Anti-aliasing Kernels, Spatial Interpolation Kernels
Last Updated	10/21/09
Application Conditions	Applicable to all compressed imagery and levels not already included in JPEG 2000 (J2K). All imagery must be uncompressed and tonally remapped (if applicable) prior to application.
Purpose	Generates high quality sub sampled versions of an image for faster, high quality exploitation at various magnification levels.

2.2.1 Overview

The Reduced Resolution Data Set (RRDS) is a set of images consisting of an original, full resolution image, known as the R_0 image, and a series of reduced resolution images at lower magnifications. The reduced resolution images are referred to as the R1, R2, R3, and Rn images, where n refers to the integer value associated with the final image produced for the dataset.

In general, the integer value used to refer to a particular resolution level image indicates the power-of-two factor used to down-sample the R0 image to create that particular reduced resolution image. Thus, the R1 image refers to a reduced resolution image that has been downsampled from the R0 image by a factor of 21 = 2 in both the row and column dimensions. Similarly, the R2 image is a reduced resolution image formed by downsampling the R0 image by a factor of 22 = 4 in both the row and column dimensions. An exception to this rule occurs for RRDS produced from R0 images containing asymmetric pixels, as will be explained in Section 2.2.5.1.

Additionally, J2K-compressed R0 images already provide access to some number of RRDS images due to the nature of the compressed R0 image codestream. Section 2.2.5.2 addresses RRDS generation issues surrounding J2K-compressed files.

Furthermore, images received by this algorithm may, in some cases, be accompanied by a previously generated full or partial RRDS. In these cases, the RRDS generation algorithm should only produce the additional RRDS images that are not currently provided in the accompanying dataset.
Note that while this algorithm is more time-consuming than simple downsampling operations, the benefits in terms of image quality far outweigh the additional computational burden.

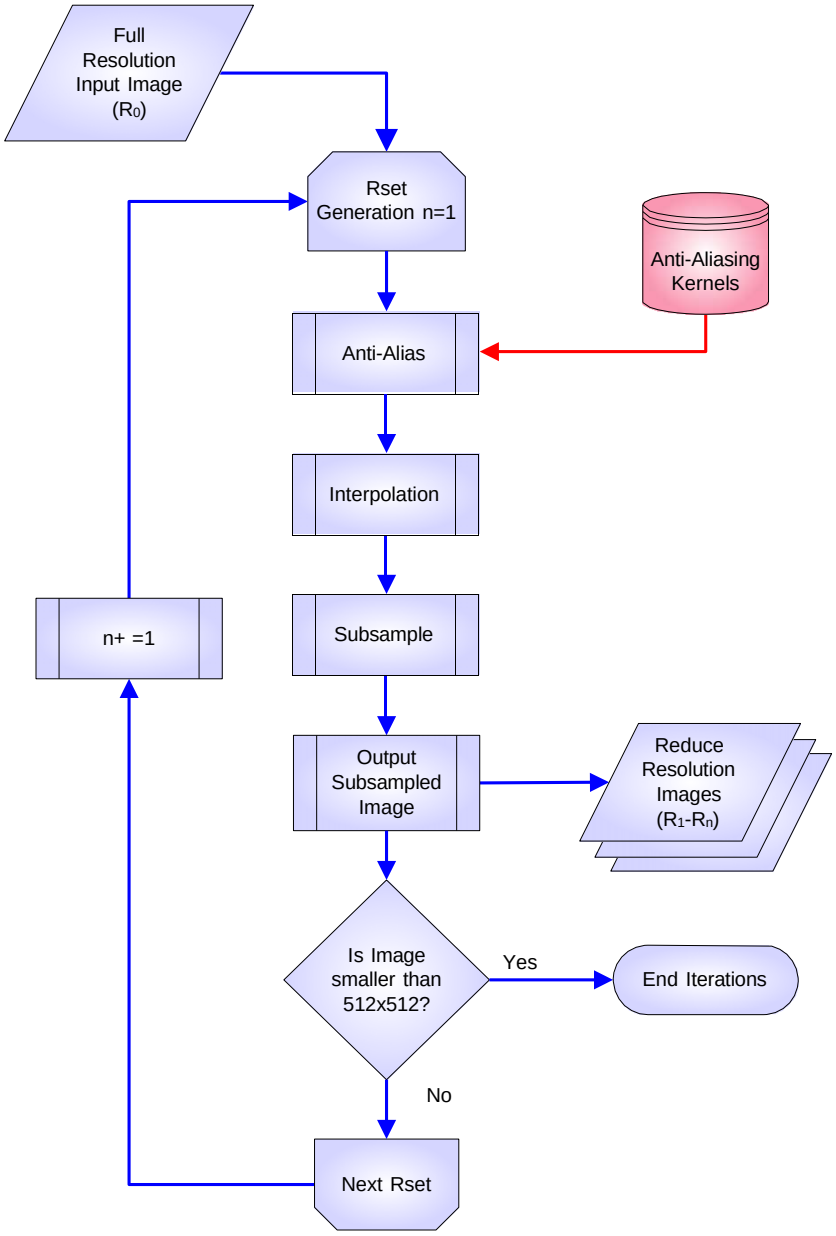
2.2.2 Implementation Options

- Other interpolation algorithms may be utilized for performance optimization.
  - A high-order interpolation algorithm (e.g. cubic, modified sinc, etc.) may be used for most Electro-Optical (EO) imagery products
  - Max-Pixel decimation should be selected for processing colorized change composite image products (see section 2.10).

2.2.3 Inputs Required

- One of the following:
  - A native-resolution image requiring full or partial RRDS generation, as determined by the presence of associated, previously-generated, reduced resolution image data
  - A J2K-compressed file requiring additional generation of uncompressed RRDS levels
- The tonal mapping (e.g. Piecewise Extended Density Format (PEDF) or Linear/Logarithmic Data Re-map used in Legacy Radar processing (Lin-Log)) used to produce the original image (J2K-compressed files only).
- The pixel aspect ratio for images with asymmetric pixels
- A database of anti-aliasing convolution kernels
- A database of spatial interpolation correlation kernels

2.2.4 Flow Diagram



2.2.5 Algorithm Details

The RRDS algorithm begins with the original, full resolution R_0 image. The first iteration of the process generates the R_1 image. The R_1 image seeds the iteration generating the R_2 image, and so on. This algorithm is applied independently to

each band of the image.

The naming convention for the RRDS images reflects the integer value used as a power-of-two factor for downsampling the R_0 image when creating a particular reduced resolution image. In this way, the R_1 image refers to a reduced resolution image that has been downsampled from the R_0 image by a factor of $2^1 = 2$ in both the row and column directions. Similarly, the R_2 image is a reduced resolution image formed by downsampling the R_0 image by a factor of $2^2 = 4$ in both the row and column directions.

There is one case where downsampling the R_0 image to form the R_1 image is not based on a power-of-two downsampling factor. This occurs when the R_0 image consists of asymmetric (non-square) pixels. For R_0 images with asymmetric pixels, the processing and downsampling used to form the R_1 image will be specifically designed to "correct" the aspect ratio of the asymmetric pixels to a symmetric (square) pixel space. The downsampling applied to the asymmetric R_0 image is equivalent to a power-of-two down-sample applied to the corresponding spatially corrected R_0. Having created the proper, symmetric-pixel R_1 image from the asymmetric-pixel R_0 image, all subsequent RRDS images will be formed starting from the symmetric-pixel R_1 image. This ensures that all levels of the RRDS beyond R_0 contain symmetric pixels.

It should be noted that the various reduced resolution images are not formed by a simple downsampling process; three steps must be performed before the image can be downsampled to preserve image quality. To preserve image quality within each level of the RRDS, pixels from the "parent" RRDS level are first remapped, as appropriate, to reverse any tonal mappings (e.g. PEDF or Lin-Log). Second, the "parent" RRDS level is convolved with an anti-aliasing kernel to remove high spatial frequency information from the scene. Third, after anti-aliasing processing, the "parent" scene is then correlated with a spatial interpolation kernel, currently based on LaGrange interpolation coefficients. Once the "parent" scene has undergone tonal remapping (if applicable), anti-aliasing, and interpolation (in that order), it is downsampled to form the next reduced resolution image in the RRDS.

The stopping criterion for the RRDS Generation algorithm should be when the smallest dimension (either row or column), of the final reduced resolution image to be included in the RRDS, is greater than or equal to 256 pixels and less than 512 pixels. The single exception to this rule potentially occurs with J2K compressed imagery, as will be discussed below. Therefore, the number of reduced resolution images making up an RRDS is entirely dependent upon the number of pixels in the row and column dimensions of the R_0 image. Note that if an R_0 image has either a row or column dimension that is less than 512 pixels, the RRDS will consist of only the R_0 image (unless the R_0 consists of asymmetric pixels, where the "dimensions" to be compared with this 512 pixel limit would be those of the corresponding "corrected" symmetric pixel space dimensions).

When downsampling a given reduced resolution level image to produce the next reduced resolution level image, the following rules shall be applied. If the

row or column dimension is even-valued, then the corresponding downsampled dimension is equal to the original dimension divided by 2. If the row or column dimension is odd-valued, then the corresponding downsampled dimension is equal to the original dimension divided by two and rounded to the nearest integer (e.g. $^{1025}/_{2}$ = 512.5 which rounds to 513). For the purposes of RRDS generation, rounding assumes that numbers with fractional portions greater than or equal to 0.5 are rounded up, and numbers with fractional portions less than 0.5 are rounded down. The dimensions of the next resolution level to be produced depend upon the dimensions of the current resolution level, not the value defined by downsampling directly from the R_0 level image plane.

2.2.5.1 Asymmetric Pixel Handling

For asymmetric pixel cases, the downsampling rules are more explicit. For example, an R_0 image with a pixel aspect ratio of 1.5:1 and having 601 pixels in the column dimension would generate an R_1 image in the following manner. After spatially correcting the asymmetric pixels, the original 601 column pixels would theoretically become 902 symmetric column pixels (601 x 1.5 = 901.5, rounded to 902). These 902 symmetric pixels would then be downsampled by a factor of 2 to a value of 451 representing the number of pixels in the column direction of the R_1 image.

A similar asymmetric R_0 image having an aspect ratio of 1.5:1 but with 599 pixels in the column dimension, would yield an R_1 image with 450 symmetric column pixels (599 x 1.5 = 898.5, rounded to 899, then divided by 2 to 449.5, and rounded again to 450). Note that the case of 600 pixels also yields an R_1 image with 450 column pixels. With only 598 pixels in the column dimension, the R_1 image would possess 449 column pixels (598 x 1.5 = 897, divide by 2 = 448.5, and round to 449). These values are based on the rules specific to the processing of an asymmetric R_0 image with an aspect ratio of 1.5:1 to create a symmetric R_1 image, as summarized in Table 2.1.

Table 2.1 correctly implies that there are three different spatial interpolation kernels used to generate a symmetric R_1 reduced resolution level image from an asymmetric R_0 image having an aspect ratio of 1.5:1. The spatially-variant correlation operation must be capable of applying the first kernel against R_0 image columns 1, 5, 9, 13, etc., the second kernel against R_0 image columns 2, 6, 10, 14, etc., the third kernel against R_0 image columns 3, 7, 11, 15, etc., and of skipping over R_0 image columns 4, 8, 12, 16, etc. The variables $F, S, T,$ and , used in Table 2.1, stand for **F**irst Kernel, **S**econd Kernel, **T**hird Kernel, and **U**nused, accordingly. Note that three specific kernels are required for the 1.5:1 asymmetric case. Each aspect ratio will require a unique set of interpolation kernels. Implementers must take care to ensure that the correct set of spatial interpolation kernels is applied to preserve image quality in the generated RRDS.

Table 2.1: R_0 Column Positions for Application of Spatially Variant Interpolation Kernels to Generate R_1 Column Values

Columns in 1.5:1 Aspect Ratio R_0 Image	Corresponding Columns in 1:1 Aspect Ratio R_1 Image
$1, 5, 9, 13, 17, \ldots, 4n + 1$ where $n = 0$ to F and F = number of R_0 columns minus 1, divided by 4, rounded down	$1, 4, 7, 10, \ldots, 3n + 1$ where $n = 0$ to F
$2, 6, 10, 14, 18, \ldots, 4n + 2$ where $n = 0$ to S and S = number of R_0 columns minus 2, divided by 4, rounded down	$2, 5, 8, 11, \ldots, 3n + 2$ where $n = 0$ to S
$3, 7, 11, 15, 19, \ldots, 4n + 3$ where $n = 0$ to T and T = number of R_0 columns minus 3, divided by 4, rounded down	$3, 6, 9, 12, \ldots, 3n + 3$ where $n = 0$ to T
$4, 8, 12, 16, 20, \ldots, 4n + 4$ where $n = 0$ to U and U = number of R_0 columns minus 4, divided by 4, rounded down	No R_1 columns are generated from these R_0 columns

The correlation operation used to perform the interpolation step must be capable of applying a spatially-variant correlation kernel to the "parent" scene when the "parent" consists of asymmetric pixels. This is accomplished by loading the appropriate kernel into the correlation calculation based on the row and/or column position in the "parent" image. For asymmetric pixel cases having normalized aspect ratios of the form, $(c = 1.0) : (r > 1.0)$, the spatially-variant correlation must change the interpolation kernel based on the row of the image being processed. For asymmetric pixel cases having normalized aspect ratios of the form, $(c = 1.0) : (r > 1.0)$,, the spatially-variant correlation must change the interpolation kernel based on the column of the image being processed.

2.2.5.2 JPEG2000

J2K-compressed R_0 images contain some number of reduced resolution images within their compressed J2K codestreams due to the nature of the J2K compression algorithm itself. For example, if the J2K encoder had specified the use of five decomposition levels, then the resulting compressed R_0 image would contain six of the resolution levels needed for an RRDS; R_0, R_1, R_2, R_3, R_4, and R_5 (the original level plus five decomposed levels). In this example J2K codestream, having five decomposition levels, the following RRDS generation processing logic must be followed:

1. If the R_5 level image, when extracted form the J2K codestream, has

dimensions larger than the RRDS Generation algorithm's stopping criterion (512 or more pixels in both dimensions), then the R_5 image should be used as the "parent" image for continuing to generate uncompressed RRDS images until the RRDS is fully populated. This implies that the R_5 image in this example would be extracted from the J2K codestream, expanded to an uncompressed pixel space, and used to generate uncompressed R_6 through R_n RRDS images.

(Note that there are additional processing issues involved with asymmetric pixel and/or PEDF or Lin-Log mapped datasets, as discussed below.)

-or-

1. If the R_5 level image, when extracted from the J2K codestream, has dimensions smaller than or equal to the RRDS Generation algorithm's stopping criterion (fewer than 512 pixels in either dimension), then no further RRDS level images will be created. It should be noted that the user of such a J2K- compressed image is allowed to access any resolution level provided in the J2K codestream, even if the resolution level is smaller than what would normally be produced by the RRDS Generation algorithm described herein.

   (Note that the stopping criterion is based on "symmetric" pixel dimensions, so if the J2K-compressed dataset contains asymmetric pixels, then the R_5 dimensions need to be "corrected" to a symmetric equivalent before determining the validity of the stopping criterion.)

2. For an archive or dissemination element, where the intention is to produce RRDS for storage or dissemination, not display, the RRDS Generation algorithm stops at this point. The archive/dissemination element now holds a J2K-compressed image, containing $R_0 \rightarrow R_5$ (in this example), and potentially, some number of additional, uncompressed R-level images, numbered $R_6 \rightarrow R_n$. The archive may need to convert the file format or type of compression applied to these R-levels based on user request, but such processing is beyond the scope of this ADS.

   -or-

2. For an ELT software package, or other imagery display tool, there may be additional processing steps with respect to RRDS Generation. The following steps speak to the processing of an RRDS that is provided, either in whole or in part, in a J2K-compressed codestream, when the intention is to process the RRDS for display.

   a. To enable a simple zooming capability between R-levels, or to perform Intra-Rset Scaling, the display tool will need to extract and expand the individual R-set level images from the J2K codestream.

b. Before passing the extracted and expanded R-levels to the remaining steps in the display processing chain, the display tool needs to determine if the dataset contains asymmetric pixels and if the dataset is in a mapped space such as PEDF or Lin-Log.
(See Appendix B (Section B) for information concerning how such determinations can be made from image file metadata structures.)

c. If the RRDS does not contain asymmetric or mapped pixels, at any R-level, then the display processing tool is finished with the RRDS Generation process.
-or-

c. If some or all of the R-levels contain asymmetric and/or mapped pixels (e.g. PEDF, Lin-Log), then the individual R-levels in question need to be spatially corrected to a symmetric pixel space and/or tonally remapped before being passed to the remainder of the enhancement processing chain.
   (i). Asymmetric pixels are corrected based on the algorithm provided in the Asymmetric Pixel Correction ADS in Section 2.8.
   (ii). PEDF and Lin-Log mapped datasets can be remapped via the algorithm provided in the One-Dimensional Look-Up Table Application ADS in Section 2.5. The appropriate Look-Up Tables (LUTs) for each case are provided on the data disc accompanying this document.
   (Note: Should a dataset be both in a mapped tonal space and consist of asymmetric pixels, the remapping should be applied first, then the asymmetry correction.)

One concern with using lower resolution RRDS images extracted from a J2K-compressed image file is the handling of compressed PEDF or Lin-Log image data. The lower resolution images contained in the J2K codestream may not have been remapped from the PEDF or Lin-Log space prior to their formation in the compressed codestream. Applying the PEDF or Lin-Log remapping to an extracted lower resolution image plane has not been studied from an image quality standpoint, and as such, has an unknown impact on image quality. Applying such a remapping is currently suggested by this standard, if RRDS imagery is to be formed from lower resolution imagery extracted from a J2K codestream, but this suggestion may be reevaluated for future versions of this standard.

An additional concern with J2K-compressed R_0 images occurs when the R_0 image consists of asymmetric pixels. In this case, the individual resolution levels pro- vided in the J2K codestream are not corrected to a symmetric pixel space, as called for by the RRDS Generation algorithm. The current solution is to apply the asymmetric correction to each individual R-level extracted from the J2K-compressed

codestream before using the R-level image for further image processing. This process has not been studied from an image quality standpoint (except in the case of the R_0 level), and as such, has an unknown impact on image quality. This is due to the fact that the image quality and even spatial dimensions of the R-levels extracted from the J2K-compressed codestream will not necessarily match that of corresponding R-levels generated from the traditional kernel-based RRDS processing algorithm.

Furthermore, continuing with the specific 5-decomposition level example, the creation of a spatially-corrected (symmetric pixel) R_6 level image, from the extracted asymmetric pixel J2K-compressed R_5 level, would need to rely on the traditional kernel-based RRDS processing approach to build the R_6 image. To do this, the Government Furnished Equipment (GFE) anti-aliasing convolution kernel followed by the interpolation kernel for the appropriate asymmetric correction aspect ratio, would have to be applied to the extracted/expanded R_5 image plane. These GFE kernels were originally optimized for the processing of asymmetric R_0 images to symmetric R_1 images; image quality impacts of applying these optimized kernels to the R_5-to-R_6 processing case are currently unknown.

Therefore, two options present themselves for producing an RRDS from J2K-compressed images at the display end of the image chain. The first option is to expand the full J2K-compressed asymmetric R_0 image, perform any PEDF or Lin-Log tonal remapping (as appropriate), and generate an RRDS from the uncompressed full-resolution asymmetric pixel image as per the standard kernel-based algorithmic rules described above.

The second option is to extract and expand the lowest resolution image plane provided in the J2K codestream having corresponding symmetric pixel dimensions that are still greater than the RRDS Generation algorithm's stopping criterion (256 pixels in either spatial dimension). For purposes of this example, this image will be referred to as resolution level, R_m, where "m" is an integer less than or equal to "n", and "n" is the final integer value for a resolution level in a standard RRDS. The R_m image then seeds the continuation of the standard RRDS generation algorithm, with the difference that the extracted asymmetric R_m image acts like an asymmetric R_0 image, requiring proper interpolation to produce a symmetric R_{m+1} image plane analogous to an R_1 in the standard processing. If the standard RRDS generation process does not call for additional resolution levels beyond the R_m level, then no RRDS images are produced by this option. An end-user interested in viewing spatially corrected (symmetric) resolution levels must correct each individual level extracted from the J2K codestream prior to performing any additional processing, such as Intra-RRDS Scaling (see Scaling ADS in Section 2.7). It should be added that if the J2K-compressed R_0 image is provided in a mapped space such as PEDF or Lin-Log, then the appropriate remapping must be applied to the R_m level prior to producing the R_{m+1} level.

Either option for generating an asymmetrically corrected RRDS from J2K code-streams is currently allowed under this standard for softcopy image processing,

though these options may be revised in future versions of the standard. From an image quality standpoint, the first option represents a known processing option when dealing with imagery that has asymmetric pixels or mapped imagery such as PEDF or Lin-Log data sources. The second option provides for a different processing paradigm, which is currently considered to be acceptable, though with an unknown image quality impact compared with the traditional kernel-based RRDS algorithm.

2.2.6 Expected Output

The output of this algorithm is a completed RRDS for a given input image (usually R_0). The RRDS must consist of all reduced resolution images, potentially levels R_1 through R_n, required by the algorithm description provided above, up to the stopping condition in effect at the time of RRDS generation (see Section 2.2.5).

Output RRDS levels generated by this algorithm for display purposes are uncompressed, spatially corrected (for asymmetric pixel source data), and in an unmapped tonal space (as applicable). (There is one additional consideration in this case; the original R_0 image, if it contained asymmetric pixels, must be retained along with the spatially-corrected R_0 image used for display to the observer. The reason for retaining the asymmetric R_0 image has to do with how the Intra-RRDS algorithm is implemented for creating imagery at scales between R_0 and R_1.)

Output RRDS levels other than R_0, generated by this algorithm for archive and/or dissemination purposes are uncompressed, spatially corrected (for asymmetric pixel source data), and in an unmapped tonal space (as applicable), if the source image is other than J2K compressed. If the source image was embedded in a J2K-compressed codestream, then the RRDS consists of the J2K codestream (all inherent R-levels; R_0 through R_{m-1}), and possibly, some number of additional, uncompressed R-level images (R_m through R_n) that have been spatially corrected and/or tonally remapped, as appropriate. RRDS images generated for archival use or dissemination may be subsequently compressed based on NSG customer requirements, though such compressions are beyond the scope of this ADS.

In both the display and archive/dissemination cases, the R_0 image that seeded the RRDS generation process remains unaltered and is simply provided to the user as is. For example, a full RRDS (R_0 through R_n) may consist of a J2K-compressed, tonally-mapped, symmetric-pixel based codestream (containing R_0 through R_{m-1}), and a series of R_m through R_n, uncompressed, tonally remapped, symmetric-pixel based, reduced resolution images. Alternatively, a full RRDS could consist of an original Tape Formats Requirement Document (TFRD) 4.3 compressed, asymmetric pixel, R_0 image and a series of R_1 through R_n uncompressed, symmetric, R-level images. The display process simply requires that this R_0 image be spatially and tonally corrected before further enhancements are applied and the image sent to the display screen.

2.2.7 Verification Strategies

ELT component level verification strategy provides an example approach and associated algorithms for verifying RRDS generation. The softcopy display vendor is encouraged to use or tailor this approach to verify RRDS component level processes and functionality. System level verification strategy identifies expected (or required) system data outputs and results.

2.2.7.1 ELT Component Level

This section provides verification examples for RRDS generation of symmetric-pixel R_0 images. Examples for asymmetric RRDS generation will be provided in future versions of this standard.

Symmetric RRDS generation verification is demonstrated with a 1026x1026 element (1, 052, 676 total pixels) input matrix containing four corner regions that are 10x10-element reflection images of the input matrix used for convolution algorithm verification in Section 2.4.7.1. The balance of the data (1, 052, 276 pixels) is set to a digital count value of 1024. A starting dimension of 1026 is used in this verification example because it is the smallest matrix that generates exactly two RRDS levels before meeting the stopping criteria and demonstrates downsampling of both an even and odd starting matrix.

Two RRDS levels are generated in this example, R_1 and R_2. The R_0 image has an even number of elements while the generated R_1 image used to seed the creation of the R_2 image has an odd number of elements.

2.2.7.1.1 Algorithm Inputs

Table 2.2: 7x7 Anti-Aliasing Kernel

0.00694389	0	-0.02777640	-0.04166500	-0.02777640	0	0.00694389
0	0	0	0	0	0	0
-0.02777640	0	0.11110900	0.16666500	0.11110900	0	-0.02777640
-0.04166500	0	0.16666500	0.25000000	0.16666500	0	-0.04166500
-0.02777640	0	0.11110900	0.16666500	0.11110900	0	-0.02777640
0	0	0	0	0	0	0
0.00694389	0	-0.02777640	-0.04166500	-0.02777640	0	0.00694389

Table 2.3: 4x4 LaGrange Interpolation Kernel for RRDS Generation

0.00390625	-0.03515625	-0.03515625	0.00390625
-0.03515625	0.31640625	0.31640625	-0.03515625
-0.03515625	0.31640625	0.31640625	-0.03515625
0.00390625	-0.03515625	-0.03515625	0.00390625

Table 2.4: R_0 Matrix

	0	1	2	3	4	5	6	7	8	9	10–1015	1016	1017	1018	1019	1020	1021	1022	1023	1024	1025
0	0	0	0	0	100	200	300	400	500	600	1024	600	500	400	300	200	100	0	0	0	0
1	0	0	100	200	300	400	500	600	700	800	1024	800	700	600	500	400	300	200	100	0	0
2	100	200	300	400	500	600	700	800	900	1000	1024	1000	900	800	700	600	500	400	300	200	100
3	300	400	500	600	700	800	900	1000	1100	1200	1024	1200	1100	1000	900	800	700	600	500	400	300
4	500	600	700	800	900	1000	1100	1200	1300	1400	1024	1400	1300	1200	1100	1000	900	800	700	600	500
5	700	800	900	1000	1100	1200	1300	1400	1500	1600	1024	1600	1500	1400	1300	1200	1100	1000	900	800	700
6	900	1000	1100	1200	1300	1400	1500	1600	1700	1800	1024	1800	1700	1600	1500	1400	1300	1200	1100	1000	900
7	1100	1200	1300	1400	1500	1600	1700	1800	1900	2000	1024	2000	1900	1800	1700	1600	1500	1400	1300	1200	1100
8	1300	1400	1500	1600	1700	1800	1900	2000	2047	2047	1024	2047	2047	2000	1900	1800	1700	1600	1500	1400	1300
9	1500	1600	1700	1800	1900	2000	2047	2047	2047	2047	1024	2047	2047	2047	2047	2000	1900	1800	1700	1600	1500
10–1015	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024
1016	1500	1600	1700	1800	1900	2000	2047	2047	2047	2047	1024	2047	2047	2047	2047	2000	1900	1800	1700	1600	1500
1017	1300	1400	1500	1600	1700	1800	1900	2000	2047	2047	1024	2047	2047	2000	1900	1800	1700	1600	1500	1400	1300
1018	1100	1200	1300	1400	1500	1600	1700	1800	1900	2000	1024	2000	1900	1800	1700	1600	1500	1400	1300	1200	1100
1019	900	1000	1100	1200	1300	1400	1500	1600	1700	1800	1024	1800	1700	1600	1500	1400	1300	1200	1100	1000	900
1020	700	800	900	1000	1100	1200	1300	1400	1500	1600	1024	1600	1500	1400	1300	1200	1100	1000	900	800	700
1021	500	600	700	800	900	1000	1100	1200	1300	1400	1024	1400	1300	1200	1100	1000	900	800	700	600	500
1022	300	400	500	600	700	800	900	1000	1100	1200	1024	1200	1100	1000	900	800	700	600	500	400	300
1023	100	200	300	400	500	600	700	800	900	1000	1024	1000	900	800	700	600	500	400	300	200	100
1024	0	0	100	200	300	400	500	600	700	800	1024	800	700	600	500	400	300	200	100	0	0
1025	0	0	0	0	100	200	300	400	500	600	1024	600	500	400	300	200	100	0	0	0	0

2.2.7.1.2 Computation Details

The first relevant step, in this example to generate an R_1 image from an R_0 image, is the convolution of the 7X7 element anti-aliasing kernel (Table 2.2) with the 1026X1026-element image matrix (Table 2.4) to produce an intermediate, appropriately blurred 1026X1026 matrix. The exact details of this step are described in Section 2.4.5 of the Convolution and Correlation ADS. The next step is to correlate the intermediate image with a 4X4 LaGrange Interpolation Kernel (Table 2.3) as described in Section 2.3.5 of the Four-Point Interpolation ADS. The last step is to sub-sample the interpolated image scene. In this example, if the rows and columns are labeled $0 \rightarrow 1025$ (zero-based indexing of row and column positions), then sub-sampling is achieved simply by removing the odd rows and odd columns. (Notice the last row and column are dropped.) The resulting R_1 image is a $513x513$ element matrix.

The R_1 image then becomes the input matrix for generating the R_2 image, and the same steps are applied once again. The resulting R_2 image is a 257X257 element matrix, which meets the exit criteria so no additional RRDS level images need be generated.

2.2.7.1.3 Algorithm Output

The R_1 and R_2 image matrices are provided in Table 2.5 and Table 2.6, respectively. Note that the R_1 output matrix generated by this example is a 513X513 element matrix, where the four 7X7 element regions at the corners of the matrix, are reflections of each other. The R_1 image matrix elements are symmetric in both the row and column dimensions. The R_2 image is a 257X257 element matrix, where the four 5X5 element regions at the corners of the matrix are not reflections of each other in either dimension. This example demonstrates the difference between reducing an even-sided matrix and an odd-sided matrix.

2.2.7.2 System Level

Successful symmetric pixel RRDS will yield a series of images scaled by a factor of 2n (n is an integer value) for R0 through Rn. For example, a full resolution image R0 of 16K X 16K pixels will have an R1 size of 8K X 8K pixels and R2 of 4K X 4K pixels, etc. The resulting RRDS imagery is to be free of down-sampled visual artifacts; such as pixel blocking, aliasing, and tonal inconsistencies.

Table 2.5: R_1 Matrix

	0	1	2	3	4	5	6	7-505	506	507	508	509	510	511	512
0	0	31	206	401	641	1020	1039	1024	1039	1020	641	401	206	31	0
1	203	424	637	842	1042	1042	1021	1024	1021	1042	1042	842	637	424	203
2	631	843	1050	1273	1438	1064	1004	1024	1004	1064	1438	1273	1050	843	631
3	1060	1280	1495	1761	1865	1087	984	1024	984	1087	1865	1761	1495	1280	1060
4	1423	1625	1827	1980	1921	1078	982	1024	982	1078	1921	1980	1827	1625	1423
5	1079	1091	1103	1098	1076	1026	1022	1024	1022	1026	1076	1098	1103	1091	1079
6	1002	993	984	979	983	1022	1026	1024	1026	1022	983	979	984	993	1002
7-505	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024
506	1002	993	984	979	983	1022	1026	1024	1026	1022	983	979	984	993	1002
507	1079	1091	1103	1098	1076	1026	1022	1024	1022	1026	1076	1098	1103	1091	1079
508	1423	1625	1827	1980	1921	1078	982	1024	982	1078	1921	1980	1827	1625	1423
509	1060	1280	1495	1761	1865	1087	984	1024	984	1087	1865	1761	1495	1280	1060
510	631	843	1050	1273	1438	1064	1004	1024	1004	1064	1438	1273	1050	843	631
511	203	424	637	842	1042	1042	1021	1024	1021	1042	1042	842	637	424	203
512	0	31	206	401	641	1020	1039	1024	1039	1020	641	401	206	31	0

Table 2.6: R_2 Matrix

	0	1	2	3	4	5-250	251	252	253	254	255	256
0	48	419	916	1066	1022	1024	1024	1035	1049	682	183	48
1	977	1603	1442	941	1028	1024	1026	983	1095	1697	1246	977
2	1328	1600	1311	959	1027	1024	1025	996	1053	1558	1456	1328
3	984	918	965	1037	1023	1024	1024	1029	1016	918	955	984
4	1025	1029	1027	1024	1024	1024	1024	1024	1025	1029	1026	1025
5-250	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024
251	1025	1027	1025	1024	1024	1024	1024	1024	1024	1026	1026	1025
252	994	969	998	1030	1024	1024	1024	1026	1022	974	982	994
253	1122	1139	1063	1014	1024	1024	1024	1020	1024	1112	1134	1122
254	1334	1877	1508	919	1029	1024	1026	977	1087	1876	1577	1334
255	417	925	1152	1011	1024	1024	1025	1011	1073	1126	626	417
256	48	419	916	1066	1022	1024	1024	1035	1049	682	183	48

This page intentionally includes only this statement.

2.3 Four-Point Interpolation

Algorithm Name	Four-Point Interpolation
Databases Required	LaGrange or Compromise Coefficients
Last Updated	8/31/09
Application Conditions	Apply to imagery during interactive processing to perform geometric transformations.
Purpose	Performs resampling of input image data to generate a desired output given a desired resampling function.

2.3.1 Overview

Four-point interpolation is a specific type of interpolation function. From an input image, an image resampler constructs pixel values of an output image by evaluating an interpolation function at each desired output sample location. The interpolator used by the reference image chain provided in this standard is the four-point interpolator described in this section. The LaGrange coefficients are used by this standard for all examples.

The choice of interpolation function is a trade-off between matching the input image to a desired degree of positional fidelity, preserving the Modulation Transfer Function (MTF) of the image when downsampling, and reducing the execution time of the function. To maintain as high a degree of positional fidelity as possible, the interpolation function is chosen to ensure the pixel values of the output image *exactly* agree with those of the input image at the overlap points of the input and output sampling grids. This is accomplished using the LaGrange coefficients with the four-point interpolator. To provide consistent MTF characteristics regardless of scaling factor the compromise coefficients may be used. To minimize execution time, the interpolation function usually has a simple form, so that it depends on only those input pixel values that lie within a small neighborhood of each out- put sample. The four-point interpolator is used by this standard for its execution speed, and its ability to be modified with new coefficients without effecting execution time.

Though the MTF of this algorithm is not constant when using LaGrange coefficients, it produces highly accurate positional results. Using the compromise coefficients helps preserve the MTF but at a loss of positional accuracy between the input and output sampling grids. This algorithm description provides the mathematical description for both one-dimensional and two-dimensional interpolations. While the description illustrates the implementation for a single point, practical instantiation of the four-point interpolator is accomplished by generating a 4x4 correlation kernel that is applied to the image for each pixel spacing desired.

Multiple correlations, therefore, may be required to accomplish a single geometric transformation such as scaling.

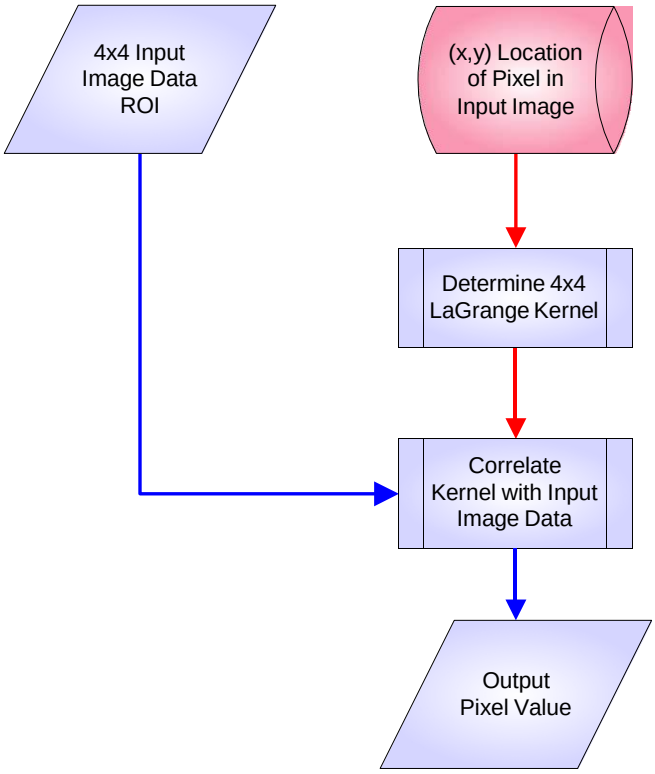
2.3.2 Implementation Options

- While equations are provided herein for LaGrange interpolation, a generic 4-point interpolation engine should be implemented to utilize a given table of coefficients (provided in Table 2.7 or Table 2.8). While small amounts of precision error may occur when using the table method, it is believed that these errors will be beneath any visual threshold for normal processing. Future versions of this standard may utilize different coefficients.

2.3.3 Inputs Required

- Input image region surrounding the desired output pixel location
- Output pixel location in input image coordinate system

2.3.4 Flow Diagram



2.3.5 Algorithm Details

To understand LaGrange interpolation in two dimensions, one must first understand the one-dimensional case. Implementation of a 1-D LaGrange interpolator must be performed using a four element kernel whose values are chosen based upon the spacing location of the interpolated pixel. A cubic polynomial is used to define the four-point LaGrange interpolator, $P(x)$. The polynomial uses the pixel locations $x_1, x_2, x_3,$ and x_4 and the respective pixel sample values $a_1, a_2, a_3,$ and a_4. Figure 2.1 depicts a 1-D graphical layout for the LaGrange parameters.

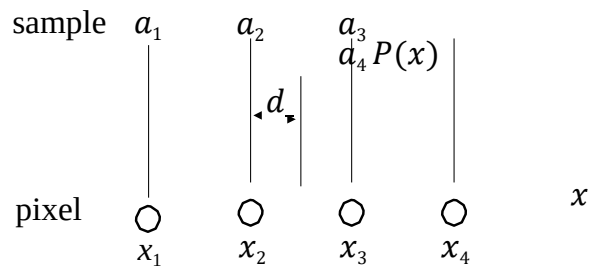


Figure 2.1: Graphical Representation of a 1-D LaGrange Interpolator

The four-point LaGrange polynomial is described in Equation 2.1,

$$\begin{aligned} P(x) = {} & \frac{(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_2)(x_1-x_3)(x_1-x_4)} a_1 + \\ & \frac{(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_1)(x_2-x_3)(x_2-x_4)} a_2 + \\ & \frac{(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_1)(x_3-x_2)(x_3-x_4)} a_{13} + \\ & \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_1)(x_4-x_2)(x_4-x_3)} a_4 \end{aligned} \tag{2.1}$$

where

- $P(x)$ is the interpolated value at location x,
- d is the distance between interpolated value location and previous pixel location, $d = x - x_2$,
- $x_1, x_2, x_3,$ and x_4 are the pixel locations, and
- $a_1, a_2, a_3,$ and a_4 are the pixel count values at the corresponding x locations.

Note: Equation 2.1 is of the form described in Equation 2.2.

$$P\,(x) \;=\; C_1a_1 \;+\; C_2a_2 \;+\; C_3a_3 \;+\; C_4a_4 \;=\; \sum_{m=1}^{4} C_m a_m \tag{2.2}$$

where the coefficients C_1 through C_4 in Equation 2.2 are defined below in Equation 2.3 through Equation 2.6:

$$C_1 = -\frac{d(d-1)(d-2)}{6} \tag{2.3}$$

$$C_2 = \frac{(d^2-1)(d-2)}{2} \tag{2.4}$$

$$C_3 = -\frac{d(d+1)(d-2)}{2} \tag{2.5}$$

$$C_4 = \frac{d(d^2-1)}{6} \tag{2.6}$$

Using $d = x - x_2$ such that $0 \le d < 1$ and $x_{i+1} - x_i = 1$.

An important implementation feature of LaGrange interpolation is that the coefficient values vary for specific sample spacings. These values are based on the interpolated pixel distance from the surrounding pixels. For example, if the spacing from the interpolated pixel to the previous pixel is a delta value of d, then the four LaGrange coefficients (c_1, c_2, c_3, and c_4) may be calculated for that specific delta value given the coefficients listed in Table 2.7.

Table 2.7 lists a sample set of coefficients calculated at 1/32 pixel spacings between 0/32 and 31/32. Actual coefficient values for other spacings may be determined through the calculations given above.

Table 2.7: Sample LaGrange Coefficients for Various Spacings

Spacing	**LaGrange Coefficients**			
d	C_1	C_2	C_3	C_4
0/32	0.00000	1.00000	0.00000	0.00000
1/32	-0.00993	0.98341	0.03172	-0.00520
2/32	-0.01892	0.96497	0.06433	-0.01038
3/32	-0.02699	0.94475	0.09773	-0.01549
4/32	-0.03418	0.92285	0.13184	-0.02051

(continued on next page)

Spacing	LaGrange Coefficients			
d	$C1$	$C2$	$C3$	$C4$
5/32	-0.04051	0.89937	0.16655	-0.02541
6/32	-0.04602	0.87439	0.20178	-0.03015
7/32	-0.05074	0.84801	0.23744	-0.03471
8/32	-0.05469	0.82031	0.27344	-0.03906
9/32	-0.05791	0.79140	0.30968	-0.04317
10/32	-0.06042	0.76135	0.34607	-0.04700
11/32	-0.06227	0.73027	0.38252	-0.05052
12/32	-0.06348	0.69824	0.41895	-0.05371
13/32	-0.06407	0.66536	0.45525	-0.05653
14/32	-0.06409	0.63171	0.49133	-0.05896
15/32	-0.06355	0.59740	0.52711	-0.06096
16/32	-0.06250	0.56250	0.56250	-0.06250
17/32	-0.06096	0.52711	0.59740	-0.06355
18/32	-0.05896	0.49133	0.63171	-0.06409
19/32	-0.05653	0.45525	0.66536	-0.06407
20/32	-0.05371	0.41895	0.69824	-0.06348
21/32	-0.05052	0.38252	0.73027	-0.06227
22/32	-0.04700	0.34607	0.76135	-0.06042
23/32	-0.04317	0.30968	0.79140	-0.05791
24/32	-0.03906	0.27344	0.82031	-0.05469
25/32	-0.03471	0.23744	0.84801	-0.05074
26/32	-0.03015	0.20178	0.87439	-0.04602
27/32	-0.02541	0.16655	0.89937	-0.04051
28/32	-0.02051	0.13184	0.92285	-0.03418
29/32	-0.01549	0.09773	0.94475	-0.02699
30/32	-0.01038	0.06433	0.96497	-0.01892
31/32	-0.005200	0.03172	0.98341	-0.00993

Once the coefficients have been determined, the interpolated pixel value is calculated by summing the products of the four nearest points with these coefficients. For the next interpolated pixel, the spacing is re-calculated to determine the next set of coefficients.

For two dimensions, product separability allows the coefficients to be expressed as follows in Equation 2.7:

$$C_{mn} = C_m(d_x)\, C_n(d_y) \tag{2.7}$$

such that $0 \leq d_x < 1$ and $0 \leq d_y < 1$.

This leads to an ultimate two-dimensional solution for calculating an output pixel value $P(x, y)$ using Equation 2.8. Implementation of Equation 2.8 may be optimized by representing the coefficients in a 4x4 kernel that may be correlated over the image in a single pass (see Section 2.4.5.2 for the algorithmic description of correlation),

$$P\,(x, y) = \sum_{n=1}^{4} \sum_{m=1}^{4} C_{mn} \cdot a_{mn} \tag{2.8}$$

where

- $P\,(x, y)$ is the interpolated value at location (x, y),
- C_{mn} is the LaGrange coefficient at location (m, n) of the kernel, and
- a_{mn} is the pixel count value at location (x_m, y_n).

Multiple kernels may be required for a single warping, scale, shear, or other geometric transformation requiring interpolation. For these operations, edge pixels are handled using edge mirroring as is described in the Convolution and Correlation ADS (see Section 2.4.5.3).

This concludes the derivation of LaGrange Interpolation using the process of four-point interpolation. If maintaining a constant MTF during scaling operations is more important than precise pixel placement, compromise coefficients may be used. Table 2.8 provides these coefficients for use with the four-point interpolator. Compromise coefficients are used by the same four-point interpolation process as the LaGrange example shown in this ADS.

Table 2.8: Sample Compromise Coefficients for Various Spacings

Spacing	**Compromise Coefficients**			
d	C_1	C_2	C_3	C_4
$0/64$	0.087	0.731	0.277	-0.095
$1/64$	0.081688	0.7285	0.282313	-0.0925
$2/64$	0.076375	0.726	0.287625	-0.09

(continued on next page)

Spacing	Compromise Coefficients			
d	$C1$	$C2$	$C3$	$C4$
3/64	0.071063	0.7235	0.292938	-0.0875
4/64	0.06575	0.721	0.29825	-0.085
5/64	0.060438	0.7185	0.303563	-0.0825
6/64	0.055125	0.716	0.308875	-0.08
7/64	0.049813	0.7135	0.314188	-0.0775
8/64	0.0445	0.711	0.3195	-0.075
9/64	0.039188	0.7085	0.324813	-0.0725
10/64	0.033875	0.706	0.330125	-0.07
11/64	0.028563	0.7035	0.335438	-0.0675
12/64	0.02325	0.701	0.34075	-0.065
13/64	0.017938	0.6985	0.346063	-0.0625
14/64	0.012625	0.696	0.351375	-0.06
15/64	0.007312	0.6935	0.356688	-0.0575
16/64	0.002	0.691	0.362	-0.055
17/64	-0.00331	0.6885	0.367313	-0.0525
18/64	-0.00863	0.686	0.372625	-0.05
19/64	-0.01394	0.6835	0.377938	-0.0475
20/64	-0.01925	0.681	0.38325	-0.045
21/64	-0.02456	0.6785	0.388563	-0.0425
22/64	-0.02988	0.676	0.393875	-0.04
23/64	-0.03519	0.6735	0.399188	-0.0375
24/64	-0.0405	0.671	0.4045	-0.035
25/64	-0.04581	0.6685	0.409813	-0.0325
26/64	-0.05113	0.666	0.415125	-0.03
27/64	-0.05644	0.6635	0.420438	-0.0275
28/64	-0.06175	0.661	0.42575	-0.025
29/64	-0.06706	0.6585	0.431063	-0.0225
30/64	-0.07238	0.656	0.436375	-0.02
31/64	-0.07769	0.6535	0.441688	-0.0175
32/64	-0.01625	0.444344	0.65225	-0.08034
33/64	-0.0175	0.441688	0.6535	-0.07769

(continued on next page)

Spacing	Compromise Coefficients			
d	C1	C2	C3	C4
34/64	-0.02	0.436375	0.656	-0.07238
35/64	-0.0225	0.431063	0.6585	-0.06706
36/64	-0.025	0.42575	0.661	-0.06175
37/64	-0.0275	0.420438	0.6635	-0.05644
38/64	-0.03	0.415125	0.666	-0.05113
39/64	-0.0325	0.409813	0.6685	-0.04581
40/64	-0.035	0.4045	0.671	-0.0405
41/64	-0.0375	0.399188	0.6735	-0.03519
42/64	-0.04	0.393875	0.676	-0.02988
43/64	-0.0425	0.388563	0.6785	-0.02456
44/64	-0.045	0.38325	0.681	-0.01925
45/64	-0.0475	0.377938	0.6835	-0.01394
46/64	-0.05	0.372625	0.686	-0.00863
47/64	-0.0525	0.367313	0.6885	-0.00331
48/64	-0.055	0.362	0.691	0.002
49/64	-0.0575	0.356688	0.6935	0.007312
50/64	-0.06	0.351375	0.696	0.012625
51/64	-0.0625	0.346063	0.6985	0.017938
52/64	-0.065	0.34075	0.701	0.02325
53/64	-0.0675	0.335438	0.7035	0.028563
54/64	-0.07	0.330125	0.706	0.033875
55/64	-0.0725	0.324813	0.7085	0.039188
56/64	-0.075	0.3195	0.711	0.0445
57/64	-0.0775	0.314188	0.7135	0.049813
58/64	-0.08	0.308875	0.716	0.055125
59/64	-0.0825	0.303563	0.7185	0.060438
60/64	-0.085	0.29825	0.721	0.06575
61/64	-0.0875	0.292938	0.7235	0.071063
62/64	-0.09	0.287625	0.726	0.076375
63/64	-0.0925	0.282313	0.7285	0.081688

2.3.6 Expected Output

The output of this operation is an interpolated value based on the input image surrounding pixel values.

2.3.7 Verification Strategies

ELT component level verification strategy provides an example approach and associated algorithms for verifying Four Point Interpolation. The softcopy display vendor is encouraged to use or tailor this approach to verify component level functionality and processes. System level verification strategy identifies expected (or required) system data outputs and results.

2.3.7.1 ELT Component Level

Four-point interpolation using LaGrange coefficients is verified in two parts. The first part demonstrates a single-point two-dimensional LaGrange interpolation case applied to a 4x4 input matrix segment. The first step in the single-point LaGrange interpolation processing is determining the composition of the 4x4 kernel from Table 2.7. A spacing of $^{16}/_{32}$ is used for the columnar direction (x-direction) and $^{8}/_{32}$ is used for the row-wise direction (y-direction). The second step is correlating the kernel with the input matrix segment.

The second part performs a LaGrange interpolation using the same kernel on a 10x10 input matrix. The 4x4 input matrix segment from the first part is embedded in the 10x10 input matrix used for the second part. The input matrix is constructed such that the pixel counts increase by 100 in the columnar direction and 200 in the row-wise dimension with the top left corner having a pixel value of 0 and the bottom right having a pixel count of 2047. The same interpolation space of $^{16}/_{32}$ in the columnar direction and $^{8}/_{32}$ in the row-wise dimension is used in this second part.

2.3.7.1.1 Algorithm Inputs

Table 2.9: 4x4 LaGrange Interpolation Kernel $\left({}^{16}/_{32} \times {}^{8}/_{32}\right)$

0.0034181250	-0.0307631250	-0.0307631250	0.0034181250
-0.0512693750	0.4614243750	0.4614243750	-0.0512693750
-0.0170900000	0.1538100000	0.1538100000	-0.0170900000
0.0024412500	-0.0219712500	-0.0219712500	0.0024412500

Table 2.10: Input 4x4 Matrix Segment for Single Point Verification

100	200	300	400
300	400	500	600
500	600	700	800
700	800	900	1000

Table 2.11: Input 10x10 Matrix for Matrix Verification

0	0	0	0	100	200	300	400	500	600
0	0	100	200	300	400	500	600	700	800
100	200	300	400	500	600	700	800	900	1000
300	400	500	600	700	800	900	1000	1100	1200
500	600	700	800	900	1000	1100	1200	1300	1400
700	800	900	1000	1100	1200	1300	1400	1500	1600
900	1000	1100	1200	1300	1400	1500	1600	1700	1800
1100	1200	1300	1400	1500	1600	1700	1800	1900	2000
1300	1400	1500	1600	1700	1800	1900	2000	2047	2047
1500	1600	1700	1800	1900	2000	2047	2047	2047	2047

2.3.7.1.2 Computation Details

2.3.7.1.2.1 Single Point Verification

A single point LaGrange interpolation is given for spacings of $^{16}/_{32}$ and $^{8}/_{32}$ in the columnar and row-wise dimensions, respectively. The 4x4 kernel is given in Table 2.9, and the 4x4 input matrix is shown in Table 2.10. Note that the input matrix is made simply by starting in the upper left corner with a value of 100 and increasing the values by 100 in the columnar dimension and 200 in the row-wise dimension.

Using the spacings of $^{16}/_{32}$ and $^{8}/_{32}$, Table 2.12 shows the input matrix expanded to visualize the spacings used for the selected kernel. The desired output pixel location $P(x, y)$ is shown in blue, and other more intuitive pixel values are calculated in red. The values in red are for illustration only, and are not calculated by the kernel used in this example.

To calculate the output pixel value, the input 4x4 matrix is correlated with the 4x4 LaGrange kernel. No edge mirroring is provided in this example, as this is a single point verification example. See the details on the 10x10 matrix example in Section 2.3.7.1.2.2 for more complete details on application to an entire image.

Table 2.12: LaGrange Single Point Verification Visualization

100				200				300				400
300				400	425	450	475	500				600
				450		$P(x,y)$ 500		550				
				500				600				
				550				540				
500				600	625	650	675	700				800
700				800				900				1000

2.3.7.1.2.2 10x10 Matrix Verification

A LaGrange interpolation on a 10x10 matrix is given for spacings of $^{16}/_{32}$ and $^{8}/_{32}$ in the columnar and row-wise dimensions, respectively. The same 4x4 kernel used for the single-point calculation, above, is used. The 10x10 input matrix is shown in Table 2.11. Note that the input matrix contains the 4x4 sample matrix used in the single-point example.

To calculate the output matrix values, the input 10x10 matrix is correlated with the 4x4 LaGrange kernel. Edge mirroring is used to handle elements of the matrix boundary. Table 2.13 illustrates the input 10x10 matrix with edge mirroring. The single-point 4x4 matrix contained in the upper left quadrant is highlighted in a rose color, while light-yellow, green, blue, and orange are used to illustrate the effect of mirroring the edge elements.

The calculations for correlating the 4x4 kernel with the 10x10 input matrix using edge mirroring can then be performed. The preliminary results are seen in Table 2.14. Assuming the output is in the 11-bit range of the input data, additional clipping must be performed on the shaded cells to produce the output (Table 2.15).

Table 2.13: 10x10 Matrix with Edge Mirroring

0	0	0	100	200	300	400	500	600	700	800	700	600
0	0	0	0	0	100	200	300	400	500	600	500	400
0	0	0	100	200	300	400	500	600	700	800	700	600
200	100	200	300	400	500	600	700	800	900	1000	900	800
400	300	400	500	600	700	800	900	1000	1100	1200	1100	1000
600	500	600	700	800	900	1000	1100	1200	1300	1400	1300	1200
800	700	800	900	1000	1100	1200	1300	1400	1500	1600	1500	1400
1000	900	1000	1100	1200	1300	1400	1500	1600	1700	1800	1700	1600
1200	1100	1200	1300	1400	1500	1600	1700	1800	1900	2000	1900	1800
1400	1300	1400	1500	1600	1700	1800	1900	2000	2047	2047	2047	2000
1600	1500	1600	1700	1800	1900	2000	2047	2047	2047	2047	2047	2047
1400	1300	1400	1500	1600	1700	1800	1900	2000	2047	2047	2047	2000
1200	1100	1200	1300	1400	1500	1600	1700	1800	1900	2000	1900	1800

Table 2.14: Preliminary LaGrange Interpolation Results

-7	0	14	73	178	278	378	478	591	591
19	87	198	300	400	500	600	700	813	813
184	300	400	500	600	700	800	900	1013	1013
388	500	600	700	800	900	1000	1100	1213	1213
588	700	800	900	1000	1100	1200	1300	1413	1413
788	900	1000	1100	1200	1300	1400	1500	1613	1613
988	1100	1200	1300	1400	1500	1600	1701	1817	1817
1188	1300	1400	1500	1600	1701	1805	1902	1994	1994
1403	1516	1616	1716	1817	1910	1991	2044	2054	2054
1509	1622	1722	1822	1925	2005	2041	2051	2051	2051

2.3.7.1.3 Algorithm Output

Below are the outputs for the two verification examples for LaGrange interpolation.

2.3.7.1.3.1 Single Point Verification

Output = 500

2.3.7.1.3.2 10x10 Matrix Verification

Table 2.15: Output 10x10 LaGrange Interpolation Matrix

0	0	14	73	178	278	378	478	591	591
19	87	198	300	400	500	600	700	813	813
184	300	400	500	600	700	800	900	1013	1013
388	500	600	700	800	900	1000	1100	1213	1213
588	700	800	900	1000	1100	1200	1300	1413	1413
788	900	1000	1100	1200	1300	1400	1500	1613	1613
988	1100	1200	1300	1400	1500	1600	1701	1817	1817
1188	1300	1400	1500	1600	1701	1805	1902	1994	1994
1403	1516	1616	1716	1817	1910	1991	2044	2047	2047
1509	1622	1722	1822	1925	2005	2041	2047	2047	2047

2.3.7.2 System Level

Successful Four Point Interpolation implementation will include La Grange two-dimensional interpolation. Other Interpolators such as Bi-linear and Nearest Neighbor should also be included as display processing options. The resulting imagery is to be free of Interpolator visual artifacts; such as MTF variability, structured noise, and compression artifacts.

2.4 Convolution and Correlation

Algorithm Name	Convolution and Correlation
Databases Required	Kernel database (MTFR, MTFE, MTFC, antialiasing, or interpolation)
Last Updated	10/12/09
Application Conditions	Apply this algorithm to a single band of image data. The nature of the image data will vary based on the purpose of the selected kernel.
Purpose	Applies a spatial kernel to a single plane of image data.

2.4.1 Overview

Convolution and correlation play a key role in many image processing chains. Both algorithms compute a weighted sum of several input pixels for a given output pixel of a single band of image data. While different implementations can achieve varying levels of performance, this section describes the required computations for accurate convolution or correlation operations.

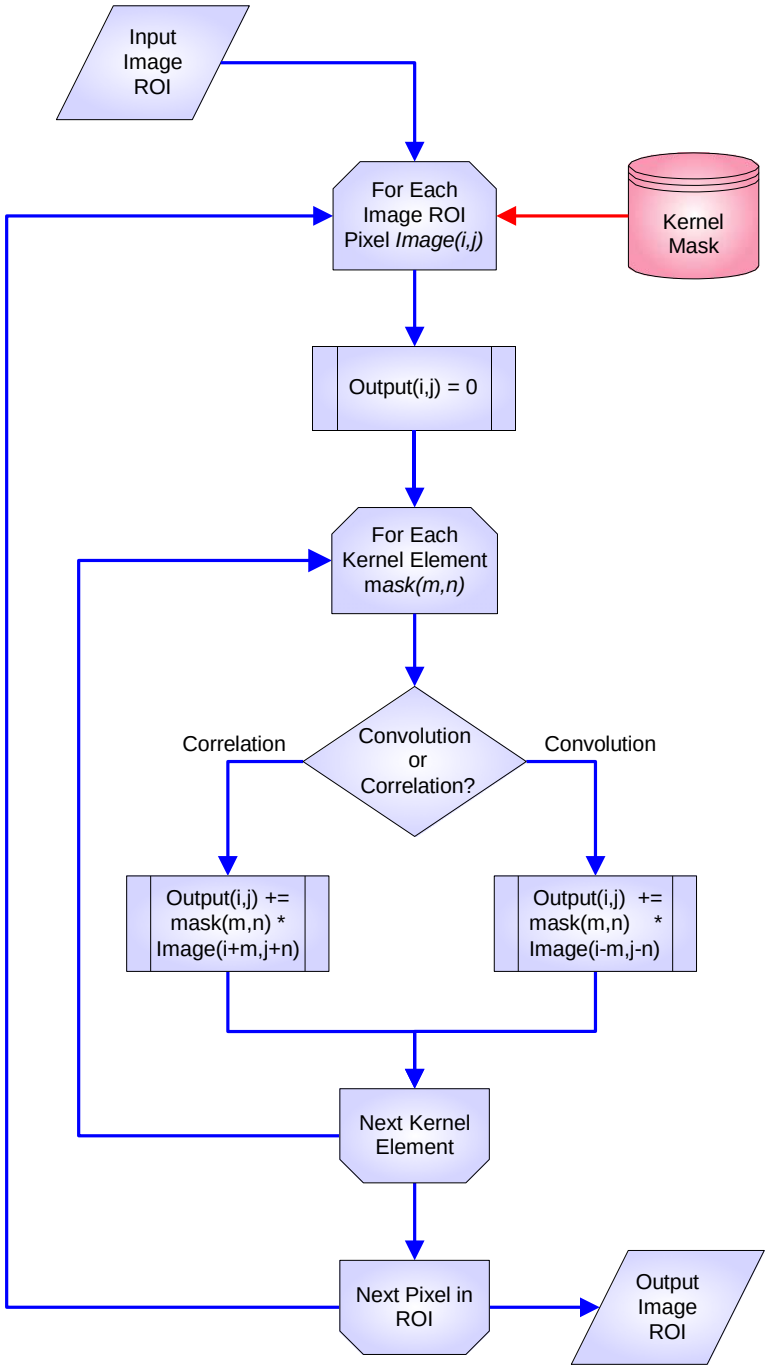
2.4.2 Implementation Options

- Not Applicable (N/A)

2.4.3 Inputs Required

- Input image (or region of input image)
- Convolution or Correlation Kernel
  - MTF Restoration (MTFR) limited to be no larger than 15X15
  - MTF Compensation (MTFC)/MTF Enhancement (MTFE) limited to be no larger than 5X5
  - Anti-aliasing kernel for RRDS or Scaling limited to be no larger than 7X7
  - Four-Point interpolator kernel

2.4.4 Flow Diagram



2.4.5 Algorithm Details

2.4.5.1 Convolution

In the discrete form, the convolution of two functions is shown in Equation 2.9

$$P\ (i,j)\ =\ \sum_{n=a}^{b}\sum_{m=c}^{d} mask(m,n)Image(i-m,j-n) \tag{2.9}$$

where

- $p(i,j)$ is the output pixel value for position i,j,
- $mask(m,n)$ is the kernel mask used to filter the image,
- $Image(i,j)$ is the input image data,
- m and n represent the position within the kernel mask,
- M and N are the dimensions of the kernel,
- a,b,c, and d are calculated according to Equation 2.10 or Equation 2.11 for even- or odd-sided kernels, respectively.

$$\begin{aligned} a &= -{}^{M}/_{2} \\ b &= {}^{M}/_{2} - 1 \\ c &= -{}^{N}/_{2} \\ d &= {}^{N}/_{2} - 1 \end{aligned} \tag{2.10}$$

$$\begin{aligned} a &= {}^{(1-M)}/_{2} \\ b &= {}^{(M-1)}/_{2} \\ c &= {}^{(1-N)}/_{2} \\ d &= {}^{(N-1)}/_{2} \end{aligned} \tag{2.11}$$

These equations are written with the convention of the center value of the kernel having coordinates $(0,0)$. Negative indices into the kernel are relative to the center pixel. Indices to pixels outside the image are generated as needed using the boundary handling convention listed below in Section 2.4.5.3.

2.4.5.2 Correlation

In the discrete form, the correlation of two real-valued functions is shown in Equation 2.12:

$$P\ (i,j)\ =\ \sum_{n=a}^{b}\sum_{m=c}^{d} mask(m,n)Image(i+m,j+n) \tag{2.12}$$

where

- $p(i,j)$ is the output pixel value for position i,j,
- $mask(m,n)$ is the kernel mask used to process the image,
- $Image(i,j)$ is the input image data,
- m and n represent the position within the kernel mask,
- M and N are the dimensions of the kernel,
- a,b,c, and d are calculated according to Equation 2.13 or Equation 2.14 for even- or odd-sided kernels, respectively.

$$\begin{aligned} a &= -{}^{M}/_{2} + 1 \\ b &= {}^{M}/_{2} \\ c &= -{}^{N}/_{2} + 1 \\ d &= {}^{N}/_{2} \end{aligned} \tag{2.13}$$

$$\begin{aligned} a &= {}^{(1-M)}/_{2} \\ b &= {}^{(M-1)}/_{2} \\ c &= {}^{(1-N)}/_{2} \\ d &= {}^{(N-1)}/_{2} \end{aligned} \tag{2.14}$$

These equations are written with the convention of the center value of the kernel having coordinates $(0,0)$. Negative indices into the kernel are relative to the center pixel. Indices to pixels outside the image are generated as needed using the boundary handling convention listed below in Section 2.4.5.3.

Correlation and convolution are very similar operations. However, they are not equivalent and there are critical differences in implementation which must be handled correctly to obtain accurate results. The primary difference is the orientation of the kernel mask relative to the image pixels (note the difference in indexing of the "Image" component in Equation 2.9 as compared with Equation 2.12). Convolution effectively 'flips' the kernel mask in the horizontal and vertical dimensions with respect to the image pixels. Correlation does NOT perform this kernel 'flip.' This orientation issue also results in differences when determining the center pixel of even-sided kernels (note the differences between Equations 2.10 and 2.13). Thus, with the exception of kernels that are quadrant symmetric, the results of correlation and convolution will differ for an arbitrary kernel.

2.4.5.3 Image Boundary Handling

The required approach for handling image boundaries is to use the implementation known as "Mirror Edge - Odd." In this implementation, original outer image pixel values are used as the pivot points, mirroring the interior data points, as depicted in Table 2.16. The gray areas are the mirrored pixels.

This is the required method for handling image boundaries. The edge information at the boundaries is maintained, allowing for a more natural image representation.

Table 2.16: Mirror Edge - Odd Boundary Handling for Convolution and Correlation

i	h	g	h	i	h	g
f	e	d	e	f	e	d
c	b	a	b	c	b	a
f	e	d	e	f	e	d
i	h	g	h	i	h	g
f	*e*	*d*	*e*	*f*	*e*	*d*
c	*b*	*a*	*b*	*c*	*b*	*a*

2.4.6 Expected Output

The output image must be the same physical dimensions as the input image. Blurring, sharpening, or other effects will vary based upon the kernel used in the convolution or correlation process. Image boundaries should be used to achieve the proper output dimension.

2.4.7 Verification Strategies

ELT component level verification strategy provides an example approach and associated algorithms for verifying Convolution and Correlation. The softcopy display vendor is encouraged to use or tailor this approach to verify component level functionality and processes. System level verification strategy identifies expected (or required) system data outputs and results.

2.4.7.1 ELT Component Level

Verification examples of convolution processing are demonstrated using both a 5x5 kernel and a 4x4 kernel. This is done to cover processing issues arising from the differences between the application of even-sided and odd-sided kernels. For verification of correlation processing, see the verification examples found in the Four-Point Interpolation (Section 2.3.7.1) and Scaling (Section 2.7.7.1) algorithm descriptions.

2.4.7.1.1 Algorithm Inputs

2.4.7.1.1.1 5x5 Example

The 5x5 kernel used for the first verification example is given in Table 2.17. This kernel is designed to have a component sum of 1 over the 25 elements. This is an asymmetric kernel and is used to ensure the convolution operation is implemented correctly according to the mathematical definition given in Equation 2.9.

Table 2.17: 5x5 Convolution Kernel

-0.0004	-0.0022	-0.0032	-0.0023	-0.0005
-0.0021	-0.0112	-0.0220	-0.0122	-0.0016
-0.0031	-0.0021	1.1180	-0.0019	-0.0029
-0.0020	-0.0102	-0.0200	-0.0092	-0.0017
-0.0003	-0.0019	-0.0030	-0.0018	-0.0002

The input matrix used for this example is shown in Table 2.18. This is an 11-bit unsigned integer matrix with an average digital count value of 1136. The input matrix is color coded to illustrate three different gray wedges separated by a thin, two-pixel wide low-density border. This input matrix provides a good visual example of the effects of the above 5x5 sharpening kernel.

Table 2.18: 5x5 Convolution Input Matrix

139	310	147	220	205	224	215	314	223	138
221	199	198	179	201	253	223	181	1982	2047
178	239	151	119	151	232	1869	1901	1965	1931
278	232	181	88	119	1901	1929	1869	1482	1588
1889	1928	2047	1989	1897	1931	1597	1655	1648	1642
1971	2028	1949	1982	1948	1545	1582	1536	1551	1563
892	1022	879	969	935	1970	1693	1656	1619	1579
876	992	970	873	1022	1838	1955	1665	1618	1608
1001	927	971	913	933	931	1951	1974	1539	1591
944	996	950	875	982	923	963	2003	1973	1499

2.4.7.1.1.2 4x4 Example

The 4x4 kernel used in the second verification example is given in Table 2.19. This kernel is constructed in a similar fashion to the 5x5 kernel. It also has a component sum of 1 over its 16 elements. This kernel is provided to illustrate the proper handling of an even-sided kernel. This is also the same kernel used for the verification of Four-Point interpolation and given in Table 2.9. However, in this instance, the kernel is being used to illustrate a convolution operation, not the correlation operation used for interpolation.

Table 2.19: 4x4 Convolution Kernel

0.0034181250	-0.0307631250	-0.0307631250	0.0034181250
-0.0512693750	0.4614243750	0.4614243750	-0.0512693750
-0.0170900000	0.1538100000	0.1538100000	-0.0170900000
0.0024412500	-0.0219712500	-0.0219712500	0.0024412500

The input matrix used in this example is shown in Table 2.19. Because this kernel is designed to shift data in a correlation operation, the input matrix is designed to be a simple gray sweep, similar to the one used in the Four-Point interpolation verification example. The convolution operation, however, will produce markedly different results.

Table 2.20: 4x4 Input Convolution Matrix

0	0	0	0	100	200	300	400	500	600
0	0	100	200	300	400	500	600	700	800
100	200	300	400	500	600	700	800	900	1000
300	400	500	600	700	800	900	1000	1100	1200
500	600	700	800	900	1000	1100	1200	1300	1400
700	800	900	1000	1100	1200	1300	1400	1500	1600
900	1000	1100	1200	1300	1400	1500	1600	1700	1800
1100	1200	1300	1400	1500	1600	1700	1800	1900	2000
1300	1400	1500	1600	1700	1800	1900	2000	2047	2047
1500	1600	1700	1800	1900	2000	2047	2047	2047	2047

2.4.7.1.2 Computation Details

2.4.7.1.2.1 5x5 Example

The first step in the convolution process is to properly mirror the edges of the input matrix or image. The input matrix for this example is given in Table 2.18 and is shown with edge mirroring applied in Table 2.21. The matrix is color coded to illustrate handling of edge data at the corners.

Table 2.21: 5x5 Convolution Matrix with Edge Mirroring Applied

151	239	178	239	151	119	151	232	1869	1901	1965	1931	1965	1901
198	199	221	199	198	179	201	253	223	181	1982	2047	1982	181
147	310	139	310	147	220	205	224	215	314	223	138	223	314
198	199	221	199	198	179	201	253	223	181	1982	2047	1982	181
151	239	178	239	151	119	151	232	1869	1901	1965	1931	1965	1901
181	232	278	232	181	88	119	1901	1929	1869	1482	1588	1482	1869
2047	1928	1889	1928	2047	1989	1897	1931	1597	1655	1648	1642	1648	1655
1949	2028	1971	2028	1949	1982	1948	1545	1582	1536	1551	1563	1551	1536
879	1022	892	1022	879	969	935	1970	1693	1656	1619	1579	1619	1656
970	992	876	992	970	873	1022	1838	1955	1665	1618	1608	1618	1665
971	927	1001	927	971	913	933	931	1951	1974	1539	1591	1539	1974
950	996	944	996	950	875	982	923	963	2003	1973	1499	1973	2003
971	927	1001	927	971	913	933	931	1951	1974	1539	1591	1539	1974
970	992	876	992	970	873	1022	1838	1955	1665	1618	1608	1618	1665

Once the edges are mirrored, Equation 2.9 is applied. *Image* is the input matrix

and *mask* is the kernel given in Table 2.17. The initial output of this operation is shown in Table 2.22. Values of the output matrix that are outside the 11-bit range of the input matrix are color coded to indicate the need for clipping. All values outside the range of the desired output are clipped to the range of the output data type. In this example, the values less than 0 are mapped to 0, and the values greater than 2047 are mapped to 2047. Final output can be seen in Table 2.25.

Table 2.22: 5x5 Convolution Interim Output Matrix

131	323	142	224	204	216	189	256	77	-51
222	197	200	179	195	226	144	68	2075	2149
158	227	132	94	105	153	1959	1977	2016	1967
187	135	77	-30	0	1980	1982	1893	1451	1567
1973	2019	2156	2090	1972	1978	1587	1652	1653	1648
2044	2107	2018	2054	2003	1528	1566	1521	1542	1557
834	978	819	916	871	2014	1695	1655	1620	1577
859	989	966	853	1007	1891	1994	1658	1613	1607
1006	923	974	909	914	884	2003	2008	1516	1579
942	1001	950	867	979	882	880	2031	2010	1483

2.4.7.1.2.2 4x4 Example

As in the 5x5 example, the first step in the convolution process is to properly mirror the edges of the input matrix or image. The input matrix for this example is given in Table 2.20 and is shown with edge mirroring applied in Table 2.23. This matrix is also color coded to illustrate handling of edge data at the corners. It is important to note that the edge mirroring process produces slightly different results when using an even-sided kernel.

Table 2.23: 4x4 Convolution Example Input Matrix with Edge Mirroring

0	0	0	100	200	300	400	500	600	700	800	700	600
0	0	0	0	0	100	200	300	400	500	600	500	400
0	0	0	100	200	300	400	500	600	700	800	700	600
200	100	200	300	400	500	600	700	800	900	1000	900	800
400	300	400	500	600	700	800	900	1000	1100	1200	1100	1000
600	500	600	700	800	900	1000	1100	1200	1300	1400	1300	1200
800	700	800	900	1000	1100	1200	1300	1400	1500	1600	1500	1400
1000	900	1000	1100	1200	1300	1400	1500	1600	1700	1800	1700	1600
1200	1100	1200	1300	1400	1500	1600	1700	1800	1900	2000	1900	1800
1400	1300	1400	1500	1600	1700	1800	1900	2000	2047	2047	2047	2000
1600	1500	1600	1700	1800	1900	2000	2047	2047	2047	2047	2047	2047
1400	1300	1400	1500	1600	1700	1800	1900	2000	2047	2047	2047	2000
1200	1100	1200	1300	1400	1500	1600	1700	1800	1900	2000	1900	1800

After mirroring the edges, Equation 2.9 is applied. *Image* is the input matrix and *mask* is the kernel given in Table 2.19. The initial output of this operation is shown in Table 2.24. Values of the output matrix that are outside the 11-bit range of the input matrix are color coded to indicate the need for clipping. All values outside the range of the desired output are clipped to the range of the output data type. In this example, the values less than 0 are mapped to 0, and the values greater than 2047 are mapped to 2047. Final output can be seen in Table 2.26.

Table 2.24: 4x4 Convolution Example Interim Output

-12	21	96	183	284	384	484	584	697	697
93	192	298	400	500	600	700	800	912	912
285	400	500	600	700	800	900	1000	1112	1112
487	600	700	800	900	1000	1100	1200	1312	1312
687	800	900	1000	1100	1200	1300	1400	1512	1512
887	1000	1100	1200	1300	1400	1500	1600	1712	1712
1087	1200	1300	1400	1500	1600	1700	1801	1919	1919
1287	1400	1500	1600	1700	1801	1908	1994	2037	2037
1509	1622	1722	1822	1925	2005	2041	2051	2051	2051
1403	1516	1616	1716	1817	1910	1991	2044	2054	2054

2.4.7.1.3 Algorithm Output

2.4.7.1.3.1 5x5 Example

Table 2.25: 5x5 Convolution Example Output Matrix

131	323	142	224	204	216	189	256	77	0
222	197	200	179	195	226	144	68	2047	2047
158	227	132	94	105	153	1959	1977	2016	1967
187	135	77	0	0	1980	1982	1893	1451	1567
1973	2019	2047	2047	1972	1978	1587	1652	1653	1648
2044	2047	2018	2047	2003	1528	1566	1521	1542	1557
834	978	819	916	871	2014	1695	1655	1620	1577
859	989	966	853	1007	1891	1994	1658	1613	1607
1006	923	974	909	914	884	2003	2008	1516	1579
942	1001	950	867	979	882	880	2031	2010	1483

2.4.7.1.3.2 4x4 Example

Table 2.26: 4x4 Convolution Example Output Matrix

0	21	96	183	284	384	484	584	697	697
93	192	298	400	500	600	700	800	912	912
285	400	500	600	700	800	900	1000	1112	1112
487	600	700	800	900	1000	1100	1200	1312	1312
687	800	900	1000	1100	1200	1300	1400	1512	1512
887	1000	1100	1200	1300	1400	1500	1600	1712	1712
1087	1200	1300	1400	1500	1600	1700	1801	1919	1919
1287	1400	1500	1600	1700	1801	1908	1994	2037	2037
1509	1622	1722	1822	1925	2005	2041	2047	2047	2047
1403	1516	1616	1716	1817	1910	1991	2044	2047	2047

2.4.7.2 System Level

Successful Convolution and Correlation implementations are verified mathematical and visually. Convolution and correlation kernels (filters) are mathematical functions when applied to an image data matrix yield results verifiable by calculation. MTFC filters are designed to selectively output a sharper or softer image

when convolved with an image data set. ELT systems are to include a series of MTFC kernels ranging from increased image blurring to increased image sharpening. Imagery will be processed and displayed using this range of MTFCs, then visually assess to determine if the expected and desired range of sharpening or blurring is achieved. The resulting imagery is to be free of MTF variability, aliasing, and structured noise artifacts.

2.5 One-Dimensional Look-Up Table Application

Algorithm Name	One-Dimensional Look-Up Table Application
Databases Required	Tone Transfer Curve (TTC) Database (d/b) or Data Remap d/b
Last Updated	12/03/08
Application Conditions	Apply selected LUT to a single plane of image data.
Purpose	Map image data to a new value based on the desired LUT curve shape.

2.5.1 Overview

One Dimensional Look-Up Tables (1-D LUTs) are implemented using the value that is input to a device, X, as an index into an array containing the device's desired output value, Y . The value, X_{max}, is determined by the possible range of the input data. A LUT for an 8-bit system using this method contains 256 entries, beginning at zero (X_{max} = 255), while a LUT for a higher-precision 12-bit system would contain 4096 entries, (X_{max} = 4095). This standard requires implementations to minimally handle 1-D LUTs for 8, 11, 12, and 16 bit images. A 1-D LUT is applicable to only a single band of image data. Multiple bands require application of band specific LUTs.

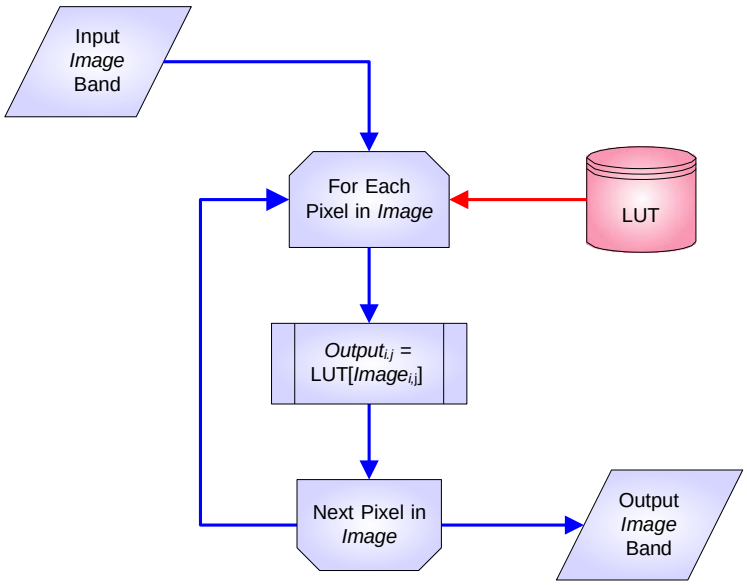
2.5.2 Implementation Options

- N/A

2.5.3 Inputs Required

- Input pixel values
- 1-D LUT file formatted in accordance with Appendix A.1.2.2

2.5.4 Flow Diagram



2.5.5 Algorithm Details

Equation 2.15 provides the implementation of a 1-D LUT operation for a complete sampling of the dynamic range. A completely sampled 1-D LUT contains an entry for every possible input digital count value.

$$Y = \begin{cases} LUT\ [0], & x < 0 \\ LUT\ [X], & 0 \leq X \leq X_{max} \\ LUT\ [X_{max}], & X > X_{max} \end{cases} \tag{2.15}$$

2.5.6 Expected Output

The output of this algorithm is a set of output values determined from a set of input values.

2.5.7 Verification Strategies

ELT component level verification strategy provides an example approach and associated algorithms for verifying One-Dimensional Look-Up Table (1-D LUT) Applications. The softcopy display vendor is encouraged to use or tailor this approach to verify component level functionality and processes. System level verification strategy identifies expected (or required) system data outputs and results.

2.5.7.1 ELT Component Level

The verification of the TTC algorithm consists of a single example. This example utilizes TTC member 0 of family 0, which may be found in the database that accompanies this document and is described in Appendix A. While a LUT can map to a different bit-depth for it's output, the 11-bit matrix used for this example produces an 11-bit output image.

2.5.7.1.1 Algorithm Inputs

Table 2.27: TTC Verification Input Matrix Data

643	1611	1578	1070	1459	985	998	2047	1755	0
1265	1446	1239	672	1172	1125	474	1383	1400	592
1303	702	901	1645	491	952	1616	571	1265	871
892	1049	1713	1751	1501	850	884	846	681	1484
935	973	1049	385	969	465	1057	1404	867	601
1518	918	1214	1413	68	1286	1091	846	1066	1759
1552	774	283	1235	744	1210	1260	943	990	660
1286	647	30	1878	1806	981	880	178	1150	1108
1408	1053	1527	825	1916	1024	960	1413	1396	727
1032	1002	1379	842	1446	897	1468	1243	930	1688

2.5.7.1.2 Computation Details

Computing the output of a TTC is straightforward, as there is no interpolation to be performed. Each element in the input matrix given in Table 2.27 is used as an index into the TTC Family 0 member 0 table available with this document. An extracted portion of the TTC is included in Table 2.28 to illustrate the mapping between input and output values. The output of this operation on the example matrix is seen in Table 2.29.

Table 2.28: TTC Family 0 Member 0 Sparsely Sampled Examples

Input	Output	Input	Output	Input	Output
0	3	918	1784	1286	1915
30	336	930	1790	1303	1919
68	594	935	1793	1379	1939
178	1008	943	1796	1383	1940
283	1232	952	1800	1396	1943
385	1383	960	1803	1400	1944
465	1474	969	1807	1404	1945
474	1484	973	1809	1408	1945
491	1501	981	1812	1413	1947
571	1573	985	1814	1446	1954
592	1589	990	1815	1459	1957
601	1597	998	1819	1468	1959
643	1628	1002	1820	1484	1963
647	1631	1024	1829	1501	1966
660	1640	1032	1832	1518	1969
672	1648	1049	1838	1527	1971
681	1654	1053	1840	1552	1976
702	1668	1057	1842	1578	1981
727	1683	1066	1845	1611	1987
744	1694	1070	1846	1616	1988
774	1711	1091	1854	1645	1993
825	1739	1108	1860	1688	2001
842	1748	1125	1866	1713	2005
846	1750	1150	1874	1751	2010
850	1752	1172	1881	1755	2011
867	1761	1210	1893	1759	2012
871	1762	1214	1894	1806	2018
880	1767	1235	1901	1878	2028
884	1769	1239	1901	1916	2033
892	1773	1243	1903	2047	2047
897	1775	1260	1908		
901	1777	1265	1909		

2.5.7.1.3 Algorithm Outputs

Table 2.29: TTC Numerical Output Verification

1628	1987	1981	1846	1957	1814	1819	2047	2011	3
1909	1954	1901	1648	1881	1866	1484	1940	1944	1589
1919	1668	1777	1993	1501	1800	1988	1573	1909	1762
1773	1838	2005	2010	1966	1752	1769	1750	1654	1963
1793	1809	1838	1383	1807	1474	1842	1945	1761	1597
1969	1784	1894	1947	594	1915	1854	1750	1845	2012
1976	1711	1232	1901	1694	1893	1908	1796	1815	1640
1915	1631	336	2028	2018	1812	1767	1008	1874	1860
1945	1840	1971	1739	2033	1829	1803	1947	1943	1683
1832	1820	1939	1748	1954	1775	1959	1903	1790	2001

2.5.7.2 System Level

Successful 1-D LUT implementation is verified numerically. The image matrix input data is translated (or changed) to the corresponding value in the TTC LUT data matrix. For example, a linear LUT (TTC-32) will translate each input value into the same output value 0 maps to 0, 128 maps to 128, and 255 maps to 255, which equates to no tonal changes. The SIPS database provides 12 Families and 64 members TTCs outputting a variety of tonal remaps of the image data. ELT vendors should verify their 1-D LUT implementation against a sufficient sample of these TTC tables.

This page intentionally includes only this statement.

2.6 Dynamic Range Adjustment

Algorithm Name	Dynamic Range Adjustment (DRA)
Databases Required	histogram results
Last Updated	8/31/09
Application Conditions	Apply to a single plane of image data that has not been previously enhanced.
Purpose	The DRA maps the desired range of image data into a predetermined range of values for further standardized processing.

2.6.1 Overview

The DRA methodology is implemented as part of the Interactive Processing step described in Section 3.4. It is used to perform tonal processing of the raw data, mapping it from the dynamic range of the imaging device to that of a well-defined metric. Once mapped, standardized enhancement processing can be applied. For this standard, the well-defined metric is the entire range of the bit-depth of the image data.

This algorithm permits the adjustment and optimization of an image's dynamic range so that the maximum amount of information can be obtained from all portions of the tonal range. This algorithm does clip information beyond the desired range.

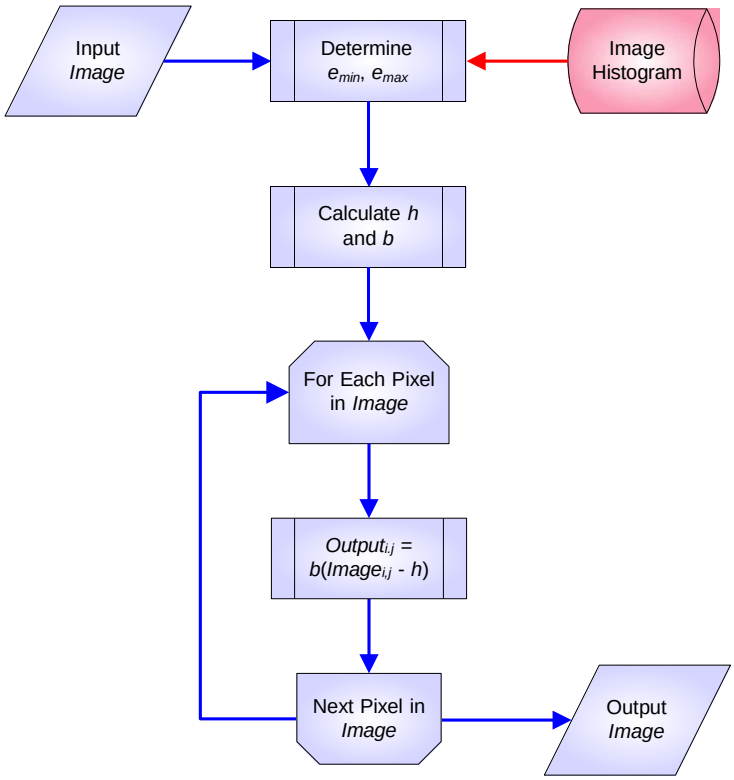
2.6.2 Implementation Options

- N/A

2.6.3 Inputs Required

- DRA parameters, p_{min} and p_{max}, for performing a DRA on a given image

2.6.4 Flow Diagram



2.6.5 Algorithm Details

As an automatic operation, p_{min} and p_{max} must be set according to the desired sensor. More information can be found in the proper appendix for the sensor type in question. In a manual mode, a mechanism must be provided to allow interactive selection of these penetration points.

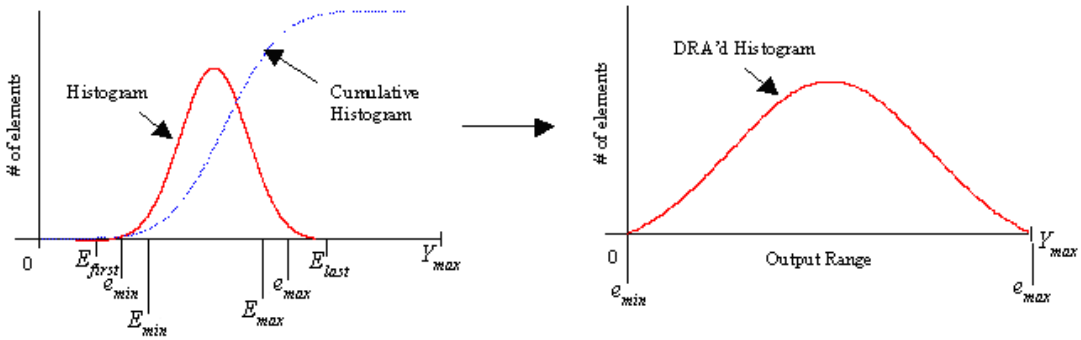


Figure 2.2: Auto-DRA Algorithm Using Penetration Points

The auto-DRA algorithm, illustrated in Figure 2.2, uses cumulative image histogram percentage points to select the input dynamic range to be displayed. Table 2.30 lists the parameters of the automatic DRA algorithm, while Equation 2.16 through Equation 2.20 are used to modify input image values;

$$e_{min} = max(E_{first}, E_{min} - A \cdot (E_{max} - E_{min})) \tag{2.16}$$

$$e_{max} = min(E_{last}, E_{max} + B \cdot (E_{max} - E_{min})) \tag{2.17}$$

$$y = b \cdot (x - h) \tag{2.18}$$

where

$$b = \frac{Y_{max}}{e_{max} - e_{min}} \tag{2.19}$$

$$h = e_{min} \tag{2.20}$$

Table 2.30: Parameters of the DRA Algorithm

DRA Parameter	Definition
p_{min}	Cumulative histogram percentage value that defines the lower end-point of the dynamic range to be displayed
p_{max}	Cumulative histogram percentage value that defines the upper end-point of the dynamic range to be displayed
A	E_{min} modifier
B	E_{max} modifier
E_{first}	Smallest occupied bin in the image histogram
E_{last}	Largest occupied bin in the image histogram
E_{min}	p^{th}_{min} percentage point of the image histogram
E_{max}	p^{th}_{max} percentage point of the image histogram
e_{min}	Adjusted E_{min} value
e_{max}	Adjusted E_{max} value
b	Multiplier value used to brighten image data
h	Subtractor value used to reduce haze in image
x	Input pixel count
y	Output pixel value
Y_{max}	Maximum output dynamic range value

2.6.6 Expected Output

The output of this algorithm is an image whose dynamic range has been adjusted to yield the maximum amount of information in not only the midtones, but also in shadow and highlight regions.

2.6.7 Verification Strategies

ELT component level verification strategy provides an example approach and associated algorithms for verifying Dynamic Range Adjustment (DRA) Applications. The softcopy display vendor is encouraged to use or tailor this approach to verify component level functionality and processes. System level verification strategy identifies expected (or required) system data outputs and results.

2.6.7.1 ELT Component Level

The verification example for dynamic range adjustment uses the default parameters listed in Table 2.31. These parameters are chosen purely for verification purposes of the algorithm. Actual values used for processing of real imagery are sensor dependent and are likely found in the proper appendix of this document. In this example, an input matrix representing an image file (Table 2.32) is used to illustrate the intermediate calculations used to derive the correct output matrix.

2.6.7.1.1 Algorithm Inputs

Table 2.31: DRA Algorithm Parameters

Parameter	Value
p_{min}	2%
p_{max}	98%
A	0.2
B	0.4

Table 2.32: DRA Input 10x10, 11-bit Matrix

694	923	915	795	887	775	778	1026	957	542
841	884	835	701	819	808	654	869	873	682
850	708	755	931	658	767	924	677	841	748
753	790	947	956	897	743	751	742	703	893
763	772	790	633	771	652	792	874	747	684
901	759	829	876	558	846	800	742	794	958
909	725	609	834	718	828	840	765	776	698
846	695	549	986	969	774	750	584	814	804
875	791	903	737	995	784	769	876	872	714
786	779	868	741	884	754	889	836	762	941

2.6.7.1.2 Computation Details

Figure 2.3 shows a histogram for the test image given in Table 2.32. Intermediate values calculated by the algorithm are listed in Table 2.33. The resulting histogram of the output image (Table 2.34) is then shown in Figure 2.4.

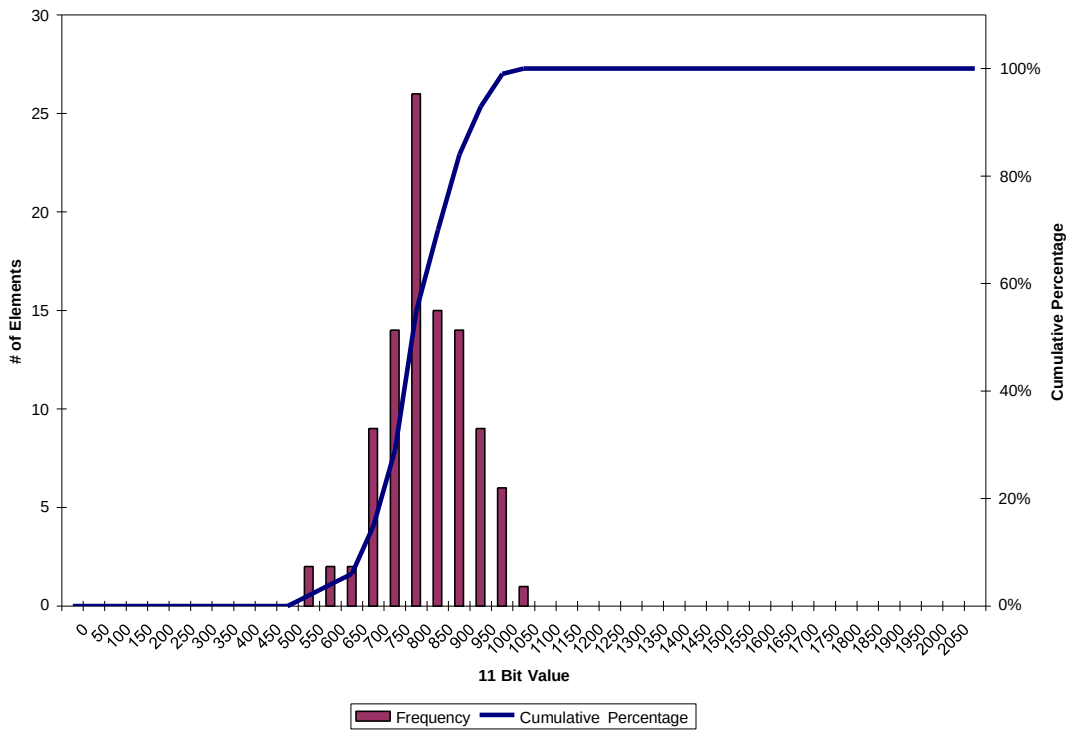


Figure 2.3: Histogram of Original 10x10, 11-bit Matrix

Table 2.33: DRA Example Interim Calculated Values

DRA Parameter	Value
E_{first}	542
E_{last}	1026
E_{min}	549
E_{max}	986
e_{min}	542
e_{max}	1026
b	4.2293388
h	542

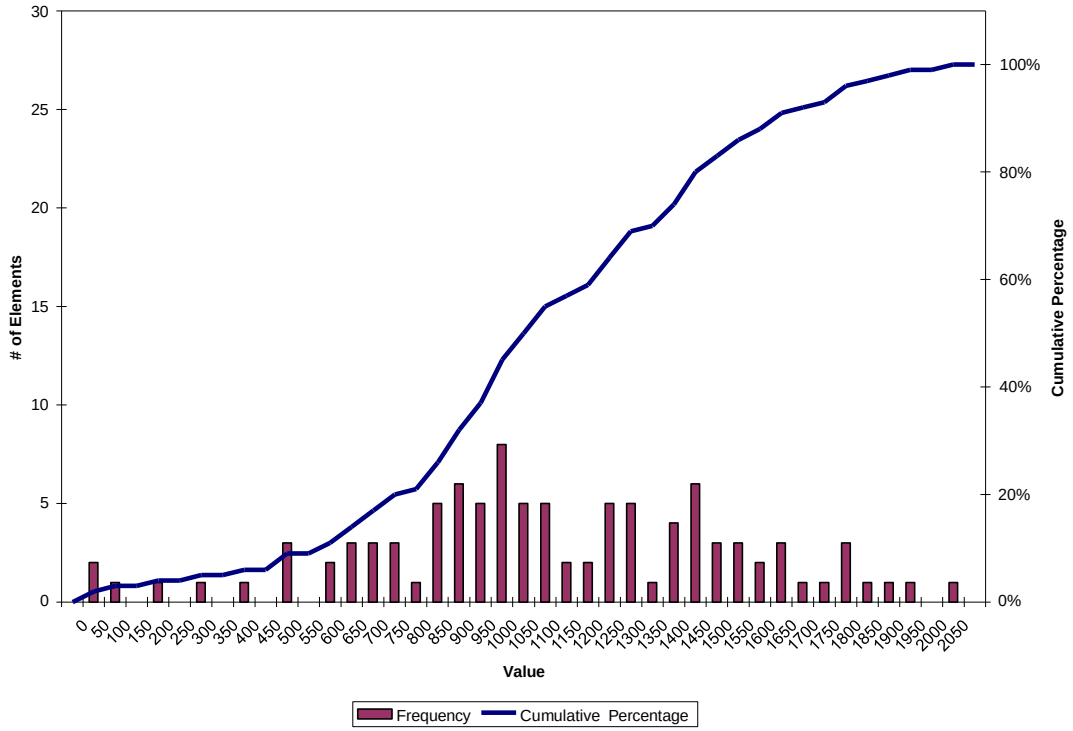


Figure 2.4: Histogram of DRA Processed 10x10, 11-bit Image

2.6.7.1.3 Algorithm Output

Table 2.34: DRA Processed 10x10, 11-bit matrix

643	1611	1578	1070	1459	985	998	2047	1755	0
1265	1446	1239	672	1172	1125	474	1383	1400	592
1303	702	901	1645	491	952	1616	571	1265	871
892	1049	1713	1751	1501	850	884	846	681	1484
935	973	1049	385	969	465	1057	1404	867	601
1518	918	1214	1413	68	1286	1091	846	1066	1759
1552	774	283	1235	744	1210	1260	943	990	660
1286	647	30	1878	1806	981	880	178	1150	1108
1408	1053	1527	825	1916	1024	960	1413	1396	727
1032	1002	1379	842	1446	897	1468	1243	930	1688

2.6.7.2 System Level

Successful DRA implementation is verified empirically. The input histogram is remapped to an output histogram such that the Percent minimum (Pmin, a.k.a. DRA low) and the Percent maximum (Pmax, a.k.a. DRA high) match the defined penetration points for the corresponding sensor/product and Interactive chain. For example, Pmin 0.01 and Pmax 0.99 means the lower 1% and the upper 1% of the input histogram are truncated before re-mapping. Intermediate points are calculated using equations in Section 2.6.5.

2.7 Scaling

Algorithm Name	Scaling
Databases Required	*R*set images (for scale factors < 1.0), Interpolation Coefficients, Intra-*R*set Anti-aliasing kernel (for scale factors < 1.0)
Last Updated	08/25/09
Application Conditions	Scaling is applied per user request in the interactive processing component.
Purpose	When exploiting imagery, it is often useful to zoom in or out of the scene. Scaling is performed to accomplish the desired zoom level.

2.7.1 Overview

Scaling is performed to obtain a desired zoom level. All scale factors are based on a scale factor of 1.0 being the native resolution of the image. Scale factors greater than 1.0 are used to perform magnification while scale factors less than 1.0 are used to perform de-magnification via intra-*R*set scaling. Both processes are described herein.

Intra-*R*set scaling as described in this document utilizes LaGrange interpolation for all examples. Other interpolation algorithms may be utilized as long as required image quality measures are achieved. To maintain consistent image quality when crossing *R*set boundaries, intra-*R*set anti-alias filtering must be used.

2.7.2 Implementation Options

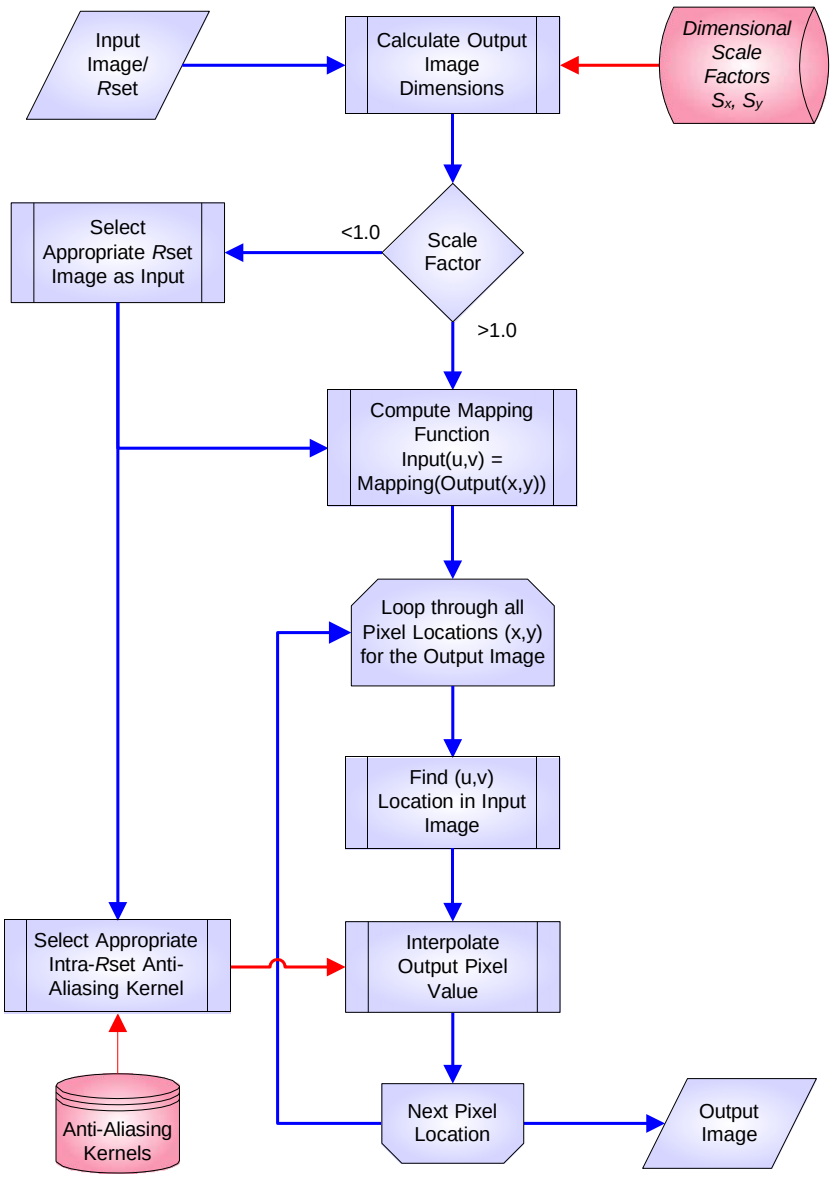
- Interpolation engines may vary. High order interpolation algorithms such as cubic, LaGrange, or modified Sinc are acceptable.

2.7.3 Inputs Required

- Scale factor
- The R_0 image, for scale factors > 1.0
- The RRDS of images necessary to perform the desired scaling, for scale factors < 1.0
- Database of intra-*R*set anti-aliasing convolution kernels, for scale factors < 1.0

- Database of interpolation coefficients

2.7.4 Flow Diagram



2.7.5 Algorithm Details

2.7.5.1 Scale Factors < 1.0

When the desired scaling factor is less than 1.0, indicating de-magnification of the full-resolution image, intra-Rset interpolation provides imagery with scale factors between RRDS levels. The intra-Rset scaling algorithm does not apply outside of the range of imagery scales bounded by the extent of the RRDS for a given source image. For a desired absolute scale factor within this range, the image plane used to seed the scaling operation is the lowest resolution R-level image that still has a higher resolution than the desired scale factor would provide. For example, to produce a scaled image between the R_1 and R_2 R-levels, one would use the R_1 image as the starting point for intra-Rset scaling and then down sample to obtain the correct image footprint.

Once the starting R-level is selected, each output image pixel is calculated using the selected high-order interpolation algorithm. The necessary interpolation is determined based on a mapping function that maps the desired output pixel co- ordinates to the input image. Figure 2.5 provides an illustration of this mapping between input and output image pixel coordinates.

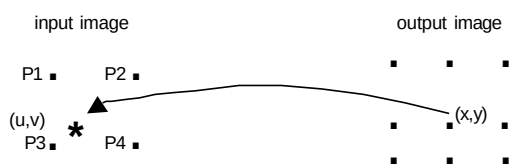


Figure 2.5: Mapping of Scaled Image Pixel into the Original Image

The interpolation process includes anti-alias filtering with intra-Rset filters to provide the appearance of a continuous zoom from one RRDS level to another. The SIPS recommends using 3-element product separable intra-Rset kernels that can be generated based on the desired magnification level between Rsets. The appropriate kernel can be generated as follows.

The first step is to determine the level of scaling between target Rsets that is required for the chosen magnification level. For example, if the target magnification is 0.4x, this requires use of the R_1 image since the target lies between 0.5x (R_1) and 0.25x (R_2). The amount of decimation required for the Rset of interest is:

$$Decimation = \frac{MAG_{RRDS} - MAG_{target}}{MAG_{RRDS}}$$

where MAG_{RRDS} is the absolute magnification level of the Rset to be scaled and MAG_{target} is the desired absolute magnification level.

Next, the anti-aliasing filter for the target decimation is built as follows: In order to obtain an MTF for the scaled image that is visually between the Rset being scaled and the next smaller Rset, the filter should be based on the anti-aliasing kernel used for RRDS generation. As decimation progresses from R_n to R_{n+1} the anti-aliasing filter should more closely approximate the RRDS filter so that when the Rset boundary is crossed, there is a negligible visual effect. A method for doing this in one dimension starts by calculating the horizontal Line-Spread Function (LSF) for the 7x7 RRDS anti-aliasing kernel. This is done by multiplying a row-vector of all ones by the 7x7 anti-aliasing kernel matrix, AA_{7x7}.

$$LSF_{AA} = [1\ 1\ 1\ 1\ 1\ 1\ 1] \cdot AA_{7x7} \tag{2.22}$$

The 7-element LSF is then truncated to a 3-element LSF by taking the center three values of the LSF and rescaling them to sum to 1.0. The result is a 1-D approximation of the RRDS kernel. Then the difference between the 1x3 truncated, scaled anti-aliasing LSF and a 1x3 do-nothing kernel (1 surrounded by 0's) is calculated and multiplied by the amount of decimation required. Finally, a 1x3 do-nothing kernel is added to the result, creating a filter that will prevent aliasing at the desired intra-Rset magnification level.

$$AA_{intraRset} = Decimation \cdot (LSF_{AA,\,1x3} - [0\ 1\ 0]) + [0\ 1\ 0] \tag{2.23}$$

where $LSF_{AA,\,1x3}$ is the 3-element truncated line-spread function.

There are several options for implementing this filtering process that produce acceptable quality results. One is to calculate the specific kernel, as just described, for each magnification level to be displayed. Another method is to define discrete breakpoints for intra-Rset decimation with their associated kernels. A single kernel is used for all decimation levels between defined breakpoints. This approach has historically been implemented with eight breakpoint levels between consecutive Rsets, giving rise to the terminology 'intra-octave filters.' Further efficiency may be achieved with a breakpoint implementation by utilizing product separability of the filters. Since the LaGrange interpolation coefficients are also product separable and specified in one dimension, the 1-D representation of the breakpoint filters can be pre-convolved with the interpolator coefficients, thereby requiring a single correlation operation for the decimation process.

2.7.5.2 Scale Factors > 1.0

Magnification is accomplished when scale factors greater than 1.0 are selected and applied to the R_0 image. This process is similar to the process for scale factors less

than 1.0, with the following differences: All interpolations are performed on the R_0 image. Scale factors between 1.0 and 8.0 are accomplished by interpolation. Scale factors greater than 8.0 may be accomplished through pixel replication rather than interpolation. This implementation provides for optimal image quality up to a scale factor of 8.0, while maintaining exploitation performance for larger scale factors.

2.7.6 Expected Output

The output of this algorithm is an image with dimensions equal to the original image multiplied by the desired scale factor.

2.7.7 Verification Strategies

ELT component level verification strategy provides an example approach and associated algorithms for verifying Scaling. The softcopy display vendor is encouraged to use or tailor this approach to verify component level functionality and processes. System level verification strategy identifies expected (or required) system data outputs and results.

2.7.7.1 ELT Component Level

Scaling is typically implemented in the interactive chain to allow continuous zooming of an image using a slider or wheel on an ELT. However, for simplicity, a static example will be used for verification. In this example, a set of 1-D intra- octave anti-aliasing breakpoint kernels will be derived and applied to a specific de-magnification example.

2.7.7.1.1 Algorithm Inputs

Target magnification level: 0.4x

Table 2.35: 7x7 Anti-Aliasing Kernel

0.00694389	0	-0.02777640	-0.04166500	-0.02777640	0	0.00694389
0	0	0	0	0	0	0
-0.02777640	0	0.11110900	0.16666500	0.11110900	0	-0.02777640
-0.04166500	0	0.16666500	0.25000000	0.16666500	0	-0.04166500
-0.02777640	0	0.11110900	0.16666500	0.11110900	0	-0.02777640
0	0	0	0	0	0	0
0.00694389	0	-0.02777640	-0.04166500	-0.02777640	0	0.00694389

Table 2.36: 11x11 R_1 Input Matrix

0	0	0	0	100	200	300	400	500	600	1024
0	0	100	200	300	400	500	600	700	800	1024
100	200	300	400	500	600	700	800	900	1000	1024
300	400	500	600	700	800	900	1000	1100	1200	1024
500	600	700	800	900	1000	1100	1200	1300	1400	1024
700	800	900	1000	1100	1200	1300	1400	1500	1600	1024
900	1000	1100	1200	1300	1400	1500	1600	1700	1800	1024
1100	1200	1300	1400	1500	1600	1700	1800	1900	2000	1024
1300	1400	1500	1600	1700	1800	1900	2000	2047	2047	1024
1500	1600	1700	1800	1900	2000	2047	2047	2047	2047	1024
1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024

2.7.7.1.2 Computation Details

2.7.7.1.3 Derive 1-D Intra-Octave Kernels

The first example will be to calculate the set of intra-octave anti-aliasing kernels in one dimension. Starting with the 7x7 RRDS anti-aliasing filter shown in Table 2.35, the horizontal line-spread function is calculated according to Equation 2.22. The resulting LSF is shown in Table 2.37.

Table 2.37: RRDS Anti-aliasing Filter LSF

-0.08333	0.0	0.33333	0.5	0.33333	0.0	-0.08333

The LSF is then truncated to three elements and rescaled so the elements sum to 1.0. This is shown in Table 2.38.

Table 2.38: Truncated, Scaled 3-Element LSF

0.28571	0.42857	0.28571

Using Equation 2.23, the set of intra-octave kernels is generated using decimation values of 0, 1/8, 2/8, 3/8, 4/8, 5/8, 6/8, 7/8 resulting in a set of 1x3 kernels shown in Table 2.39. Also shown in the Table are the associated decimation ranges for each of the kernels.

Table 2.39: 1-D Intra-Octave Anti-Aliasing Kernels

	Decimation Range	Kernel Elements		
First Octave	[0, 0.0625)	0	1	0
Second Octave	[0.0625, 0.125)	0.035714	0.928572	0.035714
Third Octave	[0.125, 0.1875)	0.071428	0.857143	0.071428
Fourth Octave	[0.1875, 0.25)	0.107142	0.785715	0.107142
Fifth Octave	[0.25, 0.3125)	0.142857	0.714287	0.142857
Sixth Octave	[0.3125, 0.375)	0.178571	0.642859	0.178571
Seventh Octave	[0.375, 0.4375)	0.214285	0.57143	0.214285
Eighth Octave	[0.4375, 0.5)	0.25	0.5	0.25

2.7.7.1.3.1 Intra-*R*set Scaling Example

The following example shows the calculations required to perform scaling for a magnification level of 0.4x. This magnification requires scaling of the R_1, since 0.4x lies between R_1 (0.5x) and R_2 (0.25x). From Equation 2.21, the decimation level required from R_1 is 0.2, which falls in the range of the fourth octave in Table 2.39, meaning the corresponding kernel will be used for anti-alias filtering in both the horizontal and vertical dimensions. A 2-D kernel is calculated as follows:

$$AA_{octave4} = \begin{bmatrix} 0.107142 \\ 0.785715 \\ 0.107142 \end{bmatrix} \cdot \begin{bmatrix} 0.107142 & 0.785715 & 0.107142 \end{bmatrix}$$

$$= \begin{bmatrix} 0.011479 & 0.084183 & 0.011479 \\ 0.084183 & 0.617348 & 0.084183 \\ 0.011479 & 0.084183 & 0.011479 \end{bmatrix} \tag{2.24}$$

Table 2.36 is an 11x11 pixel matrix with 11-bit values which will be used as the R_1 image to be scaled. The first step in the scaling operation is to do anti-alias filtering by convolving the R_1 with the 2-D intra-octave kernel, resulting in the intermediate image in Table 2.40. The "Mirror Edge - Odd" approach was used for handling edge pixels in the image (see Section 2.4.5.3).

Table 2.40: Intermediate Anti-Alias Filtered R_1 Matrix

0	2	21	51	143	243	343	443	543	673	942
13	30	111	201	300	400	500	600	700	813	976
130	201	300	400	500	600	700	800	900	992	1019
321	400	500	600	700	800	900	1000	1100	1170	1062
521	600	700	800	900	1000	1100	1200	1300	1349	1105
721	800	900	1000	1100	1200	1300	1400	1500	1528	1147
921	1000	1100	1200	1300	1400	1500	1600	1700	1706	1190
1121	1200	1300	1400	1500	1600	1700	1799	1894	1871	1230
1321	1400	1500	1600	1700	1799	1894	1979	2027	1932	1242
1447	1517	1606	1695	1785	1870	1918	1932	1937	1840	1220
1131	1147	1169	1190	1212	1232	1242	1243	1243	1220	1071

The next stage of the scaling process is LaGrange interpolation of the filtered image pixels based on the target decimation level. The mapping function from input pixel grid to output pixel grid will vary by row and column. Since the scale factor is horizontally and vertically symmetric, the same mappings will be applied in both directions. Table 2.41 shows the row and column mappings for the R_1 matrix to achieve the required interpolation.

Table 2.41: LaGrange Pixel Spacings for Rows and Columns of R_1

Input Row/Col	Interpolation Distance	LaGrange Spacing	Output Row/Col
1	0.125	4/32	1
2	0.375	12/32	2
3	0.625	20/32	3
4	0.875	28/32	4
5	Unused	N/A	N/A
6	0.125	4/32	5
7	0.375	12/32	6
8	0.625	20/32	7
9	0.875	28/32	8
10	Unused	N/A	N/A
11	0.125	4/32	9

The table indicates that four different sets of LaGrange coefficients are needed to interpolate the R_1 image for the target scale factor. Table 2.42 is the result of applying the interpolation coefficients (via spatially-variant correlation) to the rows and columns of the filtered R_1 before downsampling the matrix. Values of the output matrix that are outside the 11-bit range of the input matrix are color coded to indicate the need for clipping.

Table 2.42: Filtered R_1 Matrix with LaGrange Interpolators Applied

-2	8	42	138	151	263	388	512	659	680	926
47	114	235	360	372	485	610	734	864	878	985
249	360	487	613	625	737	862	989	1099	1103	1052
500	611	738	863	875	988	1113	1241	1331	1327	1118
525	636	762	888	900	1013	1138	1266	1354	1349	1124
750	861	987	1113	1125	1237	1362	1492	1563	1550	1184
1000	1111	1238	1363	1375	1488	1613	1745	1796	1775	1250
1255	1367	1494	1620	1632	1745	1870	1980	1957	1924	1292
1451	1554	1671	1788	1799	1896	1953	1978	1906	1872	1275
1450	1549	1662	1774	1785	1878	1926	1942	1872	1840	1266
1159	1188	1221	1254	1258	1285	1296	1297	1275	1266	1096

The final step is downsampling the image to obtain the scaled result. Based on the R_1 decimation of 0.2, the final scaled image will be 0.8x of the R_1. For an 11x11 image, the scaled size will be 9x9. As indicated in Table 2.41, every fifth row and column of the input image will be skipped in calculating the output, meaning those rows and columns will be removed in the downsampling process. Additionally, any pixel values outside the 11-bit range of the output data type will be clipped to that range. Thus, after downsampling of the filtered and interpolated R_1 matrix in Table 2.42, the final scaled image matrix is shown in Table 2.43.

2.7.7.1.3.2 Algorithm Outputs

Table 2.43: Final 9x9 Scaled Matrix at 0.4x Absolute Magnification

0	8	42	138	263	388	512	659	926
47	114	235	360	485	610	734	864	985
249	360	487	613	737	862	989	1099	1052
500	611	738	863	988	1113	1241	1331	1118
750	861	987	1113	1237	1362	1492	1563	1184
1000	1111	1238	1363	1488	1613	1745	1796	1250
1255	1367	1494	1620	1745	1870	1980	1957	1292
1451	1554	1671	1788	1896	1953	1978	1906	1275
1159	1188	1221	1254	1285	1296	1297	1275	1096

2.7.7.2 System Level

Successful Scaling allows the user the smoothly zoom in and zoom out image magnification during display, while maintaining the imagery to be free of sampling visual artifacts; such as pixel blocking, induced edge effects, aliasing, and tonal inconsistencies. Scale is verifiable by direct measurement of the image.

2.8 Asymmetric Pixel Correction

Algorithm Name	Asymmetric Pixel Correction
Databases Required	Spatial Interpolation database (LaGrange)
Last Updated	8/31/09
Application Conditions	Asymmetric pixel correction is applied, per user request, in the Geometric Transforms element of the Interactive Processing component.
Purpose	Asymmetric pixel correction is applied to images composed of asymmetric pixels (aspect ratio other than 1 : 1) in order to resample the image plane to a corrected, symmetric pixel space, for purposes of display.

2.8.1 Overview

Asymmetric pixel correction is used to resample pixels having aspect ratios other than 1 : 1 to a symmetric pixel-space representation, for purposes of image display. Asymmetric pixel correction is needed when a user desires to display an asymmetric image on a display device using a regular grid of square pixels, in such a way that the spatial resolution of the captured image is faithfully presented to the observer.

There is nothing inherently wrong with presenting spatially-uncorrected asymmetric imagery directly to an observer. Asymmetric pixel correction is only applied to an image if the user requests that the image be resampled in this manner. Different users may have different preferences for the display of asymmetric imagery. Asymmetric pixel correction is essentially a two-dimensional spatial warping operation. This spatial correction algorithm is applied via a correlation operation described in the Convolution and Correlation ADS in Section 2.4.5.2. The correlation, which is a spatially-variant correlation operation, applies a spatial interpolation kernel (currently based on LaGrange coefficients) to the image plane based on the row and column position in the input image plane. The spatial interpolation kernels to use can be database-selectable or calculable depending on the

system implementation requirements.

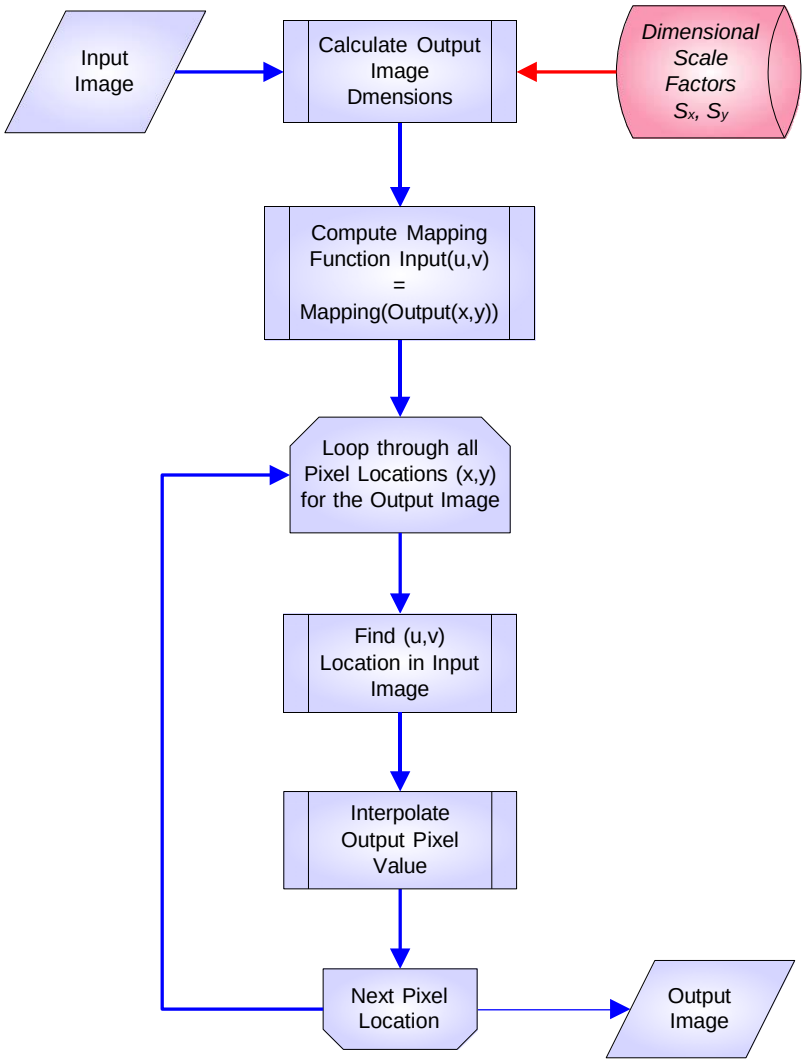
2.8.2 Implementation Options

- N/A

2.8.3 Inputs Required

- An image comprised of asymmetric pixels
- The normalized aspect ratio of the asymmetric image pixels, defined such that the smaller dimension, be it the row or column dimension, is normalized to a value of 1.0
- A calculated family of spatial interpolation kernels (designed for use with the correlation algorithm defined in Section 2.4.5.2), tailored for use with asymmetric pixels having a given aspect ratio

2.8.4 Flow Diagram



2.8.5 Algorithm Details

Asymmetric pixel correction, much like the scaling algorithm discussed in the Scaling ADS (Section 2.7), is simply one case of a generalized two-dimensional warping algorithm. The algorithm maps every pixel position in the spatially corrected, symmetric-pixel output image to a corresponding location in the input asymmetric-pixel image. This mapping depends solely upon the aspect ratio of the input image pixels. Once the mapping is determined for a given output image pixel, it is possible to calculate the proper spatial interpolation kernel coefficients

to apply at the corresponding mapped position in the input image scene. LaGrange coefficients are used in this standard.

In the aggregate, every pixel value in the output image plane can be calculated through the application of a spatially-variant correlation operation where the definition of correlation is that provided in the Convolution and Correlation ADS (Section 2.4.5.2) in this standard. The correlation is spatially-variant in that a different interpolation kernel may be required for each subsequent pixel position being processed.

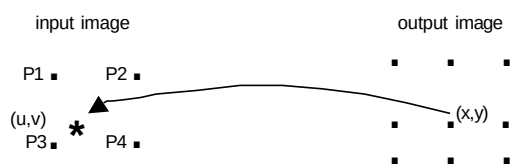


Figure 2.6: Mapping of Corrected Image Pixel into the Original Image

Figure 2.6 illustrates the mapping of a symmetric, spatially corrected, output pixel center-point to its corresponding position in the input image's asymmetric pixel grid. The mapped position in the input image plane is calculated as a floating-point value in terms of row and column position.

In a theoretical sense, every mapped pixel could require a unique set of interpolation kernel coefficients to spatially correct the image at that particular point in the image plane. To avoid the need to calculate a very large number of such interpolation kernels for use in the spatially-variant correlation operation, this algorithm designates a limit to the potential number of kernels required. This limit is based on a spatial sampling (spatial quantization) of the mapped space between the four immediately bounding pixel center-points in the input image. This sampling is based on $1/32$ of the side of a pixel in the input image plane. This sampling fidelity is implied in the Four-Point Interpolation ADS in Section 2.3 via Table 2.7. The manner in which this sampling is performed is illustrated in Figure 2.7. Note that this figure illustrates the spatial quantization concept with a sampling based on $1/4$ of the side of a pixel and not the prescribed $1/32$ sampling.

In Figure 2.7, the black X-marks denote the centers of the four bounding pixels in the input image plane. The blue box in the diagram denotes the boundary of the spatial sampling region; it connects the four center-points of the pixels in the input image and therefore encompasses an extent equal in size to a single pixel in the input image plane. The 25 small square regions, colored red and/or black, denote the spatial quantization regions based on $1/4$ of the size of a pixel. The red X-marks denote the quantized spatial positions for each of the 25 sampling regions. When the center of a pixel in the output image plane is mapped into its

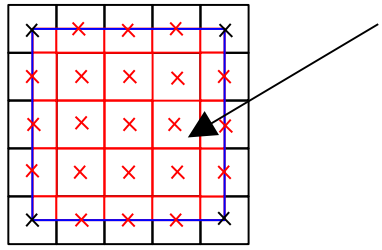


Figure 2.7: Spatial Quantization Example Based on Quarter-pixel Sampling Grid

corresponding position in the input image plane, it will fall at some point between four neighboring input pixels. One possible mapped position is provided by the arrow in Figure 2.7. This position is then quantized to the value at the center of the local sampling region. The quantized position corresponds to a particular interpolation kernel that is then used in the spatially-variant correlation operation to generate the spatially corrected output image.

In the actual asymmetric pixel correction algorithm the sampling space is based on $1/32$ of the side of a pixel. This generates 1089 sampling regions within any given four neighboring pixels of the original image. These 1089 regions are not uniform in size; the central 961 regions have equal extent, the 124 edge regions are half the size of the central regions, and the 4 corner regions are a quarter of the size of the central regions. These regions are analogous to the 25 red regions in Figure 2.7. This provides for an optimal two-dimensional sampling of the space between the four neighboring pixel centers in the input image.

2.8.6 Expected Output

The asymmetric pixel correction algorithm generates an output image consisting of symmetric pixels that displays the same footprint as the original scene. The primary differences between the output image and the input image are that the output image consists of a greater number of rows or columns than the original image, and the presentation of the two images on a standard display device is different. The output image appears to the observer to have the appropriate scale, whereas the input image appears to the observer to be spatially contracted in one dimension.

This page intentionally includes only this statement.

2.9 Color Management Module

Algorithm Name	Color Management Module (CMM)
Databases Required	ICC Profile Database
Last Updated	08/31/09
Application Conditions	The CMM is always applied to multiband imagery. The conversion is determined by the International Color Consortium (ICC) profiles provided for the specific product rendering.
Purpose	There can be many purposes for using a CMM. Common applications are to convert between "sensor" colors and "display" colors.

2.9.1 Overview

A CMM implements multiband color space transforms according to the ICC standard ICC.1:2004-10, which is also available as the International Standards Organization (ISO) standard 5076-1:2005. This standard defines a reference color space, the transformation algorithms for converting between the reference and device color spaces, and the profile format for encoding specific transformations for various devices. The two primary uses for a CMM are to provide a mechanism for transforming multiple bands of sensor data into the reference color space and to provide a mechanism to transform data from the reference color space to the desired output device color space.

2.9.2 Implementation Options

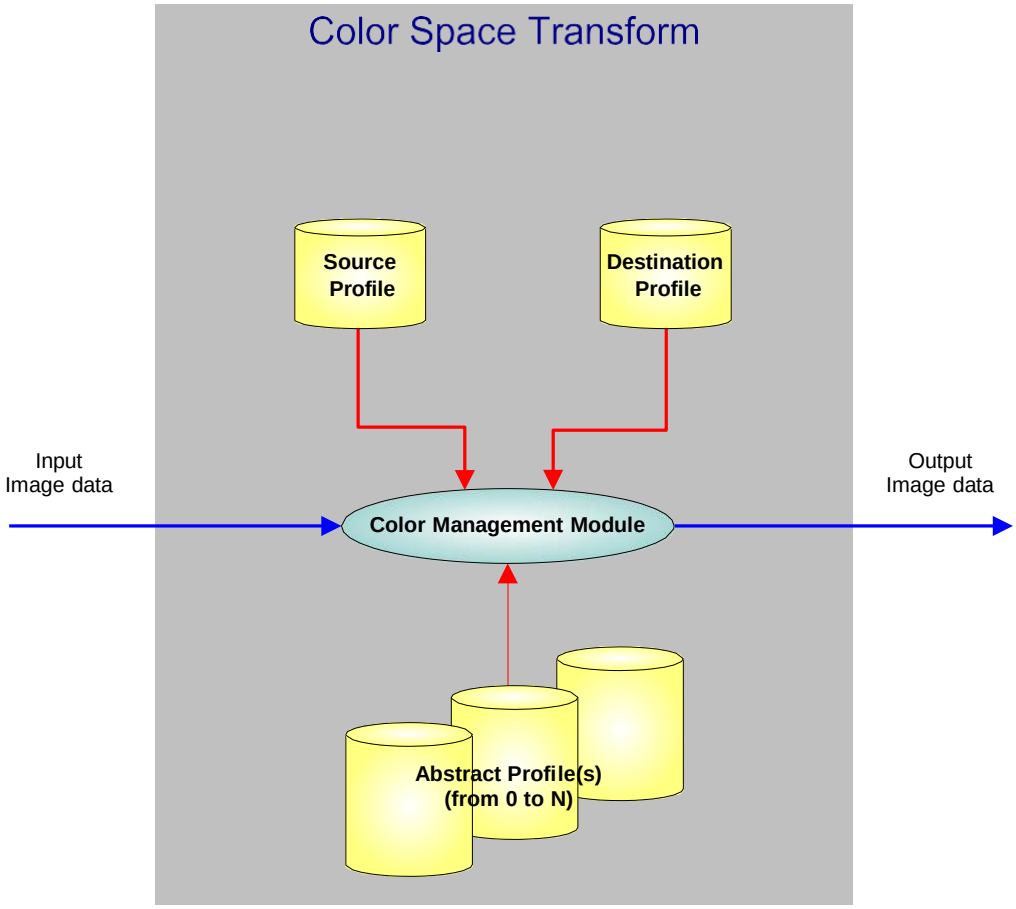
- The CMM may be provided by the operating system, purchased from a third party, or written as part of the ELT application. Any implementation must conform to the ICC standard mentioned in the overview.

2.9.3 Inputs Required

- ICC Profiles (source, destination)
- Rendering Intent (perceptual, saturation, relative colorimetric, absolute colorimetric)
- Image data corresponding to the color space described by the source profile.

– For profiles requiring a subset of bands from a given sensor, a spectral subset is obtained as part of the image chipping operation upon acquisition of the image data.

2.9.4 Flow Diagram



2.9.5 Algorithm Details

Details regarding the functionality of a CMM are contained in the ICC standard referenced in the overview section of this ADS. For the purposes of this standard, an example calculation is provided to illustrate using a CMM to perform a color space transform. The profiles used for this example are provided along with other databases detailed in Appendix A.

The simplest example of using a CMM is specifying a source profile representing one device color space and a destination profile representing a different color space.

For this example, standard Red Green Blue (sRGB) was selected as the source profile and ProPhoto RGB was selected as the destination profile. The sRGB color space was selected as it is a well-defined color space that approximates the characteristics of a Cathode Ray Tube (CRT) display. ProPhoto RGB is another well-defined color space that can represent a larger volume of colors than sRGB. It is not as common as sRGB, but it yields a noticeable difference in the values necessary to represent the same color as sRGB. Table 2.44 contains the results produced by a CMM using these profiles.

Table 2.44: CMM Numerical Example Converting Between sRGB and ProPhoto RGB

sRGB Digital Counts			ProPhoto RGB Digital Counts		
R	**G**	**B**	**R**	**G**	**B**
1.0	0.0	0.0	0.7019	0.2751	0.1031
0	1.0	0	0.5397	0.9274	0.3040
0	0	1.0	0.3356	0.1371	0.9229
0	1.0	1.0	0.6575	0.9440	0.9907
1.0	0	1.0	0.8002	0.3165	0.9328
1.0	1.0	0	0.9192	0.9842	0.3275
0.5	0.5	0.5	0.4241	0.4241	0.4241
0.5	0	0	0.2976	0.1167	0.0437
0	0.5	0	0.2289	0.3933	0.1289
0	0	0.5	0.1423	0.0581	0.3913
0.8353	0.5059	0.1294	0.6227	0.4701	0.1893
0.2706	0.7882	0.8431	0.5303	0.7061	0.7922

Because ICC profiles are either 8-bit or 16-bit encodings representing a range of [0.0, 1.0], the entries in Table 2.44 are provided as floating point values in this range. Some CMM vendors require data to be passed in this manner, while others require the data to be encoded as either 8-bit or 16-bit integer data. For the purposes of this standard, only the floating point representation is provided. Integer equivalents for alternate bit-depths can be calculated by multiplying the floating point value by the maximum value for the desired bit-depth (e.g. 255 for 8-bits, 2047 for 11-bits, 65535 for 16-bits).

At this time, it is possible for a particular CMM to produce results within one or two digital counts of this table. In the future, the ICC may provide profiles and example CMM code for testing purposes. At that time, this table will be replaced by that standard for use as a reference.

2.9.6 Expected Output

The CMM algorithm generates output image data transformed to the color space defined by the output profile. The number of bands in the output image is determined by the destination profile. The image dimensions are not affected by the CMM.

2.9.7 Verification Strategy

Successful Color Management Module implementation is verified empirically and visually. First order color management is to assign the assign each color input band to the correct display channel. For MSI systems with more than 4 bands (e.g. B, G, R, IR, SW1, SW2, MWIR1, MWIR2, LWIR, etc.), refer to the appendix for the corresponding sensor/product and Interactive chain for color band to channel selection. Due to data processing and color transformation, it is unlikely that the output color counts will match the input color counts. For example R255, G0, B0 input may be outputted as different numbers R220, G50, B20. In such an example the output is still visually very red and almost indistinguishable (but slightly less saturated).

2.10 Max-Pixel Decimation

Algorithm Name	Max-Pixel Decimation
Databases Required	N/A
Last Updated	8/25/09
Application Conditions	Applicable to colorized change composite image and Synthetic Aperture Radar (SAR) products. Imagery must be uncompressed and color remapped (if applicable) before application.
Purpose	The algorithm is recognized as the preferred decimation strategy for colorized change composite image products to produce high quality sub- sampled imagery for exploitation at various magnification levels.

2.10.1 Overview

The Max-Pixel decimation strategy is used for color image products to ensure that Red Green Blue (RGB) color changes are visually maintained as the product is reduced in size. If color changes are not maintained, the image quality of the reduced or decimated image will be degraded, and information can be missed during exploitation. For example, the max-pixel decimation algorithm has been identified by imagery analysts as the preferred method to preserve color information in colorized change composite image during RRDS generation. Thus, max-pixel decimation should be used for RRDS generation of these products.

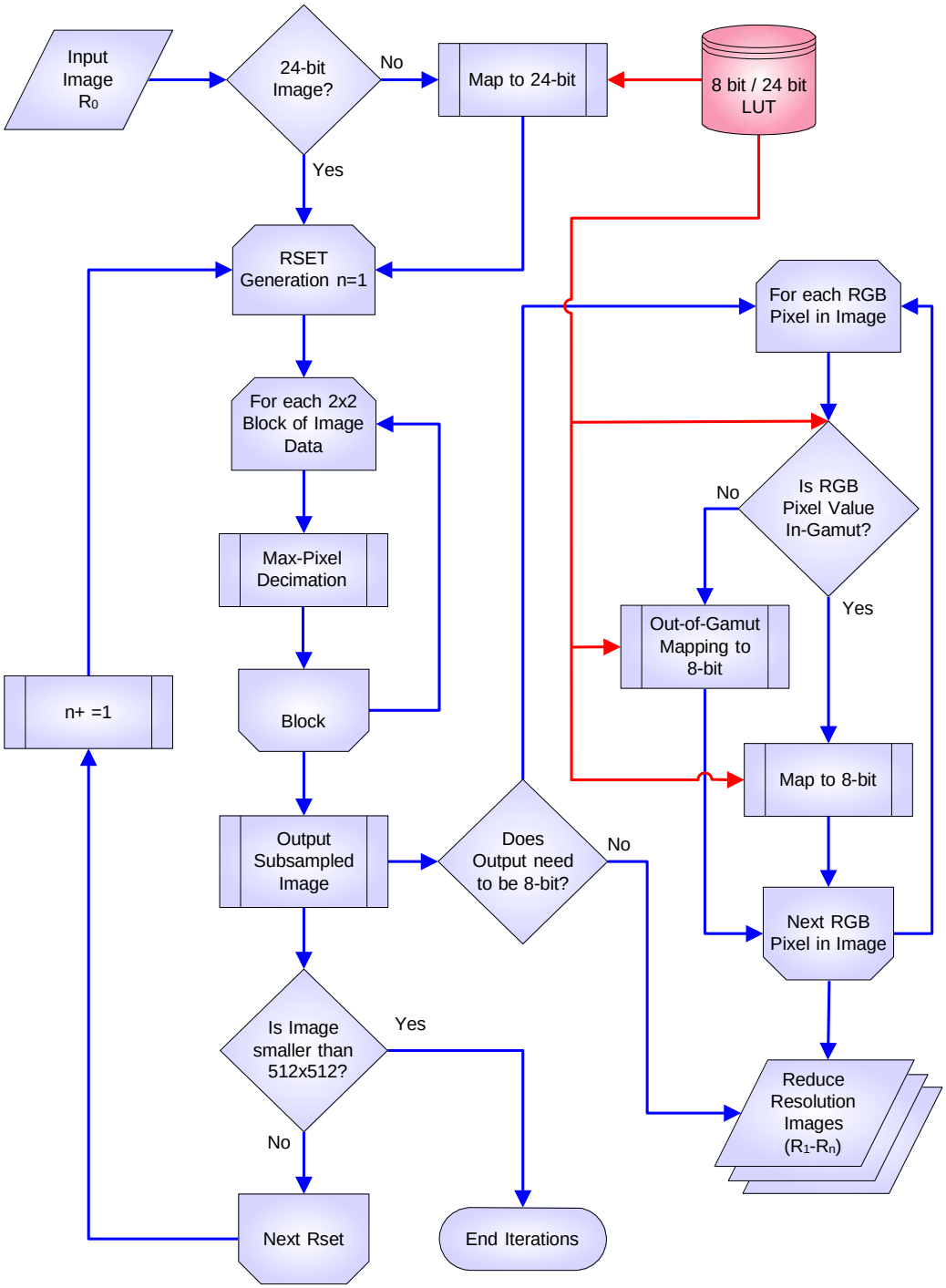
2.10.2 Implementation Options

- Not Applicable (N/A)

2.10.3 Inputs Required

- An 8-bit or 24-bit color input image
- 8-bit imagery also requires the image specific LUTs, obtained from the meta-data, needed to convert 1-byte pixel values to 24-bit color values

2.10.4 Flow Diagram



2.10.5 Algorithm Details

The implementation of the max-pixel decimation strategy differs for 8-bit color products versus 24-bit products due to the nature of the pixel data. Therefore, the algorithm first determines if the imagery is 8-bit or 24-bit. Each pixel of a 24-bit product is a three-byte value where the first byte is the value of the red color channel, the second byte is the value of the green color channel, and the third byte is the value of the blue color channel. Because the pixel values of a 24-bit color product are the RGB levels, decimation is applied directly to the image data.

An 8-bit color image, however, contains at most 256 pixel values ranging from 0 to 255. These pixel values are indices into a color map or LUT which resides in the product metadata. Each index maps to a specific 24-bit value consisting of three 8-bit color channels. To perform max-pixel decimation on 8-bit products the red, green, and blue color values corresponding to each 8-bit pixel value are retrieved from the color map which converts the image to 24-bit values.

In the next step, the algorithm uses a decimation window (a 2-pixel by 2-pixel block of image data) to produce a single output image pixel. The algorithm finds the maximum red, green and blue component from any of the four input image pixels and composites these values into the RGB values of the output pixel. In this manner, a 2X2 pixel block of image data yields a single 24-bit output pixel in the R_i image of the RRDS. Upon completion, the algorithm is implemented on the next adjacent decimation window. A decimation window is a unique Region Of Interest (ROI) and does not have data overlap with preceding ROI. Imagery containing odd numbers of rows or columns should use zero-padding to obtain even-numbered dimensions.

Upon completion of the algorithm, the number of columns and rows of the output image is reduced by a factor of two compared to the input image. Similar to the RRDS generation algorithm described in Section 2.2, the first iteration starts with the R_0 original full-resolution color product and creates the R_1 image which is then used as input during the next iteration to generate the R_2 image and so forth until a full set of images has been created. The process is repeated until the output image is less than 512X512.

If 24-bit output imagery is needed, the algorithm is complete. However if 8-bit output imagery is needed the 24-bit data must be mapped back to 8-bit according to the original color map from the R_0 image metadata. If a 24-bit pixel has a corresponding 8-bit value in the color map it is considered In-Gamut (see Section 2.10.5.1). If there is no corresponding 8-bit value in the color map it is considered Out-of-Gamut and an Out-of-Gamut algorithm must be applied to determine the best 8-bit value (see Section 2.10.5.2).

2.10.5.1 In-Gamut Example

Figure 2.8 shows an example of in-gamut max-pixel decimation on a single 2X2 block of data. 8-bit data is mapped to 24-bit data using the LUT displayed in

Figure 2.11 with a subset shown in Table 2.45. The block is decimated and mapped back to 8-bit using the same LUT.

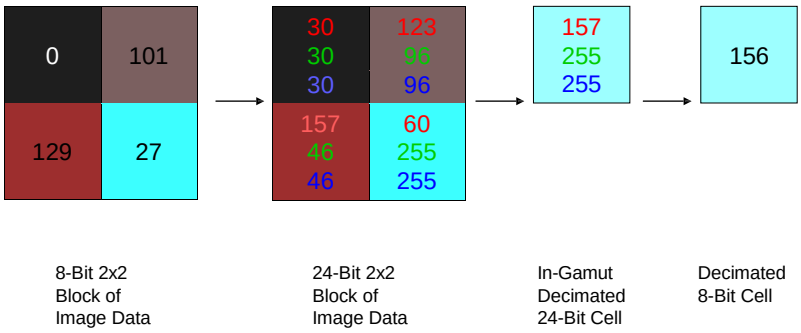


Figure 2.8: In-Gamut Max-Pixel Decimation on a Single Block of Data

2.10.5.2 Out-of-Gamut Example

Figure 2.9 shows an example of out-of-gamut max-pixel decimation on a single 2x2 block of data. 8-bit data is mapped to 24-bit data using the LUT displayed in Figure 2.11 with a subset shown in Table 2.45. The block is decimated to a 24-bit value that is not part of the LUT. To determine the correct 8-bit value the out-of-gamut 24-bit pixel needs to be transformed to the nearest defined 24-bit three color combination before mapping back to 8-bit. This transformation is defined in Section 2.10.5.2.1.

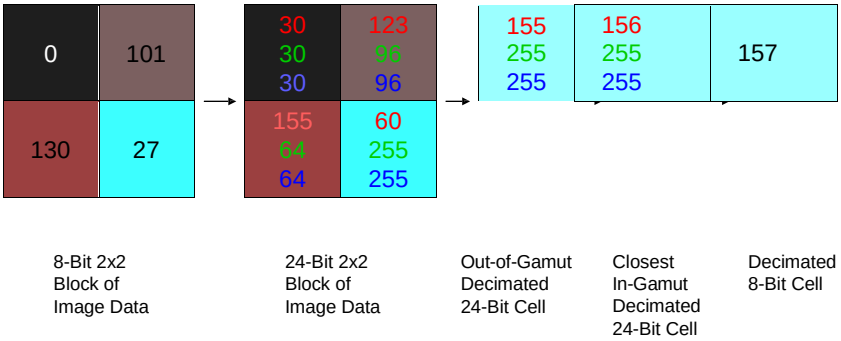


Figure 2.9: Out-of-Gamut Max-Pixel Decimation on a Single Block of Data

2.10.5.2.1 Out-of-Gamut Mapping

There are two criteria for mapping out-of-gamut pixel values:

- The pixel values should be mapped to the nearest equivalent pixel value in the LUT
- The nearest value must maintain the proper hue of the out-of-gamut value

These two criteria maintain color fidelity for the output pixel. For example, if the out-of-gamut pixel is a cyan color, the mapping algorithm should select the nearest LUT value that preserves the relative lightness and cyan hue, not red or gray.

The nearest equivalent pixel value can be determined by choosing the value with the smallest color distance between the out-of-gamut 24-bit value and in-gamut 24-bit value of the LUT. If two or more in-gamut pixel values yield the same distance, the value chosen is the one with the smallest angle between it and the out-of-gamut value.

To determine the color distance between two pixels consider each three color 24-bit pixel as a vector within 3-dimensional color space. The absolute distance d_i between the out-of-gamut pixel P_x and any LUT defined 24-bit pixel value P_i can be calculated as:

$$d_i = \|P_x - P_i\| = \sqrt{(R_x + R_i)^2 + (G_x + G_i)^2 + (B_x + GB_i)^2} \tag{2.25}$$

where R, G and B are the one-byte values of the red color channel, green color channel and blue color channel respectively.

The angle φ_i in degrees between the out-of-gamut pixel P_x and any LUT defined 24-bit pixel value P_i can be calculated as:

$$\emptyset_i = arccos\left(\frac{R_x \cdot R_i + G_x \cdot G_i + B_x \cdot B_i}{\sqrt{{R_x}^2+{G_x}^2+{B_x}^2} \cdot \sqrt{{R_i}^2+{G_i}^2+{B_i}^2}}\right) \tag{2.26}$$

Table 2.45 shows a subset of the LUT depicted in Figure 2.11 needed to solve both the in-gamut example of Section 2.10.5.1 and the out-of-gamut example of Section 2.10.5.2. The last two columns are the distance and angle calculated from the out-of-gamut pixel (155, 255, 255).

Table 2.45: Subset of LUT used with Distance and Angle from (155, 255, 255)

8-Bit Index	Red (R_i)	Green (G_i)	Blue (B_i)	Distance (d_i)	Angle (φ_i)
157	156	255	255	1.000	0.1340°
158	154	255	255	1.000	0.1342°
156	157	255	255	2.000	0.2677°
159	153	255	255	2.000	0.2687°
27	60	255	255	95.00	13.8123°
101	123	96	96	227.1	18.9174°
130	155	64	64	270.1	36.4594°
129	157	46	46	295.6	44.2345°
0	30	30	30	341.9	12.0059°

2.10.6 Expected Output

The max-pixel algorithm produces a reduced resolution version of the input image. The primary difference between the output image and the input image is the reduction in the number of columns and rows of the output image by a factor of two as compared to the input image. Color change is maintained due to the decimation strategy.

2.10.7 Verification Strategies

ELT component level verification strategy provides an example approach and associated algorithms for verifying Max-Pixel Decimation. The softcopy display vendor is encouraged to use or tailor this approach to verify component level functionality and processes. System level verification strategy identifies expected (or required) system data outputs and results.

2.10.7.1 ELT Component Level

The following section is a verification of the process for an 8-bit colorized change composite image example using an 11-pixel by 11-pixel matrix of input image data. The individual cells are shaded to the appropriate RGB color. This example is for illustration only as one would not normally decimate a matrix that was less than 512x512. Also note that the LUT has been fabricated for this example.

2.10.7.1.1 Algorithm Inputs

120	103	86	69	52	35	18	18	0	0	0
137	120	103	86	69	52	35	18	0	0	0
154	137	120	103	176	192	192	35	18	18	0
170	154	137	120	128	176	192	52	35	35	18
170	185	150	98	64	128	176	69	52	52	35
186	251	185	150	98	64	128	10	11	11	52
186	251	251	185	150	98	64	8	10	11	69
186	186	186	186	155	89	23	5	8	10	86
220	220	220	186	189	155	89	23	5	8	103
220	220	220	186	189	189	155	89	23	5	120
220	220	220	186	186	186	170	170	154	154	137

Figure 2.10: Max-Pixel Input 11x11, 8-bit Matrix

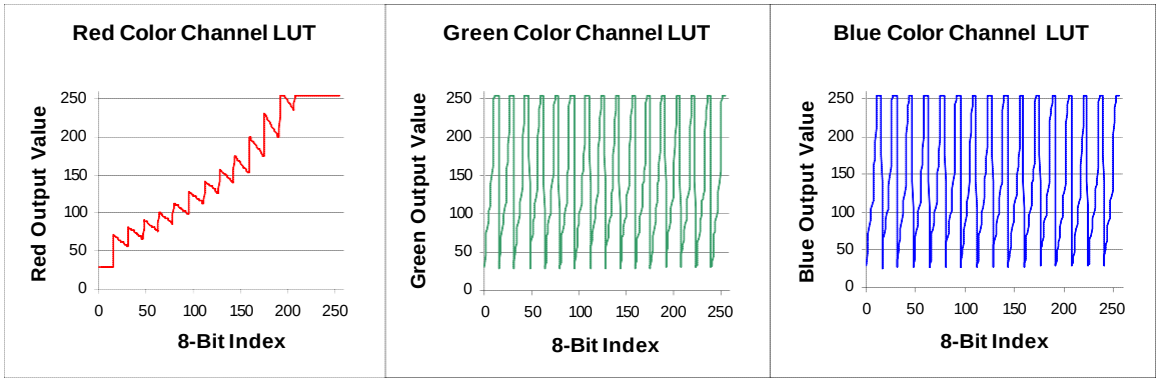


Figure 2.11: Max-Pixel Input LUT

Table 2.46: Subset of Max-Pixel LUT illustrated in Figure 2.11

8-Bit Index	Red (R_i)	Green (G_i)	Blue (B_i)	8-Bit Index	Red (R_i)	Green (G_i)	Blue (B_i)
0	30	30	30	121	133	160	160
5	30	100	100	128	158	30	30
8	30	150	150	130	155	64	64
10	30	240	240	137	148	148	148
11	30	255	255	150	166	103	103
18	69	69	69	154	160	160	160
23	64	124	124	155	159	215	215
24	63	148	148	170	184	184	184
35	78	78	78	171	182	230	230
52	87	87	87	176	230	30	30
64	101	30	30	185	212	152	152
69	96	96	96	186	210	210	210
78	87	255	255	188	206	255	255
86	107	107	107	189	204	255	255
88	105	142	142	192	255	30	30
89	104	167	167	203	243	243	243
91	102	255	255	220	255	255	255
98	126	66	66	221	255	255	255
103	121	121	121	245	255	87	87
120	134	134	134	251	255	231	231

2.10.7.1.2 Computation Details

Figure 2.12 shows the 11-pixel by 11-pixel matrix with 24-bit values mapped from the input matrix found in Figure 2.10 and the LUT depicted in Figure 2.11 (a subset of the LUT is found in Table 2.46). Each cell is shaded to the appropriate RGB color. Individual color channel values are shown with the appropriate red, green, or blue color. This matrix can be decimated using the algorithm defined in Section 2.10.5. The resulting 6-pixel by 6-pixel, 24-bit matrix can then be mapped to 8-bit using the same LUT subset found in Table 2.46 using the in-gamut and out-of-gamut strategies defined in Sections 2.10.5.1 and 2.10.5.2. Note that when there are two or more 8-bit values that map to the same 24-bit triple the smallest

value is chosen.

134 134 134	121 121 121	107 107 107	96 96 96	87 87 87	78 78 78	69 69 69	69 69 69	30 30 30	30 30 30	30 30 30
148 148 148	134 134 134	121 121 121	107 107 107	96 96 96	87 87 87	78 78 78	69 69 69	30 30 30	30 30 30	30 30 30
160 160 160	148 148 148	134 134 134	121 121 121	230 30 30	255 30 30	255 30 30	78 78 78	69 69 69	69 69 69	30 30 30
184 184 184	160 160 160	148 148 148	134 134 134	158 30 30	230 30 30	255 30 30	87 87 87	78 78 78	78 78 78	69 69 69
184 184 184	212 152 152	166 103 103	126 66 66	101 30 30	158 30 30	230 30 30	96 96 96	87 87 87	87 87 87	78 78 78
210 210 210	255 231 231	212 152 152	166 103 103	126 66 66	101 30 30	158 30 30	30 240 240	30 255 255	30 255 255	87 87 87
210 210 210	255 231 231	255 231 231	212 152 152	166 103 103	126 66 66	101 30 30	30 150 150	30 240 240	30 255 255	96 96 96
210 210 210	210 210 210	210 210 210	210 210 210	159 215 215	104 167 167	64 124 124	30 100 100	30 150 150	30 240 240	107 107 107
255 255 255	255 255 255	255 255 255	210 210 210	204 255 255	159 215 215	104 167 167	64 124 124	30 100 100	30 150 150	121 121 121
255 255 255	255 255 255	255 255 255	210 210 210	204 255 255	204 255 255	159 215 215	104 167 167	64 124 124	30 100 100	134 134 134
255 255 255	255 255 255	255 255 255	210 210 210	210 210 210	210 210 210	184 184 184	184 184 184	160 160 160	160 160 160	148 148 148

Figure 2.12: 11x11, 24-bit Matrix for Max-Pixel Example

2.10.7.1.3 Algorithm Output

Below are the 24-bit (Figure 2.13) and 8-bit (Figure 2.14) output image data for this example.

148 148 148	121 121 121	96 96 96	78 78 78	30 30 30	30 30 30
184 184 184	148 148 148	255 30 30	255 87 87	78 78 78	69 69 69
255 231 231	212 152 152	158 66 66	230 240 240	87 255 255	87 87 87
255 231 231	255 231 231	166 215 215	101 150 150	30 255 255	107 107 107
255 255 255	255 255 255	204 255 255	159 215 215	64 150 150	134 134 134
255 255 255	255 255 255	210 210 210	184 184 184	160 160 160	148 148 148

Figure 2.13: Decimated 6x6, 24-bit Max-Pixel Output Matrix

137	103	69	35	0	0
170	137	192	245	35	18
251	185	130	203	78	52
251	251	155	88	11	86
220	220	189	155	24	120
220	220	186	170	154	137

Figure 2.14: Decimated 6x6, 8-bit Max-Pixel Output Matrix

2.10.7.2 System Level

Successful Max-Pixel Decimation implementation is verified empirically and visually. Pixel decimation results in Reduced Resolution Data Set process and pixel aggregation by a factor of 2n. Output color counts are calculated by distance weighting functions identified in Section 2.10.5.2. The process is designed to maintain visual color throughout the decimation process.

This page intentionally includes only this statement.

2.11 Band Dependent Linearization

Algorithm Name	Band Dependent Linearization (BDL)
Databases Required	multiband histogram results
Last Updated	2/10/11
Application Conditions	Apply to a selected bands of multiband image data that has not been previously enhanced.
Purpose	The BDL balances and maps the desired range of each band of image data into a predetermined range of values for further standardized processing.

2.11.1 Overview

The BDL methodology is implemented as part of the Interactive Processing step described in Section 3.4.8.2. It is used to perform tonal processing of the raw data, mapping it from the dynamic range of the imaging device to that of a well-defined metric. While performing this mapping, the BDL algorithm attempts to balance the contribution from each band or channel to generate a 'neutral balanced' image. Once mapped, standardized enhancement processing can be applied. For this standard, the well-defined metric is the entire range of the bit-depth of the image data.

This algorithm is based on the histograms of each of the channels to be used in the rendered image. First, a histogram of each band is computed. The black point is defined as the point at which the image signal rises above noise in each band. For this algorithm, the last incidence of two adjacent bins in the histogram having values of 0 is used as an indicator that the signal is above noise level. Starting at the lowest bin (e.g. digital count value of 0), bins are sequentially searched until the above criteria is met and the next non-zero bin is assigned the black point. Each band will have its own black point value. In the case where there isn't a sequence of two adjacent 0 bins in the histogram, the lowest bin is considered the black point value.

Assumed-white objects in a scene are readily identified by humans as being bright and low in color (low chromaticity). In a similar way, the white DRA stretch limit is found by identifying pixels that are in the top 0.1% of the distribution in any one band, and then finding the set of pixels that were in the top 0.1% in all bands. The last criterion ensures that there is no strong coloration as any pixel meeting this criteria must be near neutral.

Once the black points and white points are determined, the input values in each band are remapped to the desired output minimum and maximum range using a subtractor of the black point value and a multiplier of the ratio of each

band's maximum range to the maximum of the output range as per the following equation:

$$DC\ (X_b) = (X_b - BP_b) \cdot \frac{DC_{max} - DC_{min}}{WP_b - BP_b} \tag{2.27}$$

where X_b is the value of pixel X for band b, BP_b is the black point for that band, WP is the white point, and DC_{max} is the desired maximum DC value of the image and DC_{min} is the minimum DC value of the image. In order to avoid clipping of an image in subsequent processing steps, DC_{max} and DC_{min} may be set inside the allowable range of digital counts.

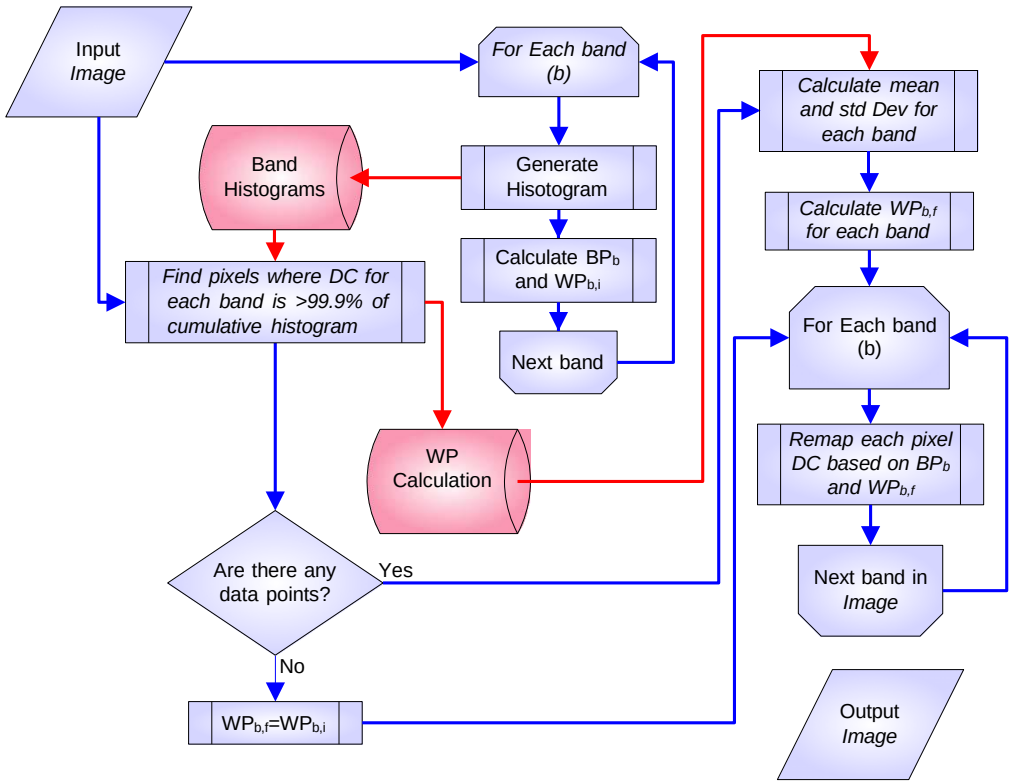
2.11.2 Implementation Options

- In order to speed up processing time, a RRDS can be used to determine the WP and BP values

2.11.3 Inputs Required

- Pixel data for all bands of input image being processed.

2.11.4 Flow Diagram



2.11.5 Algorithm Details

The BDL algorithm expects 'white' objects in the scene where white is simply the equivalent of a perfect Lambertian reflector for all the bands being processed. All pixel data consisting of the highest possible DC value should be excluded from this algorithm, as they are likely the result of clipping and could bias the results of the algorithm. Also, *BP* calculations assume that the darkest pixels in the scene belong to objects that should be perceived as 'black' during exploitation. Some scenes may violate these assumptions and cause the algorithm to fail.

2.11.5.1 Implementation Using All Image Pixels

The following sequence of operations are performed using all pixels of the bands to be processed:

1. Open target image and ingest the desired bands for processing.
2. For each band, determine a histogram. The range of the histogram bins should cover the range allowed for DC values.

3. Once the histogram is determined for each band, a black point (BP) is determined for each band (b). Initially, this value should be the minimum DC for each band. This value should be further refined based on searching the starting bins of the histogram for the last occurrence of two sequential empty bins. If this situation exists, then BP_b should be adjusted to DC value of the adjusted to DC value of the first bin following the two sequential empty bins. This is illustrated in Figure 2.15.

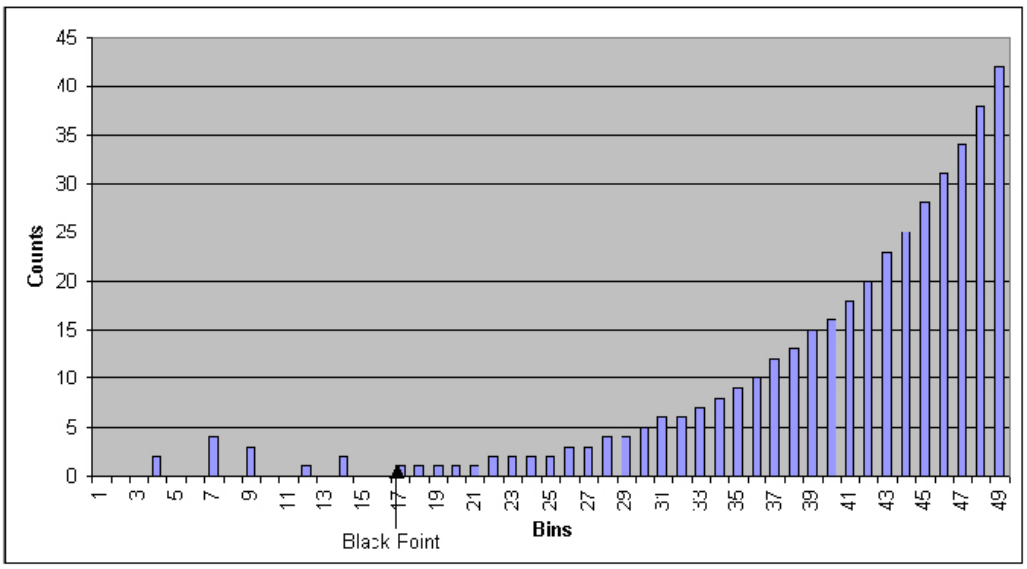


Figure 2.15: Graphical representation of calculating BP_b.

4. An initial white point ($WP_{b,i}$) is then determined for each band based on the DC value representing 99.9% point of a cumulative histogram.

5. All pixels with a DC value greater than or equal to the $WP_{b,i}$ for all bands are identified for use in calculating the final white point for each band$WP_{b,f}$. An example of data points used for calculating $WP_{b,f}$for an RGB image is shown in Figure 2.16.

6. Using the above identified data points, calculate the mean and standard deviation for each band. The mean and standard deviation of the band with the largest mean WP_{bmax} and σ_{bmax}, respectively) are used to calculate the final white point for each band ($WP_{b,f}$). The formula for this calculation is found in equation 2.28

$$WP_{b,f} = WP_{b,ave} \cdot \left(1 + \frac{2 \cdot \sigma_{bmax}}{WP_{bmax}}\right)$$

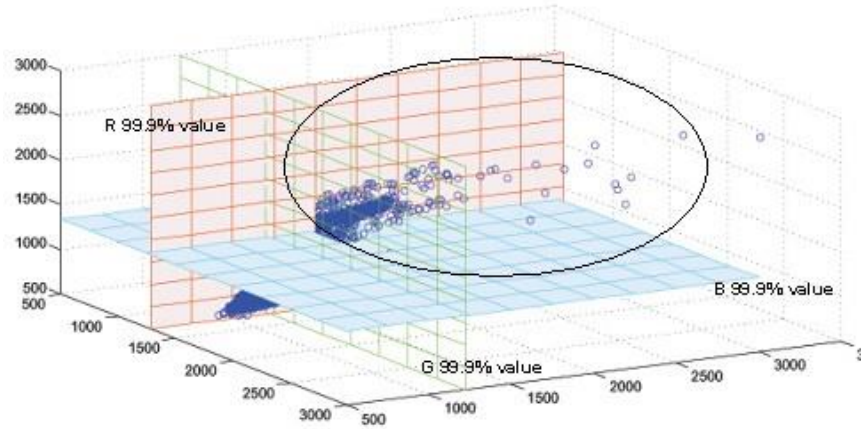


Figure 2.16: Graphical representation of data used for calculating $WP_{b,f}$

Where $WP_{b,ave}$ is the mean white point DC value for band $_{b,}$ $\sigma_{bmax}$is the maximum of the standard deviations of the bands, and WP_{bmax} is the mean white point DC value for that band.

7. Once the final white and black points are determined for each band, the data is remapped according to Equation 2.27. DC_{max} should be set to 75% of the maximum DC value for each band to ensure additional image processing steps do not clip the image. DC_{min} is set to 0.

$$WP_{b,f} = WP_{b,ave} \cdot \left(1 + \frac{2 \cdot \sigma_{bmax}}{WP_{bmax}}\right)$$

2.11.5.2 Implementation Using Reduced Resolution Data Sets for Statistics Calculation

In order to speed up processing time, a RRDS can be used to determine the white point and black point DC values. Experimentation with a small image set covering a range of image types indicates use of RRDS down to the R_3 level gives acceptable results. The algorithm is very similar to that given in Section 2.11.5.1 with the following modifications.

- Steps 1-5 are done as in the previous section on the R_3 version of the image.
- Once $WP_{R_3b,f}$ and $BP_{R_3b,f}$ values for the R_3 image have been determined, the maximum value is determined overall on the full resolution image for the bands to be displayed. The band that contained the maximum value is the reference band and the other two bands are scaled so that the ratios are the same between bands as the white point DC values found for the R_3 image using Equation 2.29.

$$WP_{b,f} = WP_{R_3 b,f} \cdot \frac{DC_{max}}{WP_{R_3 bref}}$$

Where $WP_{b,f}$ is the final white point for the R_0 image band b, DC_{max} is the maximum DC value for the R_0 image, $WP_{R_3 b,f}$ is the final white point DC value determined for band b, and $WP_{R_3 bref}$ is the final white point DC value for the 'reference' band containing DC_{max}

- The average of the R_3 white point values found in Equation 2.29 and those found from the full resolution image is determined and used in Equation 2.27. By averaging the two white point numbers, the impact of glints is reduced. The black point ($BP_{b,f}$) used for the full resolution image is the black point determined from the R_3 image.

2.11.5.3 Implementation Using Only Displayed Image Pixels

The method described above using RRDS for calculating the black and white points, can also be used to implement an auto-DRA algorithm without having to access the full image to get a white balance point. The only difference is that the maximum values used in Equation 2.29 are determined based on the displayed pixel range and not on the full resolution image. Averaging the maximum white point DC values from the entire image with those determined from the displayed data is still recommended to prevent large changes in lightness as different sections of the image are displayed.

2.11.6 Expected Output

The main output of the algorithm is a dynamic range adjusted image that can be written out in any desired image file format. In addition, the individual band histograms, white point values, and black point values can also be saved as output for use by subsequent algorithms.

2.11.7 Verification Strategies

ELT component level verification strategy provides an example approach and associated algorithms for verifying Band Dependent Linearization. The softcopy display vendor is encouraged to use or tailor this approach to verify component level functionality and processes. System level verification strategy identifies expected (or required) system data outputs and results.

2.11.7.1 ELT Component Level

The verification example for BDL does not require any parameters. This example will use a synthetic three band specifically designed to exaggerate and simplify the interim calculations for the BDL algorithm. The three band synthetic image is represented by three input matrices shown in tables 2.47, 2.48, and 2.49.

2.11.7.1.1 Algorithm Inputs

Table 2.47: BDL Input 100x100, 11-bit Matrix for Band 0

	0	1	2	3	4	5 - 94	95	96	97	98	99
0	100	101	101	104	107	1024	1024	1024	1024	1024	1024
1	104	105	105	105	1024	1024	1024	1024	1024	1024	1024
2	106	106	106	107	1024	1024	1024	1024	1024	1024	1024
3 - 96	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024
97	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024
98	1024	1024	1024	1024	1024	1024	1794	1792	1793	1796	1792
99	1024	1024	1024	1024	1024	1024	1792	1795	1790	1795	1800

Table 2.48: BDL Input 100x100, 11-bit Matrix for Band 1

	0	1	2	3	4	5 - 94	95	96	97	98	99
0	100	101	101	103	103	1024	1024	1024	1024	1024	1024
1	106	105	105	105	107	1024	1024	1024	1024	1024	1024
2	106	106	106	107	107	1024	1024	1024	1024	1024	1024
3 - 96	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024
97	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024
98	1024	1024	1024	1024	1024	1024	1778	1786	1782	1784	1790
99	1024	1024	1024	1024	1024	1024	1784	1788	1782	1780	1795

Table 2.49: BDL Input 100x100, 11-bit Matrix for Band 2

	0	1	2	3	4	5 - 94	95	96	97	98	99
0	100	101	101	102	103	1024	1024	1024	1024	1024	1024
1	104	105	105	105	107	1024	1024	1024	1024	1024	1024
2	106	106	106	107	107	1024	1024	1024	1024	1024	1024
3 - 96	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024
97	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1799
98	1024	1024	1024	1024	1024	1024	1024	1792	1793	1796	1792
99	1024	1024	1024	1024	1024	1024	1792	1795	1790	1795	1800

2.11.7.1.2 Computation Details

For the purposes of illustration, bands 0 through 2 shown in the above tables are represented as blue, green, and red colors, respectively, in the following figures. Figure 2.17 depicts a portion of the histograms relevant for calculating the black points for each of the three bands. The smallest pixel value for each band is 100. For the purpose of illustration, the histogram stops at a value of 107 and bins beyond this are assumed to be populated with values. A value of 1024 is used in the input matrices to represent pixel values that are not used for calculating the black or white points.

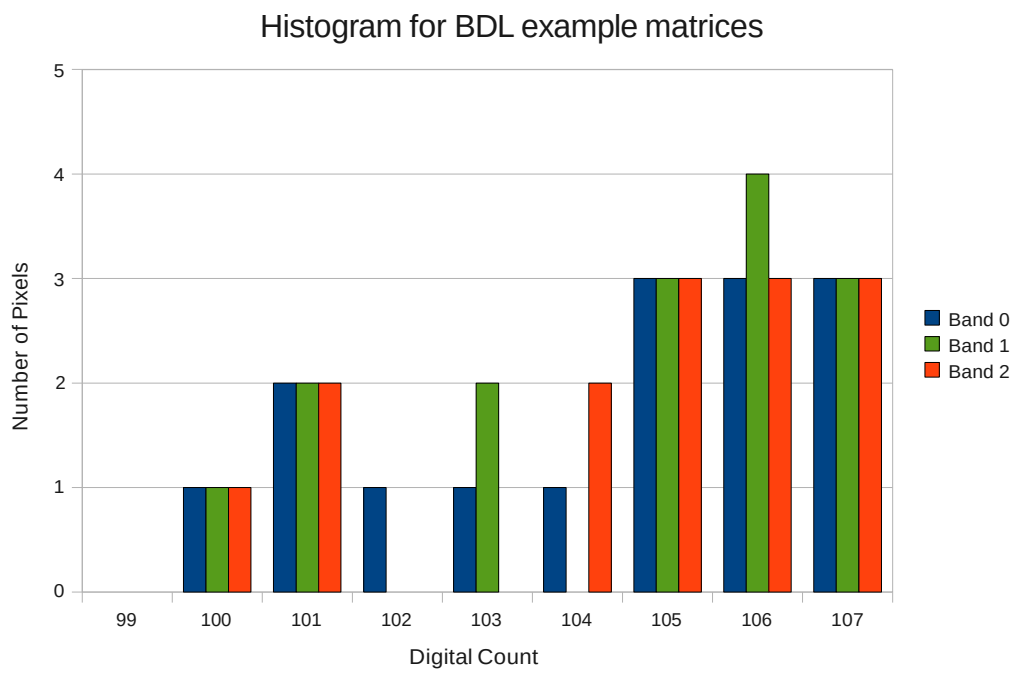


Figure 2.17: Black Point Histograms for Bands 0-2 of the input matrices

Using the values from the histograms shown in Figure 2.17, band 0 has at least one pixel in each bin starting at a DC of 100. Band 1 contains two empty bins for DC values of 102 and 104, but since these are not sequential bins, it's black point value is 100, just like to band 0. Band 2, however, has two sequential empty bins, 102 and 103, so the black point for band 2 is set to the next non-zero bin, 104. Table 2.51 shows the resulting values for BP_b for all three bands.

Pixels representing the top 0.1% of the histogram are color coded blue in the input matrix for each band. While each input matrix contains exactly 10 pixels in the lower right corner as candidates for the white point calculation process, only 9 of the 10 pixels in the top 0.1% of each band are in the top 0.1% of all the bands. Table 2.50 shows the values calculated for use in Equation 2.28 to determine the final white point ($WP_{b,f}$) for each band.

Table 2.50: Initial White Point Calculation Values

Band (b)	$WP_{b,ave}$	σ_b
0	1793.9	2.9768
1	1785.7	4.6904
2	1793.9	2.9768

Band 1, having the largest standard deviation, is used for identifying WP_{bmax} and σ_{bmax}. Applying Equation 2.28 to each of the bands can be seen in the following equations:

$$WP_{0,f} = WP_{0,ave} \cdot (1 + \frac{2 \cdot \sigma_{bmax}}{WP_{bmax}}) = 1793.9 \cdot (1 + \frac{2 \cdot 4.6904}{1785.7}) = 1803$$

$$WP_{1,f} = WP_{1,ave} \cdot (1 + \frac{2 \cdot \sigma_{bmax}}{WP_{bmax}}) = 1785.7 \cdot (1 + \frac{2 \cdot 4.6904}{1785.7}) = 1795 \quad (2.30)$$

$$WP_{2,f} = WP_{2,ave} \cdot (1 + \frac{2 \cdot \sigma_{bmax}}{WP_{bmax}}) = 1793.9 \cdot (1 + \frac{2 \cdot 4.6904}{1785.7}) = 1803$$

Table 2.51 shows the final black and white point values calculated for each band for the example image. These values are then used with Equation 2.27 to remap each pixel of each band. The final output for this example is shown in tables 2.52, 2.53, and 2.54 in the following section.

Table 2.51: Final Black Point and White Point Values

Band (b)	BP_b	WP_b
0	100	1803
1	100	1795
2	104	1803

2.11.7.1.3 Algorithm Output

Table 2.52: BDL Output 100x100, 11-bit Matrix for Band 0

	0	1	2	3	4	5 - 94	95	96	97	98	99
0	0	1	1	4	6	833	833	833	833	833	833
1	4	5	5	5	833	833	833	833	833	833	833
2	5	5	5	6	833	833	833	833	833	833	833
3 - 96	833	833	833	833	833	833	833	833	833	833	833
97	833	833	833	833	833	833	833	833	833	833	833
98	833	833	833	833	833	833	1527	1525	1526	1529	1525
99	833	833	833	833	833	833	1525	1528	1524	1528	1533

Table 2.53: BDL Output 100x100, 11-bit Matrix for Band 1

	0	1	2	3	4	5 - 94	95	96	97	98	99
0	0	1	1	3	3	833	833	833	833	833	833
1	5	5	5	5	6	833	833	833	833	833	833
2	5	5	5	6	6	833	833	833	833	833	833
3 - 96	833	833	833	833	833	833	833	833	833	833	833
97	833	833	833	833	833	833	833	833	833	833	833
98	833	833	833	833	833	833	1520	1527	1523	1525	1531
99	833	833	833	833	833	833	1525	1529	1523	1522	1535

Table 2.54: BDL Output 100x100, 11-bit Matrix for Band 2

	0	1	2	3	4	5 - 94	95	96	97	98	99
0	0	0	0	0	0	833	833	833	833	833	833
1	0	1	1	1	3	833	833	833	833	833	833
2	2	2	2	3	3	833	833	833	833	833	833
3 - 96	833	833	833	833	833	833	833	833	833	833	833
97	833	833	833	833	833	833	833	833	833	833	1532
98	833	833	833	833	833	833	833	1525	1526	1529	1525
99	833	833	833	833	833	833	1525	1528	1524	1528	1533

2.11.7.2 System Level

Successful Band Dependent Linearization implementation is verified empirically.

This page intentionally includes only this statement.

2.12 LUT Generation from Piecewise Mathematical Formulae

Algorithm Name	LUT Generation from Piecewise Mathematical Formulae
Databases Required	Piecewise Mathematical formula LUT, Applicable bit-depth
Last Updated	1/30/12
Application Conditions	Applicable to all piecewise mathematical formula defined LUTs.
Purpose	Generates a LUT of the desired bit-depth for application to an image as a one-dimensional LUT (see section 2.5).

2.12.1 Overview

The piecewise mathematical formula is used to define a functional transformation between input and output values normalized to a domain and range of [0, 1]. This normalized function is sampled and scaled to the appropriate bit-depth to generate a 1-D LUT. The format of a piecewise mathematical formula is given in appendix A. The algorithm contained here provides the mathematical mechanism to generate a 1-D LUT for a desired bit-depth. Please see section 2.5 for information about applying the resulting LUT to image data.

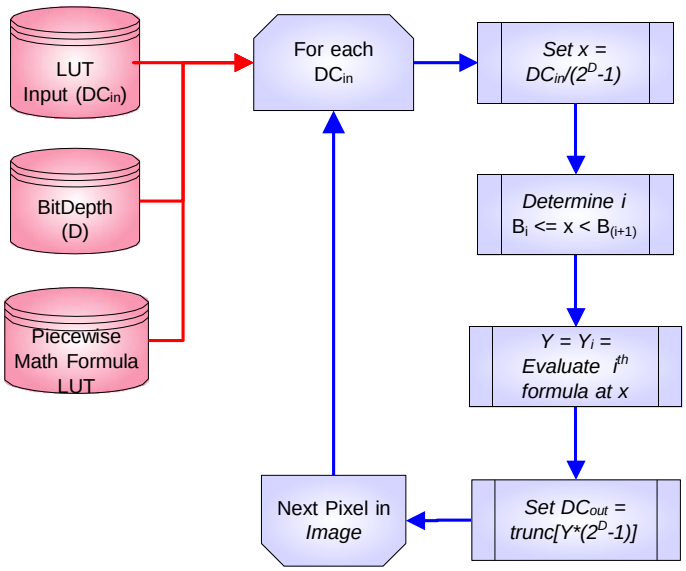
2.12.2 Implementation Options

- N/A

2.12.3 Inputs Required

- Bit-Depth of desired output LUT.
- Piecewise mathematical formula definition - typically from a database of piecewise mathematical formula defined LUTs.

2.12.4 Flow Diagram



2.12.5 Algorithm Details

A piecewise mathematical formula definition provided for this standard describes a mapping from the domain of [0, 1] to the range of [0, 1]. This allows for the generation of the mapping for the appropriate bit-depth necessary for processing the image data. The domain and range may be scaled as needed with a multiplier. Quantization error can then be minimized by sampling the function at the appropriate bit-depth for the application.

Two pieces of information are required to identify a unique piecewise mathematical formula. The 'breaks' are an array of N values given in ascending order within the domain. These values along the domain determine the intervals to which each formula will apply. A piecewise mathematical formula with N breaks will be broken up into $N - 1$ intervals. The second required piece of information

is a series of mathematical formulae. There must be $N - 1$ rows where each row contains a string of the formula to use for that segment of the LUT. The ith row contains the formula as a function of x for the ith segment. Equation 2.31 shows an example of a mathematical formula that could appear in a piecewise mathematical formula LUT.

$$"1.42 * pow(x, 5) - 1.74 * pow(x, 4) + 1.38 * pow(x, 3) - 0.18 * x * x + 0.12 * x" \quad (2.31)$$

Using the breaks array (B) and the specified mathematical formulae for each segment, an output value (Y) is calculated for an input value (x) by first determining the interval (i) containing x. This is done by selecting i to satisfy the criteria given in equation 2.32. Note that if x is only valid if it is within the bounds of the end points of the breaks array $[0, 1]$. Values outside the domain given by the endpoints of the breaks array are not valid for this algorithm.

$$B_i \leq x < B_{i+1} \tag{2.32}$$

The normalized output can be scaled for any bit-depth (D). Input values of x should first be obtained by dividing the original D-bit input value by $2^D - 1$. Similarly, output values of Y must be multiplied by $2^D - 1$ and truncated to obtain the integer mapping of the desired bit-depth.

2.12.6 Expected Output

The output of this algorithm are the values necessary to build a 1-D LUT to apply to image data of the desired bit-depth. The data may be used directly by the application, or it may be output to a file like the common LUT format given in Appendix A.1.2.2.

2.12.7 Verification Strategies

ELT component level verification strategy provides an example approach and associated algorithms for verifying LUT Generation from Piecewise Mathematical Formulae. The softcopy display vendor is encouraged to use or tailor this approach to verify component level functionality and processes. System level verification strategy identifies expected (or required) system data outputs and results.

2.12.7.1 ELT Component Level

The verification of this algorithm is provided by an example generating an 11-bit LUT using a simple piecewise mathematical formula definition given below.

2.12.7.1.1 Algorithm Inputs

- Break points (B): $0, 0.5, 1.0$
- Formula Strings:

  "pow(x,3) - 3 * pow(x,2) + 3 * x"

  "0.25 * (x - 0.5) + 0.875"

2.12.7.1.2 Computation Details

The original input values are integers between 0 and $2^{11} - 1 = 2047$, inclusive. These values are converted to real numbers in the domain of the piecewise mathematical formula, which is [0, 1.0], by dividing each value by 2047. The real numbers are then used as input to the piecewise mathematical formula. The given set of break points and formula strings can be written mathematically as Equation 2.33 to calculate the appropriate output values.

$$\begin{aligned} Y_1(x) &= \quad x^3 - 3 \cdot x^2 + 3 \cdot x, \quad 0 \le x < 0.5 \\ Y_1(x) &= \quad 0.25 \cdot (x - 0.5) + 0.875, \quad 0.5 \le x \le 1 \end{aligned} \tag{2.33}$$

The output values generated by Y_1 and Y_2 are then scaled to the range of the desired output bit-depth by multiplying by 2047 and truncating the result.

2.12.7.1.3 Algorithm Output

The following table provides a sampling of the original 11-bit input values, normalized input values, normalized output values, and output 11-bit integer values.

Table 2.55: Piecewise Mathematical Formula output LUT

DC_{in}	x	Y	DC_{out}	DC_{in}	x	Y	DC_{out}
0	0	0	0	1055	0.5154	0.8788	1799
31	0.0151	0.0447	92	1087	0.5310	0.8827	1807
63	0.0308	0.0895	183	1119	0.5467	0.8867	1815
95	0.0464	0.1329	272	1151	0.5623	0.8906	1823
127	0.0620	0.1748	358	1183	0.5779	0.8945	1831
159	0.0777	0.2154	441	1215	0.5936	0.8984	1839
191	0.0933	0.2546	521	1247	0.6092	0.9023	1847
223	0.1089	0.2925	599	1279	0.6248	0.9062	1855
255	0.1246	0.3291	674	1311	0.6404	0.9101	1863
287	0.1402	0.3644	746	1343	0.6561	0.9140	1871
319	0.1558	0.3984	816	1375	0.6717	0.9179	1879
351	0.1715	0.4312	883	1407	0.6873	0.9218	1887
383	0.1871	0.4628	947	1439	0.7030	0.9257	1895
415	0.2027	0.4932	1010	1471	0.7186	0.9297	1903

(continued on next page)

DC_{in}	x	Y	DC_{out}	DC_{in}	x	Y	DC_{out}
447	0.2184	0.5225	1069	1503	0.7342	0.9336	1911
479	0.2340	0.5505	1127	1535	0.7499	0.9375	1919
511	0.2496	0.5775	1182	1567	0.7655	0.9414	1927
543	0.2653	0.6034	1235	1599	0.7811	0.9453	1935
575	0.2809	0.6281	1286	1631	0.7968	0.9492	1943
607	0.2965	0.6519	1334	1663	0.8124	0.9531	1951
639	0.3122	0.6746	1381	1695	0.8280	0.9570	1959
671	0.3278	0.6963	1425	1727	0.8437	0.9609	1967
703	0.3434	0.7170	1468	1759	0.8593	0.9648	1975
735	0.3591	0.7367	1508	1791	0.8749	0.9687	1983
767	0.3747	0.7555	1547	1823	0.8906	0.9726	1991
799	0.3903	0.7734	1583	1855	0.9062	0.9766	1999
831	0.4060	0.7904	1618	1887	0.9218	0.9805	2007
863	0.4216	0.8065	1651	1919	0.9375	0.9844	2015
895	0.4372	0.8218	1682	1951	0.9531	0.9883	2023
927	0.4529	0.8362	1712	1983	0.9687	0.9922	2031
959	0.4685	0.8498	1740	2015	0.9844	0.9961	2039
991	0.4841	0.8627	1766	2047	1	1	2047
1023	0.4998	0.8748	1791				

2.12.7.2 System Level

Successful LUT Generation from Piecewise Mathematical Formulae implementation is verified empirically. A piecewise mathematical formula is a tonal transfer curve (TTC) designed with defined break point(s) and two end points. Every input digital values maps to a single specific output digital value as defined by the data transfer curve or data transfer table.

This page intentionally includes only this statement.

Reference Implementation

This chapter provides the details of the image chain utilized by NGA to establish a benchmark for image quality. The operators and order of operations in this reference image chain are based on fundamental scientific principles of image processing, engineering evaluations and judgment. All possible algorithm choices were not evaluated for this image chain, due to the magnitude of such an effort. Additionally users and implementers have varied requirements for a specific implementation based on primary exploitation activities, sensor and product types, etc. Thus this standard does not impose a requirement on algorithm choice or order of operations, nor does this standard address performance. Compliance with SIPS is determined based upon the ability of a candidate image chain to meet or exceed an image quality threshold as defined by the SIPS reference implementation and associated panchromatic and multiband metrics (see Section 4).

Each algorithm utilized by this reference image chain is documented in section 2. Alternate algorithms to those used by this standard are known to exist, but have not been evaluated for image quality differences. When possible, alternate algorithms have been listed to imply additional options that may provide better performance, but their impact on image quality has not been determined. The list of options is not comprehensive and should be regarded only as a suggestion. Implementers should use their best judgment in building their image chain to achieve performance and image quality goals. Implementers should contact NIQU if a desired architecture does not meet the image quality tolerances of this standard. Link to `http://t.nga.ic.gov/VVG/?org=tq` for the most current contact information.

3.1 Overview

Figure 3.1 presents a diagram of the reference image chain designed to preserve image quality of multiband imagery for exploitation. This reference image chain builds upon known image processing fundamentals to illustrate an architecture that will provide efficient rendering of single- and multiband imagery with minimal error propagation. Each component of the architecture is detailed in this section.

Database elements contain additional information required by various image processing elements to function properly on the image data.

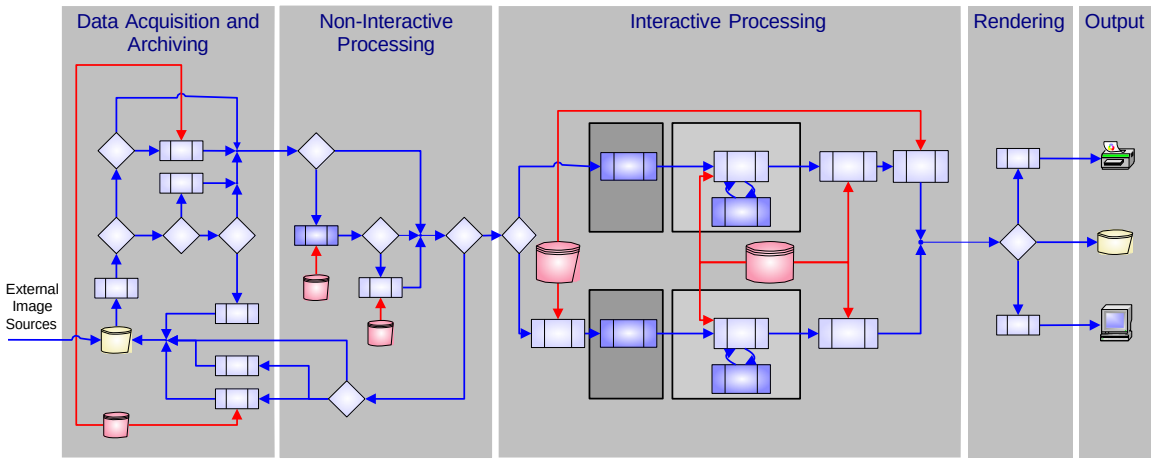


Figure 3.1: High Level Reference Image Chain

Examples of such databases include sharpening kernels and one-dimensional look-up-tables. Data- bases are provided as Government Furnished Information (GFI), and contain the information necessary for complete processing of imagery. These databases are designed to function properly within the reference implementation provided. Different database elements may be required for image chains that are significantly different from the reference implementation provided.

In some circumstances, for example selecting the correct MTFR kernel, the choice of database element is determined by the source of the image data. Other database elements like the TTC should be chosen by the user to optimize the information available for exploitation. The particular elements selected for initial display of imagery must be chosen based on the default settings provided for the given sensor. All default display operations are designed to operate correctly for the reference image chain. Changes to the order of operations or algorithm implementations can lead to image quality degradations. Appendix A provides initial display settings for a variety of commercial panchromatic and multiband imagery sources currently in use within the NSG at the time of publication of this standard.

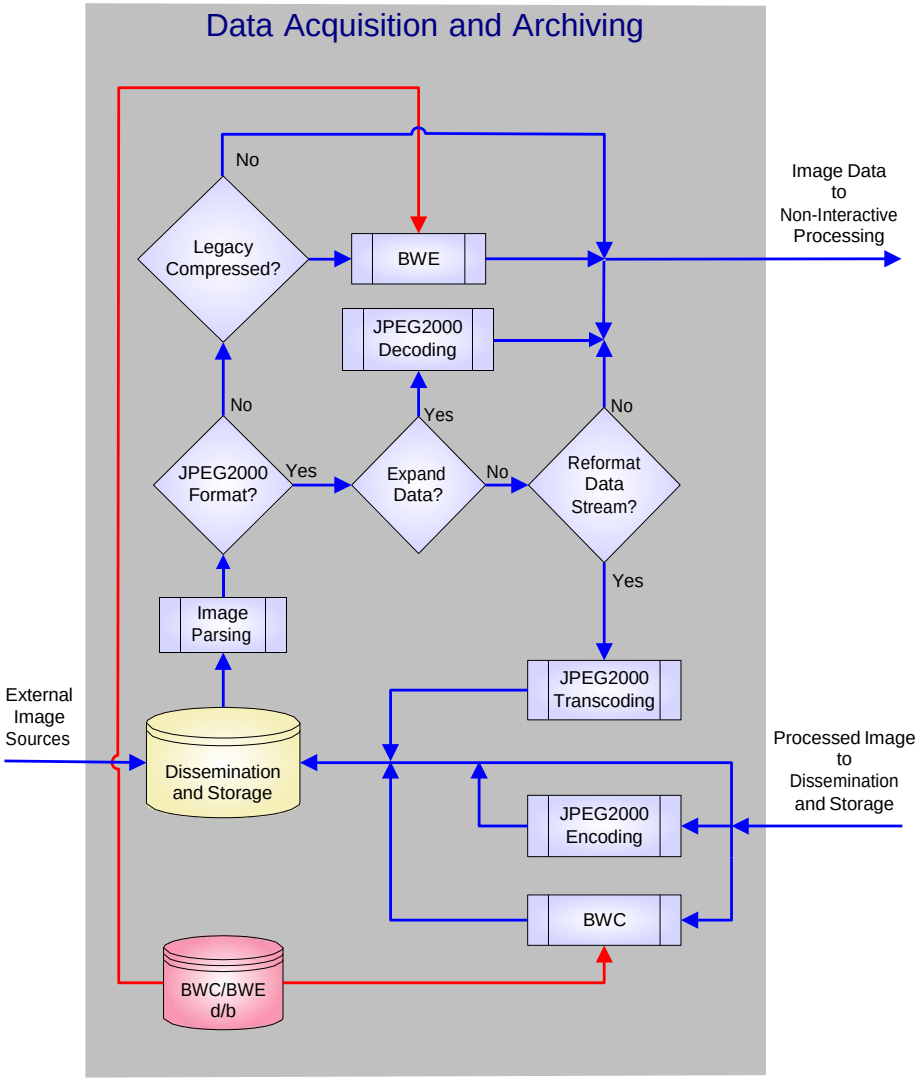


Figure 3.2: Data Acquisition and Archiving Component

The data acquisition and archiving component shown in Figure 3.2 is the portal through which images enter the processing chain. It is defined here to mean the import of an image that previously has not been enhanced for softcopy display (un-enhanced imagery) or the retrieval of a previously enhanced version of an image from a data archive. The component is comprised of eight elements plus decision points: dissemination and storage, J2K workflow (including decoding, transcoding, and encoding), image chipping, Bandwidth Compression (BWC), Bandwidth

Expansion (BWE) and the supporting BWC/BWE database.

The most common form of data being handled by this architecture is un-enhanced data. Whether compressed or un-compressed, un-enhanced data is defined here as being stored in the native space of the imaging device (or a mapped space such as PEDF or Lin-Log). If the data is compressed, it must be expanded using an appropriate BWE algorithm.

The SIPS architecture accommodates images that have already undergone enhancement. Because they are archived in a well-defined storage space, they can be provided directly to the interactive processing component. In most cases, the storage space will be such that the various elements of the interactive processing component will be null operations.

This component also contains an image-chipping element for selecting a subset of tiles from an image. A tile subset is a rectangular region consisting of an offset, width, and height that can encompass anything from a single tile (typically 1024x1024 pixels) to all of the tiles in the image. Performance is dependent upon the BWC algorithm used.

3.2.1 Dissemination and Storage Element

The dissemination and storage element represents the many possible mechanisms for acquiring and archiving image data, ranging from a single image file on a disk to an entire database of image files. Table 3.1 lists the formats that are to be supported by this standard.

Table 3.1: Supported Imagery and Compression Formats

Imagery and Compression Formats	Reference Document
National Imagery Formats	
TFRD Discrete Cosine Transform (DCT) @ 1.29 Bits Per Pixel (bpp)	S2025P Tape Format Requirements Document II
TFRD DCT @ 2.3 bpp	NGA Mapping Systems Interface Control Document (NGA-MICD) Addendum-1 IF300EAA NIMA/CITO to National IMINT Program ICD, vol. I & II
TFRD Differential Pulse Code Modulation (DPCM) @ 4.3 bpp	IF405NI-I DPF to IDS-D ICD

(continued on next page)

Imagery and Compression Formats	Reference Document
NITF 2.1 J2K Visually lossless	MIL-STD-2500C NITF v2.1 for the National Imagery Transmission Format Standard
NITF 2.1 J2K Numerically lossless	MIL-STD-2500C NITF v2.0 for the National Imagery Transmission Format Standard
NITF 2.1 Joint Photo Experts Group (JPEG) 12C3 12-bit lossy	ISO/IEC 15444-1:2004 JPEG 2000 Image Coding System - Part 1: Image Coding System: Core Coding System
NITF 2.1 JPEG 12C3 8-bit lossy	BPJ2K01.10 ISO/IEC BIIF Profile for JPEG 2000 version 01.00
NITF 2.1 uncompressed	S4821P FIA DPE Dataset Requirements Document (FDDRD)
NITF 2.1 formatted for Media as defined in NFDDD, Paragraph 80.1.1	STDI-0004-I NGA FIA Dataset Definition Document (NFDDD)
NITF 2.0 JPEG 12C3 12-bit lossy	S2035A NITF Implementation Requirements Document (NITFIRD)
NITF 2.0 JPEG 12C3 8-bit lossy	IF300EAA NIMA/CITO to National IMINT Program ICD, vol. I & II
NITF 2.0 uncompressed	MIL-STD-188-198A JPEG Image Compression for the National Imagery Transmission Format Standard National Technical Means Product Definitions (NTMPD) National Systems for Geospatial-Intelligence Product Description
NITF 2.0	MIL-STD-2500A NITF v2.0 for the National Imagery Transmission Format Standard
IDS-D Digital Output Product (IDOP)	IF405NI, Paragraph 3.2.1.1

(continued on next page)

Imagery and Compression Formats	Reference Document
Tactical Imagery Formats	
NITF 2.1 J2K Visually lossless	MIL-STD-2500C NITF v2.1 for the National Imagery Transmission Format Standard MIL-STD-2500A NITF v2.0 for the National Imagery Transmission Format Standard
NITF 2.1 J2K Numerically lossless	ISO/IEC 15444-1:2004 JPEG 2000 Image Coding System - Part 1: Image Coding System: Core Coding System
NITF 2.1 J2K lossy (with either wavelet)	BPJ2K01.10 ISO/IEC BIIF Profile for JPEG 2000 version 01.00
NITF 2.1 uncompressed	MIL-STD-188-198A JPEG Image Compression for the National Imagery Transmission Format Standard
NITF 2.0 uncompressed	S2035A NITFIRD IF300EAA NIMA/CITO to National IMINT Program ICD, vol. I & II
Commercial Imagery Formats	
NITF 2.1 J2K Visually lossless	STDI-0006 NITF v2.1 Commercial Dataset Requirements Documents (NCDRD)
NITF 2.1 J2K Numerically lossless	NGA Commercial Data Definition Document Revision-B (NCDDD-B)
NITF 2.1 uncompressed	NGA Commercial Data Definition Document Addendum-1 (NCDDD A-1)
NITF 2.0	NSG-STD-001-06 NGA NITF 2.0 Dataset Definition Document Addendum-1 (NNDDD)

(continued on next page)

Imagery and Compression Formats	Reference Document
Other Imagery Formats	
JPEG 2000 Codestreams	ISO/IEC 15444-1:2004 JPEG 2000 Image Coding System - Part 1: Image Coding System: Core Coding System
Tagged Image File Format (TIFF) Version 6.0	TIFF Revision 6.0
GeoTIFF Revision 1.0	GeoTIFF Format Specification, GeoTIFF Revision 1.0, version 1.8.2
Graphics Interchange Format (GIF)	GIF™, Graphics Interchange Format™, A standard defining a mechanism for the storage and transmission of raster-based graphics information
Commercial JPEG	ISO/IEC IS 10918-1 | ITU-T Recommendation T.81
JPEG File Interchange Format (JFIF)	JPEG File Interchange Format, Version 1.02

3.2.2 JPEG2000 Workflow

Image data stored in the J2K format experiences a different workflow for acquisition and archiving than do older data formats. This is due to the advanced capabilities of the J2K encoding that were unavailable in the past. The three elements of the architecture that handle J2K imagery are detailed in this section.

It is important to note that J2K may refer to either a file format or a data encoding. The remainder of this document uses the term J2K to refer to the data encoding, regardless of the file format being used.

3.2.2.1 JPEG2000 Decoding

The J2K decoding element extracts the desired image data from the J2K-encoded codestream. This operation will produce the desired Resolution level, spatial subset (chip), and quality layer of data suitable for further image processing operations.

3.2.2.2 JPEG2000 Transcoding

The J2K transcoding element is used primarily by image libraries to change the encoding progression of the J2K codestream. The desired encoding will be determined by the requirements of the library implementing transcoding as part of its operation.

3.2.2.3 JPEG2000 Encoding

The J2K encoding element provides the capability to encode image data using the J2K encoding standard. Actual parameters for the encoding format will be specific to the application for which the encoding is to be used.

3.2.3 Image Chipping

The image-chipping element is used to select a spatial subset of the image scene, on tile boundaries. This operation may be performed on compressed or uncompressed data. When chipping compressed image data, image chipping should occur in the compressed-pixel space, when algorithmically possible (e.g. J2K). If the chipping cannot be supported in a particular compressed pixel space, then the file must be expanded before chipping can occur.

3.2.4 Bandwidth Expansion

Bandwidth Expansion (BWE) is the element used for expanding compressed data. The BWC/BWE database contains all of the parameters used by the supported compression algorithms.

3.2.5 Bandwidth Compression

Bandwidth Compression (BWC) is the element that applies the compression algorithm and parameters to the image data. The BWC/BWE database contains parameters used by the compression algorithms. Further information regarding compression algorithms and parameters are available in the appropriate government and commercial specifications.

3.2.6 BWC/BWE Database

The BWC/BWE database element contains the parameters for performing file compression and expansion operations. All algorithms are either part of the specific file format specification referenced in Table 3.1 or are GFI algorithms historically used for National imagery.

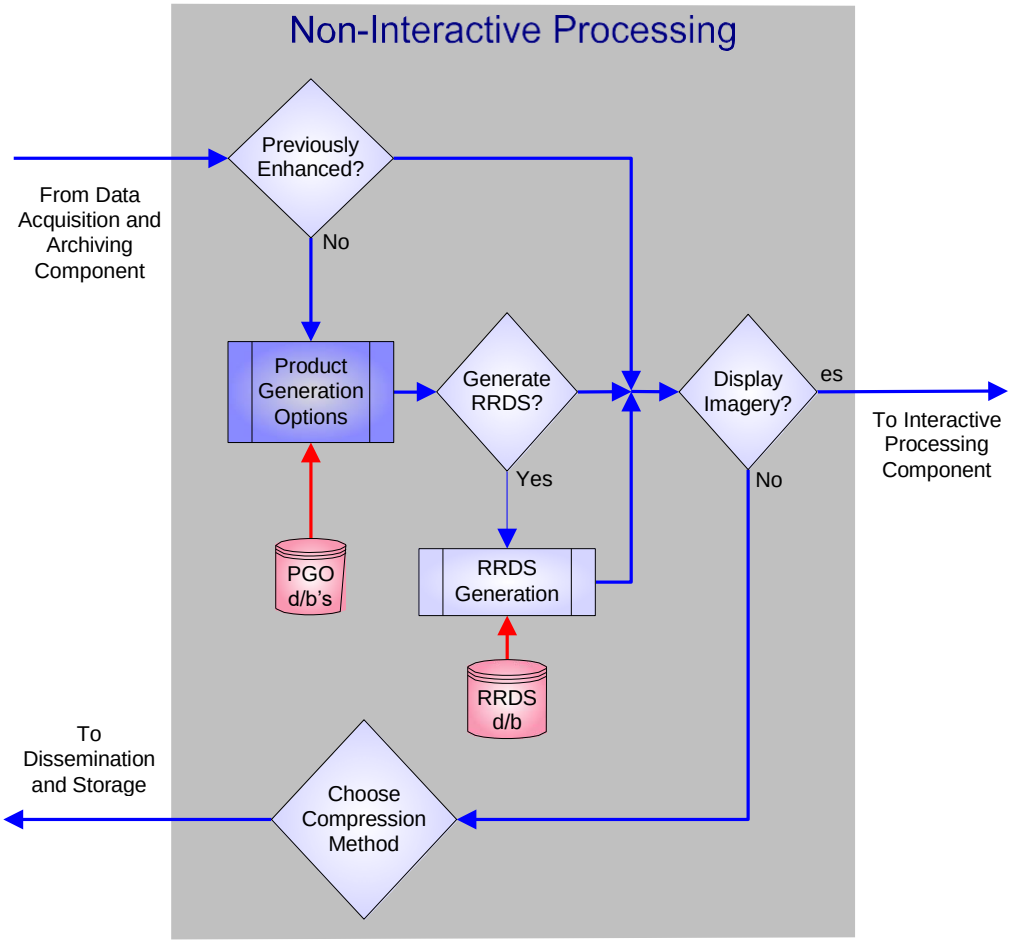


Figure 3.3: Non-Interactive Processing Component

The non-interactive processing component, shown in Figure 3.3 is comprised of four elements: Product Generation Options (PGO), PGO databases, RRDS generation, and RRDS database. Image data at this stage is un-processed, except for expansion and compression addressed in Section 3.2.

3.3.1 Product Generation Options

The PGO element performs several key actions on an image to prepare it for the necessary additional processing to achieve the desired output product. The operations performed by this element include Band Equalization, Modulation Transfer Function Restoration, atmospheric correction, data remapping, and asymmetric pixel correction.-

Only the operations necessary for further processing of the desired product should be enabled. Historical Concept of Operations (CONOPS) within the NSG may dictate which operations are turned on or off based on image acquisition system.

This element of the architecture is color-coded due to its potential for changes in future versions. These changes may include additional processing options and complexity. Many of the actions specified for this element at the time of this writing have passed a proof of concept, but do not exist in a form that is ready for production. The descriptions below offer more insight into the current state of this element.

3.3.1.1 Band Equalization

Multiband products require a band equalization procedure to achieve proper visual rendering in downstream processing. The purpose of band equalization is to ensure that real-world neutral colors have equal digital count values (i.e. are represented as neutral colors) across the dynamic range of the imaged scene.

In its most common incarnation, band equalization takes the form of a simple band-independent gain and offset mapping operation. While a well-balanced sensor may be able to apply the same gain and offset mapping to each color channel (band), it is far more likely that a unique gain and offset mapping would be needed for each individual band. An example of band equalization using gain and offset processing is illustrated in Figure 3.4.

A simple linear mapping, as provided by the application of a gain and offset to a given band, may not be sufficient in performing band equalization with certain multiband products. To accommodate this possibility, the recommended practice for implementing a band equalization process is to apply band- independent, piecewise-linear, look-up tables to the multiband dataset. These band-independent, piecewise-linear, LUTs may be provided by a predefined data- base, generated automatically during run-time processing, or created through user interaction, depending on the requirements of a given data source or processing / dissemination element. The exact procedural implementation is left to the developer. The options mentioned previously are only examples of possible implementations for band equalization processing. Figure 3.5 illustrates a band equalization process utilizing a piecewise linear breakpoint LUT.

The advantage to performing band equalization via the application of band-independent, piecewise-linear, LUTs is that the equalization process can be made as complex or as simple as desired to achieve the desired effect. In this way simple, well-balanced, linear sensors are accommodated along with more complicated, non-linear, less well-balanced sensors.

This standard provides a data format for recording piecewise-linear LUTs. Section A.1.2.2 defines a file format for a break-point LUT that suits the requirements of the conceptual band equalization process outlined above. This LUT data file

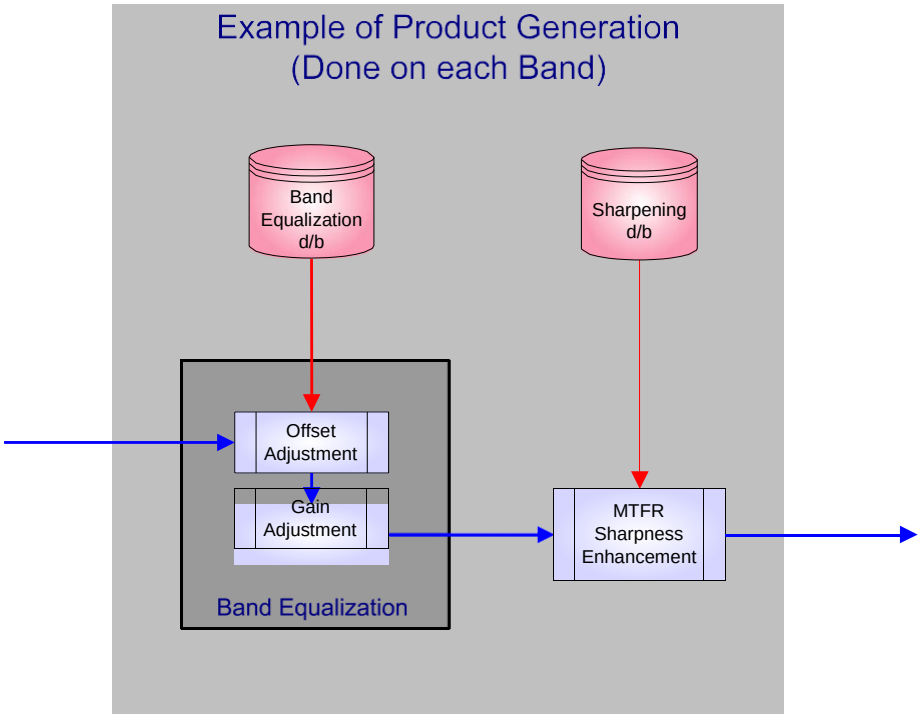


Figure 3.4: Product Generation Example 1

format is suitable for use in pre-staged databases, or to record algorithmically-generated or user-created band equalization LUT files. There is no requirement to use this data format, but developers should be aware that the SIPS document uses this LUT file format in its delivered test files and GFE processing databases; accommodating this data format will make for easier system testing and GFE database updates.

It should be noted that certain atmospheric correction algorithms may render moot the need for a separate band equalization process. This determination is left to the system developers. If specific upstream processing obviates the need for a separate band equalization process, then a null processing operation should be used for band equalization. The null processing operation could take the form of either the application of a "do-nothing" LUT, or the bypassing of the band equalization process entirely, as system requirements dictate.

Many of the multiband imaging systems serving the Intelligence Community (IC) at the time of the writing of this standard do not have well-defined band equalization processing operations established for their particular sensors. For the initial deployment of this softcopy image processing standard, systems handling multiband data from these sources should apply a band-independent null-processing operation for band equalization. As discussed in the previous paragraph,

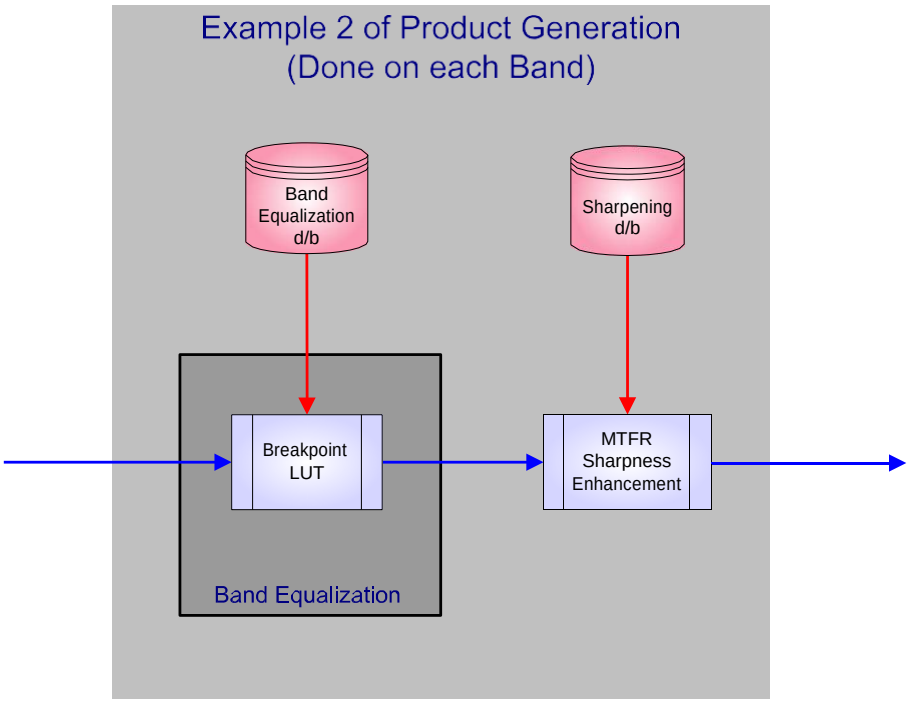


Figure 3.5: Product Generation Example 2

this null-processing band equalization process can take the form of either the application of a "do-nothing" LUT, or the bypassing of the band equalization process entirely. This recommendation is subject to change in future versions of this standard, as data producers and ground processing elements become more aware of the spectral character of the datasets they are handling, and image quality requirements drive the need for better color quality control across the IC dissemination and exploitation architectures.

3.3.1.2 Modulation Transfer Function Restoration

Products being generated from raw data may require spatial processing to compensate for degradations imparted during image collection. This MTF Restoration is typically performed by a convolution of a sharpening kernel with the raw image data. MTFR is de-coupled from the sharpening element further downstream that is used to specifically enhance edge detail in preparation for output.

The convolution algorithm for this element is given in the Convolution and Correlation ADS in Section 2.4. The kernel used for a specific image is determined based upon the characteristics of the imaging device used in the capture of that image. A unique band specific MTFR kernel is applied to each band of the image data at the native resolution of the device, and it must only be applied once. While

a "null" kernel may be applied on subsequent processing steps, software implementation must bypass this step for image data that does not require additional MTF Restoration.

Older systems utilized an MTFC kernel. Because these kernels included both the restoration and the enhancement aspects of sharpening, they should be applied downstream in the interactive processing component. The MTFR kernel that is applied here is intended for future systems utilized by the NSG.

3.3.1.3 Atmospheric Correction

Atmospheric correction may be required for some product generation options. The nature of the atmospheric correction algorithm depends on the attributes of the capture system, and the desired properties of the product to be generated. Currently, any specified atmospheric correction algorithm is covered in an appendix specific to the sensor in question.

3.3.1.4 Data Remapping

Data remapping refers to the specific need to convert the data of incoming, expanded or uncompressed image band data to non-mapped image data. In particular, this applies to the Lin-Log, PEDF encodings, and color mapped images. After expansion of Lin-Log and PEDF encodings, the product still requires the application of a One-Dimensional LUT to negate the tonal mapping applied for storage in these two formats. Expansion of color-mapped imagery requires conversion to a non-mapped, multiband image for further processing. The data remapping operation should always be performed prior to additional processing (e.g. interactive processing, RRDS generation).

Products that are delivered as color mapped images (e.g. Two-Color Multi-View) must be converted into a true multiband image before further processing is performed. The mapping table for performing this operation is always provided in the metadata of the color mapped image. The equations are provided in this document for performing the data remapping operation for PEDF and Lin-Log encodings of single band image products. These are the same equations used to produce the 1-D LUTs distributed with the other database contents listed in Appendix A. The following equations each produce integer values. Non-integer results shall be rounded to the nearest integer value.

3.3.1.4.1 PEDF Encoding

The following equations are used to expand the 8-bit PEDF data to 11-bit or 8-bit density format. Equation 3.1 has an input range of $i \in [0, 255]$ and an output range of an 11-bit integer.

$$11bDF\ (i) = \frac{2047}{382.5} \cdot \{ \begin{matrix} i & i \leq 127.5 \\ 2 \cdot (i - 127.5) + 127.5 & i > 127.5 \end{matrix} \tag{3.1}$$

Equation 3.2 has an input range of $i \in [0, 255]$ and an output range of an 8-bit integer.

$$8bDF(i) = min(max(i, 2 \cdot (i - 127.5) + 127.5), 255) \tag{3.2}$$

3.3.1.4.2 LinLog Encoding

The following equations are used to expand the 8-bit Lin-Log data to 11-bit or 8-bit log format data. Equation 3.3 has an input range of $i \in [0, 255]$ and an output range of an 11-bit integer.

$$11bLF(i) = \frac{2047}{15} \begin{cases} 0 & i = 0 \\ log_2(i) & 0 < i \leq 117 \\ log_2(2^{1/17} - 1) & i = 117 \end{cases} \tag{3.3}$$

Equation 3.4 has an input range of $i \in [0, 255]$ and an output range of an 8-bit integer.

$$8bLF(i) = \frac{255}{15} \begin{cases} 0 & i = 0 \\ log_2(i) & 0 < i \leq 117 \\ log_2(2^{1/17} - 1) & i = 117 \end{cases} \tag{3.3}$$

3.3.1.5 Asymmetric Pixel Correction

Due to the design characteristics of some data sources, pixel data from these sources may represent detectors that do not have a square aspect ratio (1 : 1, where the first element represents scale in the column direction and the second element represents scale in the row direction, $c : r$). In these cases the native resolution image plane, referred to as level R_0, consists of an array of rectangular, as opposed to square, pixels. Since display devices render images using only square pixels, an asymmetric pixel correction process is utilized to resample asymmetric image data to a symmetric pixel grid. The method for accomplishing this correction takes the form of an image warping operation, where the warping equation is dictated by the specific aspect ratio of the source data. If there are multiple bands, the same operation is applied to each band independently. In either case,

this operation corrects for asymmetry due to the physical sensor design, and not for differences in cross-scan or along-scan Ground Sampling Distance (GSD) based on collection geometry.

Often times in source data literature, aspect ratios are provided in integer form, for example $3:2$, which would indicate in this case that the extent of a pixel's width (distance along the column direction) is 1.5 times the extent of the pixel's height (distance along the row direction). For purposes of this standard, aspect ratios will always be normalized to the extent of the smaller pixel dimension. In the example above, an aspect ratio of $3:2$ would be referred to as $1.5:1$. This is an important definition from a conceptual standpoint, because the asymmetric correction process does not try to produce six pixels (three columnar pixels x two row-wise pixels) from every one pixel of a source image having an aspect ratio of $3:2$. Rather, the asymmetric correction process for this case would generate three corrected pixels in the column direction for every two original pixels in the column direction.

Once the asymmetric resampling is accomplished, the corrected R_0 image plane is presented to the observer in a spatial rendering that is visually similar to the acquired target scene. Care must be taken to ensure that any metadata provided with the image pixel data is used appropriately given the resampled image plane used for presentation purposes.

3.3.2 Product Generation Options (PGO) Databases

PGO databases currently consist of a database of kernels for modulation transfer function restoration and a database of kernel coefficients for performing asymmetric pixel correction. Future versions of this document may contain additional databases, pending the inclusion of other algorithms in this element.

3.3.2.1 Modulation Transfer Function Restoration (MTFR) Database

This database contains the sharpening kernels to be used by the MTFR action of the Product Generation Options element. The MTFR kernels are specific to the image device and are sensor-specific. MTFR kernels are designed to correct only for the characteristics of the imaging device and not to provide additional enhancement.

It is common to provide single "multi-purpose" kernels that produce edge detail enhancement for display in addition to MTF restoration. Refer to section 3.4.6 for the application of these "multi-purpose" kernels.

3.3.2.2 Asymmetric Pixel Correction Database

The interpolation coefficients necessary to perform asymmetric pixel correction are stored in this database.

3.3.3 Reduced Resolution Dataset (RRDS) Generation

The RRDS generation element is used to create a set of sub-sampled versions of an image, the purpose of which is to provide the processing chain with quick access to imagery at lower magnification values, resulting in faster computation speeds and better overall workstation and algorithm performance.

3.3.4 Reduced Resolution Dataset (RRDS) Database

The reduced resolution dataset database element contains algorithm parameters

required for RRDS generation. The database consists of Anti-Aliasing Filters (currently implemented as a single 7x7 spatial convolution kernel) and Spatial Interpolation kernels (currently implemented as a set of 4x4 spatial correlation kernels) that are applied to the original image plane before downsampling to form the reduced resolution image plane. The algorithmic choices that dictate how the database should be utilized during RRDS generation are explained in detail in Section 2.2.

3.4 Interactive Processing

The interactive processing component is comprised of seven basic elements: geometric transformation, band operations, color space transform, Alternate Color Space (ACS) operations, output preparation, an interpolator database, and an ICC profile database. A graphical representation of these components and the order in which they are applied is found in Figure 3.6. It should be noted that the ACS operations may not be used, as in the case of single band imagery. For multiband imagery, however, ACS operations may take place in more than one color space, hence the possibility of going through the color space transform one or more times.

3.4.1 Processing Order

As image data enters the interactive processing component, the first operation is determined by the desired scale factor to apply to the image data. For scale factors greater than or equal to one, all image processing operators are applied to the image data prior to magnifying the data for display. For scale factors less than one, the geometric transformation is applied to the data before the remainder of image processing is applied to the image.

3.4.2 Geometric Transform

The geometric transform element is used to perform various geometric distortions to each band of image data. These distortions include image chipping, scaling, rotation, shearing, etc. Other interactive transformations may pertain to this inter-

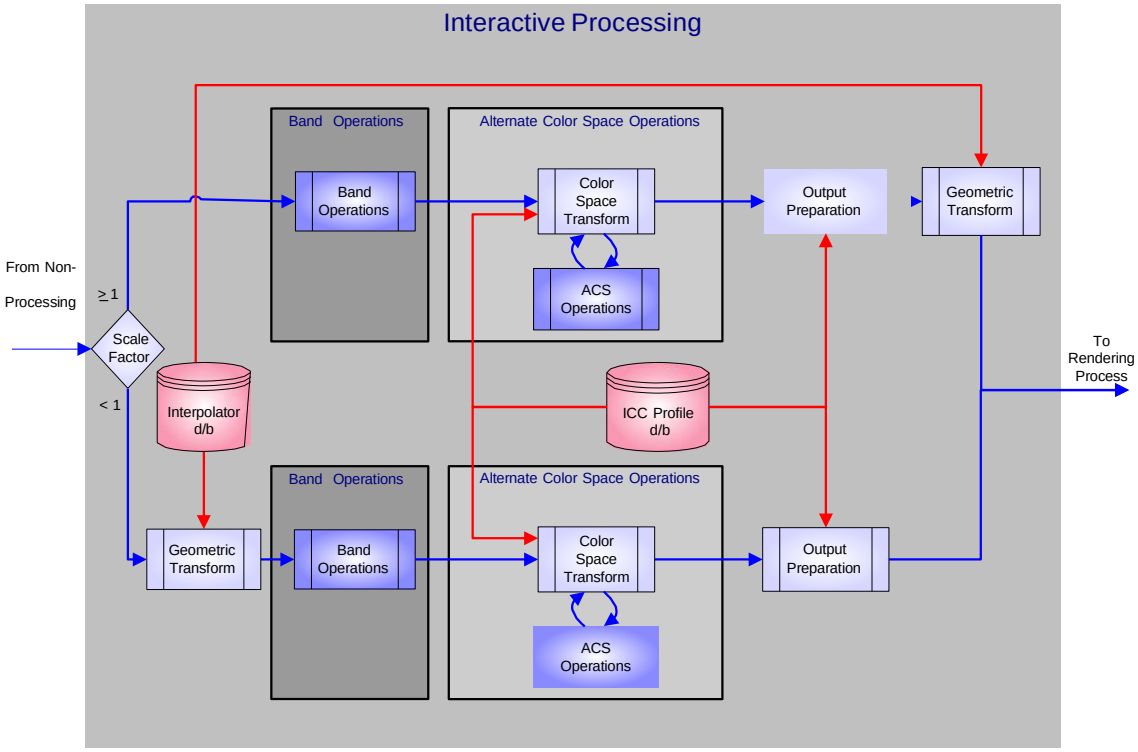


Figure 3.6: Interactive Processing Component

active element as well. Scaling, however, is the primary transformation performed by this operator.

If an image is to be displayed at a scale factor greater than or equal to 1.0, sharpness enhancement must be applied first, followed by the scaling interpolation performed in the geometric transformations element. This will prevent the accentuation of interpolation-induced artifacts. If an image is to be displayed at a scale factor less than 1.0, the scaling interpolation and down-sample must be applied first, followed by sharpness enhancement.

Image data should also be chipped to a desired spatial subset at this point. The spatial subset should consist of a specific pixel offset, width, and height. Unlike the image chipping performed in the Data Acquisition and Archiving component, this image chipping is creating an enhanced product consisting of a ROI. Therefore, image chipping should be performed on the ROI applicable for the current rendering chain, where the ROI does not have to fall on full tile boundaries.

3.4.3 Interpolator Database

The interpolator database element contains interpolation kernels and coefficients that may be applied to image data.

3.4.3.1 ICC Profile Database

The ICC profile database may be managed by the ELT application or the operating system. This collection of ICC profiles are to be used by the ELT application during the exploitation process. Profiles in this database may be of any profile type defined by the ICC standard and should be used in accordance with the ICC standard.

At this time, the sRGB profile provided as part of the SIPS databases should be used for the display profile needed for output preparation. Until otherwise specified, the display profile should be used for rendering unknown RGB imagery. Because sRGB is a color space similar to a standard RGB display, it should provide reasonable results for RGB imagery of unknown properties. Future releases of the databases for the SIPS document will contain profiles for common color spaces that may be needed for processing various imagery products.

3.4.3.2 Embedded Profiles

Most image formats allow for the inclusion of ICC profiles as part of the metadata for the image. These profiles define the color space transformation between the device space and the Profile Connection Space (PCS) used by the CMM.

At the time of the writing of this document, the NITF standard does not have a mechanism for embedding ICC profiles in the metadata. Upon adoption of such a mechanism, an updated version of this standard will be produced to provide detailed information for utilizing this data.

3.4.4 Band Operations

Band operations consist of one or more image processing operations that are performed on one or more of the bands of the imagery. Operations can include spatial transformations such as convolution or correlation, tonal operations such as applying a one-dimensional LUT, or band math operations such as a matrix. This standard provides for multiple operations to be performed at this point in the image chain. Thought must be given to the order of operations to minimize image quality degradation.

3.4.5 Color Space Transform

Upon completion of all band operations, additional operations may be performed in alternate color spaces. The Color Space Transform (CST) is used to transform

the data from one color space to another. While a CST can be invoked multiple times to process an image, it is typically invoked only once or twice. This is done primarily to maintain reasonable performance in the application, but also to prevent unnecessary interpolation or quantization errors in the final product.

A CMM should be used to perform the CST. A CMM operator is described in section 2.9. The source and destination profiles provided to the CMM define the transformation from the current color space to a new color space. The first time the CMM is invoked, the source profile should define the mathematical transform between the bands representing the device color space of the sensor and the PCS, which is described in the ICC standard ICC.1:2004-10 or ISO standard 15076-1:2005. The destination profile should describe the mathematical transform between the PCS and the desired output color space. Both profiles should be readily available in the ICC profile database[1].

At the time of the writing of this document sample ICC profiles are provided for some of the more common sensors listed in the appendices of this document. These profiles are provided along with the processing defaults for the reference image chain to provide an example of a color managed workflow. Future updates to this document should include ICC profiles for additional multiband sensors along with default parameters for using them in a color managed workflow.

3.4.6 Alternate Color Space Operations

Once the data is converted to an alternate color space, it may be processed with additional operators. These operators are the same as are used for band operations (see Section 3.4.4).

Most ingested data is in a device specific color space referred to as 'sensor' space. This means that each band represents information specific to the design of that sensor. Using Ikonos as an example, a true color product is created using the blue, green, and red bands. A true color product can also be created using the blue, green, and red bands of WorldView-2. The characteristics of the bands on these two systems, however, are different. While the names of the bands give insight into the type of data that they contain, they do not provide a precise definition. By utilizing the CST, an ICC profile is used to encapsulate a precise definition for the manner in which to represent the 'sensor' data to the human visual system.

Using the CST, it is possible to define the transformation from 'device' data to other well-defined color spaces. The purpose of the ACS operations is to manipulate the appearance of the data in a well-defined device color space or a color space that is based on human perception, since most data is ultimately exploited by a human observer. Operations that are dependent on the device characteristics should be performed prior to any color space transformations or ACS operations.

Figure 3.7 provides a n example of ACS operations being performed on image

[1]Read section 3.4.3.2 for additional details and exceptions to this rule.

data converted to a luminance/chrominance color space by the CST. In this example, sharpening and tonal adjustments are made to the luminance channel, while the chroma channels are left alone. Another instantiation of the CMM is necessary to convert to another color space or the final device space, if no further operations are required. This example represents a common use of an alternate color space to minimize image quality impacts while improving processing speed for multiband data.

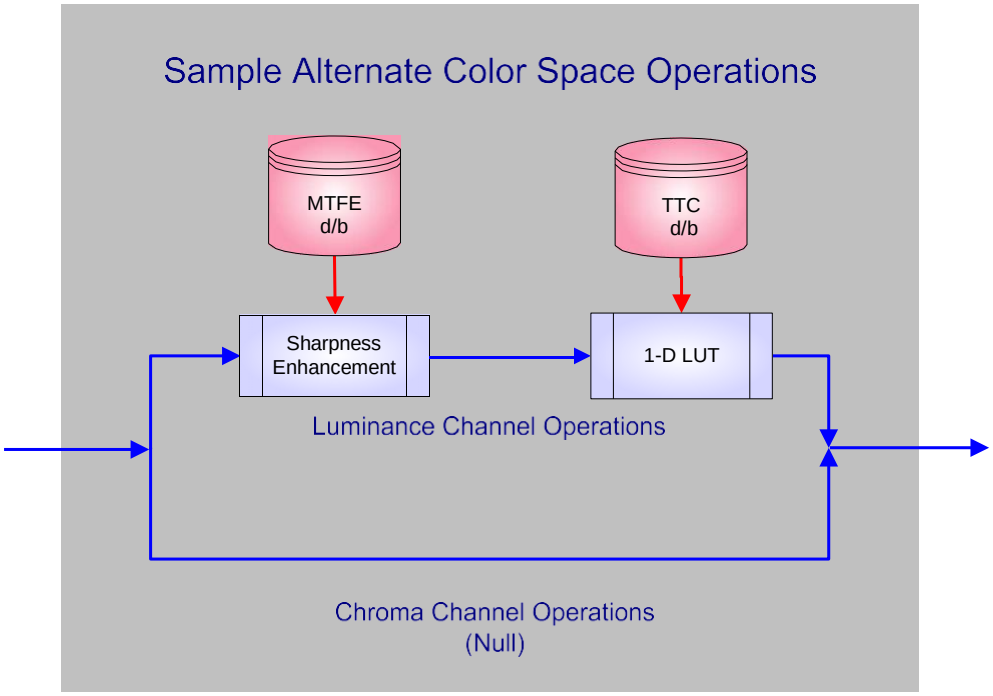


Figure 3.7: Sample Alternate Color Space Operations

3.4.7 Output Preparation

The output preparation element is identical to the CST only the destination color space is always the output device. For single-band imagery destined for display, this element quantizes the data from the bit-depth of the image chain (typically 12-bit or higher) to the bit-depth of the rendering element (typically 8-bit or 10-bit). For single-band imagery destined for storage, this element may leave the image data in 12-bit space, or a tonal compression may be applied to the image data prior to being sent to the appropriate compression algorithm to better preserve information. For example, a PEDF tonal adjustment may be applied in this element prior to compressing and storing the data.

For multiband imagery, output preparation uses a CMM to convert the image data from its current color space ('sensor' space or the last alternate color space used in processing the data) to the proper device color space for the intended output device (such as a 3-channel RGB space for a video board or a 4-channel Cyan Magenta Yellow Black (CMYK) space for a printer). If the image is to be stored, then a profile for an intermediate wide-gamut color space may be chosen to preserve color information when used by a subsequent CMM in another application. As with the single-band case, the data is mapped to the appropriate bit-depth for the intended application.

The destination profile used by the CMM must be the ICC profile generated for the desired output device configuration. The source profile used by the CMM will depend on the architecture of the image chain. The source profile must be that of the color space of the data at this point in the image chain. If no ACS operations were performed, then the source profile will be an input profile designed for the specific sensor. If ACS operations have been performed, then the destination profile of the last color space transform should be used as the source profile for output preparation.

3.4.8 Reference Implementations

For the purposes of establishing a reference image chain, two models are given below to define the order of operations used to obtain NGA/TQ Image Quality & Utility (NIQU) reference imagery. The NIQU reference imagery is utilized for comparison purposes in the verification section of this standard. This section may be updated in future versions as new algorithms, information or technology provides improved image quality results.

3.4.8.1 Reference Interactive Chain for Single-Band Imagery

Figure 3.8 illustrates the order of operations for the interactive component of the reference image chain for single-band imagery. Individual operations are explained in more detail in this section. Because single-band imagery contains no color information, there are no additional ACS operations. Single-band image processing then remains compatible with previous releases of this standard regarding databases utilized for sharpening and tonal adjustments.

3.4.8.1.1 Sharpness Enhancement

The sharpness enhancement element allows users to interactively adjust the appearance of edge detail. This convolution is applied in a manner similar to that used for MTF restoration discussed under Product Generation Options in Section 3.3.1.2.

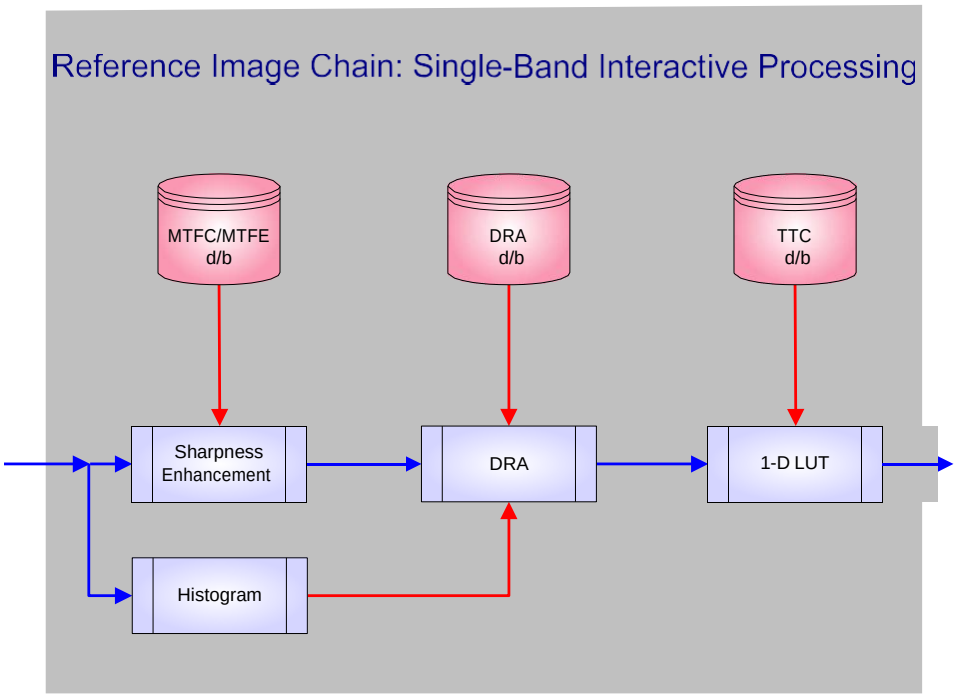


Figure 3.8: Reference Interactive Chain for Single Band Imagery

3.4.8.1.2 Sharpening Database

The sharpening database element consists of the kernels to be used for sharpness enhancement. For future systems, kernels in this database will *not* be designed to compensate for any specific MTF characteristics of a specific imaging device. For older systems, however, the MTFC kernel databases should be used.

The historical kernel databases are organized into four sets or families of sharpening kernels, each containing 64 unique kernels or members. Each family contains a number of kernels that provide an equally perceptible change in sharpness from one member to another. The various families perform different functions ranging from low-pass and high-pass boost filtering to mid-frequency-range-boost filtering.

3.4.8.1.3 Histogram

The histogram element calculates histogram statistics for the given image data. The results are then passed on to the DRA element. The data passed from this element to the DRA element must contain the information necessary to satisfy the calculations performed by the selected DRA algorithm.

3.4.8.1.4 Dynamic Range Adjustment (DRA)

The DRA element maps scene data from the dynamic range of the input image to that of a well-defined metric. Once mapped, standardized enhancement processing can be applied. For this standard, the well-defined metric is the entire range of the bit-depth of the image data.

3.4.8.1.5 Dynamic Range Adjustment (DRA) Database

This DRA database contains the parameters for performing a DRA on a given image. Histogram information is assumed available for a given image.

3.4.8.1.6 One-Dimensional Look-Up Table

The 1-D LUT element uses one or more 1-D LUTs to stretch or compress tone data in various regions within a digital image's dynamic range. Specific TTCs are provided in the TTC database. Selecting a specific TTC depends on many factors including image content and target application. Because there can be a large number of TTCs that can be built for a wide range of purposes, implementation of this element may require an interactive capability in addition to the TTC database element.

3.4.8.1.7 Tone Transfer Curve (TTC) Database

The TTC database contains GFI of various TTCs that may be applied to imagery. A variety of TTCs exist for exploiting different regions of an image's dynamic range, from shadows to mid-tones to highlights.

The TTC LUTs are grouped together into families based on shape characteristics. Each family contains 64 members designed to emphasize different tonal sections of an image. While providing desired enhancements to different regions of the tonal range, TTCs cannot recover information that is lost as a result of the DRA clipping.

3.4.8.2 Reference Interactive Chain for Multiband Imagery

If band equalization is applied to the multiband image data during Product Generation Options in the non-interactive component of the image chain, no further per-band operations are necessary on the data. If there is no automated band equalization performed during non-interactive processing, interactive band equalization must be performed before the color space transformation.

Historically, band equalization has been accomplished on multi-band imagery by applying the single-band DRA algorithm provided in section 2.6 to each band independently. This method is known to produce color shifts, as the algorithm

makes no provision for balancing the color of the image. Neutral objects in the scene may shift to red, green, etc. unpredictably due to changes in scene statistics. Beginning with version 2.3 of this standard, the reference image chain for multi-band imagery utilizes the BDL algorithm given in section 2.11. By performing linearization in a band dependent fashion, this algorithm improves the image quality of the rendered multiband scene by preserving the neutral colors desired in the final product.

Figure 3.9 illustrates the order of operations for the interactive component of the reference image chain for multiband imagery. At this time, band equalization is not performed in the non-interactive component of the reference chain, so a DRA is used here.

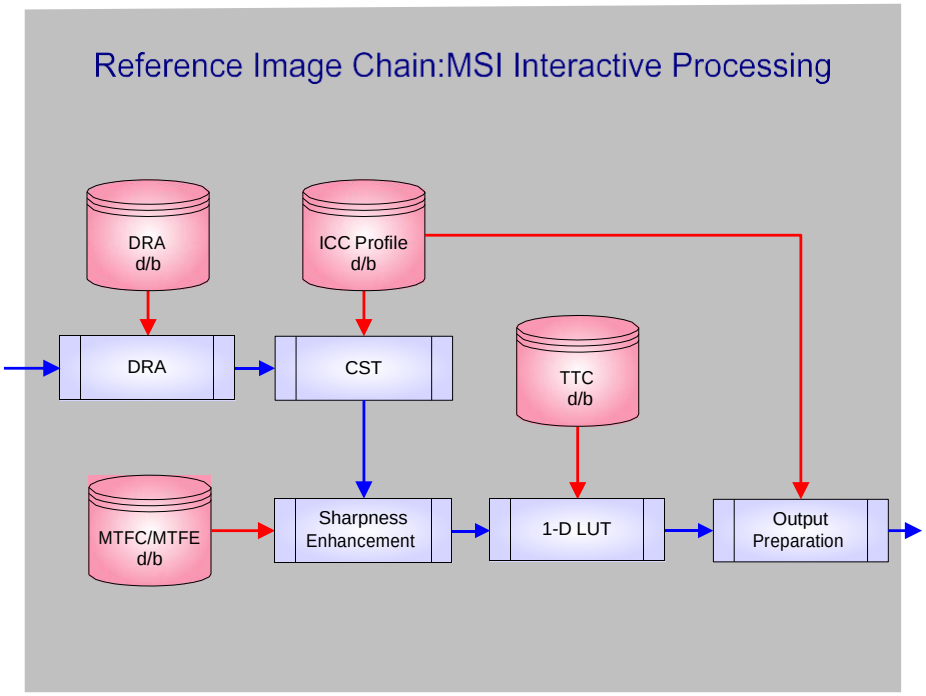


Figure 3.9: Reference Interactive Chain for Multiband Imagery

3.4.8.2.1 DRA

The DRA element balances the neutral response of the data as well as maps scene data from the dynamic range of the input image to that of a well-defined metric. Once mapped, standardized enhancement processing can be applied. For this standard, the well-defined metric is the entire range of the bit-depth of the image data.

Because multi-band imagery has the additional dimensions involved in color processing, the BDL algorithm is used by this standard for performing the DRA when a color managed workflow is used. Advantages of this algorithm are discussed in more detail in section 2.11.

Alternately, if the incoming data was previously processed, this would be a null operation.

3.4.8.2.2 DRA d/b

The DRA d/b contains algorithms and parameters for performing a dynamic range adjustment to the data.

3.4.8.2.3 CST

The CST element is employed to convert to a 'working color space.' This concept is well known in commercial venues where color management has been used for years. Because its introduction is relatively new to the field of remote sensing and image exploitation, it is currently a null operation for imagery that has not been previously processed. Operations that are often performed in an alternate or 'working' color space should continue being performed directly on the data in its native, 'sensor' space.

If imagery was previously processed and has an ICC profile associated with it, the CST is used at this point to convert the image data to a 'working' color space. While users may select a valid alternate color space to perform additional operations, for the purposes of the reference image chain at this time the 'working color space' should be color space of the output display device.

3.4.8.2.4 ICC Profile d/b

The ICC profile d/b contains the ICC profiles available to be used by the application. These may be managed by either the application or the operating system.

3.4.8.2.5 Sharpness Enhancement

The sharpness enhancement element allows users to interactively adjust the appearance of edge detail. This convolution is applied in a manner identical to the sharpness enhancement described for single-band processing and is applied to each band, independently.

3.4.8.2.6 Modulation Transfer Function Enhancement (MTFE) Database

The MTFE database element consists of the kernels to be used for sharpness enhancement. Kernels will be provided for this feature in a future version of this standard. These kernels will be true MTFE kernels and direction will be given as to what 'color' space they should be applied along with the relevant subset of bands to which they should be applied. The reference chain will apply these sharpening kernels immediately prior to the 1-D LUT to apply the desired TTC.

At this time, the recommended default sharpening for multiband sensors is a kernel from the MTFC kernel database. The specific kernel is dependent on the sensor from which the data was obtained. Default values can be found for each sensor listed in Appendix A or Appendix B.

3.4.8.2.7 One Dimensional Look-Up Table

This 1-D LUT element uses a 1-D LUT to stretch or compress tone data in various regions within the dynamic range of the data. Specific TTCs are provided in the TTC database. Selecting a specific TTC depends on many factors including image content and target application. Because there can be a large number of TTCs that can be built for a wide range of purposes, implementation of this element requires an interactive capability in selecting the appropriate TTC database element.

At this time, the recommended default TTC for multi-band sensors is listed in Appendix A or Appendix B, depending on the sensor in question. The selected TTC should be applied to each band of 'sensor' space data, independently.

3.4.8.2.8 Tone Transfer Curve (TTC) Database

This TTC database contains GFI of various TTCs. These TTCs are identical to the TTCs used in the single-band operations, and are applied to all the channels of the multiband data.

3.4.8.2.9 Output Preparation

The Output Preparation element implements a CMM to convert the data from its current color space to the output device color space. For the purposes of the reference image chain, previously unprocessed data should be in the 'sensor' space at this point in the chain. The appropriate ICC profile should be used as the 'source' profile for the CMM and the appropriate ICC profile for the output device (most likely the display, a printer, or a wide-gamut storage color space) should be used as the 'destination' profile.

3.5 Rendering

The rendering component prepares the product of the preceding image chain for the specific output device and provides the result to that device. Current output devices consist of printers for hardcopy, video monitors for softcopy, and storage. The rendering component provides the 1-D LUTs necessary to impose the proper tonal characteristic to each band or channel of data that the output device is capable of rendering. Because there is no device associated with storage, the data can be written directly to storage without further manipulations. A graphical representation of the rendering component is found in Figure 3.10.

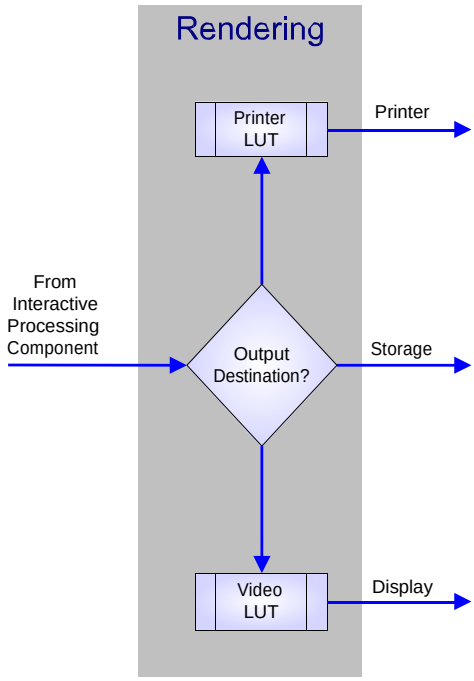


Figure 3.10: Rendering Component

Before the output profile is created the output device must be calibrated to a known state. For softcopy, calibration includes the monitor, video board and operating system setup.

The specific correction LUT applied by the rendering process is dependent upon the physical characteristics of the output device (i.e., its phosphor set, gamma, brightness, etc.). The performance of each device, and, hence, its output varies over time. For this reason, regular calibration of the output device is essential for preserving image quality (see Section 3.6.1).

3.5.1 Printer Look-Up Table

The printer LUT element processes the image data to impose the tonal characteristics desired from the hardcopy output device. A 1-D LUT is in place for each channel of the output device. Most color printers support four colors (cyan, magenta, yellow, and black), but many newer inkjet devices have started supporting additional colors to improve color rendering capabilities.

3.5.2 Dissemination & Storage

Output destined for storage is returned to Data Acquisition & Archiving (section 3.2) for file format selection and application of the desired compression algorithm.

3.5.3 Video Look-Up Table

The video LUT element processes the image data to impose the desired tonal characteristics of the softcopy output device. As with the printer LUT element, a 1-D LUT is in place for each channel of the softcopy output device.

3.6 Output

As shown in Figure 3.11, the output component may be a hardcopy device (e.g., a printer), a softcopy device (e.g., a CRT or Liquid Crystal Display (LCD)), or a storage device. With the exception of storage, the output device translates the data received from the rendering component into a set of signals that, in turn, produce a physical stimulus (i.e., a picture) that can be interpreted by the human eye.

The quality of the output when displayed is highly dependent on three predominant factors: the processing of the imagery up to the point of display, the calibrated device characteristics, and the viewing environment. This document deals specifically with the image processing piece. However, a few words are necessary with regard to the remaining factors since image quality can still be adversely affected by an un-calibrated (or out-of-calibration) output device and a poorly de- signed viewing environment.

3.6.1 Device Calibration

This standard requires calibration of output devices according to the NGA Softcopy Quality Assurance Program (SQAP). The NGA SQAP documentation covers the proper maintenance and calibration schedule of output devices in detail.

Of particular relevance to SIPS, the profile for the output device to be used in the output preparation element should be created only after completing a proper

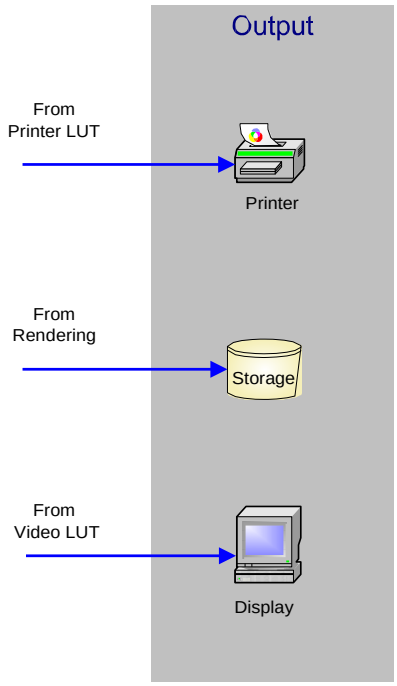


Figure 3.11: Output Component

calibration of the specific output device. Without calibration, the output from devices can vary widely, even within the same model type. Also, output devices drift and may degrade over time unless calibration is performed on a periodic basis.

The calibration procedure begins with the selection of appropriate hardware and facilities to achieve the desired level of quality. For softcopy, calibration includes the monitor, video board and operating system setup. The calibration process results in a device (printer or video) LUT that is used by the rendering element. Additionally, a profile is then generated for the calibrated device using this new LUT. This output profile is used by the CMM in the output rendering element to ensure proper color management is performed during the image processing. For the output profile to remain valid, devices must be periodically recalibrated to re- turn them to a predefined calibration state.

It should be noted here that output devices must be selected that support calibration and that perform to the desired level of quality or the final image quality will suffer. Some of the factors that affect the human visual system's ability to perceive information on a softcopy display include the luminance range and nonlinear response of the display, white and black point characteristics, sharpness, noise and artifacts. One of the most common artifacts on LCD is the viewing angle which can mask off-axis image detail. Refer to the NIQU DPS for standardized methods of evaluating these and other quality attributes of video displays and graphics boards to assess their fitness exploration use.

3.6.2 Environmental Effects

While the data produced through the reference image chain defines an extremely high quality exploitation image, and a calibrated display system faithfully renders that image without degradation, ultimate exploitability is determined by several other factors that affect the final communication of this information to the eye. The Human Visual System (HVS) varies its sensitivity to visual stimuli with various factors of the environment surrounding the stimuli. It is vitally important that these factors be addressed to provide optimal exploitation conditions, and hence maximum likelihood of extracting information from the resulting image data.

NIQU provides additional documentation regarding exploitation facilities that should be consulted to achieve optimal results. These documents detail the necessary lighting characteristics and directionality, surface properties of objects in the environment, and other design considerations including ergonomic factors that promote optimal exploitation and the comfort, health and safety of the analyst. Following these recommendations will ensure that the final link of the exploitation chain does not introduce a loss of quality in the displayed imagery.

More details regarding proper display calibration and facility setup can be found in the SFG, the SEFS, and the SQAP Ambient Light Measurement Procedure. The versions of each of these documents are detailed in the References section at the beginning of this document.

3.7 Advanced Geospatial-Intelligence Products

Advanced Geospatial-Intelligence (AGI) is a broad category of data within the NSG. Much of the data is imagery based and is therefore of interest to this image processing standard. The data is distinguished from the common, standard imagery types due to the nature of product generation and the inputs to those processes. Standard imagery begins with raw image data from a single sensor and produces a visible image, much like a snapshot from a camera. The input source and processing are treated as constants. AGI, however, allows the input and processing to be treated as variables.

AGI is characterized by advanced processing beyond (or in place of) the standard image formation chain. Such processing is used to accentuate information content of the data that cannot be seen in the standard image. This includes tar- get activity and detection, automated classification algorithms, and specialized color mapping as a few examples. AGI processing picks up where standard image formation leaves off, or it may utilize partially processed data tapped out of the image formation chain. Some AGI processing replaces the standard image formation chain completely, and utilizes multiple datasets to create a fused product. See

Figure 3.12 for a pictorial description of these differences.

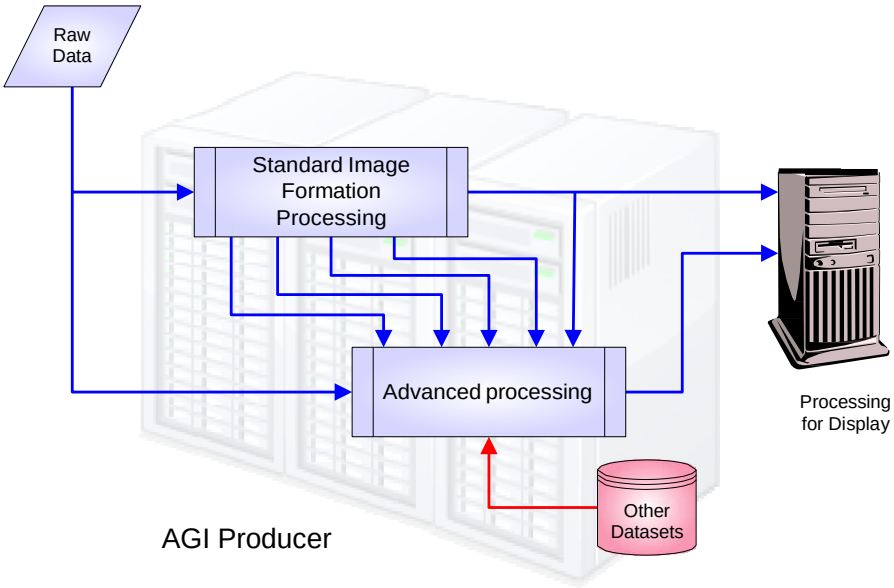


Figure 3.12: Standard Processing vs. AGI processing

Special treatment of AGI data in SIPS is warranted because, in many cases, the display processing requirements for AGI differ from those of the standard image types. These differences in processing equate to alterations in the SIPS reference implementation for specific types of AGI data.

Such alterations to the reference implementation are motivated by a combination of factors which are not unique to AGI, but come to the forefront in a brief analysis of AGI for inclusion in the standard. These factors are Product Utility, Processing History and their combined influence on Display Processing.

3.7.1 Product Utility

The intended product utility of AGI data is one factor driving modifications to display processing defaults. Product utility is concerned with the intelligence value of a product which is often characterized in terms of Essential Elements of Information (EEI) that can be satisfied by exploiting the product. EEIs are the building blocks of intelligence, the discrete, quantifiable pieces of information that together form a body of knowledge about a given intelligence scenario. An EEI involves detecting or identifying objects, characterizing changes or trends over time in a particular location, identifying evidence of a certain activity, distinguishing between similar objects and many other specific tasks.

Since image-based products convey information in spatial and spectral domains, utility is a measure of the intelligence value contained in one or both of those domains for a given product. Selection of the appropriate spatial and spectral enhancements for display is critical to preserve the EEIs in the data.

Standard datasets are produced to enable generally flexible utility, and the SIPS default display enhancements have been designed to preserve that flexibility. AGI data, however, is typically produced for a more specific purpose. In particular, AGI products are used in analysis because of their ability to satisfy EEIs that standard imagery products alone cannot. Therefore, AGI data is designed to accentuate a particular attribute or combination of attributes in the raw data coming from one or more sensors. In order to preserve the visibility of these at- tributes, specialized display processing that differs from the default chain may be required.

For each step of the reference implementation chain, there may be an alternate process that is recommended for a given AGI product. In some instances, a default processing step may be either by-passed or implemented with null parameters. In other cases, a specialized algorithm may replace a standard reference implementation step. Whatever the modification, the purpose is always to preserve the critical EEI in the product, thereby preserving product utility.

3.7.2 Processing History

The well-understood product generation chain of standard image products allows for the broad definition of default display processing. That process chain, though outside of NGA's control, is an inherent part of the standard imagery that NGA receives. With this *a priori* knowledge of processing history, the SIPS default reference implementation for standard image products was developed. However, for datasets generated using customized processing (such as AGI), the standard assumptions of image processing pedigree are not valid and default display processing will be influenced by the customized image formation chain.

Suppose, from the previous section, a product's primary utility is in the spatial domain. Spatial enhancements processing may be necessary prior to display. However, depending on the type of spatial processing involved in generating the product (processing history), additional spatial enhancement may be unwarranted and could possibly diminish product utility. This is why it is important to know both the intended utility of the product and the processing history.

The determination of processing history may not be a trivial exercise. Several options exist for determining a particular product's processing history, or pedigree, each with its trade-offs. The options are determined through *a priori* knowledge of the producer's image chain, interrogation of processing history metadata upon receipt of the image, or through reverse engineering.

A priori knowledge of the producer's image chain has the advantage of unambiguously establishing the processing chain used to generate a specific type of

image product, enabling an efficient, optimized display processing chain. This is the approach for standard products. However, this approach becomes more complicated and less desirable as the number of data producers and data types grows rapidly. Indeed, this approach may not even be feasible for some types of products from commercial or foreign producers.

Determining processing history from metadata is a promising approach because a product that contains its pedigree avoids some of the drawbacks of the other methods. However, it also requires strict adherence to a metadata standard so that the processing history can be accurately extracted by a downstream process that does not have *a priori* knowledge of the product. Once again, this may not be feasible for products whose producers do not adhere to the NGA standard.

Reverse engineering can be process-intensive and require significant user interaction and may not be able to unambiguously reverse engineer every type of previous processing. However, for a given type of product, reverse engineering is only necessary for the first product, resulting in a deduced process history that applies to subsequent products of the same type. While this may be the only option for certain types of products, it is not the most desirable approach.

3.7.3 Display Processing

Display processing, as documented here in Figure 3.13, covers the spatial and spectral domains of imagery. Spatial processing includes MTF restoration and enhancement, chipping, RRDS generation, geometric corrections, interpolations, and scaling. Spectral processing covers band equalization, atmospheric correction, data remapping, DRA, TTC, and color space transformations.

In terms of the types of data that are handled and processed by NGA, from the perspective of SIPS, there are two general divisions of data: that which is display-ready as received, and that which requires some amount of enhancement processing prior to display. Figure 3.13 shows the breakdown of display processing categories as used in SIPS.

Display-ready data requires no further processing in order to be presented to an analyst for visual exploitation. All processing necessary to optimize display properties of the data was accomplished in the production chain. Once provided to the analyst, the data can be processed through a null enhancement chain or a chain that only includes a combination of bandwidth expansion, chipping operations or CMM processing for rendering to the display. No further spatial, tonal or color processing is necessary to adequately view the desired product.

Data that requires display processing represents a large and complex category because of the many kinds of products that are encompassed. This category can be further subdivided into products that adhere to the default SIPS display processing chain and those that have unique requirements for visual display processing.

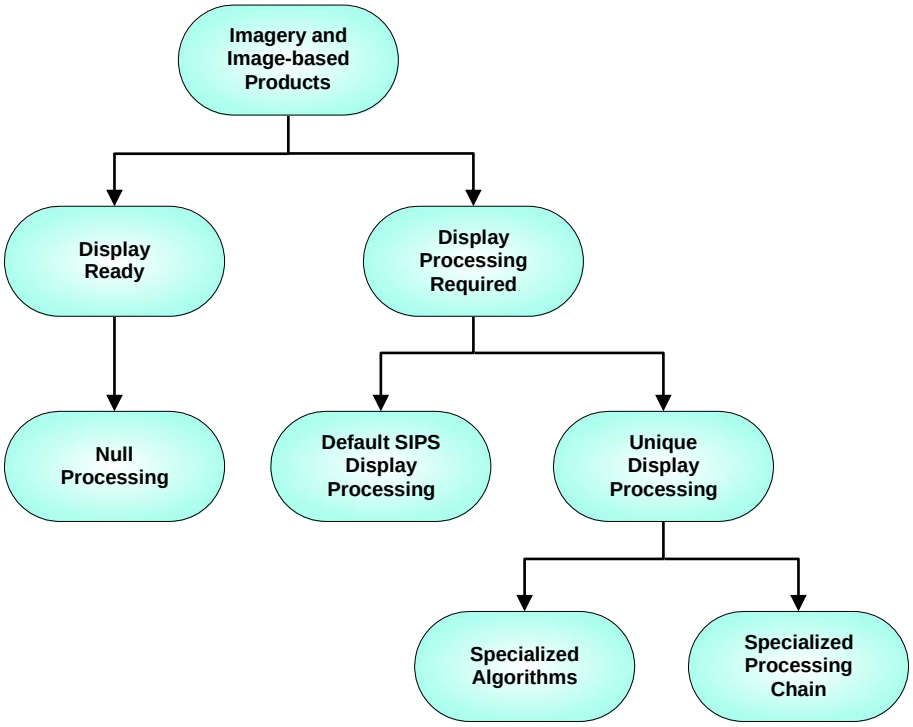


Figure 3.13: SIPS Display Processing Categories

SIPS includes a number of alternative algorithms and processing chains for display enhancement. These alternatives will be discussed in terms of their applicability to specific types of products. Further information on display processing for AGI products is located in Appendix A and Appendix B.

Verification

All software used by the NGA to store, manipulate, or disseminate image data must conform to the image quality requirements of this standard. Conformance is determined by comparing the image quality of data processed by the software to reference imagery. The software will be determined to be compliant with this standard if it produces image quality within the defined measures of this standard for still image evaluation and provides perceptually equivalent quality to still image display at roam rates up to 480 pixels per second (pps). Perceptual equivalence during roam provides the allowance to decrease the quality of imagery while roaming in order to preserve the smooth and continuous display of data by, for example, not fully expanding JPEG2000 encoded data or by using 1 RRDS-level image lower in quality than that used for static display.

The SIPS team utilized current NGA standards, legacy exploitation systems methodologies, and engineering best practices to determine a set of metrics that were applicable to image quality difference calculation for SIPS verification testing. Image quality metrics based on engineering practices and experience have been determined, and the most appropriate for quality assurance were chosen. These are provided along with acceptable tolerances for each metric specified in Sections 4.2 and 4.3.

Test images have been processed in accordance with the reference image chain provided in Section 2 (Reference Image Chain Algorithm Descriptions) and Section 3 (Reference Implementation). The parameters and input/output images from this process may be obtained by contacting NIQU. Link to `http://t.nga.ic. gov/VVG/?org=tq` for the most current contact information.

4.1 Test Imagery

Verification of SIPS version 2.4 is done with tiff images at various bit-depths. In order to test the algorithms in the imaging chain, the reference image has been constructed with a combination of real and synthetic imagery. For generic testing, a 12289x12287 pixel image size was decided on to allow for the inclusion of different test targets and real imagery, as well as being sufficiently large to allow for R-set generation.

1024x1024 test patterns encompassing various digital count ranges were added

around a piece of imagery (12-bit scanned film image) with a surround of simulated noise. The image noise was generated with an initial mean of 24000 and a standard deviation of 4096. The test image was created in 16-bit ($0 \rightarrow 65535$ DC) and downsampled to 14, 12, 11, 10, and 8-bit depths. All of the test images except 8-bit are saved in 16-bit files.

The imagery used for testing is delivered on a Digital Versatile Disc (DVD) and can be acquired along with the databases used by the reference image chain given in Section 3 of this standard. Image files along with testing parameters used to generate corresponding output images using the reference image chain are given in section A.

4.2 Pan Imagery

It follows from the intent to determine differences in imagery for visual exploitation that a metric should be linearly proportional to the magnitude of the visual difference seen by an analyst viewing the two images being compared. This goal is accomplished for single-band imagery if the imagery is displayed on a system that yields a visual response that is linearly proportional to the human visual system. The NGA DPS, SEFS, and the SFG provide for the appropriate display calibration settings and viewing environment necessary to achieve the above needed relationship between processed image data and visual response. The display non- linearity is known by the name Image Data and Exploitation (IDEX). The IDEX aim curve for displays is well defined under a very specific set of viewing conditions.

More about the details of this aim curve will be discussed later. For now, it is important that this curve is applicable over a very specific luminance range for a display. Different display technologies have different capabilities for achieving specific minimum and maximum luminance values. This is an important aspect of the display non-linearity. While the non-linearity is utilized to elicit a linear visual response to the display luminance output, it is the luminance range of the display that determines the portion of the aim curve used to define the relationship.

Differences historically have been calculated in terms of delta digital count (ΔDC) of the final processed image being readied for display. This metric has served the community well for many years for single band imagery. However, there is an implicit assumption in this metric. This metric assumes that a ΔDC of 1 is an equal number of Just Noticeable Differences (JNDs) and the values are indicative of the actual magnitude of the difference that is being measured.

The most common error metric for pan images is engineering Root Mean Square Error (RMSE), calculated on a global basis as well as on smaller image tile areas. Equation 4.1 shows the calculation used in this case,

$$RMSE = \sqrt{\frac{1}{n}\sum_1^n (DC_{proc} - DC_{std})^2} \tag{4.1}$$

where n is the number of pixels, DC_{std} is the reference pixel value, and DC_{proc} is the value of the corresponding pixel in the processed image.

Following accepted NIQU practice, the RMSE tolerance is 5 DC for 11-bit imagery (or 0.2442 normalized). The SIPS team is utilizing this methodology in a normalized fashion to account for imagery of various bit-depths (see Table 4.1) and provide a basis for comparison with the color metrics discussed later in this standard. Normalization is performed to a range of $0 \rightarrow 100$.

Table 4.1: Error Metric Tolerances for Various Bit Depths

Bit-depth	RMSE	normalized DC
8	0.625	**0.2442**
10	2.50	
11	5.0	
12	10.0	
14	40.0	
16	160.0	

The global maximum deviation (the largest difference in the entire image or tile) can be no larger than 10% of the Dynamic Range (DR).

4.3 Multiband Imagery

Determining visual differences between multiband images begins with the image data rendered through an image chain to RGB image data that can be sent directly to a video card for display. Once processed, visual differences between RGB images will vary based on display capabilities and characteristics, viewing conditions, and observer characteristics. All these factors must be taken into account to have a meaningful metric for visual differences in the imagery.

4.3.1 Color Appearance Models

The most complex factor mentioned above is the observer. The HVS is extremely complex with many psychological and physiological aspects that define the appearance

of the image data. The HVS is capable of perceiving colors in multiple dimensions. Because of the complexity of the HVS, a multi-dimensional space that can represent 'color appearance' is necessary. In order to utilize proven models in colorimetry along with time tested understanding from the community, a color model and associated color difference metric is selected that is applicable to imagery viewed by analysts.

The color appearance space known as CIELab is a relatively simple calculation and has been widely used by industry for several decades. [1]

CIELab was endorsed by the Commission Internationale de l'Éclairage (CIE) in 1976 and has been well characterized over the years. It represents color appearance in a three dimensional space where the L^* axis represents perceived lightness, and the a^* and b^* axes represent the chromatic information. Color characteristics of chroma and hue can be obtained through straightforward transformations to a polar coordinate system. An error metric, commonly known as a ΔE^*,
is simply a Euclidean distance between CIELab values. It is this color space that will be used by SIPS for determining color appearance and differences.

With the selected color appearance model, it is necessary to describe how the model will be used to make comparisons between outputs from multiband processing chains. Figure 4.1 illustrates the process of converting to the CIELab color appearance space for image data that has been through the SIPS processing chain including Output Preparation just prior to Rendering. Each stage is covered in a subsequent section.

[1] Recently developed color appearance spaces, like **CIECAM02**, are extremely involved and mathematically complex, as they attempt to account for virtually every known psychological and physiological visual effect relating to human color perception. While this space would be ideal for the uses of this project, the level of complexity was determined to be too large for this task.

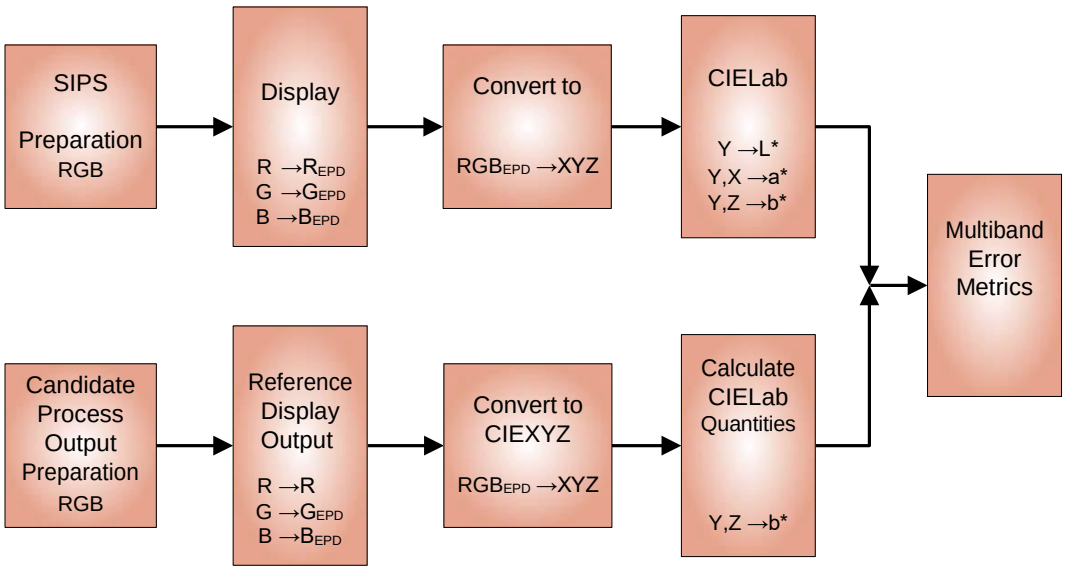


Figure 4.1: Color Modeling in SIPS

4.3.1.1 Modeling Reference Display Output

The purpose of modeling a reference display is to calculate the "stimulus" of the specified color to the human observer through the display. This "stimulus" value is measured as a point in a three-dimensional space defined by the CIE as XYZ. XYZ (aka tristimulus) values represent the visual stimulation perceived by a human observer. The 'Y' component is commonly known as the luminance value, while the 'X' and 'Z' values represent the chromatic information.

Because human perception is not linear with luminance, displays are typically calibrated with respect to human perception rather than luminance in order to maximize the number of discriminable colors the device can produce. This also means that the relationship between the RGB values being displayed and the XYZ tristimulus values produced by the display is not linear. This nonlinearity is accounted for in a simple color model of a display. The color model accounts for the nonlinearity of the display and then performs a linear transformation from RGB to XYZ. While fundamental principles of color modeling of displays are used by this standard, further details are beyond the scope of SIPS.

The first step to model the colorimetric appearance of the image data is to simulate the luminance response of the reference display. This is accomplished by mapping the image RGB values through the same nonlinear response of the calibrated device. To be consistent with SQAP and the DPS, SIPS models the reference display luminance using the IDEX aim curve to achieve an Equal Probability of Detection (EPD) response.

Once the aim curve is applied to the data, a linear transformation based on the chromatic properties of the display is used to calculate the XYZ tristimulus values. The tristimulus values are then used as the basis for calculating the "appearance" of the color in CIELab. These values may then be used to determine the magnitude of color differences between two color stimuli.

4.3.1.1.1 IDEX Aim

The IDEX aim was developed for proper calibration of the monochrome CRT display that was part of the IDEX workstation. Today, the IDEX aim is still utilized for calibrating CRT displays and LCDs, both color and monochrome. Its purpose is to convert the incoming digital count range to display output luminance values that result in equal probability of detecting perceptible differences in a scene across the dynamic range of the display.

For the original IDEX workstation and monochrome CRT displays, the achievable luminance range for calibration was $[0.2, 70]$ fL, and thus, the original IDEX polynomial was designed for this range. That polynomial is shown in Equation 4.2

$$L_{EPD}(x) = 0.2 + 8.1206638 \cdot x - 12.453941 \cdot x^2 + 96.293375 \cdot x^3 - 121.85936 \cdot x^4 + 99.699238 \cdot x^5 \quad (4.2)$$

where the input (x) for this equation is the command value normalized to the domain of $[0, 1]$ and it maps to an output device luminance range (L_{EPD}) of $[0.2, 70]$ fL.

To utilize the above equation correctly, the luminance range of the reference display must be defined to ensure that the appropriate portion of the aim curve is utilized for the transformation. This is required because the luminance output of Equation 4.2 is an absolute scale. Under the state of visual adaptation enabled by adherence to SQAP and Facilities Standard guidelines, the eye responds to absolute quantities of light in accordance with the EPD aim. For this application the reference display is defined as being calibrated according to the SQAP standard with a luminance range of $L_{\in}[0.15, 40]$ fL.

From this reference luminance range the portion of the aim curve applicable to that range can be isolated. This requires determining the input domain (x) that maps to the output range. The value of x corresponding to L_{min} will be mapped to count 0 of the display while the value of x corresponding to L_{max} will be mapped to the maximum digital count (255 for a typical 8-bit application). Since the reference maximum luminance (40fL) is below the IDEX maximum luminance (70fL), the input range will not utilize the full 0 to 1 range in x. For the reference display, an L_{max} of 40 fL can be found by setting $L = 40$ in Equation 4.2 and solving for x, yielding $x_{max} = 0.8632669$. For L_{min} of 0.15 fL, this requires a slight extrapolation of Equation 4.2 since $L_{min} < 0.2$ fL.

In similar fashion, setting $L = 0.15$ and solving gives $x_{min} = -0.00609741$. The domain of the aim curve for the reference display range is $[x_{min}, x_{max}]$. Thus, all digital count values have to be scaled to this range via a linear transformation.

$$\begin{aligned} x_{EPD,ref}(DC_{RGB}) &= (x_{max} - x_{min}) \cdot \left(\frac{DC_{RGB}}{DC_{max}}\right) + x_{min} \\ &= \left(0.86326691 - (0.00609741)\right) \cdot \left(\frac{DC_{RGB}}{DC_{max}}\right) + (-0.00609741) \\ &= 0.86936432 \cdot \left(\frac{DC_{RGB}}{DC_{max}}\right) - 0.00609741 \end{aligned} \tag{4.3}$$

where $x_{EPD,ref}$ is the value of $x_{EPD,ref}$ used as input to Equation 4.2, DC_{RGB} is the digital count from an individual channel of the RGB image, and DC_{max} is the maximum digital count based on the bit-depth of the image (e.g., 255 for 8-bit).

Using the scaled reference input range, $x_{EPD,ref}$, the EPD aim equation (4.2) may be used without altering the coefficients (i.e. renormalizing). However for certain applications, it may be desirable to have a renormalized version of EPD specific to the reference luminance. To do so, the scaled input range definition for $x_{EPD,ref}$ is substituted back into the EPD aim (equation 4.2), the polynomials are expanded and terms are recombined to generate a version of the aim that is renormalized for the reference luminance range ($L_{EPD,ef}$) of [0.15, 40] fL. This results in the following equation:

$$L_{EPD,ef}(x) = 0.15 + 7.2012824 \cdot x^2 + 65.247790 \cdot x^3 - 71.345375 \cdot x^4 + 49.510907 \cdot x^5 \tag{4.4}$$

where $x = \frac{DC_{RGB}}{DC_{max}}$

Using equations 4.2 through 4.4, the following equations show the first step of modeling the color stimulus of the RGB values.

$$\begin{aligned} R_{EPD} &= \frac{L_{EPD,ref}(R) - L_{min}}{L_{max} - L_{min}} \\ G_{EPD} &= \frac{L_{EPD,ref}(G) - L_{min}}{L_{max} - L_{min}} \\ B_{EPD} &= \frac{L_{EPD,ref}(RB) - L_{min}}{L_{max} - L_{min}} \end{aligned} \tag{4.5}$$

The $R_{EPD}, G_{EPD}, B_{EPD}$ values are now proportional to the actual luminance response of the reference display and may be used as the input to the next step of the calculation of CIE XYZ.

4.3.1.2 Convert to CIEXYZ

Once in EPD space, a 3x3 matrix is used to convert to the standard CIEXYZ color space. The SIPS team uses the RGB to XYZ matrix utilized by the International Electrotechnical Commission (IEC) 61966-2-1:1999 standard (also known as sRGB). The sRGB primaries are representative of a typical color display in use at this time. It is important to note that technology changes involving wide-gamut displays are currently being developed that require a different matrix. The specific matrix needed must be determined based on the characteristics of the wide-gamut device. The sRGB matrix used for this transformation is the following:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & X0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \cdot \begin{bmatrix} R_{EPD} \\ G_{EPD} \\ B_{EPD} \end{bmatrix} \tag{4.6}$$

where$R_{EPD}, G_{EPD}, B_{EPD}$ are the normalized values resulting from the application of the EPD aim to each channel of the image and normalizing according to Equation group 4.5.

4.3.1.3 Calculate CIELab Quantities

Once the image data is converted to XYZ, the CIE L*a*b* values are calculated. The CIELab quantities are defined in the following manner:

$$\begin{aligned} L^* &= 116\left[f\left(\frac{Y}{Y_n}\right)\right] - 16 \\ a^* &= 500\left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right] \\ b^* &= 200\left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right] \end{aligned} \tag{4.7}$$

where

$$f(t) = \begin{cases} 7.7870(t) + {}^{16}/_{116} & , \quad t \leq 0.008856 \\ (t)^{1/3} & , \quad t \leq 0.008856 \end{cases} \tag{4.8}$$

It is in this perceptual space that meaningful metrics can be taken regarding the appearance of multiband data.

4.3.2 Multiband Error Metrics

ΔE^* takes both luminance and chroma into account based on the CIELab color space, which was designed to be perceptually uniform, although it does not quite attain this uniformity. The uniformity of CIELab is considered adequate for the purposes of this standard, therefore perceptual color differences can be determined by the Euclidian distance between colors expressed in this color space.

$$\begin{aligned}\Delta E^*_{ab} &= \sqrt{\Delta L^{*2} + \Delta a^{*2} + \Delta b^2} \\ &= \sqrt{(\Delta L^*_2 - \Delta L^*_1)^2 + (\Delta a^*_2 - \Delta a^*_1)^2 + (\Delta b^*_2 - \Delta b^*_1)^2}\end{aligned} \tag{4.9}$$

An advantage of this metric is that it has been used by the color processing community for years and is well understood, including its limitations.

Engineering RMSE calculations may be performed on this metric. Tolerances are determined in similar fashion to the pan error metric. The RMSE tolerance of 0.2442 is used for the ΔE^*_{ab} metric, as a ΔE^*_{ab} calculation on a pan image is similar to the pan metric of digital count difference, since there is no chromatic information.

4.3.2.1 Comparison of Pan and Multiband Metrics

Ideally, the multiband metric would be a generalized function that reduces to the panchromatic metric when color (chromatic) information is eliminated. However, for the two metrics used in SIPS, this is not the case. The difference arises from the underlying luminance perception model of each metric. CIELab models lightness (perception of luminance) in the L^* term using a cubed-root function. The panchromatic metric relies on the use of the IDEX EPD aim, a degree-5 polynomial, for display calibration as the basis for modeling lightness.

L^* and the IDEX aim perform opposite functions: L^* maps luminance to equal perceptible gray levels while the IDEX aim maps equal perceptible gray levels to luminance. However, it is important to understand the role of each function in the respective metrics. The color metric requires modeling the display output (luminance) with the IDEX aim followed later by a conversion to lightness by calculating L^* on the luminance channel. The analogous operation for panchromatic imagery would be applying the IDEX aim to the data to model display luminance then using the inverse of the IDEX aim to model perception of lightness. However, for panchromatic imagery, the output equals the input, meaning that the original digital counts represent perceived lightness. Therefore, the panchromatic metric is based solely on digital count differences with the stipulation that the display be calibrated with the IDEX aim so that digital counts are perceived as linear in lightness.

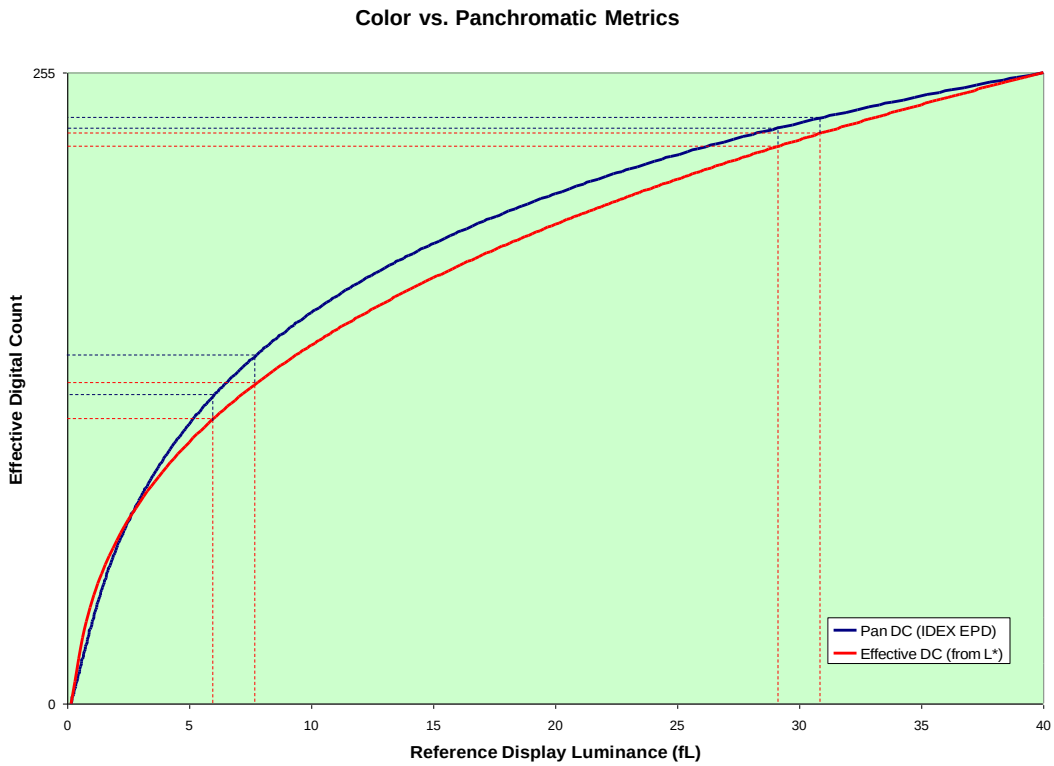


Figure 4.2: Comparison of L^* and Inverse IDEX

The two quantities to compare are L^* and the inverse of the IDEX aim. Though these are analogous in concept, they are not identical, recalling that L^* uses a cubed-root while the inverse of IDEX EPD is the inverse of a degree-5 polynomial. The curve shape of the two functions differs. Converting L^* to an effective digital count can be done by multiplying by a factor of 2.55 so that the two functions can be plotted on the same axes, as shown in Figure 4.2.

As the figure shows, the dark blue curve representing the inverse IDEX aim trends above the red curve of L^* throughout most of the range. The slope of L^* with respect to inverse IDEX fluctuates over the depicted range. L^* is slightly steeper at the very lowest portion of luminance and then again over a broader part of the highest luminance, while it is flatter than IDEX through the remainder of the midrange.

Determining the perceptual difference between two luminance values requires both values to be converted to the same perceptual scale and taking the difference in the result (which corresponds with ΔL^*). Since L^* and inverse IDEX have differently shaped curves, the perceptual differences generated using each scale will vary depending on the slope of the curve where the input luminance difference lies.

In Figure 4.2, two sets of luminance differences are mapped to L^* and the IDEX inverse to demonstrate the behavior of the two metrics. The dashed lines going from the x-axis to the respective curves and then to the y-axis represent the process of converting or mapping to a perceptual space. For the luminance difference at the lower part of the dynamic range, the perceptual difference given by L^* is less than that given by inverse IDEX because the inverse IDEX curve has a steeper slope in that region. The luminance difference at the higher end of the dynamic range, however, has a perceptual difference in L^* that is slightly larger than inverse IDEX, since the L^* slope is steeper in this region.

This page intentionally includes only this statement.

Glossary and Acronyms

5.1 Glossary

Aim Curve - Desired relationship between digital counts in an image and the luminance output of the display device.

Along Scan - In a scanning image collection platform, pixels that are collected in the direction of scan path.

Anti-Aliasing - Filtering process to remove high spatial frequency content from an image prior to resampling so as to eliminate spurious high frequency information, known as aliasing, in the resampled image.

Artifact - Spatial or tonal anomaly introduced by image processing algorithm which can degrade image quality.

Aspect Ratio - Ratio of horizontal to vertical extent of a display device or an individual pixel of an image.

Asymmetric - Imagery containing pixels which differ in horizontal versus vertical pitch (*i.e.*, non-square pixels).

Auto-DRA - Automated dynamic range adjustment based on image pixel (tonal) statistics.

Band - Discrete portion of the electro-optical spectrum sampled by an imaging sensor; see also channel.

Band Equalization - Process for multi-band imagery that ensures neutral colors maintain neutral appearance (equal digital counts) throughout the dynamic range.

Bandwidth - Amount of information (in terms of bits or bytes) contained per pixel in an image.

Bandwidth Compression - Process for reducing the bandwidth, and thus file size, of an image.

Bandwidth Expansion - Reversing the process of bandwidth compression to restore an image to its original bandwidth for viewing or other processing.

Bit-depth - Number of bits used to represent the value of a single pixel.

Brightness - Colorimetric term corresponding to an attribute of a visual sensation according to which an area appears to exhibit more or less light.

Candela - Unit of luminous intensity in the CIE photometric system; $1/60$ of the luminous intensity of $1\ cm^2$ of a blackbody radiator at the temperature of solidification of platinum.

Channel - See Band.

Chipping - Selecting a spatial subset of an image.

Chroma - Color information.

Chromatic - Exhibiting hue as distinct from white, gray, neutral, or colorless.

Chromaticity - Definition of a color without a brightness component; typically expressed in terms of the CIE chromaticity coordinates x, y.

Clipping - mapping of a value that is outside the available bit-depth range (above or below) to the minimum or maximum value for the range.

Color - (1) sensory or perceptual component of visual experience, characterized by the attributes of hue, saturation, brightness, and usually arising from, in response to stimulation of the retina by radiation of wavelengths between 380 and 760nm; related to the terms hue, tint, or shade; (2) stimulus or a visual object which evokes a chromatic response.

Colorimetric - Relating to measurement of color.

Commission Internationale de l'Éclairage - Also known as the International Commission on Illumination. An international body commissioned in 1931 to develop a worldwide standard for describing the visual perception of color.

Cones - Photoreceptors in the human eye that are responsible for color vision, active during both mesopic and photopic vision.

Contrast - Degree of brightness difference between two stimuli in an image.

Convolution - Algorithm that computes a weighted sum of several input pixels via a kernel for a given output pixel of a single band of image data; by definition, the kernel is 'flipped' horizontally and vertically during computation.

Correlation - Algorithm similar to convolution that uses a kernel to compute an output pixel as a weighted sum over a window of input pixels; unlike convolution, the kernel is not 'flipped' during computation.

Cross-scan - In a scanning image collection platform, pixels that are collected perpendicular to the direction of scan.

Decimation - Process to create a lower resolution representation of an image; see also scaling.

Decoding - Expansion of J2K data to the original bit-depth.

Decomposition level - Reduced resolution versions of an image that are part of the J2K code-stream.

Density - Light stopping property of a medium; for digital imagery, inversely proportional to the brightness of a pixel.

Display-ready - Imagery that can be viewed with no further enhancement processing required.

Down-sample - Process to reduce an image in size by eliminating algorithmically determined rows and columns.

Dynamic Range - Ratio of the maximum luminance divided by the minimum luminance.

Electro-Optical - Portion of the electromagnetic spectrum covering the visible and near Infrared (IR) wavelengths of light.

Electronic Light Table - Software used for viewing imagery and conducting image exploitation and analysis.

Embedded - Referring to extra non-image information contained within the image file.

Encoding - Process of applying J2K compression to an image.

Enhancement - Improving visibility of spatial and tonal information content of an image; processes by which those improvements are made.

Exploitation - Extracting intelligence information from an image.

Filter - See Kernel.

Filtering - Process that alters spatial information in an image to accentuate desired features or mask spurious information; typically accomplished via convolution.

Foot-candle - Unit of illuminance measurement; 1 foot candle = 1 lumen per square foot.

Foot-Lambert - Unit of luminance measurement; describes the luminance of a surface that emits or reflects one lumen per square foot; also the luminance of a Lambertian surface under an illumination of one foot-candle; 1 foot-Lambert = 3.4263 candela per square meter.

Gamma - Characteristic luminance response of a display device.

Grayscale - Monitor showing only shades of gray–*i.e.*, achromatic; (2) palette of achromatic tones available on a monitor ranging from white to black.

High-pass - Spatial filter that boosts high-frequency content of an image, producing an image that appears sharper than the original.

Highlight - Brightest region of an image dynamic range.

Histogram - Statistical tool showing the frequency of occurrence of individual digital counts in an image.

Hue - Degree to which a color stimulus can be described as similar to or different from stimuli that are described as red, green, blue, and yellow.

Kernel - Spatial mask of weighting values to be used in a convolution or correlation process; see also Filter.

Lambertian - Uniform radiance distribution.

Lightness - Brightness of a color judged relative to the brightness of a similarly illuminated color that appears to be white or highly transmitting.

Lin-Log - Non-linear tonal remapping applied to radar imagery to reduce bit-depth; consists of a "linear" and a "logarithmic" mapping for different portions of the dynamic range.

Lossless - Bandwidth Compression technique that preserves all original information (*i.e.*, fully reversible.

Lossy - Bandwidth Compression technique that allows some information loss in favor of increased compression ratio.

Low-Pass - Spatial filter that attenuates high-frequency content of an image, producing an image that appears blurred as compared to the original.

Luminance - Photometric measure of the amount of luminous intensity in a given direction; measured in foot-Lamberts.

Mask - Another term for a spatial filter or kernel.

Metadata - Non-image information which describes attributes of an image file, often contained within the file.

Midtone - Medium-range brightness levels (between the shadows and highlights) in the dynamic range of an image.

Mirroring - Replication in reverse sequence of image rows and columns at the image boundaries to support convolution or correlation.

Modulation Transfer Function - Spatial frequency response of an imaging system or process.

Modulation Transfer Function Compensation - A spatial operation that compensates for the MTF of the optical system and applies additional sharpening to edge detail for visual enhancement.

Modulation Transfer Function Enhancement - A spatial operation that applies sharpening of edge detail for visual enhancement.

Modulation Transfer Function Restoration - A spatial operation that compensates for the MTF of the optical system.

Normalize - Process by which a set of values is scaled relative to a reference value such that the reference value scales to 1.0.

Observer - Referring to the human viewer of an image.

Panchromatic - Imagery collected by sampling a broad range of the EO spectrum in a single band.

Pitch - Spacing between the centers of physical pixels.

Pixel - Smallest unit of information making up an image; a shortened term for 'picture element.'

Product Separable - Spatial filter with independent horizontal and vertical components.

ProPhoto RGB - Wide-gamut color space.

Quantize - Mapping a large or continuous set of values to a smaller finite set of values.

Raw Data - Image data that has not yet gone through PGO image processing.

Remap - Change the digital count value of a pixel to another value based on a LUT, equation, or algorithm.

Render - Prepare an image for a particular output device.

Resample - Change the native spatial sampling characteristics of an image.

Resolution - Sampling rate or pixel pitch of an image.

Resolution Level - Individual lower resolution image that is part of an RRDS.

Rset - Reduced resolution data set; set of reduced resolution images produced from a high resolution original

Rx - When X is replaced by an integer number, it refers to a specific image within an Rset; X represents the level of reduction in powers of two from the original image (e.g., R0 designates the original, full resolution image; R7 designates the image at a reduced resolution of 1 : 128).

Saturation - Quality of visual perception which permits a judgment of different purities of any one dominant wavelength; degree to which a chromatic color differs from gray at the same brightness.

Scaling - Changing the pixel pitch, and thus the size, of an image.

Shadow - Darkest region of an image dynamic range.

Spatial - Image information related to the location or position of pixels.

Spatial Frequency - Referring to the rate at which spatial information varies over given distances in an image.

Spatial Variant - Algorithm that changes processing parameters based on location within an image.

Subsample - See down-sample.

Tonal - Image information related to the brightness (digital counts) of pixels.

Transcoding - Re-arranging the J2K codestream without decoding the data.

Uncompressed - Imagery at full bandwidth or bit-depth.

Utility - Intelligence value of an image; exploitability.

Warp - Change the shape of the sampling grid of an image.

Wide-gamut - Referring to a color space gamut that covers a larger extent of chromaticities than typical display devices.

5.2 Abbreviations/Acronyms

1-D LUT	One Dimensional Look-Up Table
ACS	Alternate Color Space
ADS	Algorithm Description Sheet
AGI	Advanced Geospatial-Intelligence
AIP	ASARS-2 Improvement Program
BDL	Band Dependent Linearization
BPP	Bits Per Pixel
BWC	Bandwidth Compression
CE	Controlled Extensions
CIE	Commission Internationale de l'Éclairage
CMM	Color Management Module
CMYK	Cyan Magenta Yellow Black
CONOPS	Concept of Operations
COTS	Commercial Off-The-Shelf
CRT	Cathode Ray Tube
CST	Color Space Transform
D/B	Databases
DC	Digital Count
DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
DPE	Data Provider Element
DPS	Display Performance Standard
DR	Dynamic Range
DRA	Dynamic Range Adjustment
DVD	Digital Versatile Disc
EEI	Essential Element of Information
ELT	Electronic Light Table
EO	Electro-Optical
EDP	Equal Probability of Detection
FDDRD	FIA DPE Dataset Requirements Document
FIA	Future Imagery Architecture
FSG	Full Spectrum GEOINT

GFE	Government Furnished Equipment
GFI	Government Furnished Information
GIF	Graphics Interchange Format
GOTS	Government Off-The-Shelf
GSD	Ground Sampling Distance
HVS	Human Visual System
IC	Intelligence Community
ICC	International Color Consortium
IDEX	Image Data and Exploitation
IDOP	IDS-D Digital Output Product
IDS-D	Information Dissemination Services – Direct Delivery
IEC	Integrated Exploitation Capability
IEC	International Electro-technical Commission
IR	Infrared
ISO	International Standards Organization
J2K	JPEG 2000
JFIF	JPEG File Interchange Format
JND	Just Noticeable Differences
JPEG	Joint Photo Experts Group
KPP	Key Performance Parameter
LCD	Liquid Crystal Display
Lin-Log	Linear/Logarithmic (Data Re-map used in Legacy Radar processing)
LWIR	Long-wave Infrared
LSF	Line-Spread Function
LUT	Look-Up Table
MASINT	Measurement And Signature Intelligence
MSI	Multi-spectral Infrared
MTF	Modulation Transfer Function
MTFC	Modulation Transfer Function Compensation
MTFE	Modulation Transfer Function Enhancement
MTFR	Modulation Transfer Function Restoration
MWIR	Mid-wave Infrared

N/A Not Applicable

NGA National Geospatial-Intelligence Agency

NIQU NGA/TQ Image Quality & Utility

NITF National Imagery Transmission Format
NITFIRD NITF Implementation Requirements Document
NORD NGA Operational Requirements Document
NSG National System for Geospatial-Intelligence
NSGPDD NSG Product Description Document

PCS Profile Connection Space

PEDF Piecewise Extended Density Format
Pmax Percent maximum
Pmin Percent minimum

pps pixels per second
RGB Red Green Blue

RSME Root Mean Square Error

ROI Region of Interest

RRDS Reduced Resolution Data Set

SAR Synthetic Aperture Radar

SEFS Softcopy Exploitation Facility Standard

SFG Softcopy Facilities Guidelines

SIPS Softcopy Image Processing Standard
SQAP Softcopy Quality Assurance Program
sRGB standard Red Green Blue

SYERS Senior Year Electro-Optical Reconnaissance System
SWIR Short-wave Infrared

TFRD Tape Formats Requirement Document

TIFF Tagged Image File Format

TTC Tone Transfer Curve

Appendix A

Databases and Imagery

A.1 Databases

As a supplement to this document, a default set of databases to support the reference image chain defined in this document may be requested. These databases provide flexibility to support many sensors in use at the time this document was created. Table A.1 describes the contents of the organization of the database files available from NIQU.

A.1.1 Database Contents

Table A.1: Database Layout

Directory	Contents
RRDS_AA	The anti-aliasing kernel used for RRDS generation.
TTC	TTC Databases. Note that the file name gives the family and member number (ttc_<family>_<member>.lut). There are 12 families with 64 members to each family. This database should only be used on single-band imagery at this time.
MTFC	MTFC kernel database. Similar to the TTC directory, the files in this directory contain 10 families of kernels with 64 members for each family. This database should be used for the MTFE database for single-band applications until further notice.

(continued on next page)

Directory	Contents
Interpolator	Interpolator database. There is one file in this directory containing the coefficients for Lagrange Interpolation.
MTFR	Empty directory at the time of this release. In the future this will contain any MTF Restoration kernels available for current systems.
DATA_REMAP	1-D LUTs for PEDF and Lin-Log remapping.
ICC_Profiles	This directory currently contains the ICC profiles used by the example in the CMM ADS. In the future, this directory will contain the profiles necessary for compliance with this standard.
Band_Eq	Contains the default 1-D LUT for band equalization (Band_Eq_null.lut) as well as a band equalization 1-D LUT for testing (Band_Eq_inv.lut). This directory contains a set of input imagery that can be used for verification of SIPS

A.1.2 Database Formats

File formats for all kernels delivered with this standard utilize the common kernel file format. The definition of this format is given in Section A.1.2.1, below. 1-D LUTs are defined using the common LUT format provided in Section A.1.2.2 of this appendix. TTC and data remap files are provided in this format.

A.1.2.1 Common Kernel File Format

The common kernel file format fully specifies the kernel using the layout illustrated in Figure A.1. Comment lines are allowed at any position in the file, and they begin with # and end with a newline. The parsed format begins with a header that is enclosed in braces. This header is then followed by a data section that contains the kernel values.

```
# Comments begin with '#'and end with a new line.
{
      type = typeOfKernel [typeOfKernel]
      width = w
      height = h
      [middle = x y]
      [use = {convolution | correlation}]
}
entry1
entry2
    •
    •
    •
entryn
```

Figure A.1: Common Kernel Format Layout

type: (required) Defines the kernel's type. typeOfkernel is a mnemonic that specifies the type of kernel in the data section. The valid values are:

- general - for any kernel where the entire kernel appears in the data section.
- quadsym - for a quadrant symmetric kernel, where only the lower-right quadrant is in the data section.
- prodsep - for a kernel which is product separable, where only horizontal and vertical components are in the data section.

Only prodsep and quadsym may be specified together, in any order, to indicate a kernel which is both product separable and quadrant symmetric. In this case, horizontal and vertical components representing only the lower-right quadrant are specified in the data section.

width: (required) Defines the width of the fully expanded kernel.

height: (required) Defines the height of the fully expanded kernel.

middle: (optional) Defines the center of the kernel. The kernel's center is positioned on a pixel-of-interest after expansion but before any rotation (i.e. use = convolution). The middle value is specified as a pair of one-based indices, where the top-left is origin "1 1". These numbers are separated by a space or tab. This parameter is optional. The default value is "$(width + 1)/2 \quad (height + 1)/2$".

use: (optional) Defines how this kernel is to be used.

- convolution - the kernel is rotated 180 degrees before being applied to the pixel-of-interest. This may have implications of how the kernel overlays the pixel-of-interest and will have implications for kernels with an even width or height. **This is the default.**
- correlation - the kernel overlays the pixel-of-interest without rotation.

Classification: (optional) Defines the text string for the classification of the kernel file. The default is "Unclassified".

Kernel values contained in the data section are separated by white space (combinations of spaces, tabs and new lines). The number of values contained in the data section is dependent on the kernel type. Table A.2 shows the number of expected entries based on the kernel type.

Table A.2: Number of Kernel Values

Type	Number of Entries
general	$width * height$
prodsep	$width + height$
quadsym	$\left(\frac{width+1}{2}\right) * \left(\frac{height+1}{2}\right)$
prodsep quadsym	$\left(\frac{width+1}{2}\right) + \left(\frac{height+1}{2}\right)$

Figure A.2 depicts sample kernel files.

```
{
   type= general
   width= 3
   height= 3
}
0.1  0.1  0.1
0.1  0.2  0.1
0.1  0.1  0.1
```

```
{
   type= quadsym
   width= 5
   height= 5
}
0.11  0.08  0.02
0.08  0.06  0.01
0.02  0.01  0.01
```

```
{
   type= prodsep
   width= 3
   height= 3
}
0.0625  0.125  0. 0625
0.0625  0.125  0. 0625
```

```
{
   type= general
   width= 2
   height= 2
   middle= 1 2
}
   0  -1
   1   0
# coefficient 1 is to be at the
overlay position
```

```
{
   type= prodsep quadsym
   width= 5
   height= 5
}
# Horizontal component vector
1.218  -0.32812  -0.16406
# Vertical component vector
4.1     2.0       0.52
```

Figure A.2: Sample Common Kernel Files

A.1.2.2 Common Look-Up Table (LUT) Format

The common LUT file format specifies a look-up table that is used to convert an input value to a corresponding output value. A LUT array is an array of output values. Each input pixel value is used as an index into the LUT array. The output pixel value is the array element pointed to by this index. The common LUT format provides the definition for generating a LUT array.

Figure A.3 shows the layout of the LUT file format. Comment lines are allowed at any point in the file (except on the same line as the end of section marker), and they begin with # and end with a newline. The file starts with a header that is enclosed in braces. This header is followed by one or more data sections. Each data section is comprised of a number of lines with one entry per line followed by an end of data section marker, $. The end of section marker must be the only symbol on the line. Entries in a data section define the LUT array.

The following keywords are recognized as part of the common LUT file format.

Required fields are marked accordingly.

- **tables:** (Required) Defines the number of data sections (LUTs) contained in the file. The first data section begins immediately after the header's closing brace and ends with the end of data section marker. Subsequent data sections are delimited by the end of data section marker.
- **type:** (Required) Defines the type of LUT. Three LUT types are supported: a simple LUT that is specified by 'l', a breakpoint LUT that is specified by 'b', and a mathematical formula LUT that is specified by 'f'. If the type is l,

```
{
      tables= <number of data sections>
      type= [b | l  | f ]
      length= <a list of the number of elements in each of the data
      sections>
      zeroOffset= <a list of the input values to be matched with the
      first element in each data section>
      dataType= [U8, I16, U16, I32, U32, Float, Double]
}
entry 1
entry 2
      •
      •
      •
entry n1
$
      •
      •
      •
$
entry 1
entry 2
      •
      •
      •
entry nm
$
```

Figure A.3: LUT File Format Layout

each data section entry contains a single output value. If the type is b, each data section entry contains two values; the first is the input value and the second is the output value. In the case of a breakpoint LUT, intermediate values are interpolated in a piecewise linear fashion. A mathematical forumula LUT provides one or more formulae defining a normalized function whose output must be scaled to the appropriate bit-depth of the data being processed.

When specifying a type 'f' LUT, the first row of the data section specifies a normalized list of input breakpoints. If no breakpoints are required, the values 0 and 1 should be used on this line. For N breakpoints, there should be $N - 1$ lines containing a quoted string where each line contains a mathematical function definition for an input domain between two successive break- points. The format for such functions is described below.

- **length:** A list of the number of elements (or lines of data) in each of the data sections. The length of the list is defined by the value of the tables keyword. This keyword is not required for the mathematical formula ('f') type.

- **zeroOffset:** A list of indices to be matched with the first element in each data section.

- **dataType:** The output data type: may be one of the following: *U8, I16, U16, I32, U32, Float, Double*. If not present, for types 'l' and 'b' the output data type is inferred from the output values. If any output values have a fractional part (digits to the right of the decimal point), the data type is set to *Double*. For type 'f' the dataType (and resulting range) is explicitly chosen by the application applying the LUT. Common methods for selecting the appropriate scale factor/dataType include inference drawn from image metadata or explicit selection by the end user.

```
# A simple lut
{
    tables=1
    type=l
    length=256
}

255
254
253
                256 entries
2
1
0
$
```

```
# A simple break-
# point lut
{
    tables=1
    type=b
    length=2
}

0 255
255 0
$
```

Figure A.4: Sample LUT Files

Figure A.4 shows two LUT files, both of which reverse the values of unsigned character data. The first one explicitly remaps each input value to its inverse ($outputvalue = 255 - inputvalue$). That is, 0 maps to 255, 1 maps to 254, and so on until 255 maps to 0. The second LUT specifies the same inverse relationship using breakpoints. In this instance, 0 maps to 255 and 255 maps to 0. All intermediate values are calculated.

```
#
# Mathematical Formula LUT Example
#
{
       tables=1
       type=f
       length=1
}
0 1
"0.18071*x-0.27013*x*x+1.637335*pow(x,3)-
1.79035*pow(x,4)+1.242432*pow(x,5)"
$
```

Figure A.5: Sample Math Formula LUT

Figure A.5 shows an example of a math formula LUT. While this example uses a 5th order polynomial with a domain and range of [0,1], many other mathematical functions are permitted in the LUT definition. Figure A.6 provides a comprehensive list of scalar functions that may be used to create a mathematical formula based LUT.

Domain and Range of the LUT generated from a mathematical formula must be scaled based on the data type and min/max of the input image data. In cases where this scaling is not well-defined, the user should specify the scaling factor necessary for proper application to the data.

```
Trig Functions (angles are in radians)
    acos   cos    cosh
    asin   sin    sinh
    atan   tan    tanh
    atan2

Bessel Functions
    j0     y0
    j1     y1
    jn     yn

Math Functions
    abs    fabs   hypot  log10
    ceil   floor  ldexp  pow
    exp    fmod   log    sqrt

Statistics Functions
    max    - select the maximim value, takes 2 or more args
    min    - select the minimim value, takes 2 or more args
    mean   - compute the arithetic mean, takes 2 or more args
    median - select the (lower) middle value, takes 2 or more args

Other Functions
    clip(x, min, max) - clips x to the range [min,max]
    round(x)          - rounds x to the nearest integer, at the midpoint
                        the farther integer from zero is chosen
    trunc(x)          - integerizes by removing the fractional component

Builtins (in order of increasing precedence)
    conditional expr  ?:
    logical OR        ||
    logical AND       &&
    bitwise OR        |
    bitwise XOR       ^
    bitwise AND       &
    equality          ==  !=
    relational        <  >  <=  >=
    shift             <<  >>
    additive          +  -
    multiplicative    *  /  %
    unary             !  +  -  ~

Error Checking Functions
    finite(x)             - returns x if x is finite else errors
    finite(x,replacement) - returns x if x is finite else returns replacement
    nan(x)                - returns x if x is a valid number else errors
    nan(x,replacement)    - returns x if x is a valid number else returns
                            replacement
```

Figure A.6: Scalar Math Functions Based on "C" Programming Language

A.2 Test Imagery

As a supplement to this document, a default set of test imagery processed through the reference image chain may be obtained for compliance testing. The following is a list of input image filenames, the settings used by the reference image chain for processing, and the filename of the output generated. Any elements of the image chain that are not specifically mentioned with parameters are assumed to be turned off.

1. Input file: `Pan_Verification_Target_A_8.tif`

   Output file: `Pan_Verification_Target_A_8_processed.tif`

   Processing Parameters:

   - **DRA**: $p_{min} = 1.0\%, p_{max} = 99\%, A = 0.2, B = 0.4$
   - **CMM**:
     - **Sensor Profile:** sRGB_IEC61966-2-1_420.icc
     - **Display Profile:** sRGB_IEC61966-2-1_420.icc
2. Input file: `Pan_Verification_Target_A_10.tif`
   Output file: `Pan_Verification_Target_A_10_processed.tif`
   Processing Parameters:

   - **MTFR:** MTFR_test.ker
   - **DRA**: $p_{min} = 1.0\%, p_{max} = 99\%, A = 0.2, B = 0.4$
   - **CMM**:
     - **Sensor Profile:** sRGB_IEC61966-2-1_420.icc
     - **Display Profile:** sRGB_IEC61966-2-1_420.icc
3. Input file: `Multi_band_targetA_8.tif`
   Output file: `Multi_band_targetA_8_processed.tif`
   Processing Parameters:

   - **CMM, MTFE, TTC**
     - **Sensor Profile:** ProPhoto_420.icc
     - **Working Space Profile:** WideGamutRGB.icc
     - **Display Profile:** sRGB_IEC61966-2-1_420.icc
     - **TTC:** Family 0, Member 10
   - **DRA**: $p_{min} = 2.0\%, p_{max} = 99\%, A = 0.1, B = 0.5$
4. Input file: `Multi_band_targetA_16.tif`
   Output file: `Multi_band_targetA_16_processed.tif`
   Processing Parameters:

- **Band Equalization:** Band_eq_inv.lut (this is a breakpoint LUT that inverts the values in each band)

5. Input file: `Single_band_targetB_8.tif`
   Output file: `Single_band_targetB_8_processed.tif`
   Processing Parameters:
   - **TTC**: Family 0, Member 32
   - **DRA**: $p_{min} = 2.0\%, p_{max} = 99\%, A = 0.1, B = 0.5$
   - **CMM**:
     - **Sensor Profile:** sRGB_IEC61966-2-1_420.icc
     - **Display Profile:** sRGB_IEC61966-2-1_420.icc
6. Input file: `Single_band_targetB_14.tif`
   Output file: `Single_band_targetB_14_processed.tif`
   Processing Parameters:
   - **MTFE**: Family 0, Member 32
   - **DRA**: $p_{min} = 2.0\%, p_{max} = 99\%, A = 0.1, B = 0.5$
   - **CMM**:
     - **Sensor Profile:** sRGB_IEC61966-2-1_420.icc
     - **Display Profile:** sRGB_IEC61966-2-1_420.icc
7. Input file: `09NOV18022743-M1BS-052297602010_01_P001_chip.tif`

   Output file: `09NOV18022743-M1BS-052297602010_01_P001_chip_processed.tif`

   Processing Parameters:
   - **Bands**: RYGBC
   - **TTC**: Family 12, Member 14
   - **DRA**: Band Dependent Linearization
   - **CMM**:
     - **Sensor Profile:** WV2_HVS.icc
     - **Display Profile:** sRGB_IEC61966-2-1_420.icc
8. Input file: `Pan_Verification_Target_A_11.tif`

   Output files: `Pan_Verification_Target_A_11_R1.tif`
   `Pan_Verification_Target_A_11_R2.tif`
   `Pan_Verification_Target_A_11_R3.tif`
   `Pan_Verification_Target_A_11_R4.tif`
   `Pan_Verification_Target_A_11_R5.tif`

   Processing Parameters:
   - **Rset Generation**

A.3 Default Settings for Various Sensors

Table A.3 describes the default settings for unspecified and selected systems. These default settings define the initial display settings only and should not be construed to limit the user from alternate selections. Information for additional systems may be obtained by contacting NIQU.

The 'Unspecified' entries of Table A.3 cover the default processing recommended for products or sources that are not individually listed. Therefore, un- less specifically listed in Table A.3, a product or source should utilize the settings for an 'Unspecified' source having the band representation and per-band bit-depth corresponding to the product being processed.

Tables A.4 and A.5 contain predefined non-interactive and interactive processing chains, respectively. These chains are referenced in any appendix needing a standard set of default processing parameters for the non-interactive or interactive portion of the reference image chain. Please note that for these tables an entry of "–" means that this operation does not need to be performed for initial display.

Table A.3: Default Initial Display Settings

Source	Band Representation	bpp Per Band	Non-Interactive Chain	Interactive Chain
Unspecified	# Bands : 1 1 : Gray	8[1]	NI-0	IA-0
SAR Default	# Bands : 1 1 : Gray	8[1]	NI-0	IA-2
Unspecified	# Bands : 1 1 : Red 2 : Green 3 : Blue	8[2]	NI-4	IA-0
Unspecified	# Bands : 1 1 : Red 2 : Green 3 : Blue	>8[2]	NI-4	IA-0
Unspecified	# Bands : 3[3] 1 : Red 2 : Green 3 : Blue	8	NI-0	IA-0
Unspecified	# Bands : 3[3] 1 : Red 2 : Green 3 : Blue	>8	NI-0	IA-0
Scanned hardcopy	# Bands : 1 1 : Any	>8	NI-0	IA-0
SYERS-2 (All band combinations)	# Bands : 7 : See : Figure : A.7	8[4]	NI-0	IA-0
Global Hawk (Pan, SAR)	# Bands : 1 1 : Any	8	NI-0	IA-0

(continued on next page)

Source	Band Representation	bpp Per Band	Non-Interactive Chain	Interactive Chain
Global Hawk (IR)	# Bands : 1 1 : Any	12	NI-0	IA-0
Quickbird (1 & 2)	# Bands : 1 1 : Pan	>8	NI-0	IA-1
Quickbird (1 & 2)	# Bands : 4 1 : Red 2 : Green 3 : Blue	>8	NI-0	IA-1
Quickbird (1 & 2)	# Bands : 4 1 : NIR 2 : Red 3 : Green	>8	NI-0	IA-1
Ikonos	# Bands : 1 1 : Pan	>8	NI-0	IA-1
Ikonos	# Bands : 4 1 : Red 2 : Green 3 : Blue	>8	NI-0	IA-1
Ikonos	# Bands : 4 1 : NIR 2 : Red 3 : Green	>8	NI-0	IA-1
GeoEye-1	# Bands : 1 1 : Pan	>8	NI-0	IA-1
GeoEye-1	# Bands : 4 1 : Red 2 : Green 3 : Blue	>8	NI-0	IA-1
GeoEye-1	# Bands : 4 1 : NIR 2 : Red 3 : Green	>8	NI-0	IA-1
ClearView	# Bands : 1 1 : Pan	>8	NI-0	IA-1
ClearView	# Bands : 4 1 : Red 2 : Green 3 : Blue	>8	NI-0	IA-1
ClearView	# Bands : 4 1 : NIR 2 : Red 3 : Green	>8	NI-0	IA-1
NextView	# Bands : 1 1 : Pan	>8	NI-0	IA-1
NextView	# Bands : 4 1 : Red 2 : Green 3 : Blue	>8	NI-0	IA-1
NextView	# Bands : 4 1 : NIR 2 : Red 3 : Green	>8	NI-0	IA-1
WorldView-1	# Bands : 1 1 : Pan	>8	NI-0	IA-1

(continued on next page)

Source	Band Representation	bpp Per Band	Non-Interactive Chain	Interactive Chain
WorldView-1	# Bands : 4 1 : Red 2 : Green 3 : Blue	>8	NI-0	IA-1
WorldView-1	# Bands : 4 1 : NIR 2 : Red 3 : Green	>8	NI-0	IA-1
WorldView-2	# Bands : 1 1 : Pan	>8	NI-0	IA-1
WorldView-2[5]	# Bands : 8 1 : Red 2 : Green 3 : Blue	>8	NI-0	IA-1
WorldView-2[5]	# Bands : 8 1 : NIR 2 : Red 3 : Green	>8	NI-0	IA-1
OrbView-(1,2,&3)	# Bands : 1 1 : Pan	>8	NI-0	IA-1
OrbView-(1,2,&3)	# Bands : 4 1 : Red 2 : Green 3 : Blue	>8	NI-0	IA-1
OrbView-(1,2,&3)	# Bands : 4 1 : NIR 2 : Red 3 : Green	>8	NI-0	IA-1

[1]No color map present in image metadata.

[2]Color map present in image metadata. Color map should be applied to generate a three band image.

[3]Bands are assumed to represent Red, Green, and Blue. Unspecified source prevents additional knowledge.

[4]Syers-2 default composite band combinations are identified in Figure A.7

[5]Worldview-2 has capabilities for additional band combinations that have not been evaluated as of this version of SIPS.

BANDS	Default Composite	DESCRIPTION	RED	GREEN	BLUE
MID-WAVE INFRARED	CMC	CUSTOM MULTISPECTRAL COMPOSITE	MWIR & SW2	SW1 & NIR	RED
	IRC	INFRARED COMPOSITE	MW1/2	SW1	SW2
	MX2	MIXED COMPOSITE 2	MW1/2	NIR	SW2
	MX3	MIXED COMPOSITE 3	MW1/2	NIR	GRN
	MX4	MIXED COMPOSITE 4	MW1/2	NIR	SW1
SHORT-WAVE INFRARED	MXC	MIXED COMPOSITE	SW2	NIR	GRN
	MX1	MIXED COMPOSITE 1	SW2	NIR	SW1
VISIBLE	VSC	VISIBLE COMPOSITE	NIR	RED	GRN

Figure A.7: SYERS Band Composites

Table A.4: Default Non-Interactive Image Chain Settings

Chain ID	Band Equalization	MTFR	Data Remap	RRDS
NI-0	–	–	–	**Alg**:Standard RRDS Generation
NI-1	–	–	–	**Alg**: Max. Pixel Decimation
NI-2	–	–	Alg : 1-D LUT LUT: PEDF	**Alg**:Standard RRDS Generation
NI-3	–	–	Alg : 1-D LUT LUT: LinLog	**Alg**:Standard RRDS Generation
NI-4	–	–	Alg :Color Remap LUT: PEDF	**Alg**:Standard RRDS Generation
NI-5	–	–	Alg :Color Remap LUT: Embedded	**Alg**: Max. Pixel Decimation
NI-6	–	–	Alg : 1-D LUT LUT: PEDF	**Alg**: Max. Pixel Decimation
NI-7	–	–	Alg : 1-D LUT LUT: LinLog	**Alg**: Max. Pixel Decimation

Table A.5: Default Interactive Image Chain Settings

Chain ID	Geometric Transformation	CST	DRA	TTC	MTFE
IA-0	**Alg** : 4-point Interp. **Coeff**: LaGrange	embedded[1]	**Alg** :AutoDRA **sub** : 0.0 **mul** : 1.00	**Alg** :1-D LUT **Fam** : 0 **Mem**: 32	**Alg** :Convolution **Fam** : 0 **Mem**: 5
IA-1	**Alg** : 4-point Interp. **Coeff**: LaGrange	embedded[1]	**Alg** :AutoDRA **Pmin** : 2.50% **Pmax**: 99.00% **A** : 0.1 **B** : 0.5	**Alg** :1-D LUT **Fam** : 0 **Mem**: 10	**Alg** :Convolution **Fam** : 0 **Mem**: 5
IA-2	**Alg** : 4-point Interp. **Coeff**: LaGrange	embedded[1]	**Alg** :AutoDRA **sub** : 0.0 **Pmax**: 99.95%	**Alg** :1-D LUT **Fam** : 0 **Mem**: 32	**Alg** :Convolution **Fam** : 0 **Mem**: 5

[1] If an ICC profile is not present in the image metadata, the system display profile should be used. If the system display profile is not present, the sRGB profile should be used.

Table A.6: Default Parameter Ranges and Precisions

Variable	Range
Subtractor	$0 - 2047$
Multipler	$0.0000 - 2047.0000$
p_{min}	$0.0000 - 1.0000$
p_{max}	$0.0000 - 1.0000$
A	$0.0 - 1.0$
B	$0.0 - 1.0$

Any time that a Subtractor and Multiplier is provided in Table A.5, the automated Subtractor and Multiplier calculation of the Auto-DRA algorithm, described in the SIPS document, should be bypassed in favor of the default settings for these values. Classified Appendix B Table B.4 "DMID-Based DRA Parameters" identifies the specific Multiplier and Subtractor values to use.

A.4 Data Remapping

Global Hawk and SYERS-2A Tactical Airborne sensor platforms provide imagery having bit-depths other than 11 bits per pixel. The various bit-depths used by these sensors reflect their native bit-depths. These data sources should be linearly re-mapped to an 11-bit space prior to enhancement processing.

Multi-band products, such as the SYERS-2A Color Composite products, having 3-bands in a 24-bit color space (standard "RGB" image representation) do not require data re-mapping to 11-bits per pixel per band, and can be displayed as

standard 24-bit color imagery. Future exceptions to this rule, if they occur, will be noted specifically in future revisions of this document.

A.4.1 Global Hawk VIS, IR, and SAR Bit-Depth Re-Mapping

Global Hawk visible panchromatic (VIS) and SAR imagery should be linearly re-mapped from their native 8-bit space to an 11-bit space before applying the processing steps outlined in the SIPS document.

Global Hawk IR imagery should be linearly re-mapped from its native 12-bit space to an 11-bit space before applying the processing steps outlined in the SIPS document.

A.4.2 SYERS-2A Visible and Infrared Bit-Depth Re-Mapping

SYERS-2A green (GRN), red (RED), and near-infrared (PAN) imagery, when displayed independently and not as part of a three-band composite product (VSC, MXC, MX1, MX2, MX3, and MX4) or Multi-band Product (MBP), should be linearly re-mapped from their native 10-bit space to an 11-bit space before applying the processing steps outlined in the SIPS document.

SYERS-2A short-wave-infrared (SW1 and SW2) and mid-wave-infrared (MW1 and MW2) imagery, when displayed independently and not as part of a three-band composite product (IRC, MXC, MX1, MX2, MX3, and MX4) or MBP, should be linearly re-mapped from their native 14-bit space to an 11-bit space before applying the processing steps outlined in the SIPS document.

At this time, this appendix makes no recommendation for the processing and presentation of SYERS-2A Measurements and Signals Intelligence (MASINT) products such as MBP, and Multi-band Product Raw (MBR). These products present multiple spectral bands as separate single-band regions in an image plane mosaic, and not as a traditional "colorized" multiple-band image. Oftentimes, these single-plane multiple-image mosaics mix regions of 10-bit and 14-bit image data that do not lend themselves to a simple bit-depth re-map operation.

This page intentionally includes only this statement.

Appendix B

Reserved

This appendix and associated addendum are reserved. For more information please reference NIQU Standards on Intellipedia at https://intellipedia.intelink.ic.gov/wiki/NIQU_Standards_%26_Scales_Support (on classified website).